**Тема занятия № 44: Публикация web-сайта**

1. Публикация веб-сайта

**Публикация посредством Uvicorn**

Uvicom — ’’легкий” и быстрый веб-сервер, который специально предназначен для публикации сайтов, написанных на Python, в том числе и с применением Django.

Внимание!

Uvicom может обслуживать только сайты, написанные на Django 3.0 или более новых версиях этого фреймворка.

Преимущество Uvicom в том, что для подготовки сайта к публикации с его помощью достаточно добавить в код всего несколько выражений. Недостаток — невысокая производительность, вследствие чего этот сервер не стоит применять для обслуживания высоконагруженных решений.

На заметку!

Полная документация по Uvicorn находится здесь: https://www.uvicorn.org/.

Установка Uvicom выполняется подачей команды:

Pip install uvicorn

Помимо веб-сервера, будут установлены библиотеки websockets, click и hl 1, необходимые для работы.

**Подготовка веб-сайта к публикации посредством Uvicorn**

Веб-сайт, написанный с применением Django, успешно работает под управлением Uvicom. За одним исключением: этот веб-сервер не обрабатывает статические и выгруженные файлы. Поэтому в код сайта необходимо внести некоторые правки.

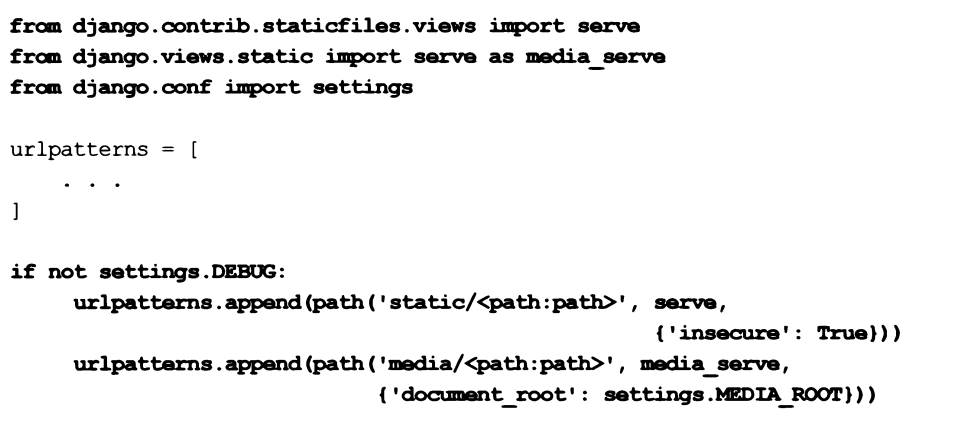
В список маршрутов уровня проекта (что хранится в модуле urls.py пакета конфигурации) следует добавить два маршрута — для обработки статических и выгруженных файлов. Оба маршрута создаются вызовом функции path о. Различаются они только контроллером-функцией, который указывается во втором параметре этой функции.

О Обработка статических файлов — будет осуществляться контроллером serve () из модуля django.contrib.staticfiles.views. Этот контроллер ’’умеет” искать статические файлы во всех папках, указанных в настройках проекта.

К сожалению, контроллер serve о работает только в отладочном режиме — в эксплуатационном он возбуждает исключение Http404. Но, к счастью, он поддерживает необязательный параметр insecure: если задать ему значение True, то контроллер успешно работает и в эксплуатационном режиме. Передать значения для параметров контроллера можно, указав их в словаре, который передается функции path () в третьем параметре.

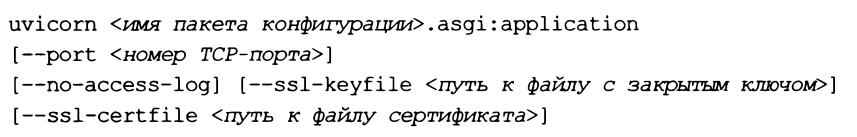
□ Обработка выгруженных файлов — будет выполняться контроллером-функцией Serve о из модуля django. Views. Static (не перепутайте с одноименным контроллером из модуля django.contrib.staticfiles.views!). Он более универсален, нежели описанный ранее, и может выдавать файлы из произвольной папки, путь к которой передается ему через необязательный параметр document root. Значение этого параметра можно передать также в словаре, указываемом в третьем параметре функции path ().

