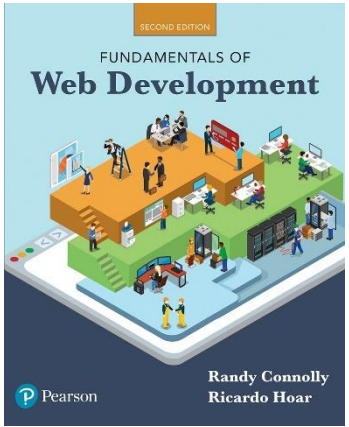
CSE 686 Internet Programming

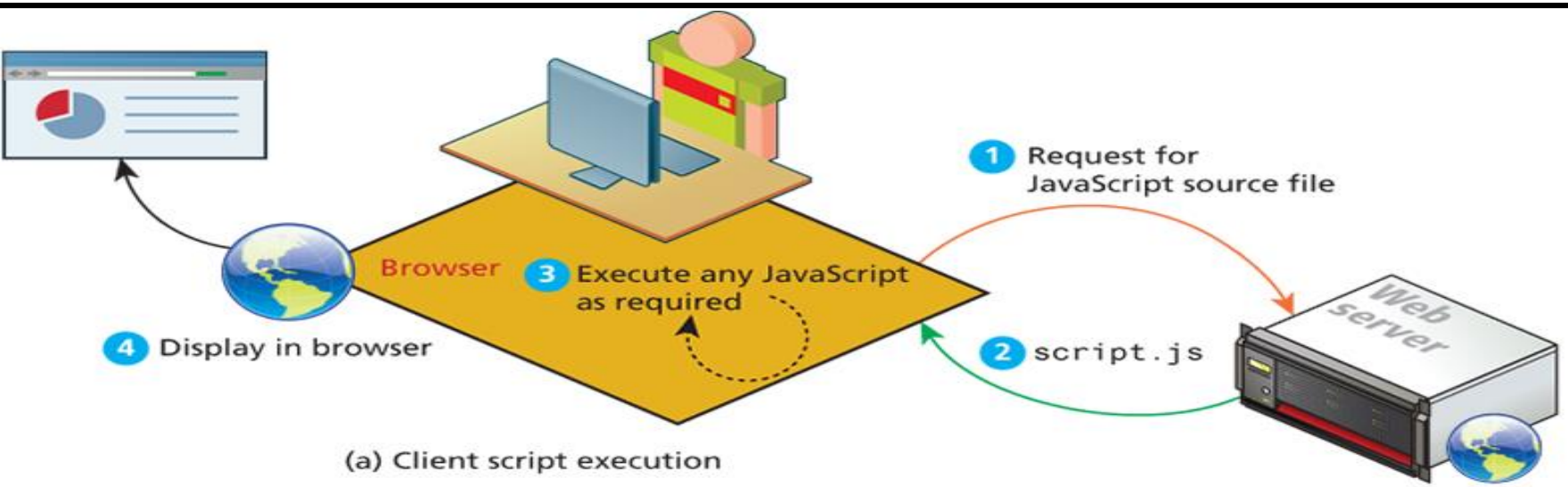
Week 13: Server-Side Scripting, Part 1: XAMPP & Apache

Edmund Yu, PhD
Associate Teaching Professor
esyu@syr.edu

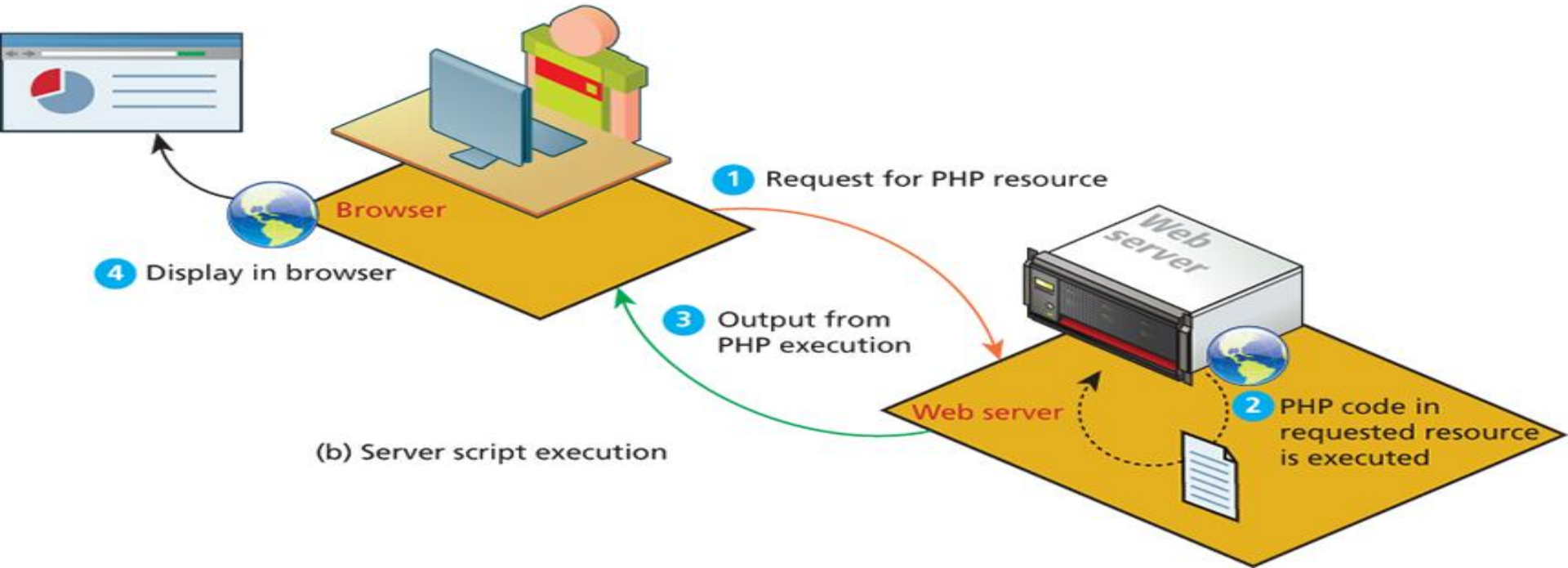
April 9 & 11, 2018



Client-Side Scripting vs Server-Side

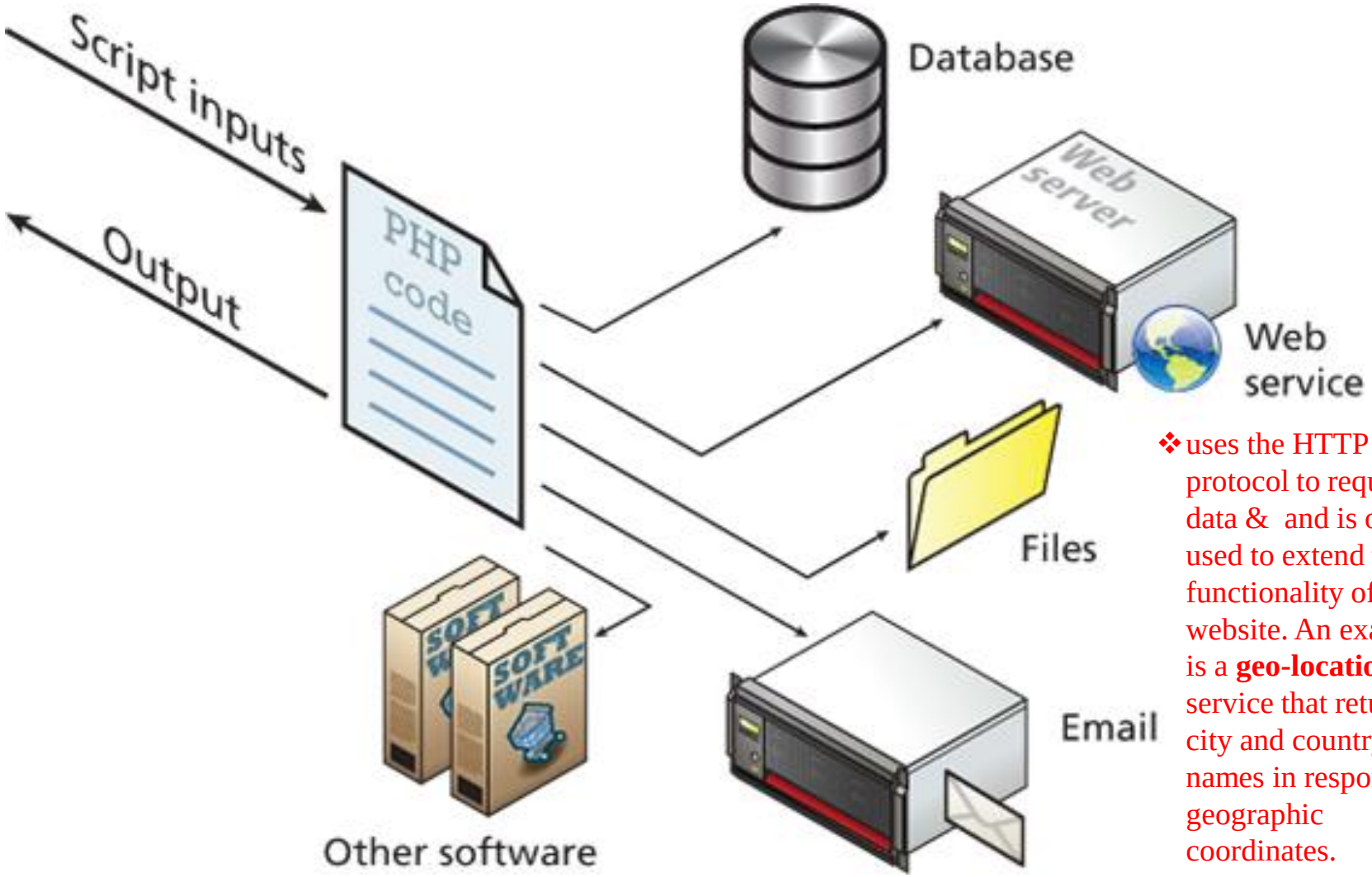


(a) Client script execution



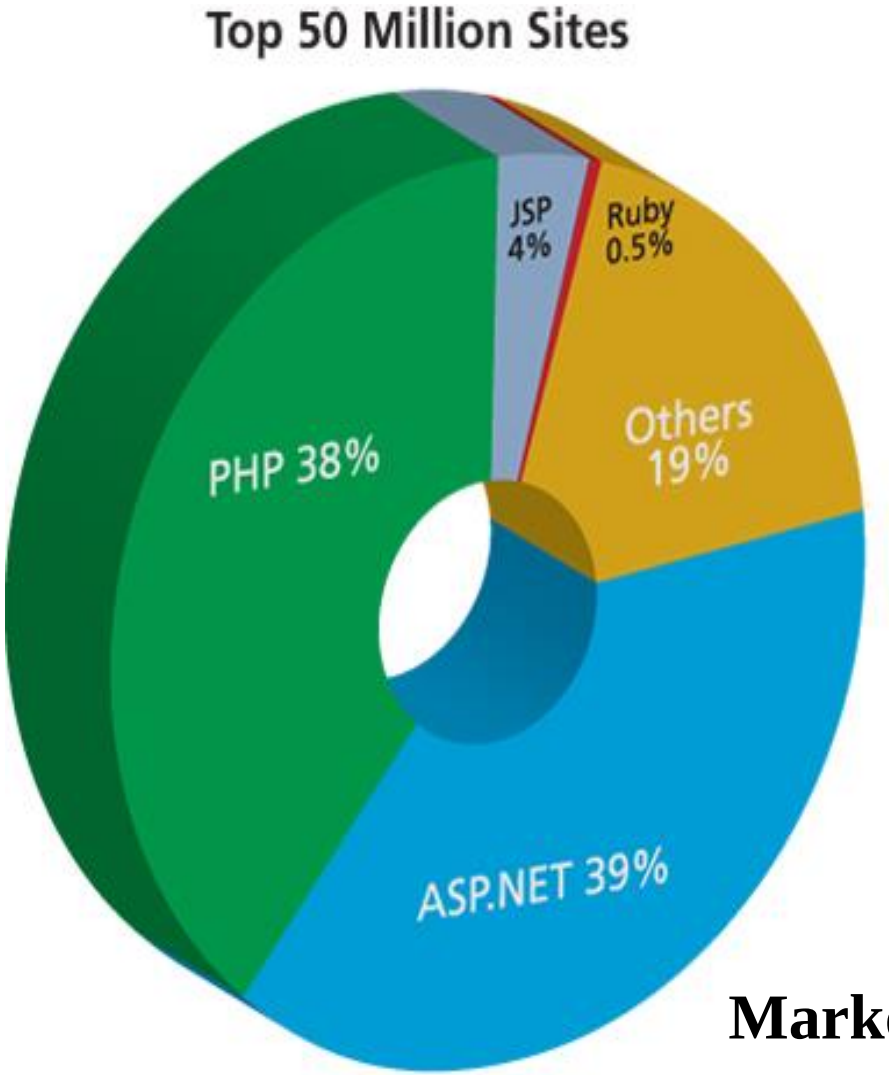
(b) Server script execution

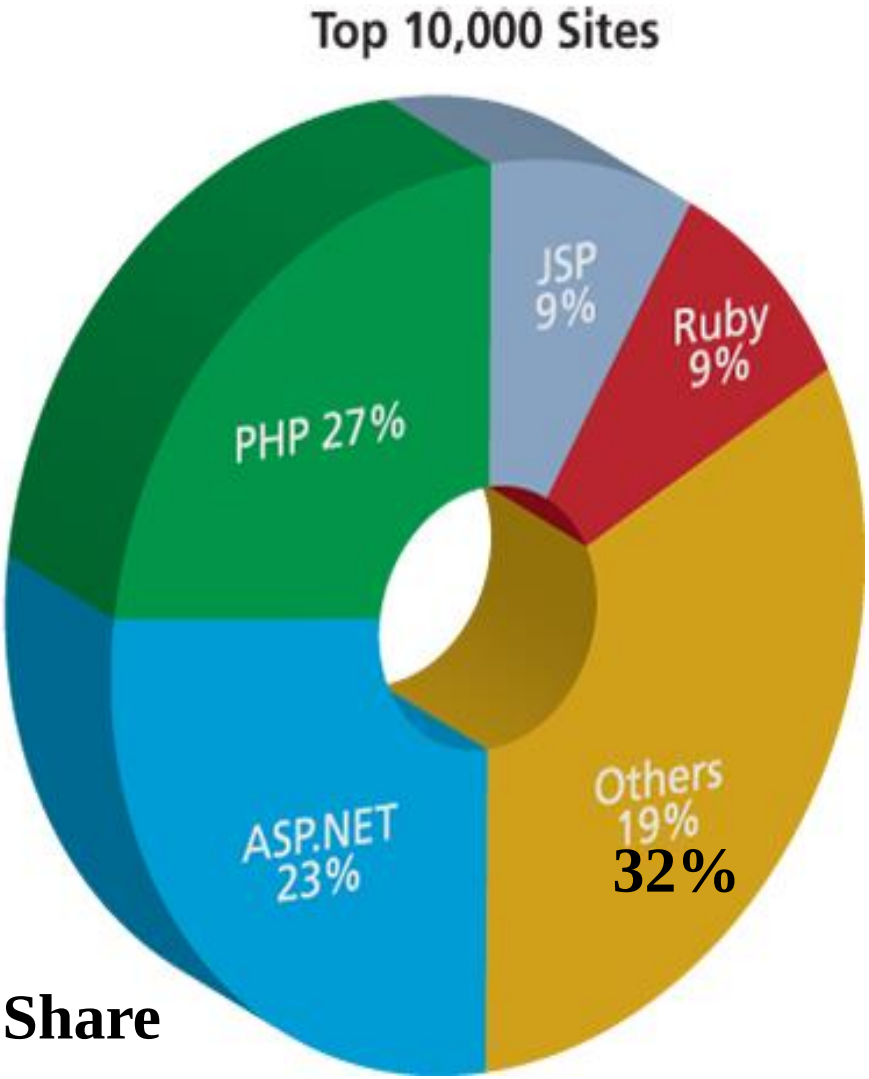
Server-Side Scripting



- uses the HTTP protocol to request data & and is often used to extend the functionality of a website. An example is a **geo-location** service that returns city and country names in response to geographic coordinates.

