Django Project

01

Создание Django проекта

$ pip install django

$ django-admin --version

$ django-admin startproject Django\_Project\_1

>>> python manage.py runserver - из папки проекта

http://127.0.0.1:8000/ или http://localhost:8000/ - в браузере

Файлы проекта в Pycharm:

Django\_Project\_1

manage.py

Django\_Project\_1

2 settings.py - INSTALLED\_APPS=[‘bboard.apps.BboardConfig’]

4 urls.py - urlpatterns=[

path(‘bboard/’, index), # 1

6 path(‘bboard/’, include (‘bboard.urls’)), # 2

]

1 bboard - (Pycharm Terminal > python manage.py startapp bboard)

apps.py

7 models.py - class Bb(models.Model):

title = models.CharField(max\_length = 50)

content = models.TextField(null=True, blank=True) #

price = models.FloatField(null=True, blank=True)

published = models.DateTimeField(auto\_now\_add=True, null=True, db\_index=True)

…

3 view.py - def index(request):

return HttpResponse(“Здесь будет текст.”)

5 создать urls.py - urlpatterns=[path(‘’, index)]

8 migrations - > python manage.py makemigrations bboard

> python manage.py migrate

Миграции

Просмотр миграций:

В текстовом редакторе открываем файл: 00001 initial.py

В виде SQL-кода: python manage.py sqlmigrate bboard 00001

Установить DataBase Navigator

Суперпользователь # 45

> python manage.py createsuperuser

Name: admin

Password: admin

В settings.py LANGUAGE\_CODE = ‘ru’

В браузере http://localhost:8000/admin

9 В файле приложения bboard/admin.py

from django.contrib import admin

from .models import Bb

admin.site.register(Bb) # регистрируем приложение bboard в

административном сайте admin

Работа с моделями

В терминале Pycharm :

> python manage.py shell

Out

>>>

Создаем первое объявление

>>> from bboard.models import Bb

>>> b1 = Bb(title=’Дача’, content=’Общество “Двухэтажники”. ’ + ‘Два этажа, кирпич, газ, свет.’, price=20000)

>>> b1.save() - сохранение объявления

>>> b1.k - узнаем, сохранилась ли модель, Узнаем ключевой номер модели.

Out

>>> 1 - атрибут класса pk хранит значение ключового поля те кущей записи.

Можно обратиться к любому полю записи:

>>> b1.title

>>> b1.content

>>> b1.price

>>> b1.published

Можно создавать объявления другими способами:

>>> b2 = Bb()

>>> b2.title = ‘Автомобиль’

>>> b2.content = ‘Жигули’

>>> b2.price = 10000

>>> b2.save()

>>> b2.k

Добавление данных:

>>> b2.content = ‘Жигули, небитая, некрашенная, год выпуска 1969.’

Еще вариант создавать объявления:

>>> Bb.objects.create(Title=’Дом’, content ‘1-ый этаж, газ, свет.’, price=30000)

Выведем ключи и заголовки всех объявлений:

>>> from b in Bb.object.all():

… print(b.pk, ‘: ‘, b.title)

Out:

1 : Дача

2 : Автомобиль

3 : Дом

Сам набор записей представляется экземпляром класса QuerySet, а отдельные записи - екземплярами соответствующего класса модели.

Сортировка модели по заголовку

>>> for b in Bb.objects.order\_by(‘title’):

… print(b.pk, ‘: ‘, b.title)

Out:

2: Автомобиль

1: Дача

3: Дом

Извлечение объявлений по продаже домов:

>>> for b in Bb.objects.filter(title=’Дом’)

… print(b.pk, ‘: ‘, b.title)

Out:

3: Дом

стр. 41

Изменяем код пакета views.py приложения bboard:

def index(request): # стр. 41

s = ‘Список об’явлений\r\n\r\n\r\n’

for bb in Bb.objects.order\_by(‘-published’):

s += bb.title + ‘\r\n’ + bb.content + ‘\r\n’

return HttpResponse(s, content\_type=’text/plain; charset=utf-8’)

…

Функция all() возвращает True, если все элементы в итерируемом объекте итерации истинны, в противном случае она возвращает False.

