Инструментальные средства информационных систем

Группа: БИН-19-1

Автор: Сафиуллина Н.Ф.

Лекция 6 Инструментальные средства web-разработки

Содержание

Введение

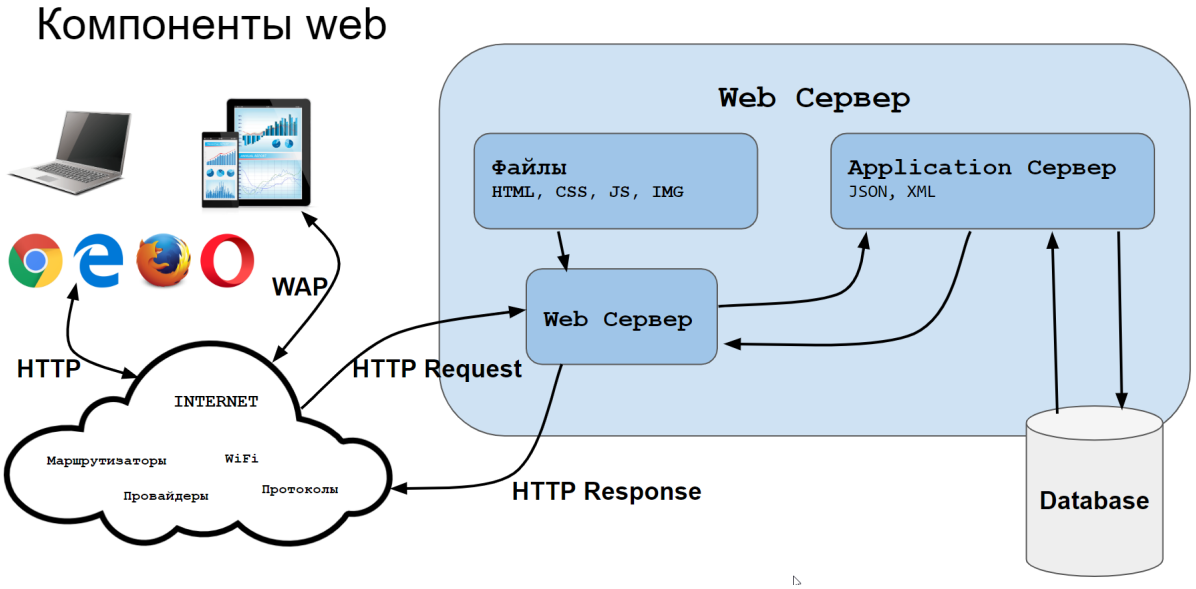
Инструменты web-разработки это еще одна тема, которой можно посвящать несколько семестров обучения в университете. Поэтому в данном курсе рассмотрим только основное, не углубляясь в подробности.

Сначала дадим определение что такое web. Известно, что web переводится с английского “сеть” или “паутина”. Слово web входит в аббревиатуру WWW – World Wide Web, но уже давно все что связано со всемирной сетью или паутиной просто называют web прибавляя определяющее слово, web-сервер, web-приложение, web-дизайн, web-страница, web-сайт, вебинар и т.д.

Коротко говоря, web – это распределенная система связанных через Интернет машин (компьютеров, в общем смысле этого слова), которая предоставляет доступ к информации, расположенной на этих машинах. Чаще всего эти машины – это сервера.

1 Компоненты web

Описать работу web можно очень подробно, включая все, от огромных систем хранения данных до мельчайших программ-служб. Рассмотрим основные компоненты web и простую схему взаимодействия этих компонентов.



1.1 Web сервер

Понятие «веб-сервер» может относиться как к аппаратному обеспечению, так и к программному обеспечению. Или к этим обеим частям, работающим совместно.

1. С аппаратной точки зрения, веб-сервер — это компьютер, который хранит файлы сайта (HTML-документы, CSS-стили, JavaScript-файлы, картинки, программы, выполняющие бизнес-логику и другое). Он обеспечивает физическую передачу данных на устройства подключенные к сети.

