**Конспект**

|  |  |
| --- | --- |
| Автор: | Горюнов Евгений С. |
| Город | Томск |
| mail: | ges1987@list.ru |
| Курс: | Fullstack разработчик на Python (SkillFactory |
| Год обучения: | 2021.08 – 2022г |
|  |  |

Оглавление

[Библиотеки для Django SkillFactory 2](#_Toc92966979)

[Команды для консоли PyCharm 3](#_Toc92966980)

[Создать виртуальное окружение (имя окруж – venv) 3](#_Toc92966981)

[Активация виртуально окружения 3](#_Toc92966982)

[Просмотр установленных приложений в вирт окруж 3](#_Toc92966983)

[Установка фреймворка Django 3](#_Toc92966984)

[Создать проект Django (имя – project) 3](#_Toc92966985)

[Переход в папку project 3](#_Toc92966986)

[Запуск сервера PyCharm 3](#_Toc92966987)

[Создание миграций 3](#_Toc92966988)

[Провести миграции 4](#_Toc92966989)

[Создание администратора (первого пользователя) 4](#_Toc92966990)

[Сигналы 6](#_Toc92966991)

[notify\_managers 6](#_Toc92966992)

[Полезное 7](#_Toc92966993)

[Подтверждение email через админку 7](#_Toc92966994)

[return в Django - варианты возвратов 8](#_Toc92966995)

[return redirect 8](#_Toc92966996)

[return render 8](#_Toc92966997)

## Библиотеки для Django SkillFactory

главный фреймворк (модуль Д1)

pip install Django

для авторизации, аутентификации пользователей (модуль Д3)

pip install django-allauth django-allauth

для работы фильтров (модуль Д4)

pip install django-filter django\_filters

для выполнения периодических задач (модуль Д6)

pip install django-apscheduler django-apscheduler

команда для обновления пакетного менеджера pip

pip install --upgrade pip

python -m pip install -U pip второй вариант, если первый вызовет ошибку

# Команды для консоли PyCharm

Все команды для windows среды

## Создать виртуальное окружение (имя окруж – venv)

python -m venv venv

## Активация виртуально окружения

venv\scripts\activate

## Просмотр установленных приложений в вирт окруж

pip list

## Установка фреймворка Django

pip install django

## Создать проект Django (имя – project)

python -m django-admin startproject project

## Переход в папку project

cd project

переход по конкретному пути

cd project\news

Переход на каталог выше

cd ..

## Запуск сервера PyCharm

py manage.py runserver

http://127.0.0.1:8000/ - порт запуска сервера

для запуска сервера на другом порту, для работы в нескольких проектах одновременно

в конце изменить номер порта

py manage.py runserver 7000

## Создание миграций

python manage.py makemigrations

## Провести миграции

py manage.py migrate

## Создание администратора (первого пользователя)

python manage.py createsuperuser

def post(self, request, \*args, \*\*kwargs):

post -

\*args -

\*\*kwargs –

request-

self-

def post-

# Сигналы

## notify\_managers

читать тут более подробно

<https://russianblogs.com/article/4399514519/>

Данный метод нужен для того, чтоб начать выполнять код после какого-либо события, например сохранение в БД информации (в нашем примере), ее изменение и так далее. Сигналов множество, на любой случай

Импортируем модули

def notify\_managers\_ appointment (sender, instance, created, \*\*kwargs):

subject = f'{instance.client\_name} {instance.date.strftime("%d %m %Y")}'

print("МЕТОД НУЖНЫЙ СРАБОТАЛ!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")

mail\_managers(

subject=subject,

message=instance.message,

)

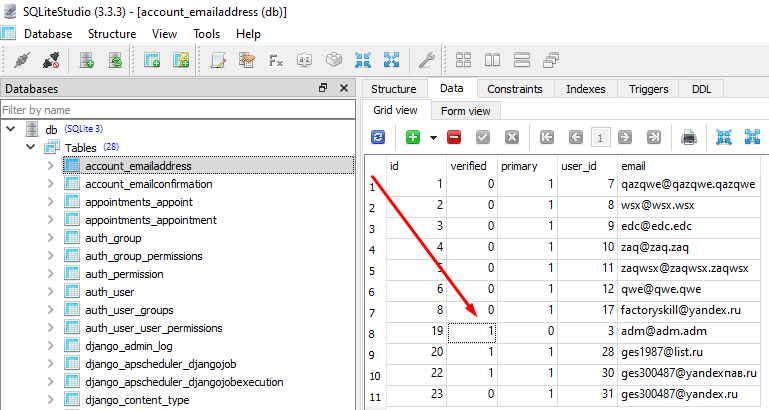
# коннектим наш сигнал к функции обработчику и указываем, к какой именно модели после сохранения привязать функцию

post\_save.connect(notify\_managers\_appointment, sender=Appointment)

