**Django Python**

В начале работы устанавливаем Фреймворк pip install Django, все как обычно.

Теперь чуть теории:

Django — это **высокоуровневый Python веб-фреймворк**, который позволяет быстро создавать безопасные и поддерживаемые веб-сайты. Созданный опытными разработчиками, Django берёт на себя большую часть хлопот веб-разработки, поэтому вы можете сосредоточиться на написании своего веб-приложения без необходимости изобретать велосипед.

Вся работа должна быть уложена в папки это важно, нужно визуальное пространство.

Итак начнем, выбираем путь нашего проекта через cd, после чего прописываем django-admin startproject test\_project, обращаемся к джанго, указываем команду о создании проекта, и его называние, создается папка нашего проекта.

Теперь пройдемся по созданным файлам в папке.

Manage.py – файл код изменять которого мы не будем, но будем очень часто к нему обращаться через терминал.

\_\_init\_\_ - как я понял инициализатор, в нем можно будет прописать какие-то действия при запуске проекта.

Asgi wsgi – они обеспечивают коректное подключение к серверу, Asgi по факту просто новый стандарт wsgi. С ними можно работать косвенно по факту мы там ничего не трогаем до тога пока не выгружаем их на сервер.

Setting – важный файл в котором прописываются все настройки нашего проекта. Рассмотрим его подробнее

В BASE.dir – записывается полный путь к нашему проекту

Secret.key – типо токен нашего вебсайта, при выгрузке на сервер его лучше изменить

Debug = True – указывает что все наши ошибки будут указываться на страничке сайта, логично при выгрузке его надо менять на false

В allowed\_host – мы можем сами указать хосты на которые нам будет разрешено выкладывать наш сайт

Installed\_aps – в нем хранятся созданные приложения, пока там базовые.

Midleware – тот лист в котором хранятся доп ПО - для обеспечения типо связи с хостом, сессией, защитой и тд

RootUrl – в нем указывается какой основной файл будет использоваться

Temlates – указывается какие шабломы мы можем использоовать для нашего проекта

Wsgi.application – опять какая технология используется для выгрузки на сервер

Database – там прописано с какой базой данных мы будем работать по умолчанию sqllite

Это все основные.

Дальше файл urls.addres – в нем указаны те приложения которые мы можем отслеживать и ссылаться к ним

urlpatterns = [

    path('admin/', admin.site.urls),

]

Вот что там в начале прописано, переведу, по вызову’admin/’ мы можем открыть встроенную панель администратора.

Начнем, попробуем открыть live server, укажим путь в терминале к manage

И прописываем **python manage.py runserver** – эта комманда даст нам ссылку на наш сайт. Вылетит ошибка что надо создать миграции, но что это узнает потом.

Важно при открытии локально сервера, чтобы он не был вечно запущен, из него по завершению работы нужно выходить, выйти можно сочетанием клавиш ctrl + c

**Создание приложений**

Лучше все основные функции сайта разбивать на приложения грубо говоря, в приложении main будет основная работа с главной страничкой сайта, в приложении password – будет работа с регистрацией и тд…

Сейчас создадим приложение в котором будем обрабатывать основную страницу и страницу про нас.

Python manage.py startapp main - создаем приложение в терминале (в названии все пишем в нижнем регистре.

Рассмотрим файлы созданного приложения.

В папке migration – там вссе файлы связанные с миграцией

Миграции нужны вообще для структуры описывания баз данных

Admin – какие таблицы должны отображаться в панели администрации.

Apps – глобальные настройки конкретно для этого приложения.

Models – для создания таблицы в бд

Test – там можем прописывать разные проверки для работы.

Vies – те методы которые будут высланы пользователем при переходе на страницу.

После создания приложения его нужно зарегистрировать в setting

В install\_apps – добавляем ‘name’,

**Теперь научимся отслеживать различные url адреса**

Заходим в urls – добавляем новый patch если надо передать два параметра

Первый сама ссылка страницы, если хотим отслеживать главную страницу то там ничего не указываем “” а вторым параметром вот это гемор, нам нужно вызвать тот же файл но из другого созданного нами приложения. По факту вторым параметром мы указываем какое приложение будет отвечать за эту сслыку.

from django.urls import path, include

подключаем новый метод include.

И в патче прописываем так

Patch(“”, include(“название\_приложения.urls”)

Грубо говоря мы обращаемся к названию приложения и внем обращаемся к файлу urls, но файл urls в папке с приложением main не создан, поэтому создаем его сами. Внутри этого файла прописываем:

from django.urls import path

urlpatterns = [

]

А вот теперь внутри urlpatterns мы будем отслеживать файл views

Но его нужно сначала импортировать.

Импортируем так **from . import views** точка значит что из этой же папки

from django.urls import path

from . import views

urlpatterns = [

    patch("", views.visual)

]

Мы через views обращаемся к методу visual который мы должны прописать во views

Теперь в функции visual чтобы вывести текст, мы должно импортировать новый метод

from django.http import HttpResponse

и в методе

def visual(request):

    return HttpResponse("<h4> Проеверка епта </h4>")

первое прописываем обязательный параметр request – запрос, потом возвращаем метод в котором тегом h4 передадим стринг

Запускаем сервер и все должно работать.

После этого давай отследим еще одну страницу.

Заходим в urls main и там отслеживаем новую сслыку которая будет обращаться к новому методу во views

from django.urls import path

from . import views

urlpatterns = [

    path("", views.visual),

    path("about", views.vis\_about)

]

Как то так, чтобы открыть эту страницу через слеш е нашему адресу добавляем /about

[127.0.0.1:8000/about](http://127.0.0.1:8000/about)

Ну а щя я побалуюсь и все попроверяю.

Можно и без тегов h4 не понятно нахуя они там