2. С точки зрения ПО, веб-сервер включает несколько компонентов, которые управляют доступом веб-пользователей к размещенным на сервере файлам. Как минимум включает HTTP-сервер. HTTP-сервер доступен через доменное имя, он понимает URL (web-адреса) и HTTP (протокол), и он доставляет содержимое веб-сайта до конечный пользователей. Web-сервер, как ПО, часто не отделим от Application-сервер и выполняет бизнес логику веб-приложения.

На самом базовом уровне, когда браузеру нужен файл, размещенный на веб-сервере, браузер запрашивает его через HTTP-протокол. Когда запрос достигает нужного веб-сервера, сервер принимает запрос, находит запрашиваемый документ (если нет, то сообщает об ошибке 404) и отправляет обратно, также через HTTP.

Чтобы опубликовать веб-сайт, необходим статический или динамический веб-сервер.

Статический веб-сервер, или стек, состоит из компьютера с сервером HTTP. Он отдает данные как они есть, т.е. есть какой-то набор статических данных. Статические веб-сайты проще всего сделать.

Динамический веб-сервер состоит из статического веб-сервера и дополнительного программного обеспечения, чаще всего сервера приложения (Application-сервер) и базы данных. Сервер называется «динамическим», потому что сервер приложений изменяет исходные файлы, динамически собирает их из различных данных, перед отправкой в браузер клиента.

Например, готовая страница в браузере в виде HTML может быть сформирована на основании данных, которые Application-сервер получил из БД, и данных, которые вычислил на основании алгоритмов бизнес-логики.

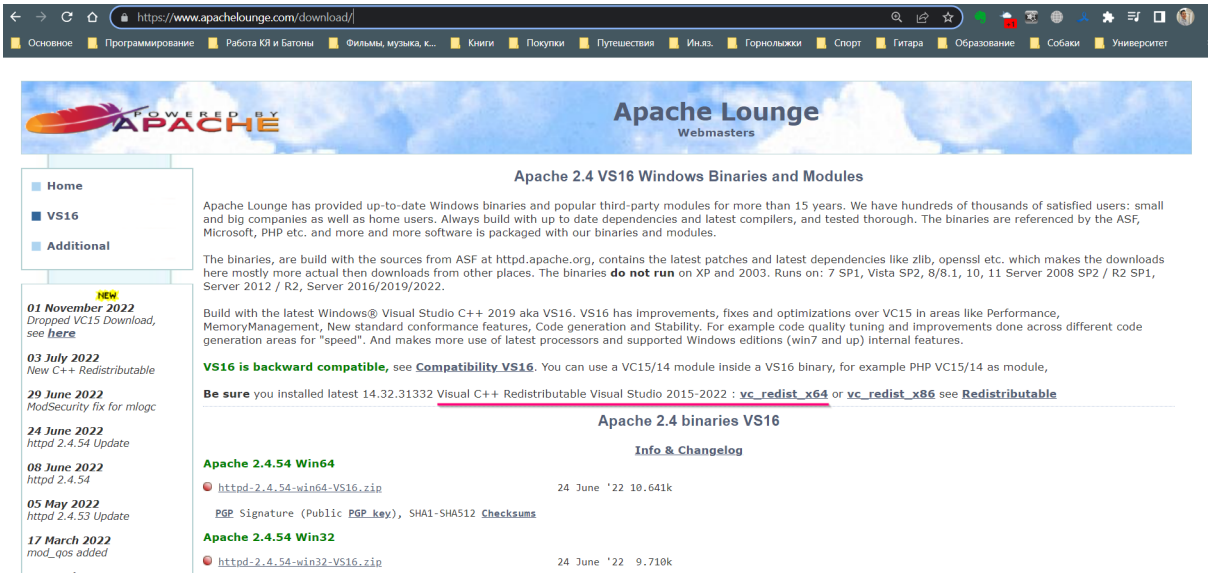
В российском интернете на 2021 год примерный процент использования веб-серверов распределился следующим образом:

1.2 HTTP-сервер Apache

Официальный сайт Apache предоставляет только исходный код своего HTTP-сервера. Но там же сказано, что если вы не можете скомпилировать HTTP-сервер Apache самостоятельно, вы можете получить бинарный пакет из многочисленных бинарных дистрибутивов, доступных в Интернете. Рекомендуют воспользоваться Apache Lounge:

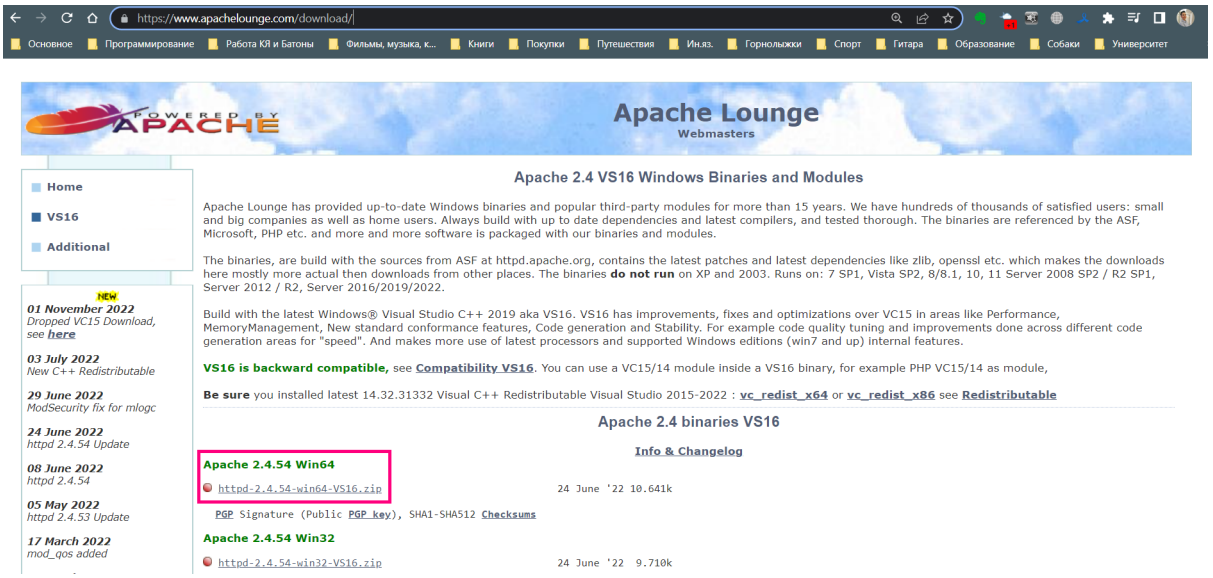
https://www.apachelounge.com/download/

Сначала нужно установить последнюю версию Visual C++ Redistributable for Visual Studio 2015-2019:



Если у вас 64 битная операционная система Windows, то скачать нужно vc_redist.x64, а если у вас 32 битная операционная система – vc_redist_x86 .

Далее скачиваем архив содержащий Apache:



Получим архив:

`httpd-2.4.54-win64-VS16.zip`

распакуем его в корень диска C:\.

Можно выбрать и другое место, но тогда измените путь в настройках Apache в файле httpd.conf параметр Define SRVROOT должен быть равен местоположению распакованного Apache24. Файл конфигурации находится по относительному такому пути:

`C:\Apache24\conf\httpd.conf`

Запустим и проверим наш север. В командной строке пишем переходим в папку Apache24\bin:

```
C:\Users\Safiullina_NF>cd C:\Apache24\bin
```

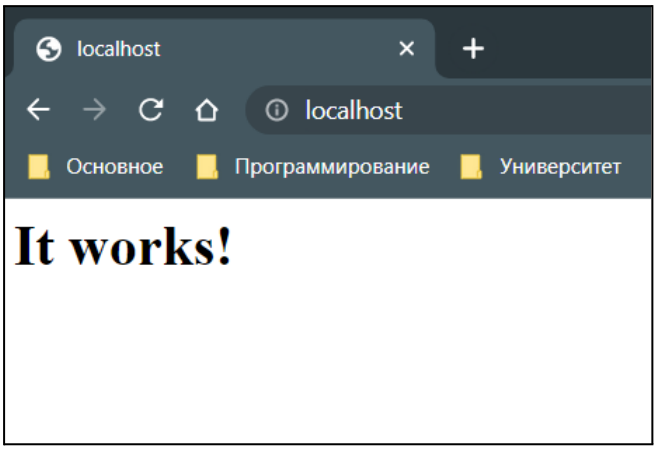
Запустим сервер:

```
C:\Apache24\bin>httpd.exe
```

