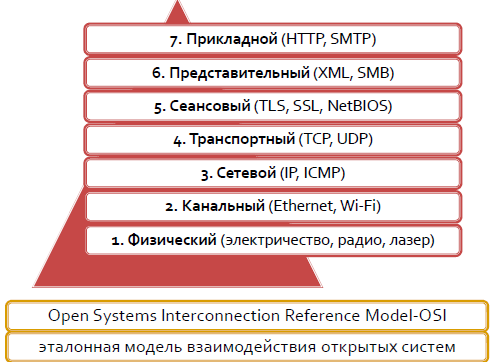
# ВЕБ-МАСТЕРИНГ - лекции Борисова

# ОСНОВЫ СЕТЕЙ

**OSI - Open System Interconnection Reference Model -** OSI (эталонная модель взаимодействия открытых систем)

Создана для шаблона передачи информации

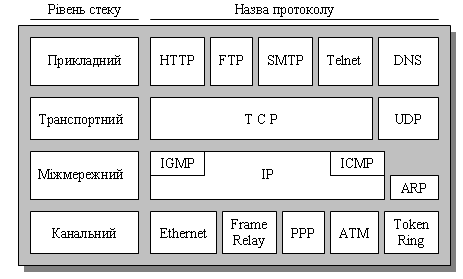
**7 уровень** - протокол высокого уровня. Описывает, что делать браузеру, который ввел пользователь. Данные преобразуются согласно протоколу и передаются на **6 уровень.** На **6 уровне** идет например преобразование картинок в какие-то данные.

**5 уровень** - шифрование

**4 уровень** - данные разбиваются на пакеты и передаются на **3 уровень**

**3 уровень -** протокол. Это правила передачи данных.

**1-2 уровень -** уровень железа. Способ передачи информации

Обычно модель OSI выражают в 4 уровня, просто объеденяя предыдущие:

**Прикладной** обьеденяет 7-5 уровни

Транспортный

Сетевой

**Канальный** обьеденяет 1-2 уровни

IP адреса и сети

**IP(v4)** - 213.180.204.8

**Маска сети** - 255.255.255.0

При обращении одной машины к другой, сравнивается класс сети. Если класс 255.255.255.0 и у одной машины, и у второй - значит номер машины хранится в последней цифре, в данном случае 8

Есть три класса сетей:

Класс А. 255.0.0.0 - для сетей, где очень много машин, но мало самих подсетей

Класс Б 255.255.0.0

Класс С 255.255.255.0 - где много подсетей, но мало машин внутри каждой

Зарезервированные адреса

255.255.255.255 - limited broadcast

x.x.x.255 - direct broadcast - обращается ко всем машинам в данной сети

Адреса для построения локальных сетей

10.х.х.х , 172.16.х.х, 192.168.х.х

Для внутреннего использования:

127.0.0.1 (localhost) - машина обращается сама к себе, ищет программу, которая готова принять запрос по этому адресу

**cmd > ipconfig** - вывод информации о сети на машине

**ipconfig /all** - вод подробных сведений

При соединении по локальной сети машины используют IP адрес и MAC-адрес.

При соединении через интернет MAC-адрес получить невозможно, т.к. он не передается браузером

Маршрутизация

Маршрутизатор имеет шлюз внутренний и шлюз внешний (**основной шлюз**).

Алгоритм передачи выглядит так:

Пользователь пишет запрос на локальной машине.

ОС обрабатывает запрос и задает ему порт. На каждый запрос задается свой порт: 192.168.1.1:2000, 192.168.1.1:2001 - это называется **сокетами**

Запрос передается маршрутизатору, который определяет, куда должен пойти запрос. Если маска сети соответствует локальной сети - обращается к соответствующей машине внутри сети. Если вне локальной сети - запрос передается на **основной шлюз** (внешний IP-адрес).

Маршрутизатор обрабатывает внутренние сокеты и делает их внешними (с использованием внешнего IP):

192.168.1.1:2000 => 83.217.212.283:3045, 192.168.1.1:2000 => 83.217.212.283:3046

Запрос уходит глобальным маршрутизаторам, где по таблицам маршрутизации определяется, какой IP надо найти для передачи

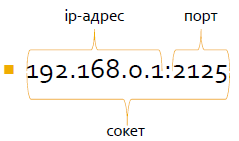
Запрос принимающим маршрутизатором, передает его нужной машине, и возвращает назад

Что бы узнать, какая машина должна принять запрос, используется посфикс к номеру IP - порт. Обращение идет от IP с портом до IP с портом, например:

от 192.168.0.1:2125 до 213.180.204.3:80

Порт

Аналог - добавочный номер телефона на основном номере

Известные порты - 0-100, например порт 80, который используется по умолчанию веб-службой. Когда мы пишем google.com по умолчанию используется порт 80.