Server-Side Technologies





Market Share

Server-Side Technologies

ASP (Active Server Pages)

- Microsoft's first server-side technology.
- ASP code (using the VBScript programming language) can be embedded within the HTML.
- Although it supported classes and some object-oriented features, most developers did not make use of these features.
- ASP programming code is interpreted at run time; hence, it can be slow in comparison to other technologies.

Server-Side Technologies

ASP.NET (replaces ASP)

- ASP.NET is part of Microsoft's .NET Framework and can use any .NET programming language (though **C#** is the most commonly used).
- ASP.NET uses an explicitly object-oriented approach that typically takes longer to learn than ASP or PHP, and is often used in larger corporate web application systems.
- It also uses special markup called **web server controls** that encapsulate common web functionality such as database-driven lists, form validation, and user registration wizards.
- A recent extension called **ASP.NET MVC** makes use of the Model-View-Controller design pattern.
- ASP.NET pages are compiled into an intermediary file format called **MSIL** that is analogous to Java's byte-code.
- ASP.NET then uses a JIT (Just-In-Time) compiler to compile the MSIL into machine executable code so its performance can be excellent.

The MVC Pattern

Name	MVC (Model-View-Controller)
Description	Separates presentation and interaction from the system data. The system is structured into three logical components that interact with each other. The Model component manages the system data and associated operations on that data. The View component defines and manages how the data is presented to the user. The Controller component manages user interaction (e.g., key presses, mouse clicks, etc.) and passes these interactions to the View and the Model. See Figure 6.5.
Example	Figure 6.6 shows the architecture of a web-based application system organized using the MVC pattern.
When used	Used when there are multiple ways to view and interact with data. Also used when the future requirements for interaction and presentation of data are unknown.
Advantages	Allows the data to change independently of its representation and vice versa. Supports presentation of the same data in different ways, with changes made in one representation shown in all of them.
Disadvantages	May involve additional code and code complexity when the data model and interactions are simple.

The Organization of the MVC

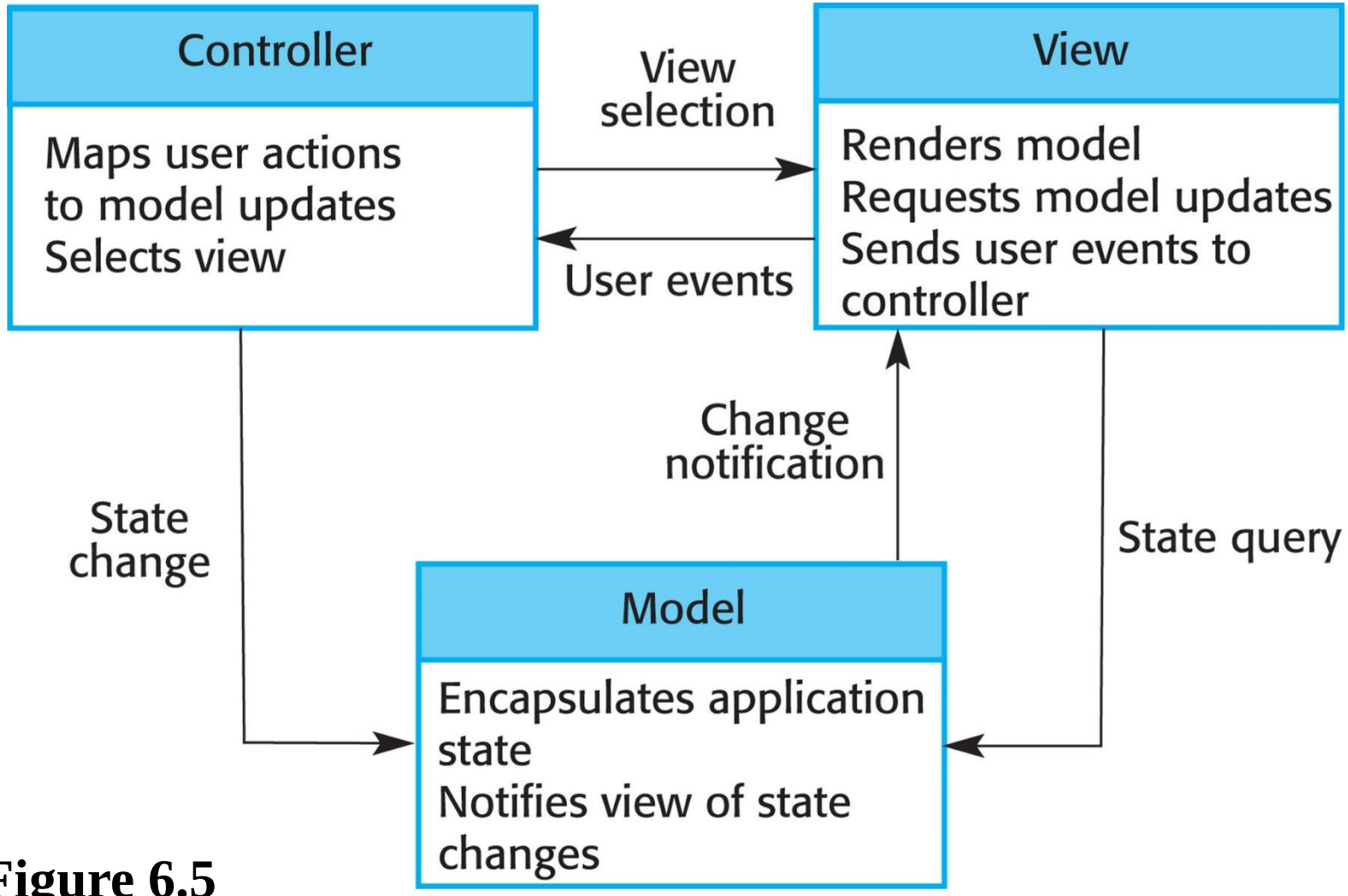


Figure 6.5

Web Application Architecture

Web application architecture using the MVC pattern

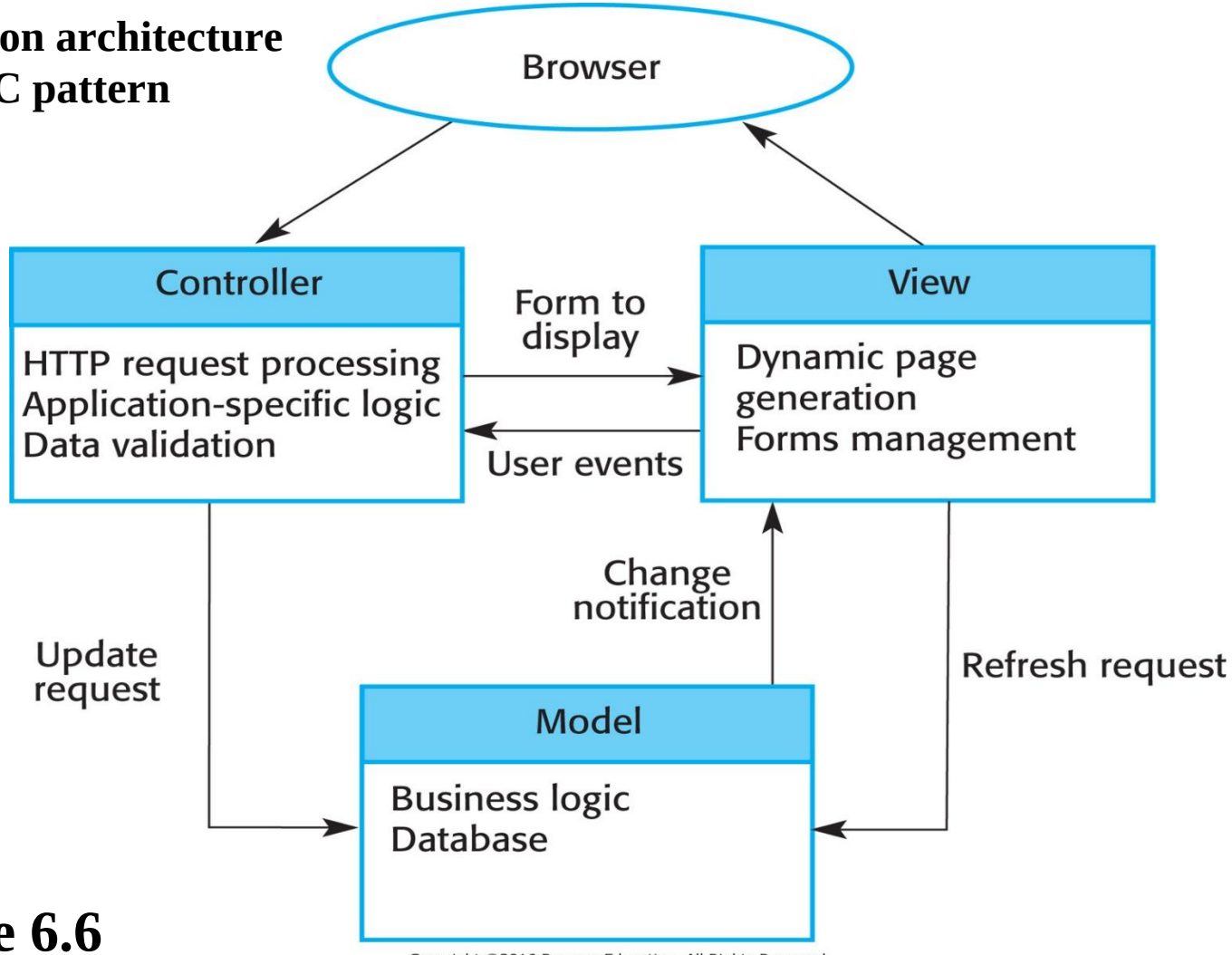


Figure 6.6

Server-Side Technologies

PHP

- PHP is a dynamically typed language (like ASP) that can be embedded directly within the HTML, and supports most common object-oriented features such as classes and inheritance.
- By default, PHP pages are compiled into an intermediary representation called **opcodes** that are analogous to Java's byte-code or the .NET Framework's MSIL.
- Originally, PHP stood for Personal Home Pages, although it now is a recursive acronym that means PHP: Hypertext Processor.

Server-Side Technologies

JSP (Java Server Pages)

- JSP uses Java as its programming language and like ASP.NET it uses an explicit object-oriented approach and is used in large enterprise web systems and is integrated into the J2EE environment.
- Since JSP uses the Java Runtime Engine, it also uses a JIT compiler for fast execution time and is cross-platform.
- While JSP's usage in the web as a whole is small, it has a substantial market share in the intranet environment, and is used on a number of very large sites.

Server-Side Technologies

Ruby on Rails

- This is a web development framework that uses the Ruby programming language.
- Like ASP.NET and JSP, Ruby on Rails emphasizes the use of common software development approaches, in particular the **MVC** design pattern.
- It integrates features such as templates and engines that aim to reduce the amount of development work required in the creation of a new site.

Server-Side Technologies

Perl

- Until the development and popularization of ASP, PHP, and JSP, Perl was the language typically used for early server-side web development.
- As a language, it excels in the manipulation of text.
- It was commonly used in conjunction with the Common Gateway Interface (CGI), an early standard API for communication between applications and web server software. (See next 3 slides, again)

Figure 8-1 depicts a kind of resource gateway. Here, the Joe's Hardware server is acting as a gateway to database content—note that the client is simply asking for a resource through HTTP, and the Joe's Hardware server is interfacing with a gateway to get at the resource.