Если курсор переместился на новую строку и строка осталась пустой, значит наш сервер запущен. Проверим его работу, откройте браузер и введите адрес:

http://localhost

Мы должны увидеть приветственную страницу:

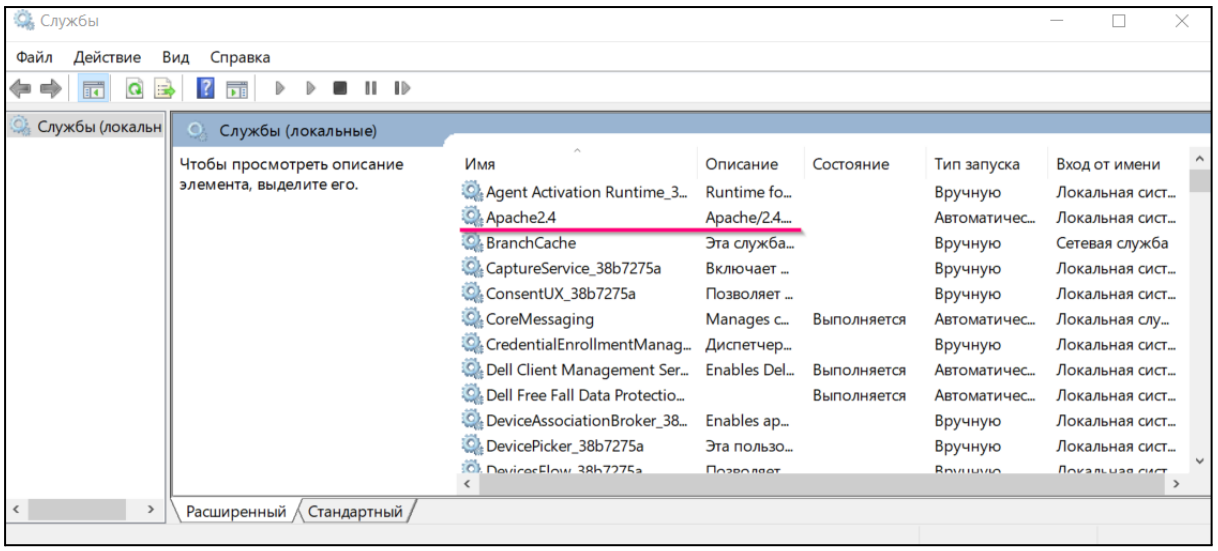


Чтобы остановить сервер Apache в командной строке нужно нажать Ctrl+C. Остановка сервера занимает несколько секунд.

Можно установить **Apache как сервис ОС Windows**. Для этого нужно выполнить командной строке запущенной от администратора:

```
httpd.exe -k install
```

Тогда, после успешной установки, служба появится в списке:



Как всегда посмотреть все доступные команды с кратким описанием можно в справке:

```
C:\Apache24\bin>httpd -h
```

Базовая настройка Apache. Файл настроек или конфигурации находится:

```
Apache24\conf\httpd.conf
```

В нем есть строки комментариев начинаются с #, а так же есть строки – директивы, которые тоже начинаются с #, это означает, что директивы отключены.

Директивами называются параметры или опции, которые создают инструкции для web-сервера.

Как и было сказано ранее первую директиву, которую нужно настроить это путь к папке, где хранятся конфигурация сервера, ошибки и протоколы:

```
Define SRVROOT "c:/Apache24"
ServerRoot "${SRVROOT}"
```

Первая строка присваивает значение переменной SRVROOT, которое по умолчанию "C:/Apache24". Вторая строка присваивает настройке ServerRoot значение переменной ${SRVROOT}.

Значение в первой строке нужно заменить на то, куда распаковывали архив Apache.

Директива Listen позволяет вам привязать Apache к определённому IP адресу и/или порту. Значение по умолчанию:

```
Listen 80
```

Оно означает, прослушивать 80 порт на любом IP адресе (т.е. любом сетевом интерфейсе), доступном в вашей системе. Вы можете указать конкретный IP адрес, который нужно прослушивать и, следовательно, на запросы с которого отвечать:

```
Listen 127.0.0.1:80
```

Можно указать несколько портов:

```
Listen 80
Listen 8080
```

Или несколько IP и портов:

```
Listen 127.0.0.1:80
Listen 127.0.0.1:8080
```

Вы можете использовать любые сочетания, главное правило – порт на указанном интерфейсе (IP) не должен быть занят другой программой.