Условно-известные порты: сервер MySQL занимает по умолчанию 3306

При каждом запросе с локальной машины, формируется новый порт

**cmd => ping** google.com - пингует указаный адрес. Посылает 3 пакета сразу. Можно проверить любой IP-адрес, а не конкретный порт

**cmd => tracert** google.com - пингует каждый узел на пути к конечной цели. Посылает 3 пакета на каждый узел

**TTL** - число прыжков. При команде **ping**, TTL изначально задается большой, что бы конечные узлы не успели его обнулить

При **tracert,** TTL задается локальной машиной каждый раз в 1, после чего пакет с этим TTL=1 отправляется на следующий узел. Если следующий узел - не конечный, он отнимает единицу и возвращает 0, что означает успех передачи

Локальная машина обрабатывает пакет с TTL=0, и посылает уже TTL=2. Первый узел обрабатывает, отнимает единицу (TTL = 2-1), и пересылает дальше. Каждый последующий узел отнимает единицу от TTL, пока не дойдет до конечного.

Если узел не успевает отвечать, локальная машина не ждет ответа и шлет следующий запрос c TTL++, но указывает прочерк в принятых пакетах от текущего узла, который пакет не возвратил

**Порты служб, задействованых в Интернет:**

FTP - 21

Telenet - 23

SMTP - 25

DNS - 53

Web - 80

POP3 - 110

IMAP4 - 143

MySQL - 3306

Если две программы пытаются использовать один порт, пользоваться ей будет та, которая заняла его первой

Что бы проверить, свободен ли порт используется команда netstat

**cmd => netstat -an** - вывод текущих подключений со всеми IP

**netstat -an | FIND "LIST" - более короткий список**

-a - отображение всех подключений и ожилающих портов

-n отображение адресов и номеров портов в числовом формате

-o отображение ID процесса каждого подключения

TCPView - маленькая программа, которая в режиме реального времени отслеживает текущие подключения

**DNS - Domain Name System**

При отправке запроса, например с google.com отправляется запрос на DNS сервер, который сопоставляет с именем IP сайта

Назначаются обычно два сервера - один основной и один запасной

forum.specialist.ru. - точка является корнем. В реальности в самом конце точку можно не ставить

Считывание адреса идет справа налево:

самый корневой сервер **.** => корневой сервер **ru** => **specialist** сервер => поддомен **forum**

Все DNS кешируют информацию, потому нужный адрес можно получить сразу от первого сервера

Так же ОС кеширует информацию, потому если запрос был сделан недавно, он отложится туда

**Файл hosts** C:\windows\system32\drivers\etc - это мини DNS сервер. Адреса, прописанные там автоматически кешируются в ОС

**cmd => nslookup google.com -** посмотреть IP адреса указанного сайта, так же, какому доменному имени принадлежит IP адрес

**cmd =>** **ipconfig /displaydns** - посмотреть текущий кеш DNS

**cmd =>** **ipconfig /flushdns -** чистит весь кеш ОС

****

**HTTP HyperText Transfer Protocol**

**Запрос клиента**

В нем описано что делать на верхнем уровне OSI - уровень приложений

После установки соединения происходит запрос клиента такого вида:

GET /folder/index.html HTTP/1.1

GET - метод запроса. Каждый раз когда что-то вбивается в строку браузера, используется метод GET

/folder/index.html - адрес запрашиваемого ресурса. Слеш ( **/** ) - это корень сайта, доменное имя отсутствует, т.к. соединение уже установлено ранее

HTTP/1.1 - используется постоянно, оставляется по умолчанию

Признаком конца строки служит возврат каретки и перевод на новую строку (enter)

Далее идут заголовки

Host: www.specialist.ru

Это обязательный заголовок, при отсутствии которого сервер может отклонить запрос как непонятный. Сервер сортирует все заголовки в массив, называя переменные приставкой http\_ к каждому заголовку:

“http\_host”: “www.specialist.ru”

Следующие заголовки посылаются клиентом по желанию. Заголовки формирует браузер

Концом передачи заголовком служит пустая строка

**Сервер сохраняет запрос в переменных:**

$\_SERVER[“REQUEST\_METHOD”] = “GET”

$\_SERVER[“REQUEST\_URI”] = “/folder/index.html”

**Ответ сервера:**

HTTP/1.1 200 OK

Далее идут заголовки, концом заголовков служит пустая строка, после чего передается html-код

Date: Sun. 17 Aug 2008 07:47:24 GMT

Server: Microsoft-IIS/6.0

Last modified: Mon, 07 Jul 2008 11:07:04 GMT

Accept-Ranges: bytes

Content-Length: 34234

Content-Type: text/html

<html>

...

