## Веб-сервер: функционал, задачи, администрирование

Веб-сервер — это сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными.

Веб-сервером называют как программное обеспечение, выполняющее функции веб-сервера, так и непосредственно компьютер, на котором это программное обеспечение работает.

Клиент, которым обычно является веб-браузер, передаёт веб-серверу запросы на получение ресурсов, обозначенных URL-адресами.

Ресурсы — это HTML-страницы, изображения, файлы, медиа-потоки или другие данные, которые необходимы клиенту. В ответ веб-сервер передаёт клиенту запрошенные данные. Этот обмен происходит по протоколу HTTP.

Дополнительные функции

Веб-серверы могут иметь различные дополнительные функции, например:

• Автоматизация работы веб страниц;

• Ведение журнала обращений пользователей к ресурсам;

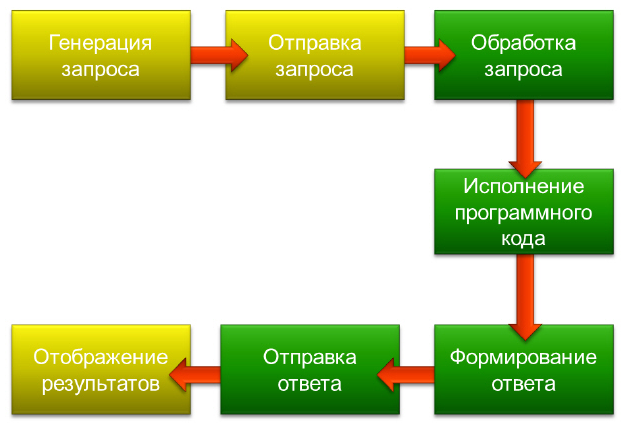
• аутентификация и авторизация пользователей;

• поддержка динамически генерируемых страниц;

• поддержка HTTPS для защищённых соединений с клиентами.

Часто на компьютере вместе с веб-сервером устанавливается также и почтовый сервер.

Алгоритм работы веб-сервера:



После того, как пользователь обратился к определенному ресурсу по протоколу HTTP, клиент (обычно браузер) формирует HTTP-запрос к веб-серверу. Обычно указывается символическое имя сервера (например, "[http://www.microsoft.com](http://www.microsoft.com/)") – в этом случае браузер предварительно преобразует это имя в IP-адрес при помощи сервисов DNS.

После этого по протоколу HTTP на веб-сервер отправляется сформированное HTTP-сообщение. В этом сообщении браузер указывает какой ресурс необходимо загрузить и всю дополнительную информацию. Задача веб-сервера – прослушивать определенный TCP-порт (обычно порт 80) и принимать все входящие HTTP-сообщения. Если входящие данные не соответствуют формату сообщения HTTP, то такой запрос игнорируется, а клиенту возвращается сообщение об ошибке.

В простейшем случае при поступлении HTTP-запроса веб-сервер должен считать содержимое запрашиваемого файла с жесткого диска, упаковать его содержимое в HTTP-ответ и отправить клиенту. В случае если требуемый файл не найден на жестком диске, то веб-сервер сгенерирует ошибку с указанием статусного кода 404 и отправит это сообщение клиенту.

Такой вариант работы веб-сервера принято называть статическими сайтами. В этом случае на стороне сервера не запускается никакой программный код, кроме программного кода самого веб-сервера. Однако подобные сценарии работы все чаще оказываются непригодными, а им на смену приходят полноценные веб-приложения. Отличие таких приложений состоит в том, что HTML-документы и другие ресурсы не хранятся на сервере в виде неизменяемых данных. Вместо этого, на сервере хранится программный код, который способен сгенерировать эти данные в момент обработки запроса. Разумеется, некоторые ресурсы (такие как файлы каскадных стилей, изображения и т.д.) могут храниться как статическое содержимое, но основные страницы HTML генерируют в процессе обработки. В таком случае веб-сервер при обработке запроса HTTP должен обращаться к программному коду, который должен сгенерировать содержимое.

**Классификация Web-серверов**

* **1. Серверы (сайты) управления трафиком (Навигационные сайты)**

Основная задача: перенаправление потребителей конечным серверам (напр.: поисковые системы, каталоги)

* **2. Конечные серверы.**

Основное назначение: получение, как можно большего числа повторных посещений. Предоставляют пользователям разнообразные ресурсы, ради которых пользователи посещают Интернет .

Конечные серверы (сайты), предоставляющие ресурсы из области деловой информации, делят на: корпоративные сайты, информационные сайты, торговые площадки.