Пример задания обоих путей в модуле uris.py пакета конфигурации:

****

**Запуск и остановка Uvicorn**

Чтобы запустить Uvicorn, следует перейти в папку проекта и задать в командной строке команду формата:

****

Uvicorn "общается" с Django-сайтом через интерфейс ASGI (Asynchronous Server Gateway Interface, асинхронный интерфейс серверного шлюза), поддержка которого появилась в Django 3.0. "Связкой" между Uvicorn и сайтом выступает модуль asgi.py пакета конфигурации; application — это переменная модуля asgi.py, хранящая объект, который представляет сайт.

Поддерживаются следующие наиболее полезные ключи:

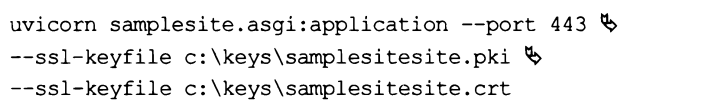
□ —port — указывает номер тср-порта, через который будет работать веб-сервер (по умолчанию: 8000);

□ —no-access-log — запрещает выводить журнал работы сервера (по умолчанию журнал выводится непосредственно в командной строке).

Пример запуска Uvicorn для обслуживания сайта через стандартный TCP-порт № 80 без вывода журнала:



Чтобы запустить Uvicom для работы через защищенный протокол HTTPS, следует дополнительно задать параметры —ssi-keyfile и —ssi-certfile, указывающие пути к файлам соответственно закрытого ключа и сертификата. Пример:



Чтобы остановить Uvicom, достаточно переключиться в окно командной строки, в которой он запущен, и нажать комбинацию клавиш <Ctrl>+<Break>.

**Публикация посредством Apache HTTP Server**

Apache HTTP Server— высокопроизводительный, универсальный и популярнейший на данный момент веб-сервер.

При публикации сайта с его помощью обработка статических и выгруженных файлов будет выполняться самим веб-сервером, не затрагивая Django. Если сайт содержит много файлов такого рода, это заметно повысит производительность.

Внимание!

К сожалению, из-за ошибки в одном из модулей стандартной библиотеки Python 3.8 сайт, публикуемый с помощью Apache HTTP Server, не работает. Возможно, в более новых версиях Python эта ошибка будет устранена.

Однако под более старыми версиями Python сайт полностью функционирует.

**Подготовка веб-сайта к публикации посредством Apache HTTP Server**

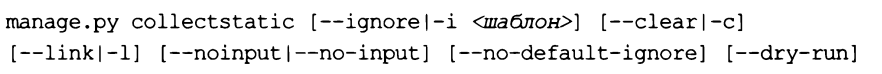
Подготовка сайта к публикации включает три шага:

1. Создание папки, в которой будут собраны все статические файлы, имеющиеся в составе Django-сайта. Обычно этой папке дают имя static.

2. Указание пути к этой папке в параметре static root настроек проекта. Вот пример задания пути к папке static, находящейся непосредственно в папке проекта:



3. Сбор всех статических файлов в указанной папке подачей команды collectstatic утилиты manage.py:



По умолчанию все статические файлы, найденные в папках static пакетов приложений и папках, пути к которым указаны в параметре staticfiles dirs, копируются в папку, заданную в параметре static root настроек проекта. Структура папок, в которые вложены эти файлы, при этом сохраняется.

Приложения просматриваются в том порядке, в котором они перечислены в списке зарегистрированных в проекте. Если в разных папках static присутствуют файлы с одним и тем же именем, будет использован файл, найденный первым.

При последующих вызовах команды collectstatic в папку будут скопированы только новые и изменившиеся после предыдущего копирования файлы. Перед перезаписью имеющегося в папке назначения файла утилита выдаст предупреждение и предложит ввести слово ”yes" для перезаписи или "по” для отказа от этого.

По умолчанию копируются все статические файлы, за исключением файлов с именами CVS, а также именами, которые совпадают с шаблонами\* и поддерживаемые командные ключи:

• —ignore или -i — указывает шаблон для имен файлов, которые не должны копироваться:



Можно задать произвольное количество таких шаблонов— каждый в своем

Ключе:



Начиная c Django 2.2, можно указывать шаблоны имен папок, содержимое которых не должно копироваться;

• —clear или -с — перед началом копирования очистить папку назначения;

• —link или -1 — вместо копирования файла создать символическую ссылку на него;

• —noinput или -no-input — имеющийся в папке назначения файл будет перезаписан без выдачи предупреждения;

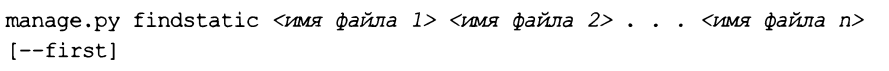
• -no-default-ignore —также копировать файлы с именами CVS \*

• -dry-run — выводит на экран сведения о файлах, подлежащих копированию, но не копирует их.