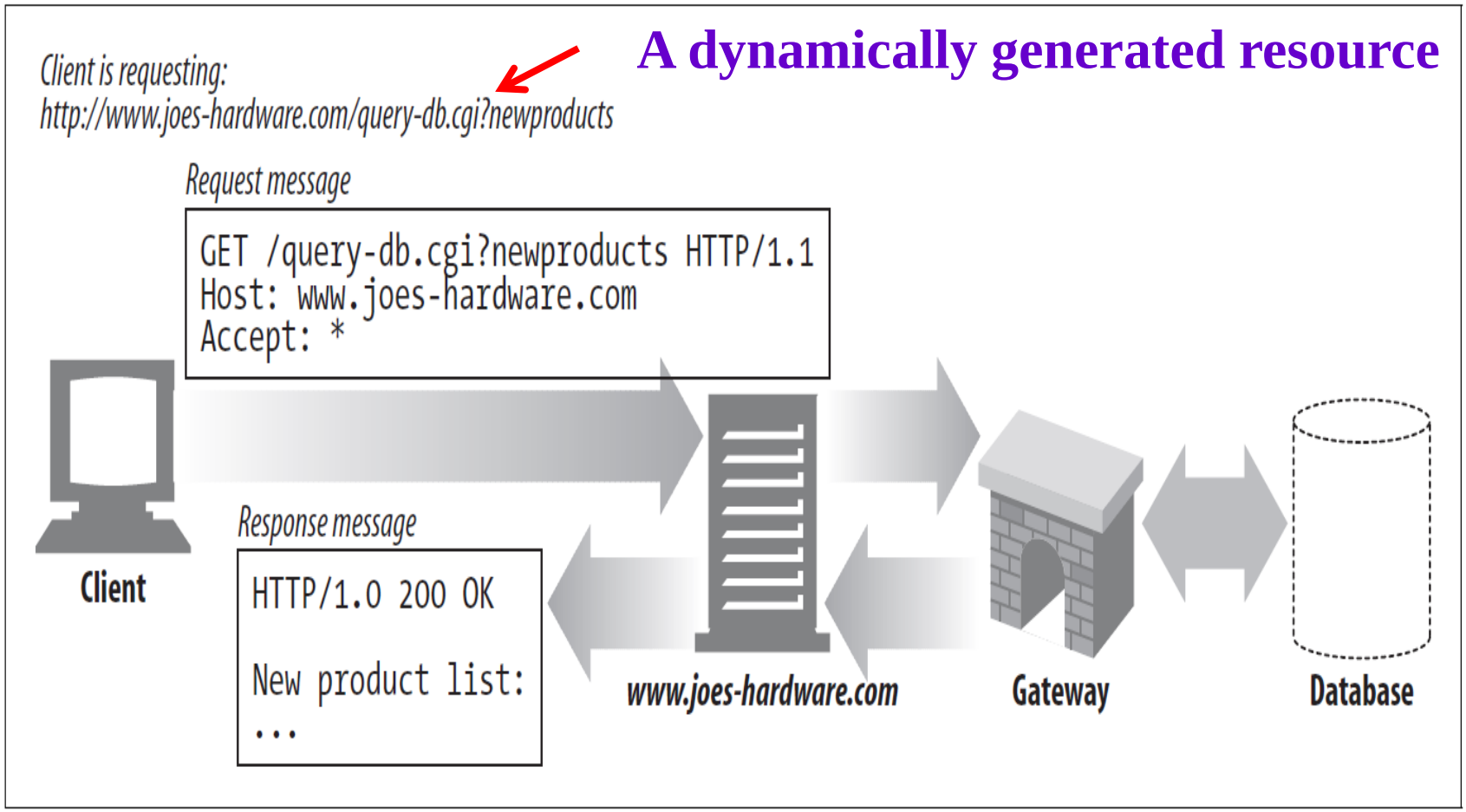


Figure 8-1. Gateway magic

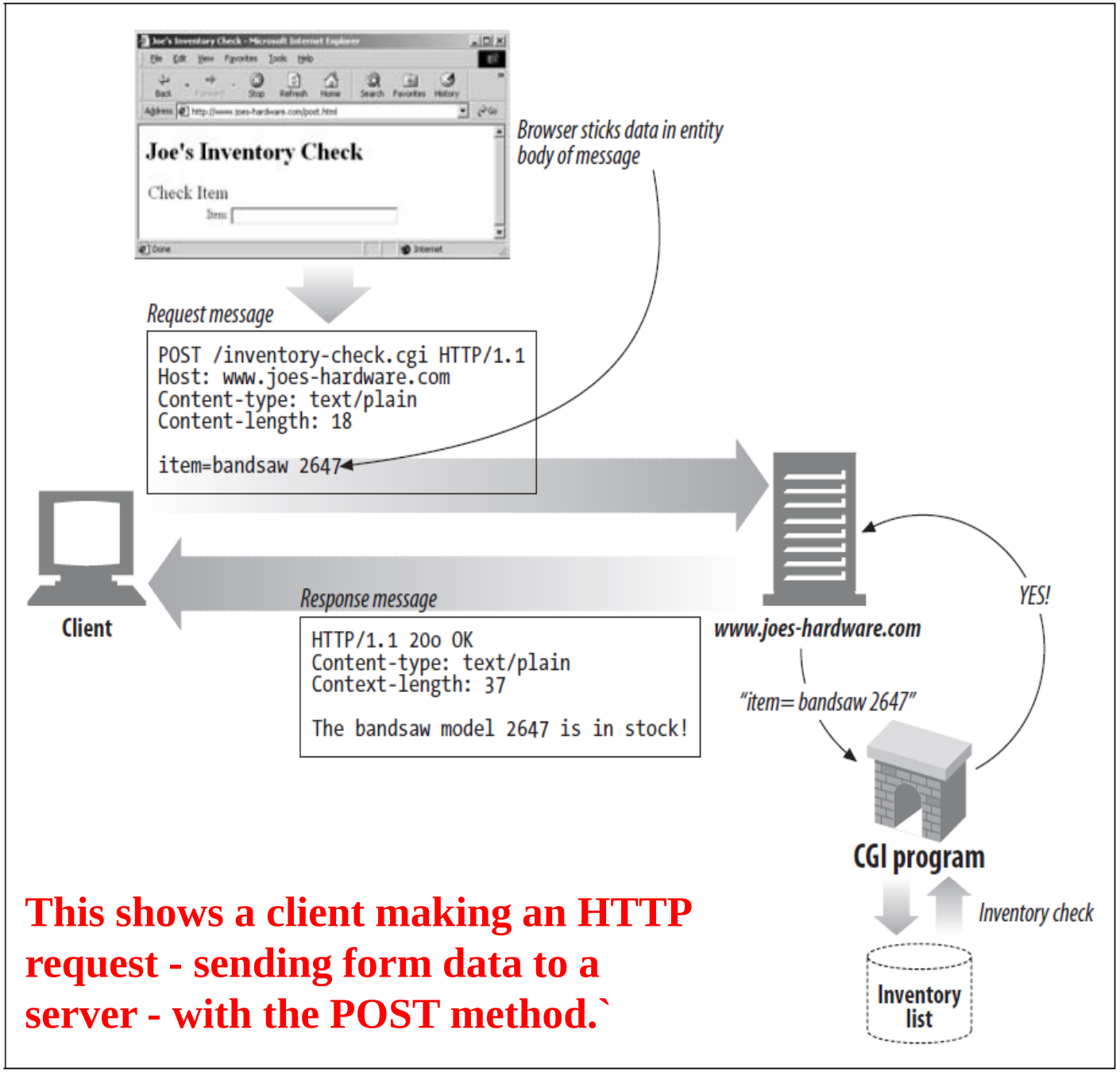


Figure 3-10. POST example

CGI (Common Gateway Interface)

- CGI's contribution was to standardize the relationship/interface between the web server and a program (a child process of the Web server program) that is responsible for generating the dynamic content.
  - The CGI specification defined the data that would get passed to the program, and the formats for these data.
  - It also defined how the web server would take the output from the program, add a **HTTP header** and return these data as the response sent back to a client.
- The programs responsible for handling the form and generate the dynamic content are known as **CGI scripts,** which are usually written in a **scripting language**.

Server-Side Technologies

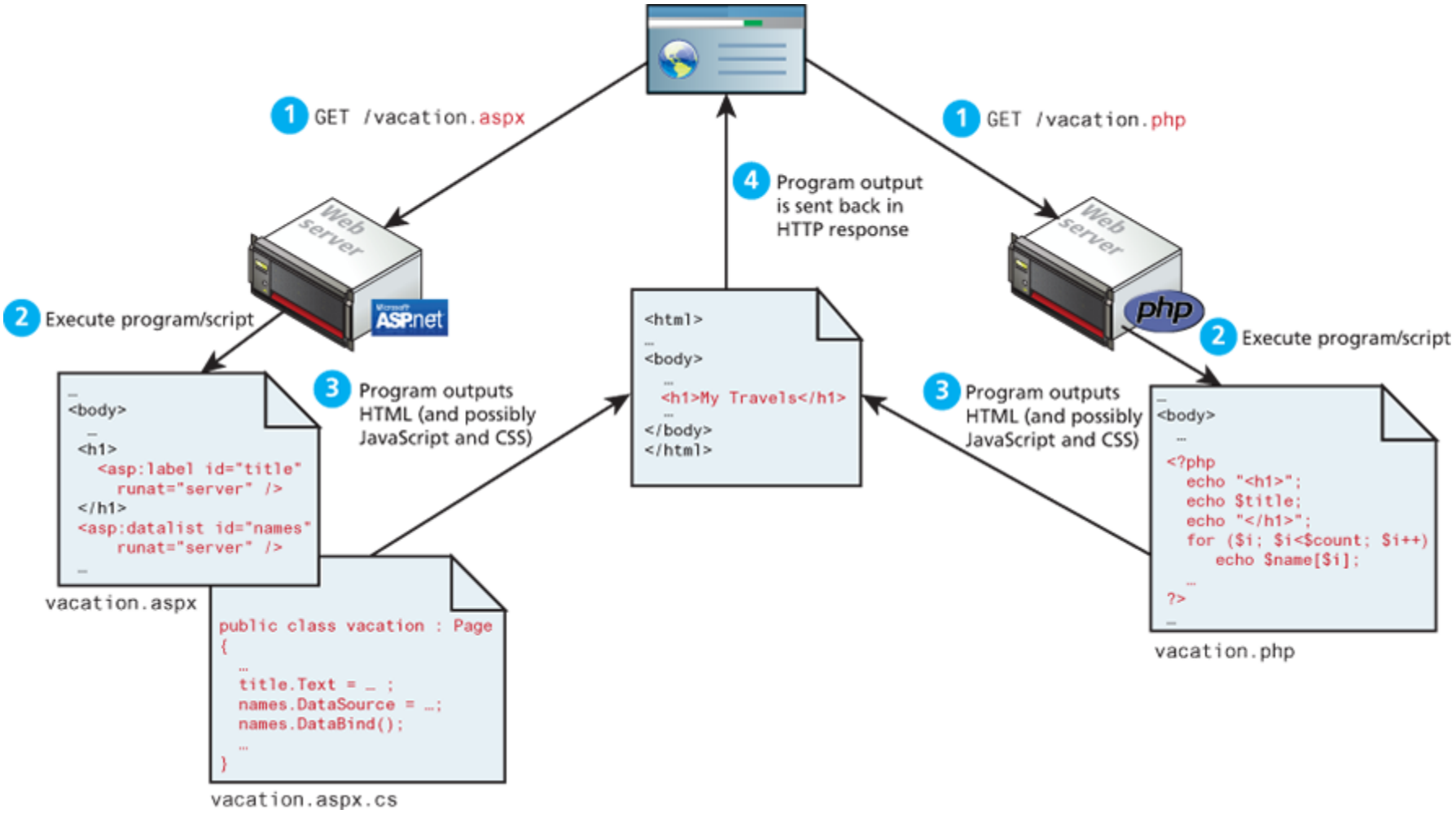
Python

- This terse, object-oriented programming language has many uses, including being used to create web applications.
- It is also used in a variety of web development frameworks such as **Django** and Pyramid.

Node.js

- This is a more recent **server environment** that uses JavaScript on the server side, thus allowing developers already familiar with JavaScript to use just a single language for both client-side and server-side development.
- Unlike the other development technologies, node.js is also its own web server software, thus eliminating the need for **Apache**, IIS, or some other web server software.

PHP vs ASP.NET



The Web Server Revisited

- As you learned at the beginning of this course, in the client-server model the server is responsible for answering all client requests.
- No matter how static or simple the website is, there must be a web server somewhere configured to answer requests for that domain.
- Once a web server is configured and its IP address associated through a DNS server, it can then start listening for and answering HTTP requests.
- In the very simplest case the server is hosting static HTML files, and in response to a request sends the content of the file back to the requester.

The Web Server Revisited

- However, a web server has many responsibilities beyond responding to requests for HTML files, include handling:
  - Managing HTTP connections
  - Responding to requests for static and **dynamic** resources
  - Managing permissions and access for certain resources
  - Encrypting and compressing data
  - Managing multiple domains and URLs
  - Managing database connections, cookies, and sessions
  - Uploading and managing files

The Web Server Revisited

- Our Textbook # 3 uses the LAMP software stack, which refers to the **L**inux operating system, the **A**pache web server, the **M**ySQL DBMS, and the **P**HP scripting language, while I use XAMPP in my Web server related courses.

- Since the Apache web server is an essential part of the web development pipeline, one should have some insight into how it works and how it interacts with PHP.

Беларуская
Български
Català
Česky
Deutsch
Español
فارسی
Français
한국어
Bahasa Indonesia
Italiano
עברית
Latviešu
Magyar
Nederlands
日本語
Polski
Português
Русский
Simple English
Српски / srpski
Українська
中文

- Comparison of lightweight web servers
- Comparison of web server software
- Embedded HTTP server
- Web server

A

- Adobe Atmosphere
- Adobe JRun

C

- C10k problem
- CGIProxy
- Cherokee (web server)
- CL-HTTP

D

- DAMP (software bundle)

E

- Eagle (Mainframe Application Server)
- Elemenope

H

- Helicon Ape

I

- IBM HTTP Server
- IBM Lotus Domino
- IBM OD390
- IBM WebSphere Application Server
- IBM WebSphere Application Server Community Edition
- In-kernel web server
- Internet Information Services

J

- Jaminid

K

- Kerio WebSTAR

L

- LAMP (software bundle)
- Lotus Foundations

M

- MacHTTP
- MAMP
- Microsoft Personal Web Server
- Mod deflate
- Mod gzip
- Mod proxy
- Mod ssl
- Mod wsgi
- Mongoose (web server)

N

- NCSA HTTPd
- NetDynamics Application Server
- NetPresenz
- Netscape Application Server
- Netscape Enterprise Server

O

- Oracle Application Server
- Orion Application Server
- Oracle HTTP Server

O cont.