Значение по умолчанию вполне подходит для локального веб-сервера – т.е. здесь можно просто ничего не менять.

Далее идет список модулей. Некоторые из них включены, некоторые выключены. Нам нужно включить модуль **mod_rewrite**. Ищите строку:

```
LoadModule rewrite_module modules/mod_rewrite.so
```

Раскомментируйте ее.

Этот модуль выполняет множество функций, одна из них, например, это превращение URL в ЧПУ – в человекопонятные URL.

Следующая директива, которая нам нужна:

```
ServerAdmin admin@example.com
```

Это адрес администратора сервера. Замените на свой адрес электронной почты.

Директива:

```
ServerName www.example.com:80
```

Это имя и порт, которые сервер использует для идентификации самого себя. Замените значение на localhost:

```
ServerName localhost
```

Следующий блок директория:

```
<Directory />
    AllowOverride none
    Require all denied
</Directory>
```

Запрещает доступ к файловой системе вашего сервера. Чтобы открыть доступ к папкам с содержимым, необходимо явно указать их в блоках директориев ниже в файле конфигурации.

Следующий блок перезаписывает предыдущий блок для конкретной папки:

```
DocumentRoot "${SRVROOT}/htdocs"
<Directory "${SRVROOT}/htdocs">
        Options Indexes FollowSymLinks
        AllowOverride None
       Require all granted
</Directory>
```

Директива DocumentRoot – это путь, где расположен ваш сайт. Если ваш контент будет находится в другом месте, нужно не забыть поменять его адрес в данном блоке.

Как видно тут используется переменная SRVROOT, для относительного задания пути. Если контент сайта будет находится в другом месте, упоминание переменной нужно удалить.

Директива Options может принимать значения “None”, “All” или любую комбинацию из следующих значений: “Indexes”, "Includes”, “FollowSymLinks”, “SymLinksifOwnerMatch”, “ExecCGI”, “MultiViews”. В нашем примере “Index” говорит, что если запрос пришел без указания имени файла, который надо открыть, то откроется файл с именем index (в него можно как раз поместить главную страницу). Опция FollowSymLinks говорит, что разрешение следовать символьным ссылкам.

AllowOverride контролирует какие директивы могут быть указаны в .htaccess (это тоже конфигурационный файл сервера, который заменяет параметры указанные в основном конфигурационном файле, при этом мы не меняем основной файл).

Директива Require all granted открывает доступ к файлам контента.

Блок:

```
<IfModule dir_module>
    DirectoryIndex index.html
</IfModule>
```

говорит Apache какие файлы индексные, т.е. открываются по запросу сайта без указания конкретного файла.

1.3 Application сервер

Сервер приложений (Application Server, App-Server) – это программный комплекс, предназначенный для доставки контента и средств его представления для клиентских приложений. Клиентами могут быть веб-приложения, браузеры и мобильные приложения.

Серверы приложений предоставляют для клиентов бизнес-логику, то есть, преобразуют данные в динамический контент и обеспечивают функционал приложений. Примеры такого контента:

- Результаты транзакций;
- Поддержка принятия решений;
- Аналитика в реальном времени, и др.

Сервер приложений – это связующее звено между клиентом и программным кодом физического сервера. Типичные задачи сервера приложений:

- Управление транзакциями;
- Безопасность;

- Внедрение зависимости DI (Dependency injection);
- Одновременность выполнения процессов (Concurrency).

Серверы приложений также обрабатывают такие процессы, как кластеризация, исправление отказов и балансировка нагрузки.

1.4 LAMP и WAMP и XAMPP

LAMP — это стек программного обеспечения, устанавливаемого на сервер и предназначенного для сайтов и веб-приложений. LAMP – это аббревиатура, она расшифровывается как: Linux, Apache, MySQL и PHP.

Рассмотрим каждый элемент LAMP подробнее:

Linux используется в качестве ОС на сервере, часто это различные дистрибутивы Ubuntu и Debian.