Шаблоны # 42

Создать bboard/templates/bboard/index.html

<!DOCTYPE!>

<html>

<head>

<meta charset=”UTF-8”>

<title>Главная :: Доска об’явлений</title>

</head>

<body>

<h1>Оъявления</h1>

{% for bb in bbs %}

<div>

<h2>{{ bb.title }}</h2>

<p>{{ bb.content }}</p>

<p>{{ bb.published|date:’d.m.Y. H:i:s’ }}</p>

</div>

{% endfor %}

</body>

</html>

В модуле views.py вносим изменения:

def index(request): # 43

template = loader.get\_template(‘bboard/index.html’) # загружаем шаблон

# с помощью функции get\_template

bbs = Bb.objects.order\_by(‘-published’) #

context = {‘bbs’: bbs} # контекст - в виде словаря Питона

# элемент bbs будет преобразован в переменную bbs

return HttpResponse(template.render(context, request)) #

# выполняем рендеринг шаблона

В модуле views.py вносим изменения - с функцией сокращения render:

def index(request): # 45

bbs = Bb.objects.order\_by(‘-published’) # с функцией сокращения render

return render(request, ‘bboard/index.html’, {‘bbs’: bbs})

Запускаем сервер.

from . models import Bb

admin.site.register(Bb) # регистрируем приложение Bb на административном

# сайте admin

стр.50

Добавляем русские названия полей модели

В модуле models.py:

class Bb(models.Model):

title = models.CharField(max\_length = 50, verbose\_name=’Товар’)

content = models.TextField(null=True, blank=True, verbose\_name=’Описание’) #

price = models.FloatField(null=True, blank=True, verbose\_name=’Цена’)

published = models.DateTimeField(auto\_now\_add=True, null=True, db\_index=True,

verbose\_name=’Опубликовано’)

class Meta:

verbose\_name\_plural = ’Объявления’

verbose\_name = ’Объявление’

ordering = [‘-published’]

Параметры моделей

с помошью

Редактор модели с помощью класса-редактора.

В модуле admin.py:

from django.contrib import admin

from .models import Bb

class BbAdmin(admin.ModelAdmin): #

list\_display = (‘title’, ‘content’, ‘price’, ‘published’) # имена полей, которые

# выводятся в списке записей

list\_display\_links = (‘title’, ‘content’) # имена полей, которые

# преобразуются в гиперссылки

search\_fields = (‘title’, ‘content’) # по этим полям по которым

# будет делаться фильтрация

admin.site.register(Bb, BbAdmin) #

Изменяем код пакета views.py приложения bboard:

стр. 51

# уберем из старого выражения, извлекающего список записей, указание сортировки:

def index(request): # стр. 51

bbs = Bb.objects.all()

Редактор модели стр. 51

В admin.py

from django.contrib import admin  
from .models import Bb  
from .models import Rubric  
  
class BbAdmin(admin.ModelAdmin): # стр. 51  
 list\_display = ('title', 'content', 'price', 'published', 'rubric') # имена полей, которые выводятся в списке записей  
 list\_display\_links = ('title', 'content') # имена полей, которые преобразуются в гиперссылки  
 search\_fields = ('title', 'content', ) # по этим полям по которым будет делаться фильтрация  
  
  
admin.site.register(Bb, BbAdmin)  
# admin.site.register(Rubric) # стр 55

**Копируем весь проект на GitHub**

**1 способ**

В Pycharm - settings - Version control - Directory mappings

(В окне справа выбираем папку, которую будем опубликовывать на GitHub)

VCS - выбираем Git

или

В Pycharm - settings - Version control - GitHub

нажать (+) и залогиниться с помощью Гитхаба (Git in via GitHub) или токена (Git in with Token)

**2 способ**

В Pycharm - VCS (это ве́ршенл контро́л) -

у меня В Pycharm - view - Tools - Version control -

В открывшемся окне - Share project on - GitHub

**Связи, ввод даннях. Статические файлы стр. 53**

Связи между моделями

В models.py приложения bboard

# Создаем модель “рубрика”

class Rubric (models.Model): # 53

name = models.CharField(max\_length=20, db\_index=True, verbose\_name=’Название’)

class Meta:

verbose\_name\_plural = ’Рубрики’

verbose\_name = ’Рубрика’

ordering = [‘name’]

Теперь добавляем в модель Bb поле внешнего ключа, устанавливаем связь один-со-многими с моделью Rubric.