- Oracle iPlanet Web Server

P

- Personal web server
- PHP-FPM
- PoorMan

R

- Run BASIC

S

- SAP NetWeaver Application Server
- Server-side redirect
- Server2Go
- Skill evaluation lab

W

- WAMP
- Oracle WebLogic Server
- WebLogic (company)
- WebSitePro
- WIMP (software bundle)
- WISA software bundle
- Wt (web toolkit)

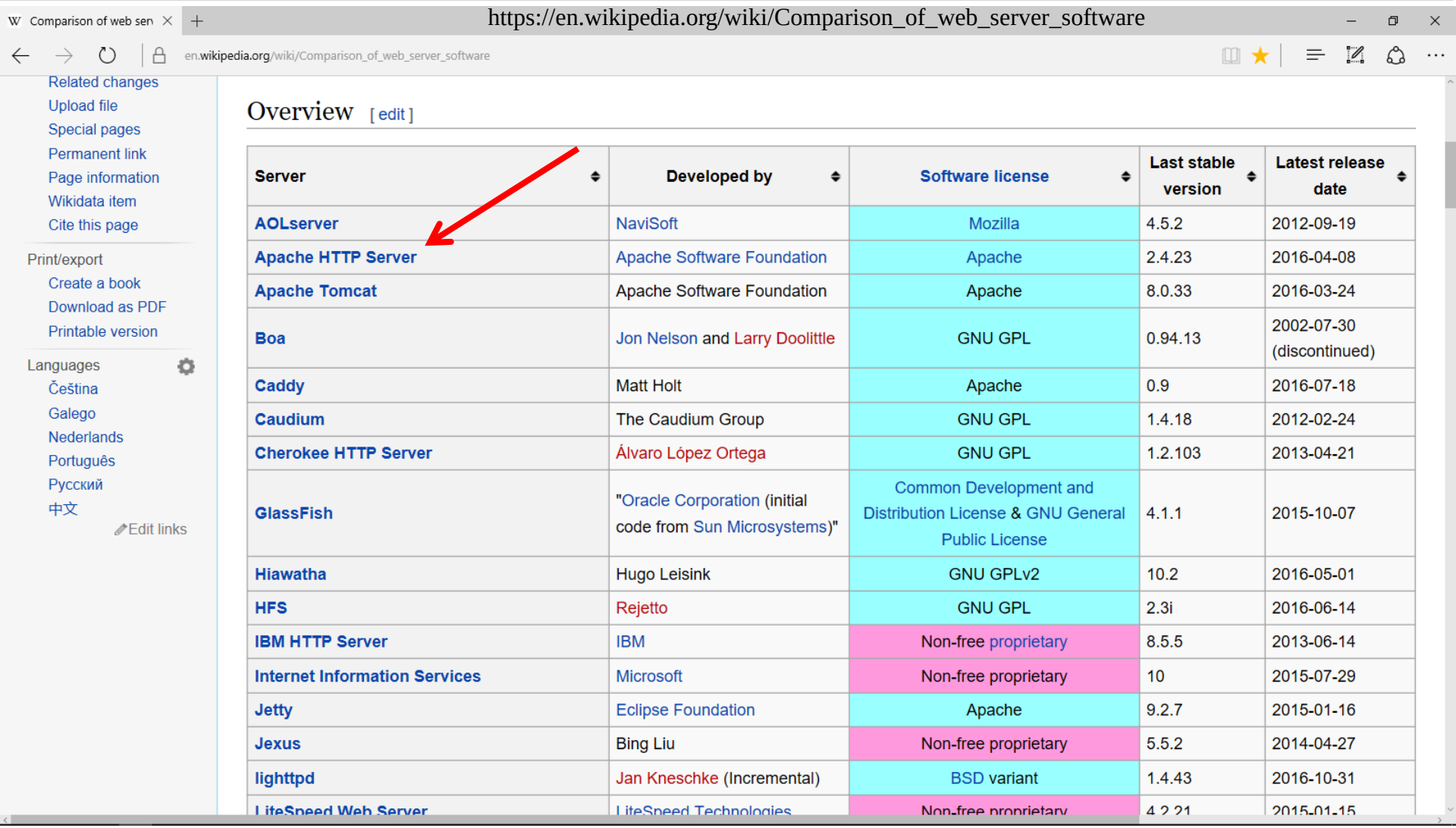
X

- XAMPP

Z

- Zend Server
- Zeus Web Server

Web Server Software

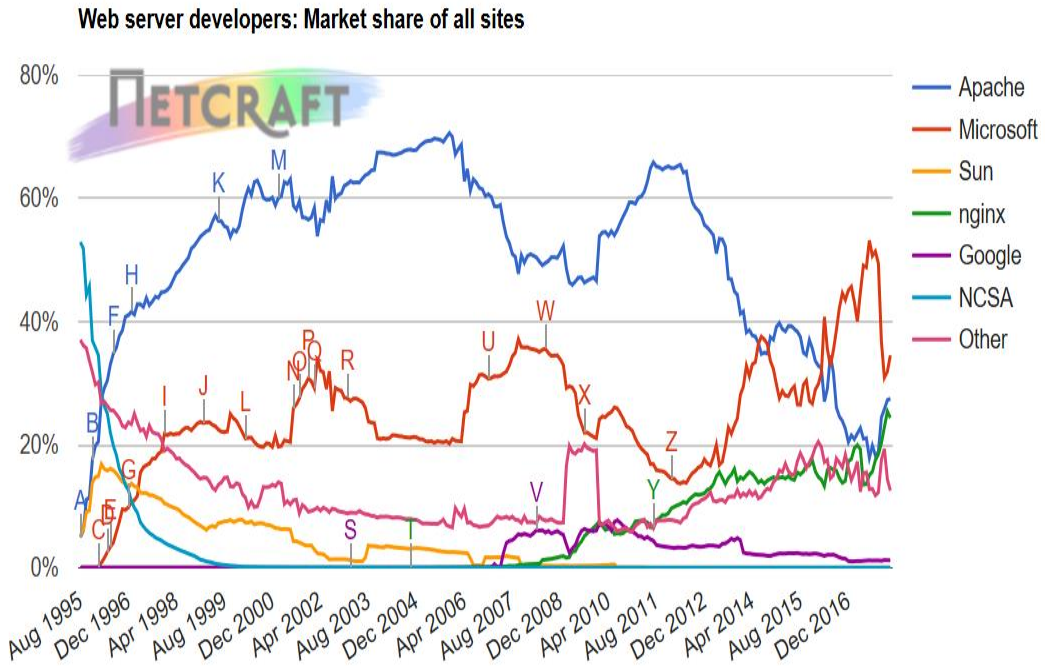


Overview [edit]

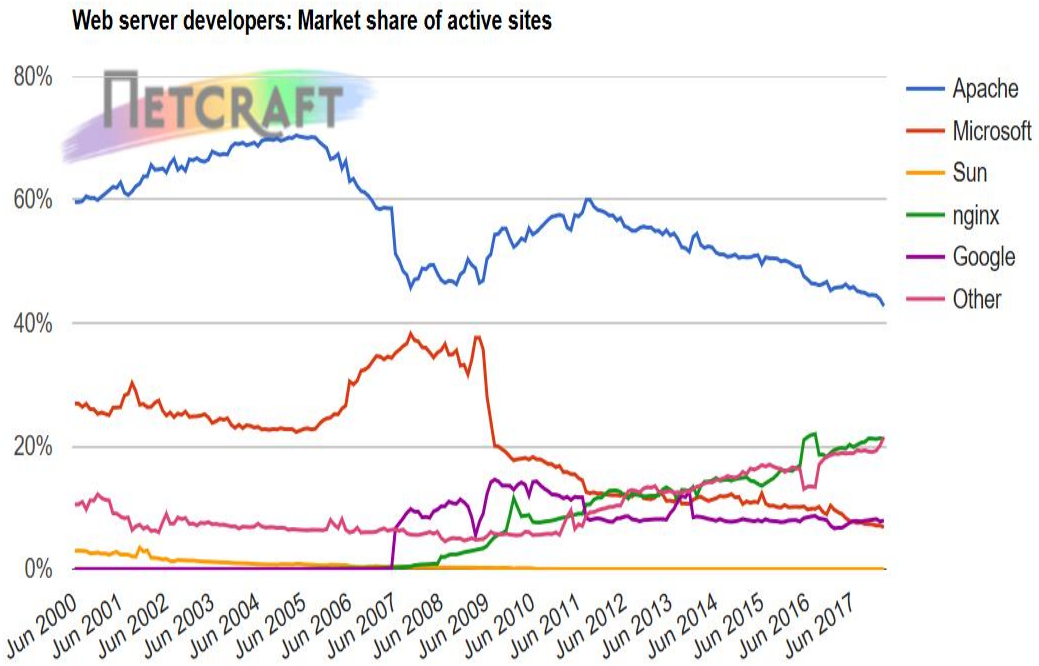
Server	Developed by	Software license	Last stable version	Latest release date
AOLserver	NaviSoft	Mozilla	4.5.2	2012-09-19
Apache HTTP Server	Apache Software Foundation	Apache	2.4.23	2016-04-08
Apache Tomcat	Apache Software Foundation	Apache	8.0.33	2016-03-24
Boa	Jon Nelson and Larry Doolittle	GNU GPL	0.94.13	2002-07-30 (discontinued)
Caddy	Matt Holt	Apache	0.9	2016-07-18
Caudium	The Caudium Group	GNU GPL	1.4.18	2012-02-24
Cherokee HTTP Server	Álvaro López Ortega	GNU GPL	1.2.103	2013-04-21
GlassFish	"Oracle Corporation (initial code from Sun Microsystems)"	Common Development and Distribution License & GNU General Public License	4.1.1	2015-10-07
Hiawatha	Hugo Leisink	GNU GPLv2	10.2	2016-05-01
HFS	Rejetto	GNU GPL	2.3i	2016-06-14
IBM HTTP Server	IBM	Non-free proprietary	8.5.5	2013-06-14
Internet Information Services	Microsoft	Non-free proprietary	10	2015-07-29
Jetty	Eclipse Foundation	Apache	9.2.7	2015-01-16
Jexus	Bing Liu	Non-free proprietary	5.5.2	2014-04-27
lighttpd	Jan Kneschke (Incremental)	BSD variant	1.4.43	2016-10-31
LiteSpeed Web Server	LiteSpeed Technologies	Non-free proprietary	4.2.21	2015-01-15

- There are thousands of different web server programs available

2015
2014
2013
2012
2011
2010
2009
2008
2007
2006
2005
2004
2003
2002

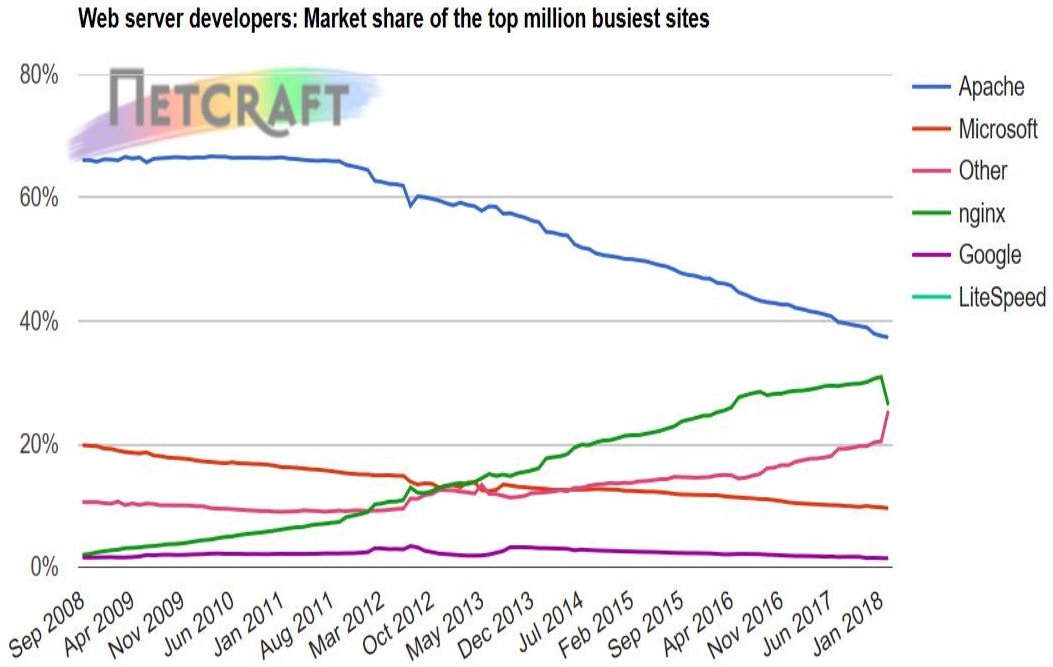


Developer	January 2018	Percent	February 2018	Percent	Change
Microsoft	575,026,648	31.85%	634,359,419	34.50%	2.65
Apache	491,259,918	27.21%	504,701,560	27.45%	0.24
nginx	458,386,423	25.39%	447,224,456	24.32%	-1.07
Google	21,657,796	1.20%	22,022,633	1.20%	-0.00