Apache — веб-сервер. Он обрабатывает все запросы к страницам сайта и выдает соответствующие ответы.

MySQL — СУБД (система управления базами данных). Иногда в LAMP используется MariaDB. Здесь хранятся все данные сайта.

PHP — скриптовый язык для генерации страниц.

Такой набор самостоятельных по отдельности компонентов стал очень популярен. Существуют готовые пакеты LAMP, которые можно скачать, к ним также прилагаются подробные инструкции к установке.

LAMP, упакованный Bitnami, представляет собой полную, полностью интегрированную и готовую к запуску среду разработки LAMP. Помимо PHP, MariaDB и Apache, он включает в себя phpMyAdmin, openSSL, ModSecurity, PageSpeed, Varnish, SQLite, ImageMagick, xDebug, Xcache, OpenLDAP, Memcache, OAuth, PEAR, PECL, APC, GD, cURL и Composer.

Существует множество вариаций LAMP, где какие-либо из компонентов заменяются на другие, например LEMP — вместо Apache в нем используется веб-сервер Nginx.

Например, WAMP – это такой же стек программного обеспечения, но вместо Linux в нем используется Windows.

- XAMPP – это кроссплатформенная сборка программного обеспечения, которая включает:
- Web-сервер Apache с поддержкой SSL
- СУБД MySQL
- PHP
- Perl
- FTP-сервер FileZilla
- POP3/SMTP сервер
- утилиту phpMyAdmin.

2 Инструменты создания web приложений

Рассмотрим несколько конкретных технологий применяемых в web разработке.

2.1 Веб-фреймворки Python

Python фреймворки содержат наборы инструментов, библиотек, полезных для разработчика модулей. В них реализованы часто встречающиеся задачи и заложена четкая структура кода. Это облегчает работу: позволяет сосредоточиться на логике проекта, не тратить время на повторяющиеся действия, которые уже имеются в ассортименте инструментов.

Фреймворки для веб-разработки бывают full-stack (полнофункциональные) и non full-stack (не полнофункциональные).

Full-stack фреймворки содержат комбинацию из базового сервера приложений HTTP, механизма хранения, такого как база данных, механизма шаблонов, диспетчера запросов, модуля аутентификации и набора инструментов AJAX. Они могут быть отдельными компонентами или предоставляться вместе, такой фреймворк называется высокоуровневым.

Non Full-stack фреймворки предоставляют базовый сервер приложений, работающий либо как отдельный сервер, либо совместно с Apache, либо с другими веб-серверами. Во многих из них можно самостоятельно добавлять необходимые механизмы.

2.1.1 Django

Django — это высокоуровневый full-stack (полнофункциональный) веб-фреймворк Python, который помогает быстрой разработке с меньшим количеством кода. Бесплатный и с открытым исходным кодом, который позволяет создавать приложения любого уровня.

Ключевые особенности Django:

- наличие собственного ORM (от англ. Object-Relational Mapping, «объектно- реляционное отображение»);
- встроенный административный интерфейс (простым языком — админка);
- шаблон проектирования MVC;
- библиотека работы с формами;
- система кэширования и интернализации;
- API;
- простой диспетчер URL;
- система авторизации и аутентификации.

2.1.2 Flask

У фреймворка Flask небольшой размер исходной кодовой базы, поэтому его называют микрофреймворком. Но это не значит, что у него меньше возможностей, чем у того же Django. По умолчанию он включает в себя только обработчик запросов и шаблонизатор, а простейшее приложение на Flask может состоять всего из нескольких строк. Разработчики этого фреймворка осознанно хотели сохранить ядро простым, но расширяемым.

Т.е. вы можете использовать любые расширения, которые могут вам понадобиться. Но нестандартные функции необходимо подбирать, устанавливать и настраивать.