модель Rubric - первичная

модель Bb - вторичная

В модели Bb создаем поле rubric:

class Bb(models.Model): #53

…

rubric = models.ForeignKey(‘Rubric’, null=True,

on\_delete=models.PROTECT, verbose\_name=’Рубрика’)

class Meta:

…

> python manage.py makemigrations bboard

> python manage.py migrate

Регистрируем новую модель на административном сайте

В модуле admin.py пакета приложения bboard

from .models import Rubric

admin.site.register(Rubric)

стр. 55

запускам сервер.

Строковое представление модели

Чтобы в браузере http://127.0.0.1:8000/admin/bboard/rubric/ рубрики не отображались в виде

<имя класса модели>object(<значение ключа записи>),

переделаем отображение в строковом виде.

В модуле models.py добавим в объявление класса модели код:

class Rubric(model.Models):

…

def \_\_str\_\_(self):

return self.name

class Meta:

…

Чтобы рубрики выводились в списке записей объявлений:

добавим в bboard/admin.py поле rubric:

стр. 56

class BbAdmin(admin.ModelAdmin): #

list\_display = (‘title’, ‘content’, ‘price’, ‘published’, ‘rubric’)

# имена полей, которые

# выводятся в списке записей

**URL - параметры и параметризованные запросы**

Сделаем разбиение по рубрикам не только при хранении, но и при выводе на экран.

Контроллер выводящий объявления относящиеся только к выбранной рубрике должен получить ключ рубрики. Его можно передать через GET-параметр: /bboard/?rubric=<ключ рубрики>

Джанго позволяет поместить параметр непосредственно в составе интернет-адреса bboard/<ключ рубрики>/ т.е. URL-параметр.

Укажем маршрутизатору, какую часть интернет-адреса считать URL-параметром, тип значения этого параметра и имя.

В модуле приложения urls.py

urlpatterns = [ # 57

path(‘<int:rubric\_id>/’, by\_rubric),

path(‘’, index),

]

# <int:rubric\_id/> URL-параметр

# rubric\_id - имя параметра контролера, котрому будет присвоено значение

# by\_rubric - контроллер-функция, который сейчас напишем

#

# Получив запрос по интернет-адресу http://localhost:8000/bboard/2/ маршрутизатор выделит путь bboard/2/, удалит из него префикс bboard и выяснит, что полученный путь совпадает с первым маршрутом из приведенного выше списка. Запустит контроллер by\_rubric передав ему ключ 2

Напишем контролер

В views.py:

def by\_rubric(request, rubric\_id): # 57

bbs = Bb.objects.filter(rubric=rubric\_id)

rubrics = Rubric.objects.all()

current\_rubric = Rubric.objects.get(pk=rubric\_id)

context = {‘bbs’: bbs, ‘rubrics”: rubrics,

‘current\_rubric’:current\_rubric}

return render(request, ‘bboard/by\_rubric.html’, context)

В папке templates/bboard создадим шаблон by\_rubric.html:

стр. 58

<!DOCTYPE html>

<html lang="en">

<head>  
 <meta charset=”UTF-8”>

<title>{{ current\_rubric.name }} :: Доска объявлений</title>

</head>  
 <body>  
 <h1>Оъявления</h1>  
 <h2>Рубрика: {{ current\_rubric.name }}</h2>

<div>

<a href=”/bboard/”>Главная</a>

{% for rubric in rubrics %}

<a href=”/bboard/{{ rubric.pk }}/”{{ rubric.name }}</a>>

{% endfor %}

</div>

{% for bb in bbs %}

<div>

<h2>{{ bb.title }}</h2>

<p>{{ bb.content }}</p>

<p>{{ bb.published|date:'d.m.Y. H:i:s' }}</p>

</div>

{% endfor %}

</body>  
</html>

current - текущий

charset - кодировка

Исправим контроллер index():

def index(request): # стр. 59

bbs = Bb.objects.all()  
 rubrics = Rubric.objects.all()

context = {'bbs': bbs, ‘rubrics’: rubrics}   
 return render(request, ‘bboard/index.html’, context)