Внимание!

Статических файлов в папке назначения может оказаться довольно много, особенно если в проекте используются сложные приложения и библиотеки, наподобие административного веб-сайта Django и Django REST framework. Поэтому во многих случаях имеет смысл рассмотреть вариант с созданием символических ссылок на статические файлы вместо их копирования.

Также может оказаться полезной команда findstatic утилиты manage.py, которая ищет статические файлы с указанными именами и выводит на экран полные пути к ним:

****

Если задать командный ключ —first, то будет выведен только путь к первому обнаруженному файлу.

**Подготовка платформы для публикации посредством Apache HTTP Server**

Подготовка платформы для публикации сайта с помощью Apache HTTP Server заключается в выполнении следующих шагов:

1. Поиск и загрузка программного модуля modwsgi, выступающего в качестве коннектора между веб-сервером и Django-сайтом.

Дистрибутивные комплекты различных редакций этого модуля находятся по интернет-адресу https://www.lfd.uci.edu/-gohlke/pythonlibs/#mod\_wsgi. Имена файлов с этими комплектами включают в себя следующие фрагменты символов:

• ар<две цифры>—где две цифры обозначают версию Apache;

• чс<одна или две цифры>— одна или две цифры показывают внутреннюю версию среды разработки Microsoft Visual C++, в которой компилировался модуль-коннектор;

• ср<две цифры> —две цифры обозначают версию Python;

• Win32 —если это 32-разрядная редакция модуля;

• win\_amd64 —если это 64-разрядная редакция модуля.

Например, mod\_wsgi-4.7.0+ap24vc14-cp36-cp36m-win32.whl — это редакция для 32-разрядного Python 3.6 и Apache 2.4, откомпилированного в Microsoft Visual C++ версии VC14.

Следует выбрать ту редакцию mod wsgi, которая соответствует версии и редакции установленного на платформе Python. Так, если установлен 64-разрядный Python 3.7, следует загрузить файл mod\_wsgi-4.7.0+ap24vc15-cp37-cp37mwin\_amd64.whl.

С расширением whl сохраняются файлы формата WHL (от англ, wheel) — дистрибутивные пакеты дополнительных библиотек для Python.

2. Установка модуля из загруженного файла WHL отдачей команды формата:



3. Загрузка и установка веб-сервера Apache HTTP Server. Найти его дистрибутивный комплект можно на сайте https://www.apachelounge.com/ или https://www.Apachehaus.com/.

На заметку!

Полная документация по Apache находится на его домашнем сайте:

Http://httpd.apache.org/.

При выборе дистрибутива сервера необходимо учесть две очень важные вещи:

• Следует выбрать ту редакцию, которая откомпилирована в той же версии Microsoft Visual C++, что и установленный ранее модуль-коннектор Mod wsgi. Так, если был установлен модуль mod\_wsgi-4.7.0+ap24vc15-cp37-сp37m-win\_amd64.whl, откомпилированный в Microsoft Visual C++ версии VC15, следует выбрать дистрибутив Apache 2.4.x openssl 1.1.1 VC15.

• В случае установки 32-разрядной редакции Python нужно выбирать только 32-разрядную редакцию Apache, а в случае 64-разрядной редакции Python — только 64-разрядную редакцию Apache.

Внимание!

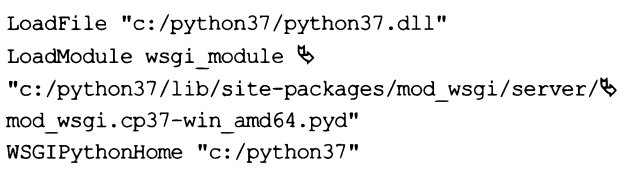
Если установить не подходящие друг другу редакции Python, mod\_wsgi и Apache, скорее всего, ничего не заработает.