**Web-сервера:**

* Apache —свободный веб-сервер, наиболее часто используемый в UNIX-подобных операционных системах.
* nginx — свободный веб-сервер, разрабатываемый Игорем Сысоевым с 2002 года и пользующийся большой популярностью на крупных сайтах
* lighttpd — свободный веб-сервер.
* Google Web Server — веб-сервер, основанный на Apache и доработанный компанией Google.
* Resin — свободный веб-сервер приложений.
* Cherokee — свободный веб-сервер, управляемый только через web-интерфейс.
* Rootage — веб-сервер, написанный на java.
* THTTPD — простой, маленький, быстрый и безопасный веб-сервер.

**Apache**

Apache HTTP-сервер ( *сокращение от англ. a patchy server* ) — свободный веб-сервер.

Apache является кроссплатформенным ПО, поддерживает операционные системы Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS.

Основными достоинствами Apache считаются надёжность и гибкость конфигурации. Он позволяет подключать внешние модули для предоставления данных, использовать СУБД для аутентификации пользователей, модифицировать сообщения об ошибках и т. д. Поддерживает IPv6.

Архитектура

Ядро Apache включает в себя основные функциональные возможности, такие как обработка конфигурационных файлов, протокол HTTP и система загрузки модулей. Ядро (в отличие от модулей) полностью разрабатывается Apache Software Foundation, без участия сторонних программистов.

Теоретически, ядро apache может функционировать в чистом виде, без использования модулей. Однако, функциональность такого решения крайне ограничена.

Ядро Apache полностью написано на языке программирования C.

Система конфигурации

Система конфигурации Apache основана на текстовых конфигурационных файлах. Имеет три условных уровня конфигурации:

* Конфигурация сервера (httpd.conf).
* Конфигурация виртуального хоста (httpd.conf c версии 2.2 extra/httpd-vhosts.conf).
* Конфигурация уровня директории (.htaccess).

Имеет собственный язык конфигурационных файлов, основанный на блоках директив. Практически все параметры ядра могут быть изменены через конфигурационные файлы, вплоть до управления MPM. Большая часть модулей имеет собственные параметры.

Часть модулей использует в своей работе конфигурационные файлы операционной системы (например /etc/passwd и /etc/hosts).

Помимо этого, параметры могут быть заданы через ключи командной строки.

Система модулей

Apache HTTP Server поддерживает модульность. Существует более 500 модулей[7], выполняющих различные функции. Часть из них разрабатывается командой Apache Software Foundation, но основное количество — отдельными open source-разработчиками.

Модули могут быть как включены в состав сервера в момент компиляции, так и загружены динамически, через директивы конфигурационного файла.

Механизм виртуальных хостов. Apache имеет встроенный механизм виртуальных хостов. Он позволяет полноценно обслуживать на одном IP-адресе множество сайтов (доменных имён), отображая для каждого из них собственное содержимое.

Для каждого виртуального хоста можно указать собственные настройки ядра и модулей, ограничить доступ ко всему сайту или отдельным файлам. Некоторые MPM, например Apache-ITK позволяют запускать процесс httpd для каждого виртуального хоста с отдельными идентификаторами uid и guid.

Безопасность

* Ограничение доступа к определённым директориям или файлам.
* Механизм авторизации пользователей для доступа к директории по методу HTTP-Авторизации (mod\_auth\_basic) и digest-авторизации (mod\_auth\_digest).
* Ограничение доступа к определённым директориям или всему серверу, основанное на IP-адресах пользователей.
* Запрет доступа к определённым типам файлов для всех или части пользователей, например запрет доступа к конфигурационным файлам и файлам баз данных.
* Существуют модули, реализующие авторизацию через СУБД или PAM.

**IIS**

IIS ( *Internet Information Services, до версии 5.1 — Internet Information Server* ) — проприетарный набор серверов для нескольких служб Интернета от компании Майкрософт. IIS распространяется с операционными системами семейства Windows NT. Основным компонентом IIS является веб-сервер, который позволяет размещать в Интернете сайты. IIS поддерживает протоколы HTTP, HTTPS, FTP, POP3, SMTP, NNTP.

Служба WWW в составе IIS

* Основным компонентом IIS является веб-сервер — служба WWW (называемая также W3SVC), которая предоставляет клиентам доступ к сайтам по протоколам HTTP и, если произведена настройка, HTTPS.
* Один сервер IIS может обслуживать несколько сайтов (IIS 6.0 и выше). Каждый сайт имеет следующие атрибуты:
* IP-адрес сайта;
* TCP-порт, на котором служба WWW ожидает подключений к данному сайту;
* Заголовок узла (Host header name) — значение заголовка Host запроса HTTP, указывающее обычно DNS-имя сайта.