</html>

Поскольку у html нет явного окончания строки, окончанием служит указанный в заголовках Content-Length

Браузер получает файл и начинает его парсить. Во время парсинга, он может наткнутся на src, url и т.д., что даст ему команду на отсылку нового запроса, например переслать картинку:

<img src="img/car.png">

После чего процедура запроса-ответа повторяется

**Метод HEAD**

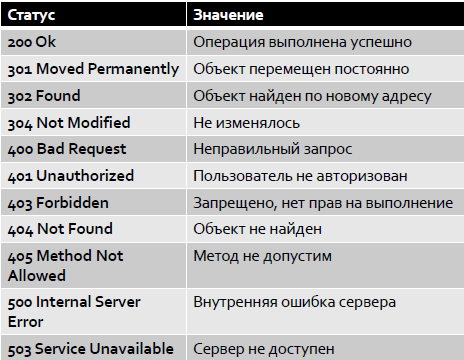
Используется для того, что бы получить заголовки без содержимого. Например, проверить вес файла, который планируется передать

// можно проверить в **cmd > telnet**

open l95068ow.bget.ru 80

HEAD / HTTP/1.1

Host: [www.l95068ow.bget.ru](http://www.l95068ow.bget.ru)

**Статусы ответа сервера**

**1хх** - Информационное сообщение (крайне редко встречаются)

**2хх** - успешное выполнение команды

**3хх** - переадресация (как правило, нужен еще один запрос)

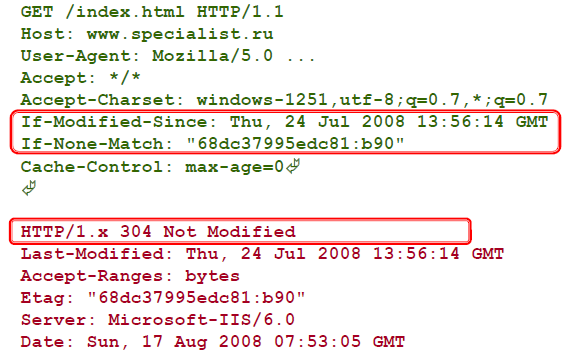
**4хх** - ошибка на стороне клиента

**5хх** Ошибка на стороне сервера

**Запрос с валидатором**

Для экономии трафика, браузер кэширует страницы. Потому при коннекте на ту же страницу, в заголовках браузер уточняет, были ли изменения с даты прошлого посещения, по сохраненному в памяти заголовку с прошлого посещения:

Last modified: Mon, 07 Jul 2008 11:07:04 GMT

 Статус 304 означает, что изменений не было, и браузер покажет закэшерированную страницу.

<=Если менялся с - уточняет

<=Если не менялся - взять из кэша

Если ресурс менялся, вернется статус 200 ОК

Большинство программистов не ставят Etag, хватает простого Modified

**Метод передачи GET**

При отправке формы, браузер запрашивает дополнительную информацию, которая перечисленна после знака вопроса (?) в адресной строке. Это строка содержится в переменной $\_SERVER["QUERY\_STRING"]

**Метод передачи POST**



POST безопасен при передаче данных, т.к. скрывает их

Отличие от GET:

- дополнительные заголовки:

referer - адрес (из адресной строки браузера), откуда пришел запрос

Content-Type - тип данных

Content-Length - длинна строки

тело метода, которе содержит скрытыю информацию. Отделяется пустой строкой

**Cookies**

Когда человек заполняет форму или совершает какие-то действия, сервер обрабатывает запрос и посылает команду браузеру сохранить информацию в cookies. Во следующий раз браузер передает cookies серверу, который тот обрабатывает и выдает нужную информацию

<= запрос из браузера

<= сервер отдает информацию со своими сохраненными cookie

<= в следующий коннект браузер уже посылает куки с идентификацией пользователя

**Web-server Microsoft IIS - Internet Infotmation Services**

Если сервер стоит - в корне диска C создается папка wwwroot. Если поставлен ftp сервер - будет папка ftproot, если почтовый - mailroot