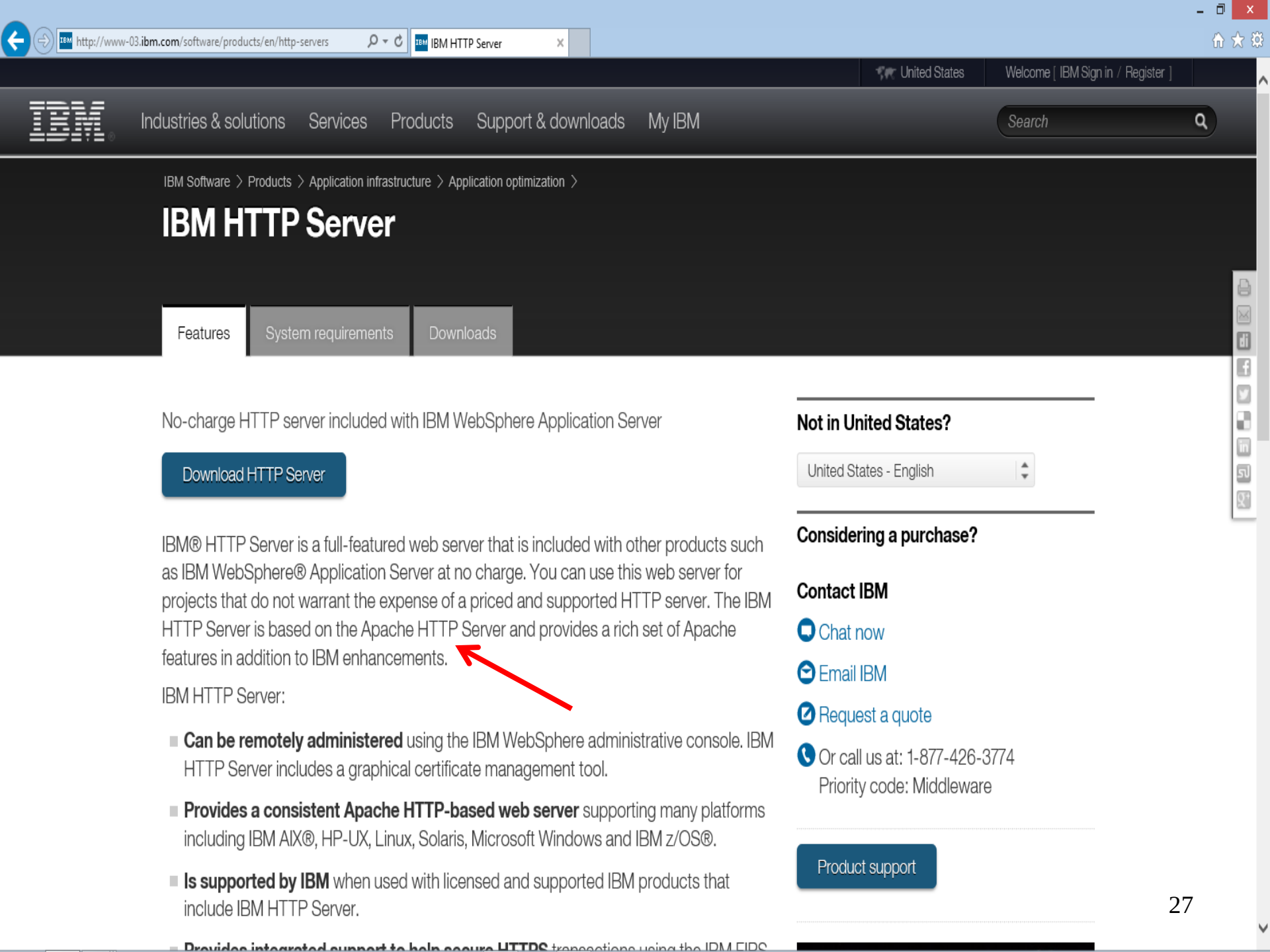


Developer	January 2018	Percent	February 2018	Percent	Change
Apache	75,212,932	43.82%	77,324,937	42.72%	-1.10
nginx	36,504,577	21.27%	38,239,250	21.13%	-0.14
Google	13,270,339	7.73%	14,288,736	7.89%	0.16
Microsoft	12,147,278	7.08%	12,301,688	6.80%	-0.28

For more information see Active Sites



Developer	January 2018	Percent	February 2018	Percent	Change
Apache	370,293	37.03%	367,859	36.79%	-0.24
nginx	304,629	30.46%	259,178	25.92%	-4.55
Microsoft	95,676	9.57%	94,063	9.41%	-0.16
LiteSpeed	13,924	1.39%	14,392	1.44%	0.05


http://www-03.ibm.com/software/products/en/http-servers
IBM HTTP Server
United States
Welcome [IBM Sign in / Register]
IBM
Industries & solutions
Services
Products
Support & downloads
My IBM
Search
IBM Software > Products > Application infrastructure > Application optimization >
IBM HTTP Server
Features
System requirements
Downloads
No-charge HTTP server included with IBM WebSphere Application Server
Download HTTP Server
IBM® HTTP Server is a full-featured web server that is included with other products such as IBM WebSphere® Application Server at no charge. You can use this web server for projects that do not warrant the expense of a priced and supported HTTP server. The IBM HTTP Server is based on the Apache HTTP Server and provides a rich set of Apache features in addition to IBM enhancements.
IBM HTTP Server:
Can be remotely administered using the IBM WebSphere administrative console. IBM HTTP Server includes a graphical certificate management tool.
Provides a consistent Apache HTTP-based web server supporting many platforms including IBM AIX®, HP-UX, Linux, Solaris, Microsoft Windows and IBM z/OS®.
Is supported by IBM when used with licensed and supported IBM products that include IBM HTTP Server.
Not in United States?
United States - English
Considering a purchase?
Contact IBM
Chat now
Email IBM
Request a quote
Or call us at: 1-877-426-3774
Priority code: Middleware
Product support

Portal
Reports
Service Bus
Service Registry
Virtual Assembly Builder
Web Services Manager
WebCenter Interaction
WebLogic Integration
WebLogic Portal
Business Activity Monitoring
BI Publisher
Crystal Ball
Oracle Big Data Discovery
Endeca Information Discovery
Essbase
SOA Governance
Business Intelligence Beans
Integration Adapters
MapViewer
Performance Management
Performance Scorecard
Workforce Planning
Financial Management
Financial Close Management
Strategic Finance
Profitability and Cost Management
Hyperion Planning
Capital Expense Planning
Business Intelligence

Oracle HTTP Server

Oracle HTTP Server (OHS) is an enterprise grade Web Server software - based on open source Apache HTTP Web Server - designed to deliver the following benefits:

- Deliver HTTP Listener for Oracle WebLogic Server through built-in WebLogic Web Server Proxy Plug-In.
- Deliver Web Server component for Fusion Middleware.
- Serve static web content such as HTML, JavaScript, Images etc, and dynamic web content built with CGI/FastCGI based applications.

What's New in Oracle HTTP Server 12.1.3

Oracle HTTP Server 12.1.3 leverages WebLogic Management Framework to provide an administration experience consistent with the rest of the Oracle Fusion Middleware 12c products. For more information, refer to What's new in Oracle HTTP Server 12c section within the product documentation.

Oracle HTTP Server 12c includes the following management tools:

- The Configuration Wizard, which allows you to create and delete Oracle HTTP Server instances. For more information, see *Installing and Configuring Oracle HTTP Server*.
- Fusion Middleware Control, which is a browser-based management tool. For more information, see *Administering Oracle Fusion Middleware*.
- The WebLogic Scripting Tool, which is a command-driven scripting tool. For more information, see *Understanding the WebLogic Scripting Tool* and *Using WebLogic Scripting Tool for OHS*.

For more information, refer to Understanding Oracle HTTP Server 12c Management Tools.

Oracle HTTP Server 12c supports provisioning via two different topologies:

- Administering OHS 12c through WebLogic Domain :- This scenario requires users to install Oracle Fusion Middleware Infrastructure 12c and then install Oracle HTTP Server 12c in the same installation location. For more information, please refer to Standard Installation Topology for OHS 12c in WebLogic Domain. Here, administrators can use either Enterprise Manager Fusion Middleware Control 12c browser based console or WebLogic Scripting Tool (WLST) to administer Oracle HTTP Server 12c.
- Administering OHS 12c through Standalone Domain :- This scenario allows users to install Oracle HTTP Server without any additional dependency. For more information, refer to Standard Installation Topology for OHS 12c in Standalone Domain. Here, users can perform server

XAMPP

- XAMPP stands for Cross-Platform (X), Apache (A), MySQL (M), PHP (P) and Perl (P).
  - Everything you need to set up a web server:
    - The web server program: Apache
    - The database: MySQL (recently replaced with MariaDB)
    - The scripting language: PHP

- XAMPP is also cross-platform, which means it works equally well on Linux, Mac and Windows.

XAMPP Installers and D

https://www.apachefriends.org/index.html

Apache Friends | Download | Add-ons | Hosting | Community | About | Search.. | Search | EN

XAMPP Apache + MariaDB + PHP + Perl

What is XAMPP?

XAMPP is the most popular PHP development environment

XAMPP is a completely free, easy to install Apache distribution containing MariaDB, PHP, and Perl. The XAMPP open source package has been set up to be incredibly easy to install and to use.

Download
Click here for other versions

XAMPP for **Windows**
7.2.3 (PHP 7.2.3)

XAMPP for **Linux**
7.2.3 (PHP 7.2.3)

XAMPP for **OS X**
XAMPP-VM (PHP 7.2.3)

XAMPP Control Panel

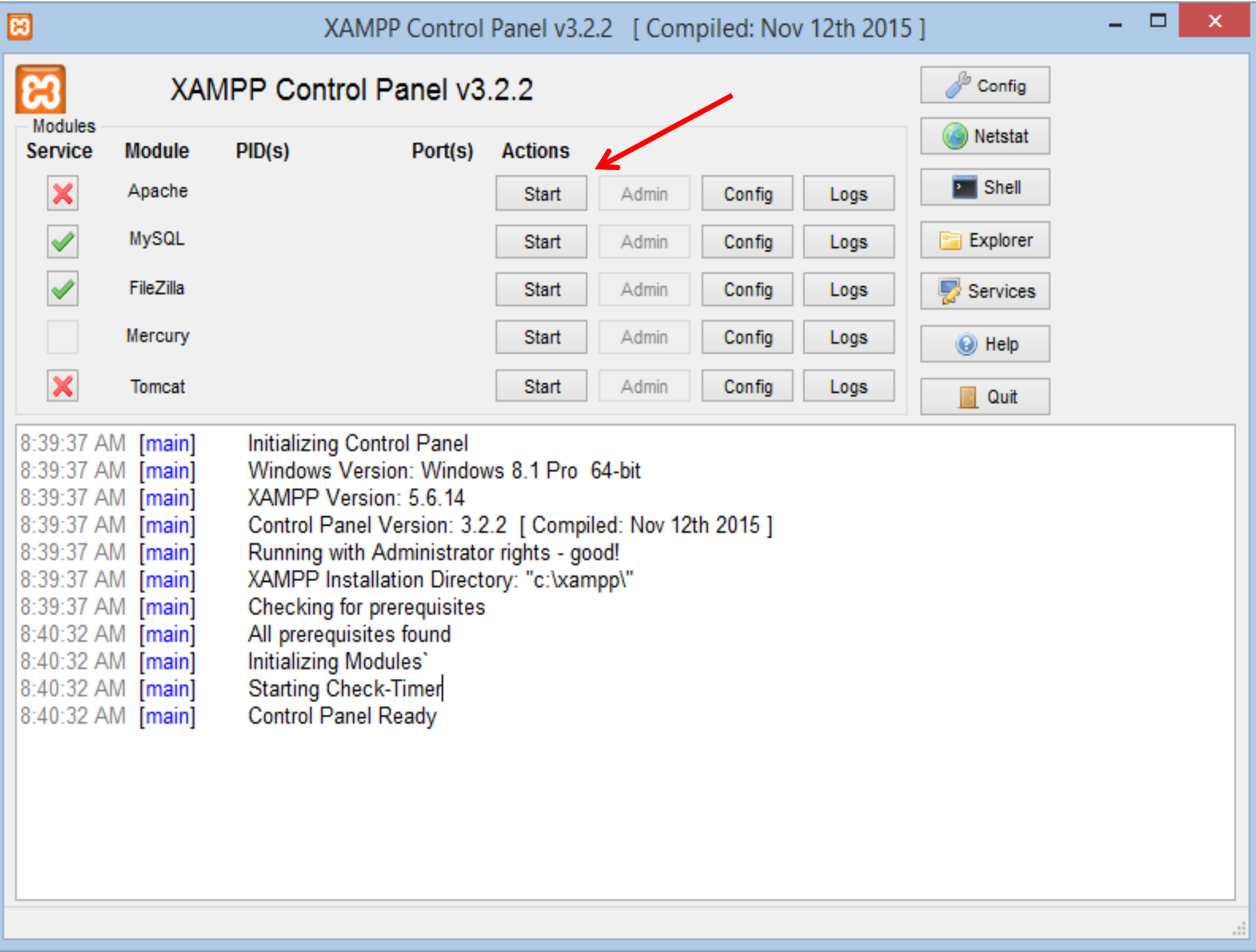
- The XAMPP control panel gives you complete control over all installed XAMPP components.

- Here is a quick overview of the Control Panel.

Testing Your XAMPP Installation

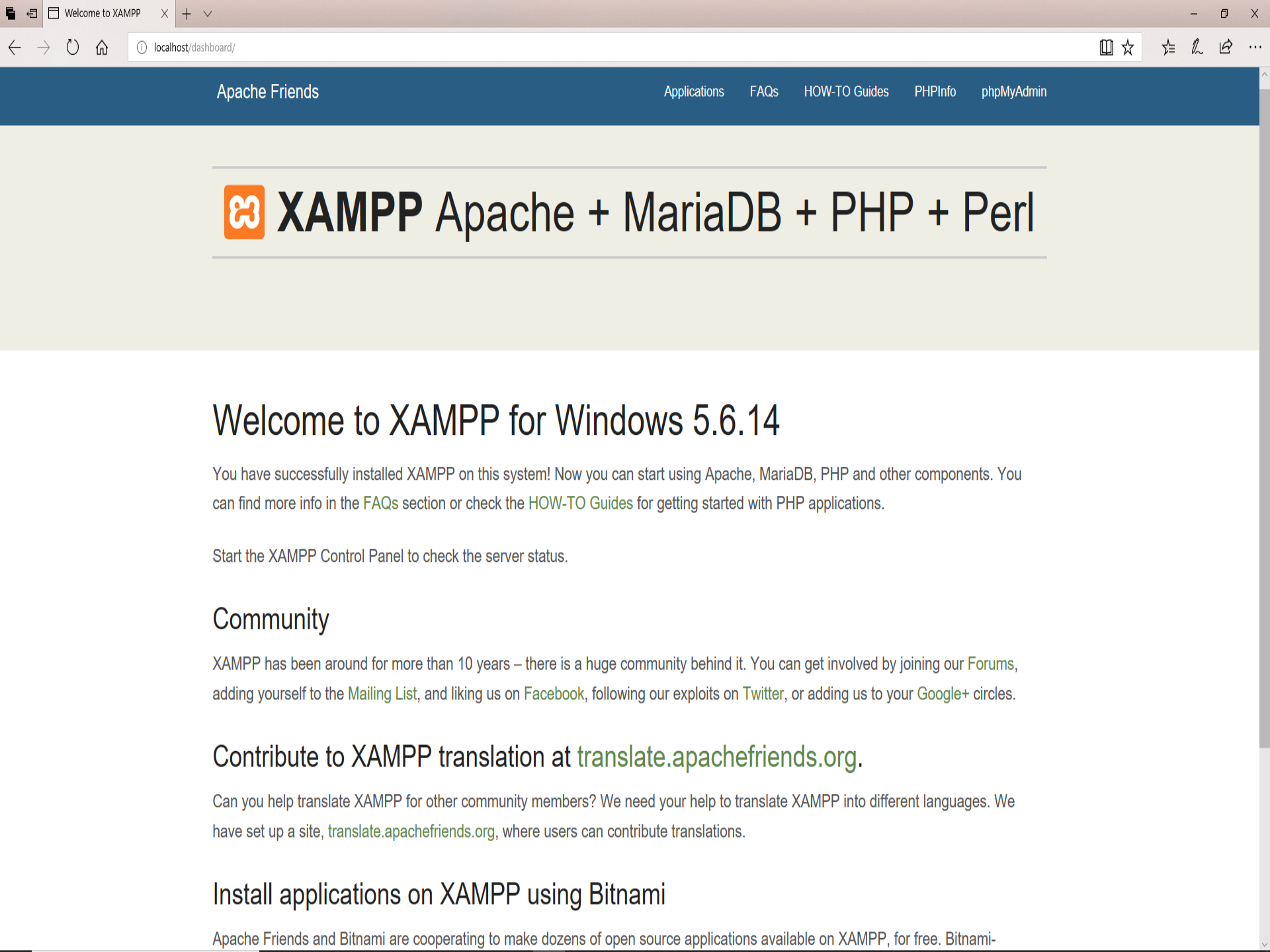
- Follow these steps to test your XAMPP installation by launching the Apache web server:

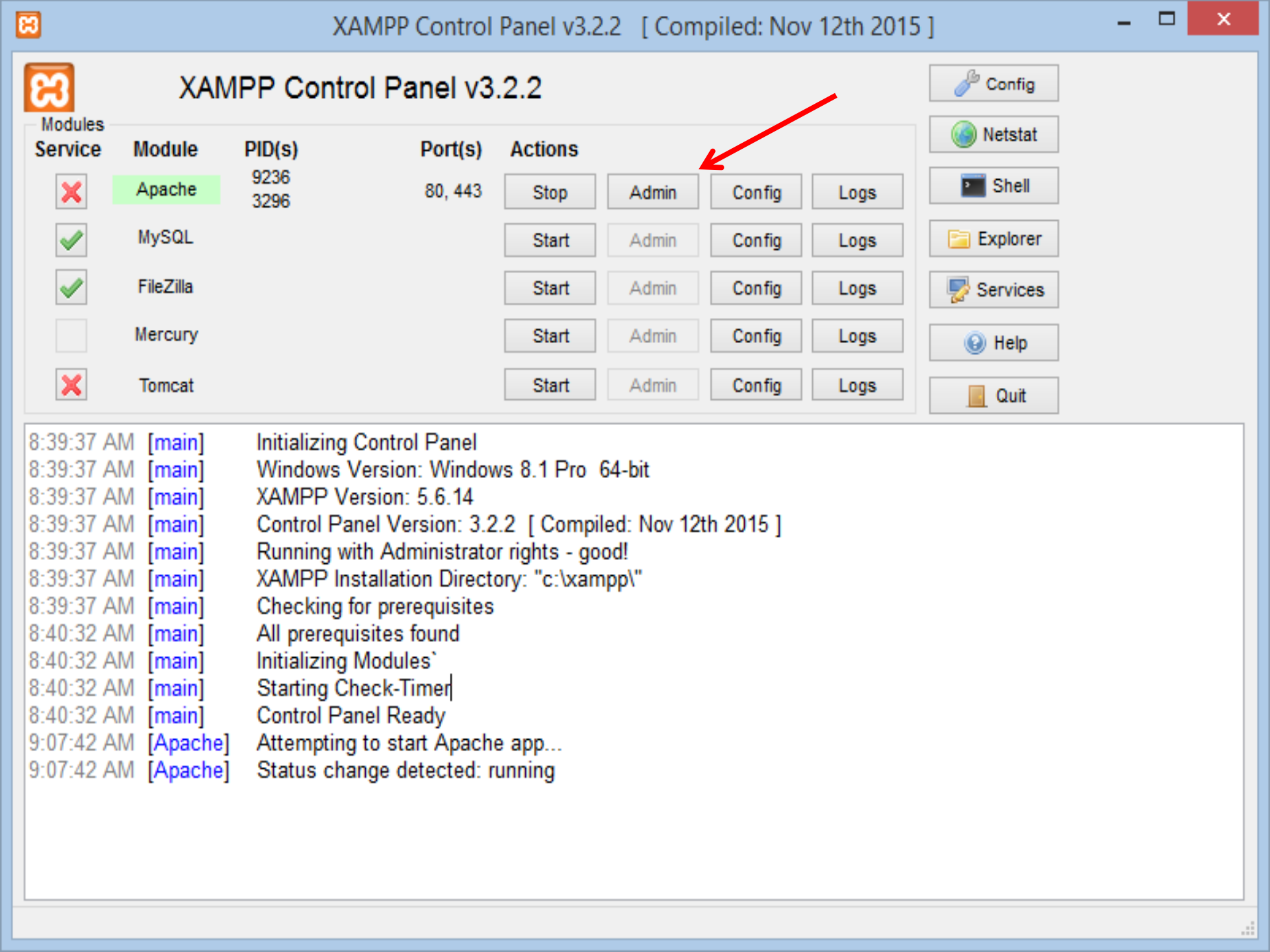
  - **Step 1:** In the XAMPP control panel, click on ‘Start’ under ‘Actions’ for the Apache module, if it’s not already started. This instructs XAMPP to start the Apache webserver. (See next slide)


XAMPP Control Panel v3.2.2 [Compiled: Nov 12th 2015]
XAMPP Control Panel v3.2.2
Modules
Service
Module
PID(s)
Port(s)
Actions
Apache
MySQL
FileZilla
Mercury
Tomcat
Start
Admin
Config
Logs
Config
Netstat
Shell
Explorer
Services
Help
Quit
8:39:37 AM [main] Initializing Control Panel
8:39:37 AM [main] Windows Version: Windows 8.1 Pro 64-bit
8:39:37 AM [main] XAMPP Version: 5.6.14
8:39:37 AM [main] Control Panel Version: 3.2.2 [Compiled: Nov 12th 2015]
8:39:37 AM [main] Running with Administrator rights - good!
8:39:37 AM [main] XAMPP Installation Directory: "c:\xampp\"
8:39:37 AM [main] Checking for prerequisites
8:40:32 AM [main] All prerequisites found
8:40:32 AM [main] Initializing Modules`
8:40:32 AM [main] Starting Check-Timer
8:40:32 AM [main] Control Panel Ready

Testing Your XAMPP Installation

- Follow these steps to test your XAMPP installation by launching the Apache web server:
  - **Step 2:** Open your web browser and type in: **http://localhost** or **127.0.0.1** (Default port: 80)
  - If you see the following page, you have successfully installed XAMPP.

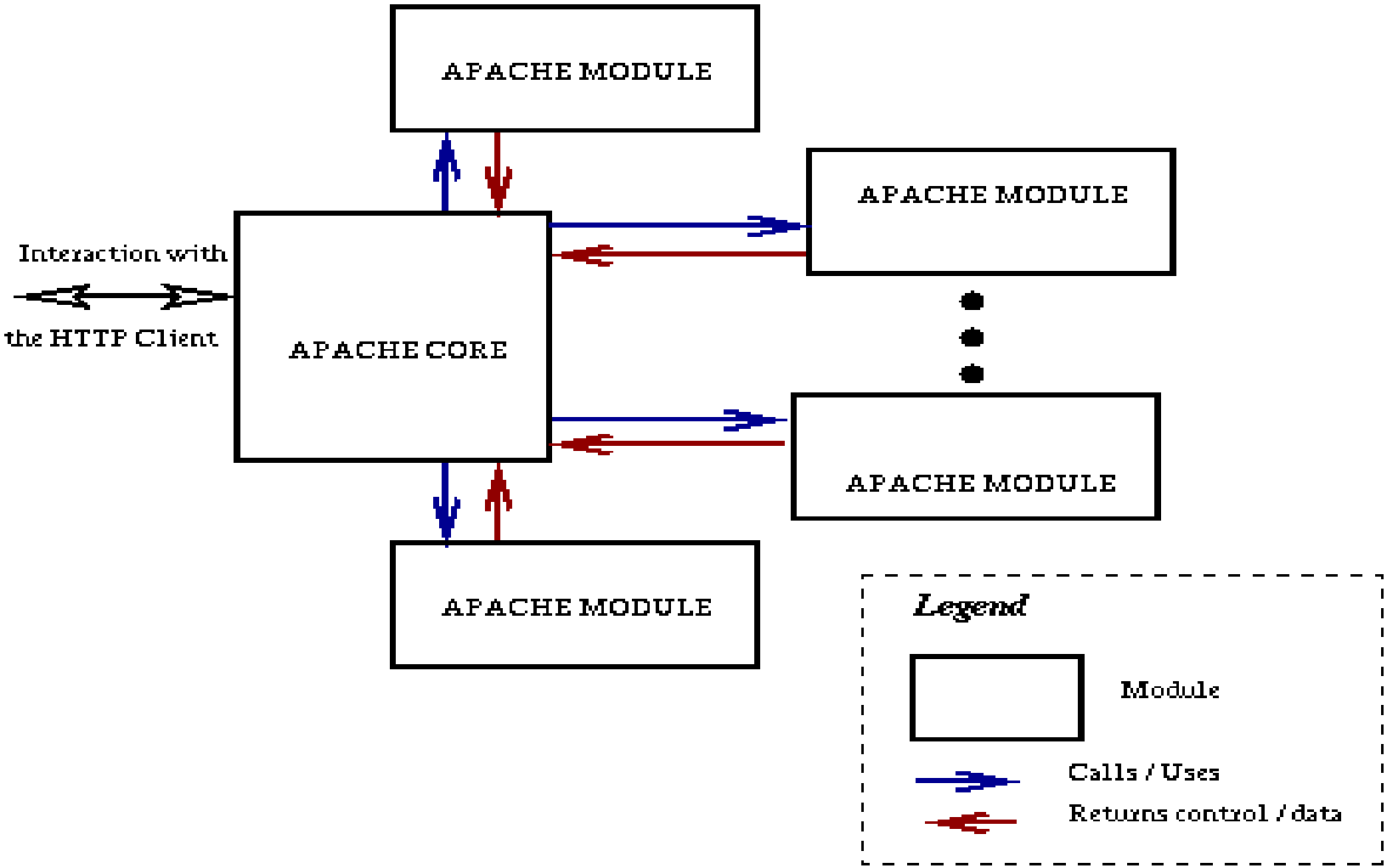

Welcome to XAMPP
localhost/dashboard/
Apache Friends
Applications
FAQs
HOW-TO Guides
PHPInfo
phpMyAdmin
XAMPP Apache + MariaDB + PHP + Perl
Welcome to XAMPP for Windows 5.6.14
You have successfully installed XAMPP on this system! Now you can start using Apache, MariaDB, PHP and other components. You can find more info in the FAQs section or check the HOW-TO Guides for getting started with PHP applications.
Start the XAMPP Control Panel to check the server status.
Community
XAMPP has been around for more than 10 years – there is a huge community behind it. You can get involved by joining our Forums, adding yourself to the Mailing List, and liking us on Facebook, following our exploits on Twitter, or adding us to your Google+ circles.
Contribute to XAMPP translation at translate.apachefriends.org.
Can you help translate XAMPP for other community members? We need your help to translate XAMPP into different languages. We have set up a site, translate.apachefriends.org, where users can contribute translations.
Install applications on XAMPP using Bitnami
Apache Friends and Bitnami are cooperating to make dozens of open source applications available on XAMPP, for free. Bitnami-


XAMPP Control Panel v3.2.2 [Compiled: Nov 12th 2015]
XAMPP Control Panel v3.2.2
Modules
Service
Module
PID(s)
Port(s)
Actions
Apache
9236
3296
80, 443
Stop
Admin
Config
Logs
MySQL
Start
Admin
Config
Logs
FileZilla
Start
Admin
Config
Logs
Mercury
Start
Admin
Config
Logs
Tomcat
Start
Admin
Config
Logs
Config
Netstat
Shell
Explorer
Services
Help
Quit
8:39:37 AM [main] Initializing Control Panel
8:39:37 AM [main] Windows Version: Windows 8.1 Pro 64-bit
8:39:37 AM [main] XAMPP Version: 5.6.14
8:39:37 AM [main] Control Panel Version: 3.2.2 [Compiled: Nov 12th 2015]
8:39:37 AM [main] Running with Administrator rights - good!
8:39:37 AM [main] XAMPP Installation Directory: "c:\xampp\"
8:39:37 AM [main] Checking for prerequisites
8:40:32 AM [main] All prerequisites found
8:40:32 AM [main] Initializing Modules`
8:40:32 AM [main] Starting Check-Timer
8:40:32 AM [main] Control Panel Ready
9:07:42 AM [Apache] Attempting to start Apache app...
9:07:42 AM [Apache] Status change detected: running

Apache HTTP/Web Server

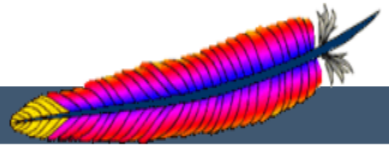
- Apache runs as a daemon on the server.
  - A **daemon** is an executing instance of a program (also called a **process**) that runs in the background, waiting for a specific event that will activate it.
- When a request arrives, Apache then uses modules to determine how to respond to the request.
- In Apache, a **module** is a compiled extension (usually written in the C programming language) to Apache that helps it handle requests.
  - For this reason, these modules are also sometimes referred to as **handlers**.

Apache's Modules



Modular Design

- The Apache web server is **modular**. It consists of:
  - A **core** Apache that deals with **requests** for files.
  - **Common modules** supporting CGI, controls on file access, HTTP content negotiation, and debugging and tracing.
  - **Extension modules** such as the modules with Perl and PHP interpreters and the module for Secure Socket Layer SSL communications.
  - You select the modules that you require, and then you build your Apache server from the parts that you want.
    - The Windows version of Apache supports dynamic linked libraries (.so files)
    - Unix/Linux versions also have shared object libraries & dynamic linking.



Apache HTTP Server Version 2.4

Module Index

Available Languages: de | en | es | fr | ja | ko | tr | zh-cn

Below is a list of all of the modules that come as part of the Apache HTTP Server distribution. See also the complete alphabetical list of all Apache HTTP Server directives.