Дистрибутив Apache поставляется в виде обычного архива формата ZIP, содержимое которого следует распаковать в корневую папку диска.

4. Получение настроек, необходимых для связывания веб-сервера и публикуемого сайта посредством modwsgi. Для этого следует задать в командной строке команду:



Утилита mod\_wsgi-express.exe устанавливается в составе модуля mod\_wsgi, а команда module-config этой утилиты выведет на экран строки, содержащие необходимые настройки. Например:

****

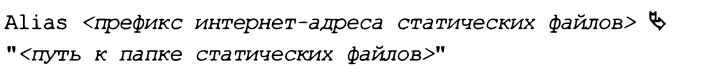
Первая строка предписывает веб-серверу при запуске загрузить программное ядро исполняющей среды Python, необходимое для успешной работы mod wsgi.

Вторая строка выполняет загрузку самого этого модуля-коннектора. Третья строка указывает коннектору путь, по которому установлен Python.

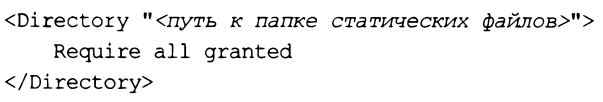
5. Добавление полученных ранее строк с настройками в файл конфигурации Apache. Этот файл носит имя httpd.conf и находится по пути <путь, по которому установлен Apache>\conf

**Конфигурирование веб-сайта для работы под Apache HTTP Server**

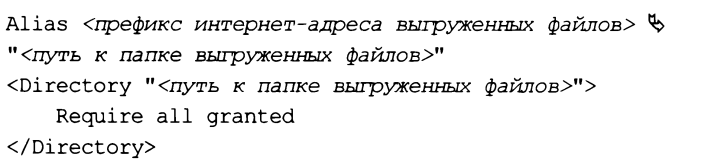
В файл конфигурации httpd.conf также нужно внести параметры самого публикуемого Django-сайта. Они указываются посредством следующих директив:

****

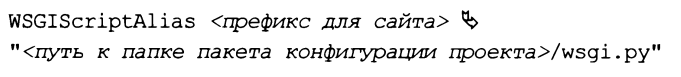
Директива Alias указывает серверу искать файлы, интернет-адреса которых имеют заданный префикс, в папке с указанным путем. После ее указания вебсервер станет сам обслуживать статические файлы, минуя Django.

****

Директива Directory устанавливает права Apache на содержимое папки с заданным путем. Команда Require ail granted дает веб-серверу доступ ко всем файлам из этой папки. Если данную директиву не указать, то сервер не сможет загрузить ни один файл из папки.

****

Аналогичным образом конфигурируется обработка файлов, выгруженных посетителями.

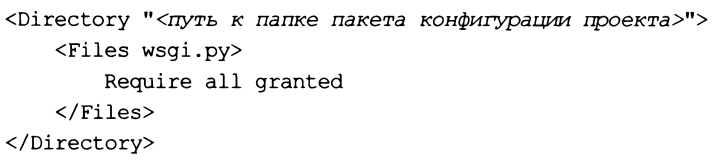


Эта директива свяжет указанный префикс с самим Django-сайтом. Например, если задать префикс /bboard, то сайт станет доступен по интернет-адресу Ъир://<интернет-адрес xocma>/bboard/. Чтобы связать сайт с ’’корнем”, следует задать префикс / (слеш).

Модуль wsgi.py, хранящийся в пакете конфигурации проекта, служит ’’связкой” между mod wsgi и сайтом. Взаимодействие между ними выполняется через интерфейс WSGI (Web Server Gateway Interface, интерфейс шлюза веб-сервера), поддерживаемый Django начиная с самых первых версий.



Очень важная директива, задающая путь к папке проекта, чтобы исполняющая среда Python ’’знала”, где находятся все пакеты и модули, составляющие сайт;

****

Эта директива разрешает веб-серверу доступ к модулю wsgi.py. Если ее не указать, то Apache не сможет запустить этот файл;



Эта директива указывает модулю-коннектору пропускать любые заголовки, содержащие имя и пароль пользователя, а не удалять их, как это он делает по умолчанию. Ее следует указать, если в состав сайта входит веб-служба, обрабатывающая AJAX-запросы и реализующая аутентификацию.

Приведен готовый код, конфигурирующий Django-сайт. Его можно использовать как шаблон для написания своей конфигурации.

****