Устанавливается через панель управления - программы и компоненты - компоненты Windows - службы IIS

Настройки IIS наследуются начиная от родительской папки и ниже. Потому настройки сайта лучше раздавать непосредственно в папках самих сайтов (из панели управления IIS), чем менять глобальные

**Привязки к адресам**

Default web server - справа "привязки". В привязках можно добавлять адреса, которые будут учитываться веб-сервером

**Путь к папке**

Сайты - Default Web Server - справа "Дополнительные параметры" - можно указать физический путь к папке с сайтом

**Журналирование**

Ведение журнала - тут можно выбрать поля, в какие будут выводится при ведении логов, как часто он будет создавать новые файлы и где будет их хранить

**Документы по умолчанию**

Для каждой папки можно указать список файлов (по приоритету) которые отображаются при обращении к папке

**Заголовки ответов HTTP**

Тут можно создать свои заголовки, которые будут посылаться клиенту

**Перенаправление протокола HTTP**

Можно задать другой адрес для перенаправления (например, на <https://google.com>)

**Просмотр каталога**

Можно включить отображение папок в каталоге, который будет отображен при отсутствии индексного файла в папке

**Создать второй сайт в IIS**

В левой панели клик правой кнопки по сайтам - вводим там название сайта, каждому сайту задается другой IP:

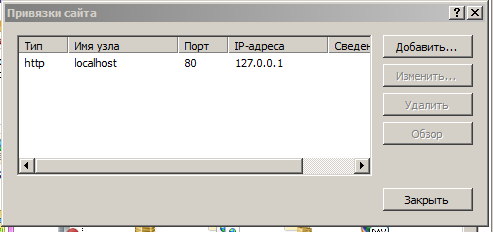
IP: 127:0.0.2 port: 80

Потом идет в hosts и прописываем:

127.0.0.2 petya.ru

**Вариант с одним IP - через заголовок Host**

Поскольку URL-адреса являются ссылками на IP адрес, можно создать сколько угодно привязок

для одного IP адреса:

Заходим в Привязки, пишем:

IP - 127.0.01, имя узла - petya.ru

После этого в hosts надо будет тоже изменить адрес на

127.0.0.1 petya.ru

**Создание подкатегории:**

Правой кнопкой по иконке сайта справа - добавить виртуальный каталог

* псевдоним - что нужно будет дописать в адресной строке браузера после имени основного сайта, напимер mysite.local/err
* Физический путь - путь к папке с этим каталогом

**Создание собственной страницы с ошибками:**

Панель управления сайтом - страница ошибок - выполнить URL на этом сайте

ввести например /404.html, если 404 страничка лежит в корне сайта. Потом кликнуть в правой панели "Изменить параметры", там выбрать "Страницы настраиваемых ошибок"

Если не показывает собственную страничку, ищем в папке локального сайта файл web.config и меняем там строку:

<httpErrors errorMode="Custom">

**MIME-типы**

Это спецификация, которая описывает форматирование не-текстовых типов файлов через Интернет.

**Алгортим обработки запросов:**

* Браузер посылает запрос со страничкой, например <http://mysite/test.asp>
* Сервер обрабатывает запрос, читает расширение, ищет соответствующий обработчик
* Передает файл в обработчик, получает от обработчика результат и передает в браузер с соответствующим заголовком, (например Content-Type: asp)
* Когда браузер парсит полученный файл и видит соответствующий заголовок, он подключает свой парсер под этот конкретный тип. Если заголовок Content-type не соответствует содержимому, (например указан text/html, а внутри лежит картинка), может быть выдана ошибка.

Content-Type: text/html

Content-Type: image/jpeg

Content-Type: application/rtf

В IIS их можно задать в соответствующем подразделе в контрольной панели сервера

**Кэширование**

Метод POST не кешируется, GET - кешируется

После запроса, браузер проверяет статус ответа. Если статус 304, он поднимает показывает страницу из кэша.

Кэш может сохраняться как браузером, так и прокси-серверами, через которые проходит трафик.