Core Features and Multi-Processing Modules

core
Core Apache HTTP Server features that are always available

mpm_common
A collection of directives that are implemented by more than one multi-processing module (MPM)

event
A variant of the `worker` MPM with the goal of consuming threads only for connections with active processing

mpm_netware
Multi-Processing Module implementing an exclusively threaded web server optimized for Novell NetWare

mpmt_os2
Hybrid multi-process, multi-threaded MPM for OS/2

prefork
Implements a non-threaded, pre-forking web server

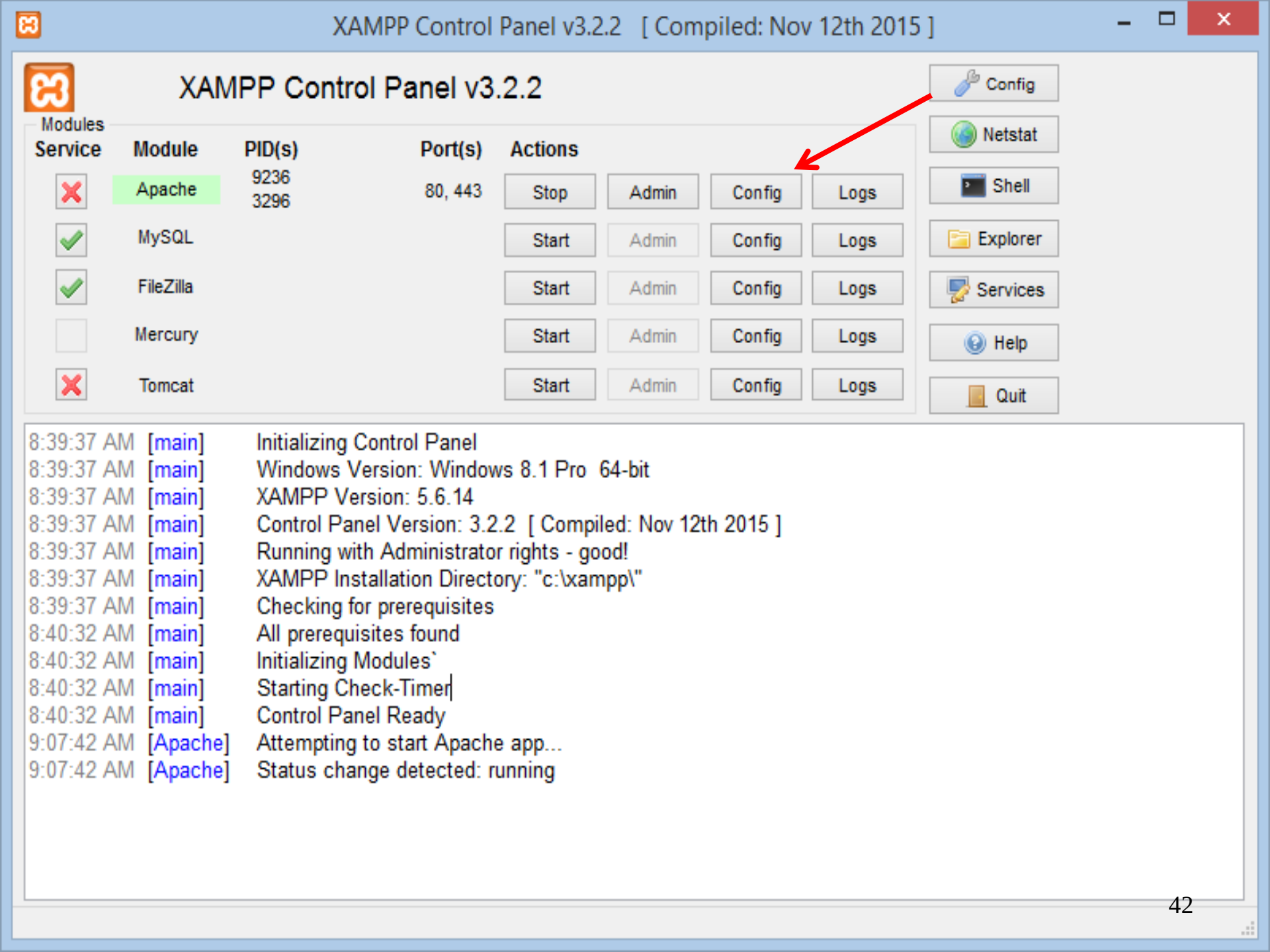
mpm_winnt
Multi-Processing Module optimized for Windows NT.

worker
Multi-Processing Module implementing a hybrid multi-threaded multi-process web server

Apache Web Server

- The Apache web server is also **scalable**.
  - You can configure it so that it can run on a typical home/office PC, providing access to a small company's web site.
  - Apache can also be configured to run well on a multi-CPU servers. Such a configuration would suit a service company that provides web services for a limited number of client companies.
  - You can scale Apache up further
    - One of IBM's server products is basically a large mainframe machine running Apache (It's equivalent to a server farm with hundreds of PC-based web servers).
  - Apache is configured at start up, using data read from a file:

    httpd.conf (next slide)

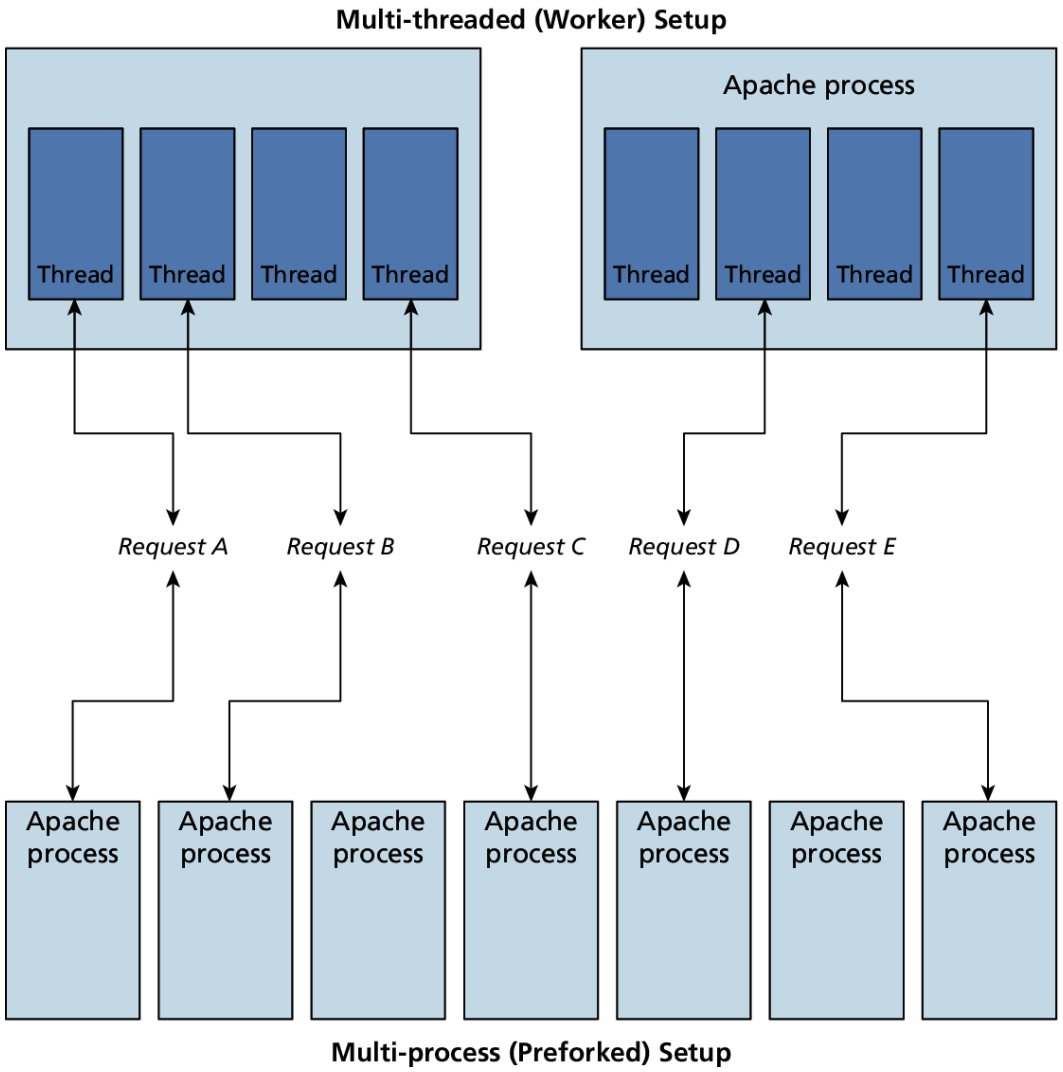

XAMPP Control Panel v3.2.2 [Compiled: Nov 12th 2015]
XAMPP Control Panel v3.2.2
Modules
Service
Module
PID(s)
Port(s)
Actions
Apache
9236
3296
80, 443
Stop
Admin
Config
Logs
MySQL
Start
Admin
Config
Logs
FileZilla
Start
Admin
Config
Logs
Mercury
Start
Admin
Config
Logs
Tomcat
Start
Admin
Config
Logs
Config
Netstat
Shell
Explorer
Services
Help
Quit
8:39:37 AM [main] Initializing Control Panel
8:39:37 AM [main] Windows Version: Windows 8.1 Pro 64-bit
8:39:37 AM [main] XAMPP Version: 5.6.14
8:39:37 AM [main] Control Panel Version: 3.2.2 [Compiled: Nov 12th 2015]
8:39:37 AM [main] Running with Administrator rights - good!
8:39:37 AM [main] XAMPP Installation Directory: "c:\xampp\"
8:39:37 AM [main] Checking for prerequisites
8:40:32 AM [main] All prerequisites found
8:40:32 AM [main] Initializing Modules`
8:40:32 AM [main] Starting Check-Timer
8:40:32 AM [main] Control Panel Ready
9:07:42 AM [Apache] Attempting to start Apache app...
9:07:42 AM [Apache] Status change detected: running

Apache Threads

- Apache runs in two possible modes:
  - **multi-process** (also called **preforked**)
  - **multi-threaded** (also called **worker**)

- The default installation of Apache runs using the multi-process mode.

Apache Threads



Apache's Processes

- Apache uses a collection (pool) of 'pre-forked' processes to reduce the time delays and costs that are associated with the creation of new processes.
  - There is a **principal process** (the 'chief') that monitors the port/socket combination where TCP/IP connection requests are received from clients.
    - This 'chief' process never handles any HTTP requests from the clients.
    - It inly distributes this work to subordinate processes (the 'tribesmen').
    - Each Apache 'tribesman' acts as a serial server, dealing with one client at a time.
    - When a tribesman process finishes with a client, it returns to the pool managed by the chief.

Apache Processes

- As well as being responsible for the distribution of work, the chief process is also responsible adjusting the number of ‘tribesmen’ (child processes).
  - If there are too few tribesmen, clients’ requests will be delayed
  - If there are too many tribesmen, system resources are wasted.

Apache Processes

- Each tribesman process does an '**accept**' on the server socket, which gives it a socket that can be used to read data sent by a client, and to write data back to that client.
  - The tribesman then reads the HTTP 'GET' or 'POST' request submitted by the client.
  - The tribesman process handles a request for a simple static page, or for a page with dynamic content that will be produced by an internal Apache modules, such as mod_include (**server-side includes**), **PHP script,** etc.

httpd.conf

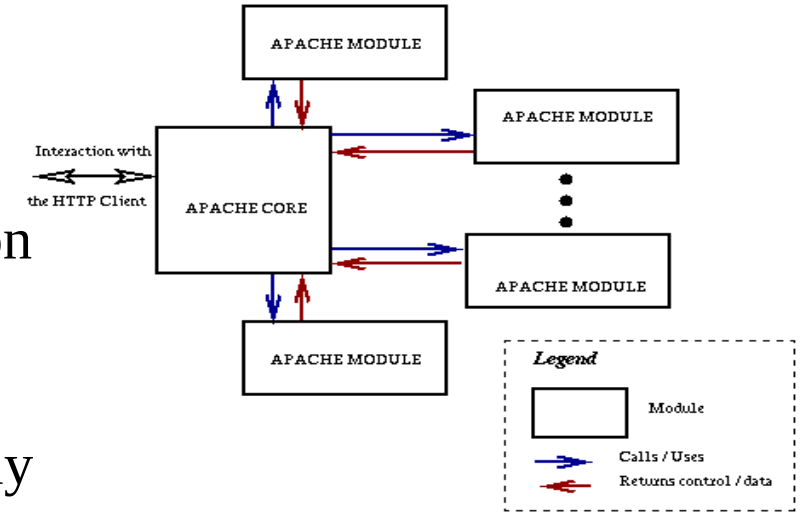
- These details of process behavior are all controlled via a configuration file, **httpd.conf**, that must be edited by the server's administrator.
  - It is in "C:\xampp\apache\conf\extra\httpd-mpm.conf" for XAMPP
- Entries in the file include the following that control the number of Apache processes:
  - **StartServers -** This defines the number of tribesman processes that the chief creates at start-up.
  - **MaxClients** (MaxRequestWorkers) **-** This is an upper limit on the number of processes that you are prepared to run.
  - **MinSpareServers**, **MaxSpareServers -** These values control the chief's behavior with regard to idle tribesmen
    - If there are fewer than MinSpareServers, more are created
    - If there are more than MaxSpareServers, some are terminated.

The Windows Version

- The Windows version of Apache works slightly differently, requiring just two distinct processes. (**WinNT MPM**)
  - The first has roughly the same role as that just described for the ‘chief” in a Linux/Unix system.
  - Instead of a separate process for each tribesman, Windows Apache just has a thread in a second multi-threaded process.
  - Some of the controls that apply in the Linux/Unix world are irrelevant
    - The **MaxRequestsPerChild** (MaxConnectionsPerChild) control does not apply: the threads are not terminated in this way.
    - The **MaxClients** control is replaced by a limit on the number of threads. →**ThreadsPerChild**

Apache's Modules

- The Apache server has a relatively small **core** that can handle HTTP 'GET' requests for static pages and modules that provide all the other services of a full HTTP-compliant server.
  - The default configuration for an Apache server incorporates the modules for dynamic pages (CGI and SSI), for controlled access to resources, for content negotiation, and so forth.
  - You can adjust this default configuration to meet your specific needs.
  - If another module is needed for a Windows version of Apache, you simply un-comment a commented-out **Load-Module** directive **httpd.conf.**



Apache's Modules

- Apache's modules include:
  - Core web server functionality
    - **mod_cgi**, **mod_env**: These modules support CGI-style generation of dynamic content.
    - **mod_include**: This module supports 'server-side includes' that allow the web server itself to create limited forms of dynamic content that are to be inserted into an HTML page prior to its return.
    - **mod_log_config**: This module handles the basic logs for the server
      - They record all accesses to web resources and also all errors (such as requests for non-existent files that might indicate bad links in your web site).