Архитектура службы WWW

В IIS 6.0, доступном в составе систем Windows Server 2003, служба WWW претерпела серьёзные изменения. Был добавлен новый режим обработки запросов, называемый режимом изоляции рабочих процессов (англ. worker process isolation mode).

Безопасность в службе WWW

Веб-сервер IIS предоставляет несколько способов разграничения доступа к сайтам и веб-приложениям. Служба WWW в составе IIS отличается от других веб-серверов тем, что функции обеспечения безопасности в ней тесно интегрированы с системой Windows NT, на основе которой она работает.

Служба WWW поддерживает три основных метода аутентификации, то есть определения личности пользователя по имени и паролю:

* Базовая аутентификация (basic authentication) — имя и пароль передаются по сети открытым текстом.
* Сжатая аутентификация (digest authentication) — пароль обрабатывается хеш-функцией перед отправкой по сети, что делает невозможным его прочтение в случае перехвата злоумышленником.
* Встроенная аутентификация Windows (integrated Windows authentication) — выполняется попытка входа на сервер с теми же учётными данными, под которыми работает браузер пользователя.

Реализация веб-приложений для IIS

Веб-сервер IIS поддерживает несколько различных технологий создания веб-приложений:

* ASP.NET — разработанная Microsoft технология;
* ASP — предшествовавшая ASP.NET технология создания динамических веб-страниц на основе сценариев. Входит в поставку IIS начиная с версии 3.0.
* CGI — стандартная межплатформенная низкоуровневая технология создания динамических веб-страниц.
* FastCGI — клиент-серверный протокол взаимодействия веб-сервера и приложения.
* ISAPI — низкоуровневая технология, аналогичная интерфейсу модулей Apache, предоставляющая полный доступ ко всем возможностям IIS,
* SSI — включение в одни страницы текста из других страниц.

**Nginx**

Nginx ( *англ. engine x* ) ( *по-русски произносится как энджин-экс* ) — веб-сервер и почтовый прокси-сервер, работающий на Unix-подобных операционных системах (тестировалась сборка и работа на FreeBSD, OpenBSD, Linux, Solaris, Mac OS X, AIX). Начиная с версии 0.7.52 появилась бинарная сборка под Microsoft Windows.

Основные функции

HTTP-сервер

* обслуживание статических запросов, индексных файлов, автоматическое создание списка файлов, кеш дескрипторов открытых файлов
* акселерированное проксирование без кэширования, простое распределение нагрузки и отказоустойчивость
* поддержка кеширования при акселерированном проксировании и FastCGI
* акселерированная поддержка FastCGI и memcached серверов, простое распределение нагрузки и отказоустойчивость
* модульность, фильтры, в том числе сжатие (gzip), byte-ranges (докачка), chunked ответы, HTTP-аутентификация, SSI-фильтр
* несколько подзапросов на одной странице, обрабатываемые в SSI-фильтре через прокси или FastCGI, выполняются параллельно
* поддержка SSL
* экспериментальная поддержка встроенного Perl

Архитектура

В nginx рабочие процессы обслуживают одновременно множество соединений, мультиплексируя их вызовами операционной системы select, epoll (Linux) и kqueue (FreeBSD). Рабочие процессы выполняют цикл обработки событий от дескрипторов (см. Событийно-ориентированное программирование).

Nginx содержит модуль географической классификации клиентов по IP-адресу. В его основу входит база данных соответствия IP-адресов географическому региону, представленная в виде Radix tree (сжатое префиксное дерево или сжатый бор) в оперативной памяти. nginx предварительно распределяет первые несколько уровней дерева, таким образом, чтобы они занимали ровно 1 страницу памяти. Это гарантирует, что при поиске IP-адреса для первых нескольких узлов при трансляции адреса всегда найдётся запись в TLB.

**Клиенты веб-сервера**

В качестве клиентов для обращения к веб-серверам могут использоваться различные программы и устройства:

• веб-браузер, работающий на настольном компьютере или переносном устройстве (например, карманном ПК);

• разнообразные программы, самостоятельно обращающиеся к веб-серверам для получения обновлений или другой информации (например, антивирус может периодически запрашивать у определённого веб-сервера обновления своих баз данных);

• мобильный телефон, получающий доступ к ресурсам веб-сервера при помощи протокола WAP;

• другие цифровые устройства или бытовая техника.