# Полезное

## Подтверждение email через админку

Нужно поставить 1 в поле verified таблицы account\_emailaddress



Для чего это нужно, так как при подключении верификации пользователей через почту на сайт не зайдешь, пока почта не будет подтверждена, а почта, как видно из скрина, почти у всех недействительная, тупо для экспериментов служит. Поэтому подтверждаем ее таким способом.

## return в Django - варианты возвратов

## return redirect

return redirect ('test')

вернет (загрузит) представление, указанное в utls.py после исполнения кода

(path('appoint/', AppointView.as\_view(), name='test'), )

Важная особенность, если мы загружаем представление, то загружается и get запрос и весь код, связанный с ним, чтоб это не делать, можно вызывать метод render (см ниже)

return redirect('/news/')

перейдет по указанному адресу

## return render

return render(request, 'test.html', {})

Более продвинутый вариант, данный вариант позволяет не вызывать представление целиком, а загружать страничку самостоятельно, конкретно, например страничку test.html, данные запроса и переменные, которые нам нужны на нашей страничке, которые также можно вычислять в пределах post функции, очень удобно для отладки приложения, для формирования запросов в БД и получения из нее инфы.

----1-------2---------3------------4-------------------5-----------------6-------

return render(request, 'test.html', {'pole\_test\_html': pole\_test, })

1 – return, вернуть (или загрузить), то есть что в итоге должна нам отдать функция, результат нашей функции

2 – render, это метод, то есть функция должна вызвать данный метод render

3 – request, данные запроса (то есть вся инфа, поступившая с POST формы, с нашей странички), по мимо наших данных в теле (в объекте) request содержится куча всего, посмотреть можно так print(dir(request)), и обраться и извлечь инфу из любого атрибута можно такими командами

<имя переменной> = request.POST['category'], либо так, тоже действует

<имя переменной> = request.POST.get('category')

4 – 'test.html', шаблон (страничка), которую следует загрузит (именно с окончанием .html, а не имя представления в урлах!)

Следующие атрибуты не обязательные, то есть для нормальной работы у нас уже все есть, это данные запроса (request), и шаблон, который нужно грузить пользователю, но если мы хотим еще какие-то данные закинуть на нашу страничку, то создаем словарь (5+6), который может иметь неограниченное количество переменных

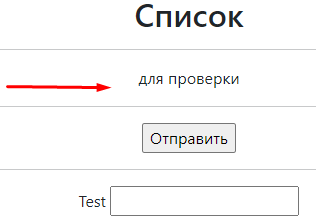
5 и 6 – {'pole\_test\_html': pole\_test, } это обязательно должен быть словарь, то есть ключ (key) и его значение (value). Идея тут следующая, мы на своей страничке размещаем метку (ключ, в нашем случае 'pole\_test\_html', то есть первый аргумент), а после двоеточия – значение метки, то есть переменная в нашем коде в файле представления. Метка на нашей страничке выглядит так {{ pole\_test\_html }}, то есть в любое место в коде воткнули, обязательно в двойных {{ }} фигурных скобках (чтоб html код мог понять, что тут значение переменной из нашего кода джанги), а в джанге в файле представления нашей переменной присваиваем значение, например такое pole\_test = request.POST['test'].

5 - 'pole\_test\_html' – это первый ключ словаря, key

6 - pole\_test – это значение первого ключа словаря, value {key: value, key1: value1, и так далее}

В данном случае, мы вводим в поле Test на нашей страничке любое значение, и тут же выводим его на нашей страничке сверху, либо можно снизу, как удобно

Изображение выглядит как текст

Автоматически созданное описание 

Все что связано с шаблонами менять тут (файл settings.py):

Изображение выглядит как текст

Автоматически созданное описание