Что браузеру и транзитным серверам делать с кэшем, можно указать на сервере через заголовок **Cache-Control**:

Cache-Control: no-cache, no-store, max-age=0

* no-cache - браузеры очищают кэш только после закрытия браузера, но пока браузер открыт, кэш будет сохранятся в памяти
* no-store - вообще запрещает кэширование
* max-age=0 - время актуальности контента на странице, в секундах
* private - запрещает кэширование серверами, но браузер может кэшировать. Так же, браузер кэшиурет в папку конкретного пользователя, не кэшируя в папку для общих пользователей
* public - можно кэшировать в общую папку
* max-revalidate - проверка, не изменилась ли эта страница

Заголовок **Expires** указывает когда истекает срок кэша, определенное время. Если указан max-age, Expires учтен не будет.

Expires: Thu, 01 Dec 1994 16:00:00 GMT

В IIS кэширование настраивается в *Заголовках HTTP => правая колонка, "Настройки стандартных заголовков*

В IIS Сопоставление обработчиков - указываются все обработчики расширений, которые подключены в данный момент, можно добавлять свои

Под IIS есть интерпретатор Администратор PHP, который подключается и появляется в контрольной панели

**Хэширование**

Хэширование - это односторонее шифрование.

Хэширование - преобразование входного массива данных произвольной длины в выходную битовую строку фиксированной длины.

Все введенные данные хешируются и сохраняются в базе данных в виде хэш-кода. На сервере принято "сорить": перед сохранением хэш-строки в нее добавляются случайные символы, для осложнения возможности взлома

Одна и та же строка дает тот же хэш постоянно:

"Hello, world" - в MD5 это выглядит как bc6e6f1b8a077ef5fbc8d59dob931b9

**Нессиметричная криптография**

* Каждый клиент генерирует два ключа: закрытый и открытый.
* Клиент А шифрует сообщение открытым ключом Клиента B и посылает его
* Теперь сообщение может расшифровать только Клиент B с помощью его закрытого ключа

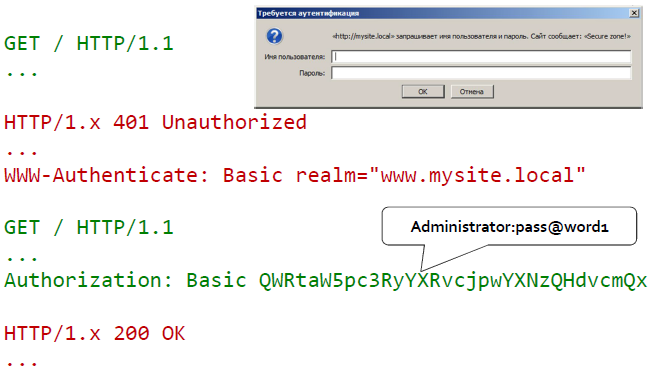
Проблема 3-го лица: сообщения могут перехватываться.

Потому появилась технология подписанного сообщения, когда доверенному третьему лицу - Тренду, пересылаются открытые ключи Клиента А и Клиента Б, и все сообщения идут через Тренд

В IIS завести пользователя можно только через сторонний плагин, если чистый IIS - надо заводить пользователя на уровне системы, просто через windows - новый пользователь.

**Проверка подлинности** - можно указать пользователь который будет передаваться браузером для сервера, обмен данными идет на уровне сервера.

В заголовках это выглядит так:



* Запрос клиента для сервера
* Сервер представляется через заголовок

WWW-Authenticate

* Браузер передает анонимную авторизацию либо администратора системы, либо гостя

Что бы обезопасить передаваемые данные, авторизация оборачивается в **SSL**

**https - приставка "s" обозначет то, что используется сертификат**

**Сертификаты**

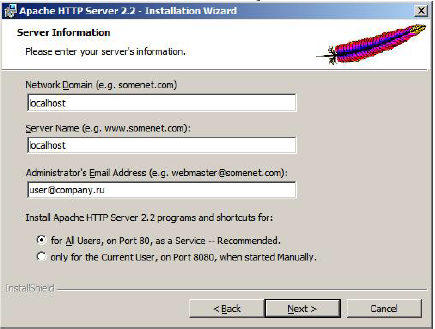
Бывают двух видов: подписанные или самоподписанные

* Подписанные - в сертификат внедряется цифровая подпись доверенного центра. В браузере есть список доверенных центров.
* Самоподписанные - выписываются сами себе. Зачем нужно? SSL не установится без сертификата.

**APACHE WEB SERVER**

Официальный сайт: <https://httpd.apache.org/>

Страница с бинарниками для Windows: <http://httpd.apache.org/docs/current/platform/windows.html#down>



<= Установка чистого Апача

Разница между версиями 2.2 и 2.4 минимальна.