Исправим шаблон bboard/index.html:

стр. 59

<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta charset=”UTF-8”>

<title>Главная :: Доска объявлений</title>

</head>  
 <body>  
 <h1>Объявления</h1>

<div>

<a href=”/bboard/”>Главная</a>

{% for rubric in rubrics %}

<a href=”/bboard/{{ rubric.pk }}/”>{{ rubric.name }}</a>

{% endfor %}

</div>

{% for bb in bbs %}

<div>  
<h2>{{ bb.title }}</h2>

<p>{{ bb.content }}</p>

<p><a href=”/bboard/{{ bb.rubric.pk }}/”>{{ bb.rubric.name }}</a></p>

<p>{{ bb.published|date:'d.m.Y. H:i:s' }}</p>

</div>  
 {% endfor %}

</body>

</html>

Обратное разрешение интернет-адресов

стр. 60

Рубрики формируются с помощью <p><a href=”/bboard/{{ rubric.pk }}/”>{{ rubric.name }}</a></p>. Здесь страница указана префиксом явно - /bboard. Чтобы формировать другие страницы, например bulletinboard, нам не обходимо Обратное разрешение интернет-адресов.

Сначала даем маршрутам имена - именованные маршруты.

В bboard/urls.py:

urlpatterns = [ # стр 61  
 path('<int:rubric\_id>/', by\_rubric, name=’by\_rubric’),

path('', index, name=’index’),

Вставляем в код гиперссылок теги шаблонизатора url, которые и выполнят обратное разрешение интернет-адресов.

В bboard/index.html:

было:

<a href=” /bboard/{{ rubric.pk }}/”>

заменим на:

<a href=”{% url ‘by\_rubric’ rubric.pk %}”></a>

<!--

‘by\_rubric’ - имя маршрута,

rubric.pk - значение URL-параметра

-->

в этом же файле найдем: строка 10:

<a href='/bboard/'>

Заменим на:

<a href=”{% url ‘index’ %}”>

<!-- маршрут index не является параметризованным, потому URL-параметры здесь не указываются

-->

получается bboard/index.html: стр.61

</head>  
<body>  
 <h1>Объявления</h1>  
 <div>  
 <!-- <a href='/bboard/'>Главная</a> -->  
 <a href='{% url "index" %}'>Главная</a>  
 {% for rubric in rubrics %}  
 <!-- <a href='/bboard/{{ rubric.pk }}/'>{{ rubric.name }}</a> -->  
 <a href='{% url "by\_rubric" rubric.pk %}'></a>  
 {% endfor %}  
 </div>

Проверяем изменения в браузере.

Формы, связанные с моделями.

стр. 61

Создаем страницу форму для добавления новых объявлений. Дадим ее

ее классу имя BbForm.

Создать bboard/form.py, в нем:

from django.forms import ModelForm

from .models import Bb

class BbForm(ModelForm): # стр 61

class Meta:

model = Bb

fields = (‘title’, ‘content’, ‘price’, ‘rubric’)

Обрабатывать формы можно в контролерах-функциях, но лучше использовать высокоуровневый контролер-класс. он будет носить имя BbCreateView

from django.views.generic.edit import CreateView

from .forms import BbForm

class BbCreateView(CreateView): # стр 62

template\_name = ‘bboard/create.html’

form\_class = BbForm

success\_url = ‘/bboard/’

def get\_context\_data(self, \*\*kwargs):

context = super().get\_context\_data(\*\*kwargs)

context[‘rubrics’] = Rubric.objects.all()

return context

# template\_name - путь к шаблону файла

# form\_class - ссылка на класс формы, связвнной с моделью

# success\_url - интернет-адрес для перенаправления после успешного сохранения данных (у нас - адрес главной страницы)

# context = super().get\_context\_data(\*\*kwargs) т.к. на каждой странице должен находиться перечень рубрик, мы переопределили get\_context\_data формирующего контекст шаблона. В теле метода мы получаем контекст шаблона от метода базового класса, добавляем в него список рубрик и возвращаем его в качестве результата.