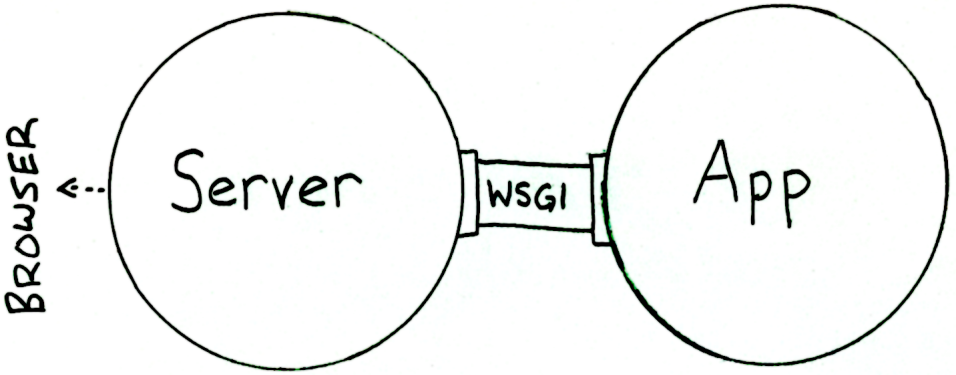
Тем не менее с помощью Flask можно реализовать практически любую задачу: от простого одностраничного сайта до серьёзного проекта с авторизацией, аутентификацией и другими возможностями. Flask подходит для задач, которые подразумевают гибкость в выборе компонентов.

Ключевые особенности Flask:

- встроенный сервер разработки и отладчик;
- диспетчеризация запросов в RESTful-стиле;

- встроенная поддержка модульного тестирования;
- зависит от WSGI и шаблонизатора Jinja2;
- множество расширений, предоставляемых сообществом.

WSGI – Web Server Gateway Interface – стандарт взаимодействия между Python-программой, выполняющейся на стороне сервера, и самим веб-сервером, например Apache.



2.1.3 Bottle

Bottle – быстрый и простой микрофреймворк для небольших веб-приложений. Он предлагает диспетчеризацию запросов (Routes) с поддержкой параметров URL, шаблоны, базы данных ключ-значение, встроенный HTTP-сервер и адаптеры для многих сторонних WSGI/HTTP-серверов и шаблонов. Все в одном файле и без каких-либо зависимостей, кроме стандартной библиотеки Python. Изначально он разрабатывался исключительно для создания веб-интерфейсов API.

Ключевые особенности Bottle:

- встроенный шаблонизатор в pythonic-стиле;
- встроенный веб-сервер;

- механизм маршрутизации запросов, с поддержкой параметров URL;
- адаптер для WSGI;
- легкий доступ к cookies, загрузке файлов, заголовкам запросов и другим метаданным.

2.1 Python Telegram бот

Одним из популярных направлений web-разработки является использование готовых сервисов и приложений. Например, использование API Vk или Telegram ботов. Сейчас многие пишут Telegram ботов и зарабатывают на них деньги, в виде донатов, например.

Бот может показывать погоду или курсы валют, или может взаимодействовать с каким-то сервисом через API и предоставлять возможность отправлять показания счетчиков в коммунальные службы, может извлекать для вас файлы из какой-то базы данных файлов (например электронные книги).

Главное задачей бота является автоматический ответ после введенной ему пользователем команды. При этом, работая непосредственно через интерфейса Telegram, программа имитирует действия живого пользователя, за счет чего использование ботов удобно и понятно.

Рассмотрим порядок создания бота. Первое, что нужно сделать – это создать бот. Делается это через бота, который можно найти по имени:

```
@BotFather
```

Который умеет:

— Создавать и удалять новых ботов.

— Выдавать и отзывать токены авторизации — это API-ключи, при помощи которых бот подключается к каналу и может работать.

— Редактировать ботов: менять имя, описание, аватарку, команды.

— Изменять настройки встроенной обратной связи и конфиденциальности в группах.

Когда вы отправите ему команду “старт”, последует следующий диалог:

```
                                                        /newbot

Alright, a new bot. How are we going to call it? Please choose a
name for your bot.
                                                  natashabotv1
```

```
Good. Now let's choose a username for your bot. It must end in
`bot`. Like this, for example: TetrisBot or tetris_bot.

                                                    NatashaV1Bot

Done! Congratulations on your new bot.
...
Use this token to access the HTTP API:
555333693:AAE2GTwQ555wcT66jjzUEZcJh1St7w9u-V8
```

В результате мы зададим боту:

– имя, name – это имя по которому можно будет найти и использовать бот,

– логин, username.

Далее следует выполнить команду:

```
/setprivacy
```

если нужно, чтобы бот отвечал не только на команды начинающиеся с “/”.