Для разработчика по сути не имеет значения, чем пользоваться

При установке Apache особенность - надо указать localhost при установке

По умолчанию создается папка в Program Files.

Лучшее ее перенести в другую папку, что бы хватило прав для изменения настроек

Я качал с ApacheHaus: скачать-распоковать, положить папку в корневую директорию диска. Сервер можно запускать из bin - *httpd.exe*

Что бы запустить, как службу: открываем cmd под **правами админа,** там переходим в bin папку апача:

cd C:\Apache24\bin

httpd -k install

Далее идет установка его как службы, после чего становятся доступны следующие команды:

|  |  |
| --- | --- |
| httpd -k start | Start Apache |
| httpd -k stop | Stop Apache |
| httpd -k restarе | Restart Apache |
| httpd -k uninstall | Uninstall Apache Service |
| httpd -t | Test Config Syntax |
| httpd -V | Version Details |
| httpd -h | Command Line Options List |

**Запуск Apache из командной строки:**

cmd (с правами админа) > net start apache2.4 // или другое название службы

**Быстрый запуск программ windows из cmd:**

Правой кнопкой по "мой комп" => свойства дополнительные параметры системы => Переменные Среды => НИЖНЕЕ ОКНО С ПЕРЕМЕННЫМИ => **Path**, два клика по ней => вставляем точку с запятой, дальше вставляем путь: C:\Apache24\bin

Если есть **ошибки с запуском**, необходимо запускать апач из-под командной строки, она укажет на ошибки

cmd > httpd -k start // если путь к bin указан в windows-переменной path

**Главный конфигурационный файл Apache**

apache/conf/httpd.conf - основной конфигурационный файл

apache/conf/extra - подключаемые редкоиспользуемые настройки конфига

**Синтаксис конфигурационных файлов:**

* Строки лучше писать в кавычках, что бы не было ошибок (типа пробелов)
* Комментарий - начинается со слеша, если хотим запретить какую-то команду

# Mutex default:logs

* Однострочная директива - это глобальные команды

ServerRoot "${SRVROOT}"

* Блочная директива - это команды с локальной областью видимости

<Files ".ht\*">

Require all denied

</Files>

* Условные директивы - если подуль подключен, исполни следующую команду

<IfModule log\_config\_module>

* Разрешить запись **.htaccess** для каждой папки - ищем в httpd.conf строчку

AllowOverride None и меняем на AllowOverride All

Теперь для каждой папки можно настроить доступы. Синтаксис .htaccess такой же, как главного конфигурационного файла

**Создать виртуальный хост**

<VirtualHost \*:80> - указывает порт, по которому будет прослушиваться хост

ServerName [myite.local](http://www.myite.local) - основное виртуальное серверное имя

ServerAlias [www.mysite.local](http://www.mysite.local) - псевдоним основного виртуального имени, ссылка

DocumentRoot "C:/sites/mysite/htdocs" - указывается корень сервера

Directory "C:/sites/mysite/htdocs"> - сразу идут настройки этой папки

Options Indexes FollowSymLinks - если файл не найден, будет показана папка. FollowSymlinks нужен для корректной работы REWRITE\_MODULE

AllowOverride All - разрешает настройки локальных папок через .htaccess

Require all granted - сайт будет доступен со всех IP адресов

<VirtualHost \*:80>

# Setting the server

ServerName mysite.local

ServerAlias www.mysite.local

ServerAdmin admin@mysite.local

# Journaling

ErrorLog "/logs"

CustomLog "C:/Apache24/mysite/logs/access.log" common

# Root Directory

DocumentRoot "C:/Apache24/mysite/htdocs"

<Directory "C:/Apache24/mysite/htdocs">

Options Indexes FollowSymLinks

AllowOverride All

Require all granted

</Directory>

</VirtualHost>

Вышеописанные настройки так же можно подключить к глобальному файлу через указание include:

Include conf/extra/mysite.conf

**После создания конфига** не забыть прописать настройки в host: 127.0.0.1 mysite.local

localhost начинает направлять на новый сайт, т.к. по умолчанию у него не прописано дефолтного сайта, это просто папка, куда сервер перенаправляет при обращении кнему

Если при запуске не выдает никаки ошибок в командной строке, но и не запускается -

99% что неправильно указана папка для журналирования - ее либо не существует, либо ошибка в написании:

ErrorLog "/logs"