Пишем шаблон bboard.create.html:

стр. 63

<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta charset=”UTF-8”>

<title>Главная :: Доска объявлений</title>

</head>  
 <body>  
 <h1>Добавление объявления</h1>

<div>

<a href=”{% url ‘index’ %}”>Главная</a>

{% for rubric in rubrics %}

<a href=”{% url ‘by\_rubric’ rubric.pk %}”>{{ rubric.name }}</a>

{% endfor %}

</div>

<form method=”post”>

{% csrf\_token %}

{{ form.as\_p }}

<input type=”submit” value=”Добавить”>

</form>

</body>

</html>

добавим в bboard/urls.py

CreateView

…

urlpatterns = [

path(‘add/’, BbCreateView.as\_view(), name=’add’) # стр 64

]

Создать на панели всех страниц гиперссылку на страницу доски объявления:

<a href=”{% url ‘add’ %}”>Добавить</a>

Запустить сервер

стр. 65

Исправим в классе BbCreateView интернет-адрес перенаправления (success\_url) с непосредственного на адрес обратного разрешения.:

…  
from django.urls import reverse\_lazy

class BbCreateView(CreateView): # стр.64

…

success\_url = reverse\_lazy(‘index’)

…

Наследование шаблонов

стр. 66

Создать папку templates/layout в ней basic.html

<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <meta charset=”UTF-8”>  
 <title>{% block title %}Главная{% endblock title %} :: Доска объявлений</title>  
 </head>  
 <body>  
 <header>  
 <h1>Объявления</h1>  
 </header>  
 <nav>  
 <a href="{% url 'index' %}">Главная</a>  
 <a href="{% url 'add' %}">Добавить</a>  
 {% for rubric in rubrics %}  
 <a href="{% url 'by\_rubric' rubric.pk %}">{{ rubric.name }}</a>  
 {% endfor %}  
 </nav>  
 <section>  
 {% block content %}  
 {% endblock %}  
 </section>  
 </body>  
</html>

Изменяем bboard/index.html

стр . 67

<!DOCTYPE html>  
<html lang="en">  
  
{% extends "layout/basic.html” %}  
{% block content %}  
  
 {% for bb in bbs %}  
 <div>  
 <h2>{{ bb.title }}</h2>  
 <p>{{ bb.content }}</p>

<p><a href='{% url “by\_rubric” bb.rubric.pk %}'>{{ bb.rubric.name }}></a></p>  
 <p>{{ bb.published|date:'d.m.Y. H:i:s' }}</p>  
 </div>  
 {% endfor %}  
{% endblock %}  
</html>

Изменяем bboard/by\_rubric.html

стр . 67

<!DOCTYPE html>  
<html lang="en">  
  
{% extends "layout/basic.html” %}  
{% block title %}{{ current\_rubric.name }}{% endblock %}  
{% block content %}  
 <h2>Рубрика: {{ current\_rubric.name }}</h2>  
 {% for bb in bbs %}  
 <div>  
 <h2>{{ bb.title }}</h2>  
 <p>{{ bb.content }}</p>  
 <p>{{ bb.published|date:'d.m.Y. H:i:s' }}</p>  
 </div>  
 {% endfor %}  
{% endblock %}  
</html>

Изменяем bboard/create.html

стр . 67

<!DOCTYPE html>  
<html lang="en">  
{% extends "layout/basic.html” %}  
{% block title %}Добавление объявления{% endblock %}  
{% block content %}  
 <h2>Добавление объявления</h2>  
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <input type="submit" value="Добавить">  
 </form>  
{% endblock %}

Статические файлы

CSS

header h1 {

font-size: 40pt;

text-transform: uppercase;

text-align: center;  
background: url ("1.png") left / auto 100 % no-repeat;

}

nav {

font-size: 16pt;

width: 150px:

float: left;

}

nav a {

display: block;

margin: 10px 0px;

}

section {

margin-left: 170px;

}

Создадим папку bboard/static в ней папку bboard - здесь будем хранить файлы .css , .jpg

Изменяем layout/basic.html стр. 68

{% load static %}  
  
<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <meta charset=”UTF-8”>  
 <title>{% block title %}Главная{% endblock title %} :: Доска объявлений</title>  
 <link rel="stylesheet" type="text/css"  
 href="{% static 'bboard/style.css' %}">  
 </head>

…