**Интерфейсы взаимодействия**

Поскольку программный код веб-приложения обычно упаковывается в отдельные модули и поставляется независимо, то требуются механизмы взаимодействия этих двух частей, т.е. интерфейс взаимодействия. В данном случае под интерфейсом взаимодействия понимается набор правил, по которым веб-сервер и приложение будут взаимодействовать друг с другом. Фактически, схема обработки запроса может выглядеть следующим образом.

Исторически сложилось так, что существует два главных типов интерфейс взаимодействия внешнего приложения и веб-сервера - CGI и ISAPI.

CGI (Common Gateway Interface) – наиболее ранний способ взаимодействия веб-сервера и веб-приложения. Основная идея, которая лежит в основе CGI заключается в том, что при поступлении очередного HTTP-запроса, веб-сервер инициирует создание нового *процесса* и передает ему все необходимые данные HTTP-запроса.

После того, как этот процесс отработает, он завершается, передав при этом результат обратно веб-серверу. Поскольку веб-сервер и приложение – это разные процессы с точки зрения операционной системы, то для обмена информации между ними используются средства межпроцессного взаимодействия (IPC) – зачастую это переменные окружения, именованные каналы и т.д. Основным преимуществом CGI является то, что процесс веб-сервера и приложения изолированы друг от друга и в случае неполадок в веб-приложении, завершится с ошибкой именно процесс приложения, при этом процесс самого веб-сервера будет продолжать функционировать.

С другой стороны, необходимость создания каждый раз нового процесса влечет за собой дополнительные накладные расходы на создание процесса (создание процесса – дорогостоящая операция с точки зрения операционной системы) и передачи данных через границы процессов. Этот факт является серьезным недостатком и оказывает существенное влияние на масштабируемость веб-приложения (возможность обрабатывать большее количество поступающих запросов).

ISAPI (Internet Server API) – альтернативный способ взаимодействия веб-сервера и веб-приложения. В отличии от CGI, при взаимодействии в рамках интерфейса ISAPI, при поступлении очередного запроса, веб-сервер инициирует создание нового *потока* в рамках основного процесса, в котором работает веб-сервер. Поскольку с точки зрения операционной системы создание потока – это менее дорогостоящая операция, чем создание процесса, то такие приложения на практике оказываются более масштабируемыми.

Кроме того, упрощается взаимодействие веб-сервера и веб-приложения, поскольку в этом случае используется единое адресное пространство в рамках операционной системы (поскольку весь код работает в одном и том же процессе). Однако, в случае серьезных неполадок в веб-приложении, которое взаимодействует с веб-сервером в рамках ISAPI, веб-сервер также потенциально подвергается риску быть завершенным.

Поскольку веб-сервер и веб-приложение работают в одном и том же процессе, это действительно так. Поэтому разработчикам программного кода веб-сервера, поддерживающего ISAPI следует уделить этому вопросу особое внимание.

На сегодняшний день наиболее распространенным способом взаимодействия веб-сервера и веб-приложения является интерфейс ISAPI, поскольку обеспечивает наиболее оптимальные показатели по накладным расходам и масштабируемости. Однако, при работе нескольких веб-приложений на одном и том же веб-сервере, в этом случае существует потенциальная опасность влияния одного приложения на другое.

Если говорить о компаниях, размещающих веб-приложения на своих серверах, то может случиться такая ситуация, что на одном и том же веб-сервере одновременно размещаются веб-сайты компаний-конкурентов. В этом случае теоретически одна из компаний может намеренно загрузить код, который будет завершать работу веб-сервера с ошибкой и, таким образом, все веб-сайты размещенные на этом веб-сервере окажутся недоступными. Для того, чтобы избежать подобной ситуации используется совмещенный подход – для каждого приложения может создаваться пул приложения (application pool), который представляет из себя отдельный процесс, в котором функционируют потоки для обработки входящих HTTP-запросов от пользователей. В этом случае, если какое-то из приложений будет содержать код, который завершает работу процесса с ошибкой, то будет завершаться процесс только этого приложения. Более того, каждый пул приложения содержит набор заранее созданных и подготовленных потоков. Это необходимо для того, чтобы не тратить время на создание потока в момент поступления входящего запроса. Такой набор заранее созданных потоков называется пулом потоков. Как правило, веб-сервер следит за каждым пулом приложения и если оно завершает свою работу с ошибкой, то веб-сервер перезапускает его процесс.

**Источники:**

<https://seti.ucoz.ru/index/lekcija_13_servisy_setevykh_operacionnykh_sistem_web_server/0-55>

<https://intuit.ru/studies/courses/4455/712/lecture/21291?page=3>

<https://kopilkaurokov.ru/prochee/presentacii/liektsiia_vieb_sierviery_klassifikatsiia_iis_apache_ngnx>