**.htaccess**

Options Indexes // разрешает просмотр папки, если не найден файл по умолчанию

Options -Indexes // запрещает просмотр папки, если не найден файл по умолчанию

DirectoryIndex index.html test.html // приоритет файлов, открывающихся по умолчанию

**Настройка папки журналов**

CustomLog - лог доступов к сайту

common **-** значит берутся настройки по умолчанию (наследуемые). Если указать метку после создания своего CustomLog, то сначала метка будет искаться в конфигурационном файле, в строках LogFormat, который ближе всего к целевой папке (mysite.conf), если там не найдено - будут подставляться общие значения из httpd.conf

CustomLog "/logs/access.log" common

CustomLog "/logs/agents.log" agent

CustomLog "/logs/refs.log" referer

LogFormat "%h %l %u %t \"r\" %>s %b common

LogFormat "%{User-Agent}i" agent

LogFormat "%{Referer}i" agent

**Создание виртуальной папки**

Alias **-** как будет доступна виртуальная папка относительно корня сайта, послее идет физический адрес. Если поставить после имени алиаса слеш ( **/** ), он будет рассматриваться как часть алиаса!

Так же в адресе директории, в конце должен стоять слеш **/**

**Alias /err "C:/Apache24/mysite/errors/"**

**<Directory "C:/Apache24/mysite/errors/">**

**Options Indexes FollowSymLinks**

**AllowOverride All**

**Require all granted**

**</Directory>**

**Настройка сообщений об ошибках**

Ошибки пишутся через директиву ErrorDoсument, после чего идет номер ошибки и путь к файлу, который должен быть показан

ErrorDocument 500 <http://www.site.com/tester>

ErrorDocument 404 /bin/bad\_urls.html

ErrorDocument 403 "Sorry can't allow you to access today"

**Настройка отправляемых заголовков**

Приоритет скриптов выше, чем приоритет .htaccess, потому php будет заменять настройки .htaccess.

Для возможности отправки заголовков, в **httpd.conf** надо проверить, что бы были подключены

(в стандартной поставке Apache по умолчанию выключены):

LoadModule headers\_module modules/mod\_headers.so

LoadModule expires\_module modules/mod\_expires.so

Затем в своем файле, например **.htaccess** своей папкиможно писать:

set - добавить свой заголовок или перезаписать существующий

add - добавить значение к существующему

unset - удалить заголовок

Header set MyHeader "Value of header"

Header add MyHeader "Value of header + 2"

Header unset MyHeader

Header set Cache-Control "private, no-cache, no-store, max-age=0"

ExpiresActive - директива, которая позволяет включить настройки кэширования, идентичен **max-age**

ExpiresDefault - устанавливает время кеширования:

* now - не кешируется
* access - от времени доступа
* modification - от времени модификации файла

ExpiresByType - какой тип сколько времени будет хранится в кэше

ExpiresActive On

ExpiresDefault "now"

ExpiresDefault "access plus 1 year 5 mounth 3 weeks 2 days 10 hours 25 minutes"

ExpiresDefault "modification plus days"

ExpiresByType text/html "acess plus 2 days"

ExpiresByType image/gif "acess plus 2 days"

**Настройка MIME-типов**

Папка conf > mime.types

В apache по умолчпнию тип высталвен в text\plain (**DefaultType text/plain**), потому если он не знает расширения, он выдаст чистый текст

Добавить новый тип -

Addtype text/html .zzz

**Привязка обработчиков в apache**

Можно прикруть обработчик двумя способами:

* как CGI-программу (exe-файл). При запросе определенного файла, сервер запускает обработчик, передает ему файл, получает ответ и выключат обработчик. Для безопасного запуска скриптов, apache запускает его в специальной папке **cgi-bin.** Что бы указать серверу, какие расширения надо обрабатывать в этой папке, прописываются такие настройки

ScriptAlias /cgi-bin/ /www/cgi-bin/

AddHandler cgi-script .php

Что бы указать exe для сервера, обработчик надо прописывать каждый раз в самом скрипте, в первой строчке, начиная с символов **#!**

#!c:php/bin/php.exe

* как модуль. Можно подключить в httpd.conf как обычный модуль:

LoadModule php5\_module "C:/OpenServer/modules/php/PHP-5.6/php5apache2\_4.dll"

AddType application/x-httpd-php .php

По историческим причинам, php обычно подключается как модуль. Если apache x32, то php тоже должен быть x32

**Аутентификация пользователей**

Аутентификация - это не авторизация. Аутентификация - представление пользователя, который сидит за браузером в данный момент

Файлы Apache обычно хранятся в файле .htpasswd. Открывается он через утилиту htpasswd.exe.