Самое главное мы получим ID бота, его надо сохранить и хранить в тайне. Теперь в Telegram можно найти бот по имени и начать с ним общаться. НО это не очень получится, так как бот ничего не умеет. Нужно написать его начинку.

Python имеет библиотек для работы с ботами. Чтобы создать обработчик событий бота, нужно создать новый проект в Python и импортировать в него библиотеку aiogram:

```
from aiogram import Bot, types
from aiogram.dispatcher import Dispatcher
from aiogram.utils import executor
```

Инициализируем объекты бота и диспетчера:

```
bot = Bot(token='555333693:AAE2GTwQ555wcT66jjzUEZcJh1St7w9u-V8')
dp = Dispatcher(bot)
```

Теперь добавим хэндлер.

Handler - это механизм, который позволяет работать с очередью сообщений.

Добавляем обработчик команды /start:

```
@dp.message_handler(commands=['start'])
async def process_start_command(message: types.Message):
    await message.reply("Привет!\nНапиши что-нибудь!")
```

Добавляем ещё один хэндлер для команды /help:

```
@dp.message_handler(commands=['help'])
async def process_help_command(message: types.Message):
    await message.reply("Я умею переворачивать слова!")
```

А теперь запишем основную логику бота, создадим хэндлер для любого сообщения:

```
@dp.message_handler()
async def answer_message(message: types.Message):
    a = message.text[::-1]
    await message.reply(a.lower())
```

Чтобы получать сообщения от серверов Telegram воспользуемся поллингом (polling, to poll - опрашивать) - постоянным опросом сервера на наличие новых обновлений. Для этого дописываем в код следующее:

```
if __name__ == '__main__':
    executor.start_polling(dp)
```

Чтобы бот работал скрипт Python должен быть запущен.

Телеграм использует собственный протокол шифрования MTProto. MTProto API (он же Telegram API) — это API, через который приложение Телеграм связывается с сервером. Telegram API полностью открыт, так что можно написать свой клиент мессенджера.

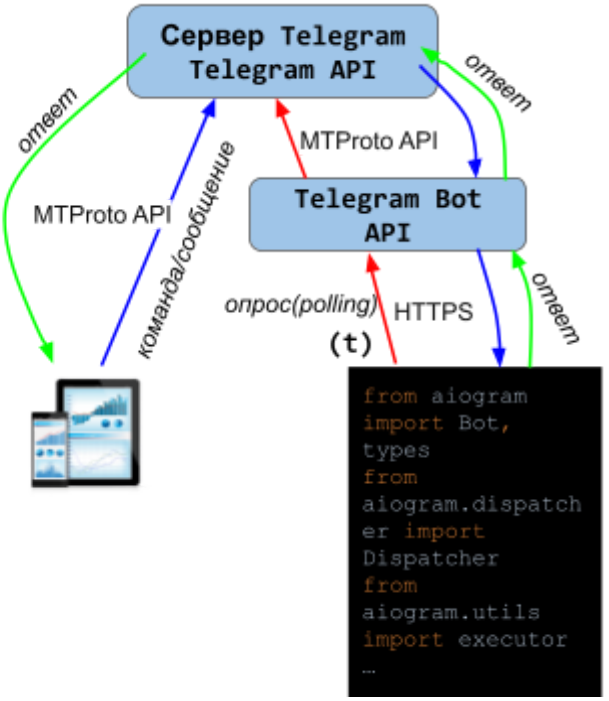
Для написания ботов был создан Telegram Bot API — надстройка над Telegram API. На официальном сайте сказано: “Чтобы использовать Bot API, вам не нужно знать о том, как работает протокол шифрования MTProto — наш

вспомогательный сервер будет сам обрабатывать все шифрование и связь с Telegram API. Вы соединяетесь с сервером через простой HTTPS-интерфейс, который предоставляет простую версию Telegram API”.

Боты могут работать напрямую через Telegram API. Более того, таким образом можно обойти некоторые ограничения, которые дает Bot API.

Работа бота

```
                    /start

Привет!
Напиши что-нибудь!

                    /help

Я умею переворачивать
слова!

          А в Енисее синева

авенис еесине в а
```



13