Создание нового файла. Ключ -c создается только при создании файла, дальше его писать не надо, иначе файл будет перезаписан. Папку, в которую будет записан файл, нужно предварительно создать (secure, в данном случае)

cmd => htpasswd -c c:\secure\.htpasswd admin

cmd => htpasswd c:\secure\.htpasswd vasya

В файле **.htaccess** в папке secure пишем (что бы сделать запароленным доступ в папку secure):

AuthType Basic

AuthName "Restricted Resource"

AuthUserFile "C:/Apache24/secure/.htpasswd"

Require valid-user

Authtype - тип аутентификации

AuthName - строка, можно писать что угодно, которая будет выводится при доступе в папку

AuthUserFile - путь к файлу с паролями. Надо указывать полный путь

Ложим файл .htaccess в ту папку, которую хотим запаролить

Что бы запароль файл

<FilesMatch "\.(zip|rar)$">

AuthType Basic

AuthName "Secure Folder"

AuthUserFile "c:/secure/.htpasswd"

Require valid-user

</FilesMatch>

Эффект не будет заметен, если файл находится в папке, которая уже требовала аутентификацию ранее

**Редирект на Apache**

Синтаксис редиректа, разделяетя пробелами. Тут 3 переданных аргумента

Redirect 301 /old.html <http://site.ru/new.html>

1. Redirect - команда
2. 301 - статус, не обязателен к указанию. Если опустить, будет Редирект под статусом 302
3. /old.html - старая страничка. Всегда указывается относительно корня сайта!
4. Адрес, куда была перемещена страничка

Пакетный редирект - когда было перемещено сразу много страничек:

RedirectMatch 301 /(.\*)\.html <http://site.ru/$1.php>

Данная операция перемещает все файлы по другому адресу и переименовывает html в php

**REWRITE\_MODULE . ЧПУ - человекопонятный URL.**

В Apache отвечает за это модуль rewrite. По умолчанию отключен

LoadModule rewrite\_module modules/mod\_rewrite.so

Принцип работы - подмена адресных строка

* site.ru => раздел /about, пользователь хочет туда зайти
* /about передается на сервер. Сервер сразу же передает /about в rewrite\_module
* rewrite преобразует /about в /index.php?id=about и отдает серверу
* Сервер ищет страничку по данным параметрам и отдает пользователю, у которого все так же остается красивый адрес в строке

Например строку **/news/2013/08/21**

надо превратить в **/index.php?category=news&year=2013&mon=08&day=21**

Пишем в .htaccess главной папки сайта:

Options Indexes FollowSymLinks

RewriteEngine On

RewriteRule ^([a-zA-Z]+)/([0-9]{4})/([0-9]{1,2})/([0-9]{1.2})/?$

index.html?category=$1&year=$2&mon=$3&day=$4

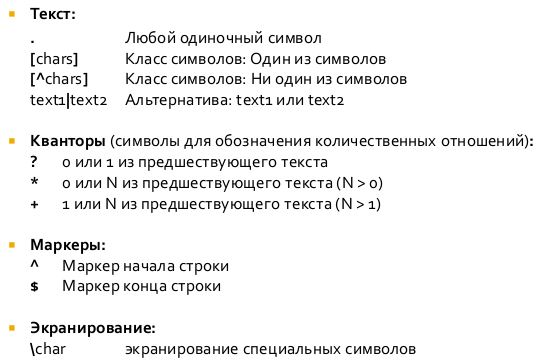
RewriteCond

RewriteEngine On - включение мода

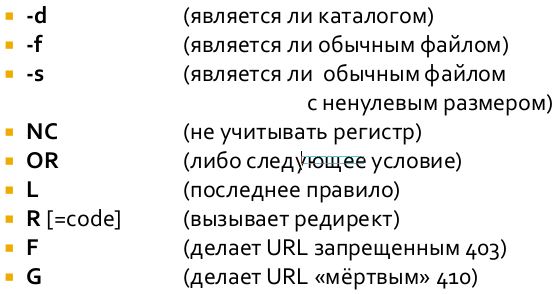
RewriteRule - 1й аргумент - шаблон, который будет применен ко второму аргументу, 2й аргумент - строка-донор

RewriteCond - аналог if/else: если что-то попало под правило, примени правило

**Шаблоны регулярных выражений:**

****

**Модификаторы и флаги**



**Примеры использования REWRITE\_MODULE**