Apache's Modules

- **mod_access**: This module allows access to selected web resources to be restricted to clients whose IP addresses satisfy specified constraints. (Renamed **mod_authz_host**)
- **mod_auth_basic**, **mod_authn_dbd**, **mod_authn_dbm**: These modules all support access controls that require a client to supply a password before access is granted to specified web resources.
- **mod_mime**: This module determines content type from file extension – so allowing the server to handle a get request for picture.gif by correctly returning a response with the HTTP header **Content-type=image/gif**.
- **mod_negotiation**: This module deals with HTTP Accept request headers that specify preferred content type.

Apache's Modules

- Server administrator options
  - **mod_status**: This generates an HTML page that displays information about the server; data shown include the number of processes currently handling requests, the number that have finished with their clients but which are still writing log data, and the number of idle processes.
  - **mod_info**: This displays a page with details of the configuration options for the server.

Apache Server Status for localhost

Server Version: Apache/2.2.21 (Win32) mod_ssl/2.2.21 OpenSSL/1.0.0e PHP/5.3.8 mod_perl/2.0.4 Perl/v5.10.1
Server Built: Sep 10 2011 11:34:11

Current Time: Tuesday, 31-Mar-2015 11:22:39 Eastern Daylight Time
Restart Time: Tuesday, 31-Mar-2015 10:20:27 Eastern Daylight Time
Parent Server Generation: 0
Server uptime: 1 hour 2 minutes 11 seconds
Total accesses: 24 - Total Traffic: 248 kB
.00643 requests/sec - 68 B/second - 10.3 kB/request
2 requests currently being processed, 148 idle workers

_____
_____
_____W__R_.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Apache Server Information

Subpages:
Configuration Files, Server Settings, Module List, Active Hooks

---

Sections:
Server Settings, Startup Hooks, Request Hooks

Loaded Modules:
mod_perl.c, mod_php5.c, mod_status.c, mod_ssl.c, mod_setenvif.c, mod_rewrite.c, mod_proxy_ajp.c, mod_proxy.c, mod_negotiation.c, mod_mime.c, mod_log_config.c, mod_isapi.c, mod_info.c, mod_include.c, mod_headers.c, mod_env.c, mod_dir.c, mod_dav_lock.c, mod_cgi.c, mod_autoindex.c, mod_authz_user.c, mod_authz_host.c, mod_authz_groupfile.c, mod_authz_default.c, mod_authn_file.c, mod_authn_default.c, mod_auth_digest.c, mod_auth_basic.c, mod_asis.c, mod_alias.c, mod_actions.c, mod_so.c, http_core.c, mpm_winnt.c, mod_win32.c, core.c

---

Server Settings

Server Version: Apache/2.2.21 (Win32) mod_ssl/2.2.21 OpenSSL/1.0.0e PHP/5.3.8 mod_perl/2.0.4 Perl/v5.10.1
Server Built: Sep 10 2011 11:34:11
Server loaded APR Version: 1.4.5
Compiled with APR Version: 1.4.5
Server loaded APU Version: 1.3.12
Compiled with APU Version: 1.3.12
Module Magic Number: 20051115:30
Hostname/port: localhost:80
Timeouts: connection: 300 keep-alive: 5
MPM Name: WinNT

Apache's Modules

- More exotic modules
  - **mod_imagemap**: This module supports server-side image-map processing. (Most web pages now rely on browsers to handle image-map interactions at the client side, so you shouldn't really need this.)
  - **mod_proxy**: This allows your Apache to act as a proxy server. Apache can filter requests by blocking access to named sites, and forwarding other requests to the actual remote server. You can also enable caching.
  - **mod_speling**: Tries to guess what the client meant if there is no resource with the name appearing in the client's request.

Logs

- In its standard configuration, Apache records all access attempts by clients (**access_log** file) and all server-side errors (**error_log** file).
  - The **access_log** file tracks client requests . Analyzing the data in the **access_log** file may help you better organize and market your site.
  - The **error_log** file records important events to help you find problems with your site (debug).
- These files don’t exist until you start Apache the first time.
- The names of the files are **access.log** and **error.log** for Windows platforms.

Standard Apache Access Logging

- Using Apache's basic logging features, you can keep track of who visits your websites by logging accesses to the servers hosting them.
  - You can log every aspect of the browser requests and server responses, including the IP address of the client, user, and resource accessed.
  - You need to take three steps to create a request log:
    1. Define **what** you want to log—your log format.
    2. Define **where** you want to log it—your log files, a database, an external program.
    3. Define **whether** to log—conditional logging rules.

What to Log

- You can define how your log entries appear by creating a log format.
  - A **log format** is a string that contains text mixed with log-formatting **directives**.
  - Log-formatting directives start with a **%** followed by a directive name or identifier, usually a letter indicating the piece of information to be logged.
  - When Apache logs a request, it scans the string and substitutes the value for each directive.
    - For example, if the log format is:

      This is the client address **%a**

    - Then the log entry would be something like:

      This is the client address 10.0.0.2

Formatting Options	Explanation
Data from the Client	
%a	Remote IP address, from the client.
%h	Hostname or IP address of the client making the request. Whether the hostname is logged depends on two factors: The IP address of the client must resolve to a hostname via a reverse DNS lookup, and Apache must be configured to do that lookup using the `HostNameLookups` directive, explained later in this chapter. If these conditions are not met, the IP address of the client is logged rather than the hostname.
%l	Remote user, obtained via the `identd` protocol. This option is not very useful because the majority of the client machines do not support this protocol.
%u	Remote user, from the HTTP basic authentication protocol.
Data from the Server	
%A	Local IP address, from the server.
%D	Time it took to serve the request, in microseconds.
%{*env_variable*}e	Value for an environment variable named *env_variable*. (There are many.)
%{*time_format*}t	Current time. If {*time_format*} is present, it is interpreted as an argument to the UNIX `strftime` function. See the `logresolve` Apache manual page for details.

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item

Ident protocol

From Wikipedia, the free encyclopedia

The **Ident Protocol** (**Identification Protocol**, **IDENT**), specified in RFC 1413, is an Internet protocol that helps identify the user of a particular TCP connection. One popular daemon program for providing the ident service is **identd**.

Contents [hide]

Internet protocol suite

Application layer
BGP · DHCP · DNS · FTP · HTTP · IMAP · LDAP · MGCP · NNTP · NTP · POP · ONC/RPC · RTP · RTSP · RIP · SIP · SMTP · SNMP · SSH · Telnet · TLS/SSL · XMPP · *more...*

Transport layer
TCP · UDP · DCCP · SCTP · RSVP · *more...*

Internet layer
IP (IPv4 · IPv6) · ICMP · ICMPv6 · OSPF · ECN · IGMP · IPsec · *more...*

Link layer
ARP · NDP · Tunnels (L2TP) · PPP · MAC (Ethernet · DSL · ISDN · FDDI) · *more...*

V · T · E

How ident works [edit]

The Ident Protocol is designed to work as a server daemon, on a user's computer, where it receives requests to a specified port, generally 113. In the query, a client specifies a pair of TCP ports (a local and a remote port), encoded

Formatting Options	Explanation
`%T`	Time it took to serve the request, in seconds.
`%v`	Canonical name of the server that answered the request.
`%V`	Server name according to the `UseCanonicalName` directive.
`%X`	Status of the connection to the server. A value of `x` means the connection was aborted before the server could send the data. A `+` means the connection will be kept alive for further requests from the same client. A `-` means the connection will be closed.
Data from the Request	
`%{`*`cookie_name`*`}C`	Value for a cookie named *`cookie_name`*.
`%H`	Request protocol, such as HTTP or HTTPS.
`%m`	Request method such as `GET`, `POST`, `PUT`, and so on.
`%{`*`header_name`*`}i`	Value for a header named *`header_name`* in the request from the client. This information can be useful, for example, to log the names and versions of your visitors' browsers.
`%r`	Text of the original HTTP request.
`%q`	Query parameters, if any, prefixed by a `?`.
`%U`	Requested URL, without query parameters.
`%y`	Username for the HTTP authentication (basic or digest).
Data from the Response	
`%b`, `%B`	Size, in bytes, of the body of the response sent back to the client (excluding headers). The only difference between the options is that if no data was sent, `%b` will log a `-` and `%B` will log `0`.
`%f`	Path of the file served, if any.
`%t`	Time when the request was served.
`%{`*`header_name`*`}o`	Value for a header named *`header_name`* in the response to the client.
`%>s`	Final status code. Apache can process several times the same request (internal redirects). This is the status code of the final response.

Common Log Format

- **The Common Log Format** (CLF) is a standard log format.
  - Most websites can log requests using this format, and many log processing and reporting tools understand the format.
  - Its format is the following:

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

  - That is, it includes:
    - the hostname or IP address of the client
    - remote user via identd
    - remote user via HTTP authentication
    - time when the request was served
    - text of the request
    - status code
    - size in bytes of the content served

Where to Log

- Logging to files is the default way of logging requests in Apache.
  - You can define the name of the file using the **TransferLog** and **CustomLog** directives.

Where to Log

- The **CustomLog** directive enables you to specify the logging format explicitly.
  - It takes two arguments: destination file & a logging format file.
  - The logging format can be specified as a nickname or as a logging string directly.
  - For example:

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{User-agent}i\"" **myformat**
CustomLog logs/access_log **myformat**

Or, equivalently:

CustomLog logs/access_log "%h %l %u %t \"%r\" %>s %b \"%{User-agent}i\""

Where to Log

- The **TransferLog** directive takes a file argument and uses the latest log format defined by a LogFormat directive with a single argument (the nickname or the format string).
  - If no log format is present, it defaults to the CLF.
  - Example:

```
LogFormat "%h %l %u %t \"%r\” %>s %b \"%{User-agent}i\""
TransferLog logs/access_log
```

Whether to Log

- You can specify whether to log specific elements in a log entry, by inserting a list of status codes, separated by commas.

- If the request status is one of the listed codes, the parameter is logged; otherwise, a **-** is logged.

- For example:

  %400,501{User-agent}i

  This logs the browser name and version for malformed requests (status code 400) and requests with methods not implemented (status code 501)

The Webalizer

100% Proud to be Microsoft free!

What is it?

The Webalizer is a fast, free web server log file analysis program. It produces highly detailed, easily configurable usage reports in HTML format, for viewing with a standard web browser.

Features

- Is written in C to be extremely fast and highly portable. On my 1.6Ghz laptop, it can process close to 70,000 records per second, which means a log file with roughly 2 million hits can be analyzed in about 30 seconds.
- Handles standard Common logfile format (CLF) server logs, several variations of the NCSA Combined logfile format, wu-ftpd/proftpd xferlog (FTP) format logs, Squid proxy server native format, and W3C Extended log formats. In addition, gzip (.gz) and bzip2 (.bz2) compressed logs may be used directly without the need for uncompressing.
- Generated reports can be configured from the command line, or more commonly, by the use of one or more configuration files. Detailed information on configuration options can be found in the README file, supplied with all distributions.
- Supports multiple languages. Currently, Albanian, Arabic, Catalan, Chinese (traditional and simplified), Croatian, Czech, Danish, Dutch, English, Estonian, Finnish, French, Galician, German, Greek, Hungarian, Icelandic, Indonesian, Italian, Japanese, Korean, Latvian, Lithuanian, Malay, Norwegian, Polish, Portuguese (Portugal and Brazil), Romanian, Russian, Serbian, Slovak, Slovene, Spanish, Swedish, Thai, Turkish and Ukrainian are available.
- Unlimited log file sizes and partial logs are supported, allowing logs to be rotated as often as needed, and eliminating the need to keep huge monthly files on the system.
- Fully supports IPv4 and IPv6 addresses. Includes built-in distributed DNS lookup capability and native Geolocation services.
- Distributed under the GNU General Public License, complete source code is available, as well as binary distributions for some of the more popular platforms. Please read the Copyright notices for additional information.

Download
Whats New
Mirrors
Credits
FAQ

Webalizer

From Wikipedia, the free encyclopedia

The **Webalizer** is a GPL application that generates web pages of analysis, from access and usage logs, i.e. it is web log analysis software. It is one of the most commonly used web server administration tools. It was initiated by Bradford L. Barrett in 1997. Statistics commonly reported by Webalizer include hits, visits, referrers, the visitors' countries, and the amount of data downloaded. These statistics can be viewed graphically and presented by different time frames, such as by day, hour, or month.

Webalizer

Stable release	2.23-08 / August 26, 2013; 18 months ago
Written in	C
Operating system	Cross-platform
Available in	Over 30 languages
Type	Web analytics
License	GNU General Public License
Website	www.webalizer.org

Contents [hide]

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here

Simpletons Guide to Web Server Analysis

Welcome to the wonderful world of web server usage analysis! This guide is intended to provide the necessary background and insight to how web server analysis works, things to look for and things to watch out for. Specifically, this guide is intended for the users of the Webalizer, but can be applied to most any analysis package out there. If you are new to web server analysis, or just want to find out how things work, then this guide is for you.

Hit me Please!

Ok, so you got a web site and you want to know if anybody is looking at it, and if so, what they are looking at and how many times. Lucky for you, (most) every web server keeps a log of what it's doing, so you can just go look and see. The logs are just plain ASCII text files, so any text editor or viewer would work just fine. Each time someone (using a web browser) asks for one of your web pages, or any componant thereof (known as URLs, or *Uniform Resource Locator*), the web server will write a line to the end of the log representing that request. Unfortunately, the raw logs are rather crpytic for everyday humans to read. While you might be able to determine *if* anybody was looking at your web site, any other information would require some sort of processing to determine. A typical log entry might look something like the following:

```
192.168.45.13 - - [24/May/2005:11:20:39 -0400] "GET /mypage.html HTTP/1.1" 200 117
```

This represents a request from a computer with the IP address 192.168.45.13 for the URL **/mypage.html** on the web server. It also gives the time and date the request was made, the type of request, the result code for that request and how many bytes were sent to the remote browser. There will be a line similar to this one for each and every request made to the web server over the period covered by the log. A '**Hit**' is another way to say '*request made to the server*', so as you may have noticed, each line in the log represents a '**Hit**'. If you want to know how many Hits your server received, just count the number of lines in the log. And since each log line represents a request for a *specific* URL, from a *specific* IP address, you can easily figure out how many hits you got for each of your web pages or how many hits you received from a particular IP address by just counting the lines in the log that contain them. Yes, it really is that simple. And while you *could* do this manually with a text editor or other simple text processing tools, it is much more practical and easier to use a program specifially designed to analyze the logs for you, such as the Webalizer. They take the work out of it for you, provide totals for many other aspects of your server, and allow you to visualize the data in a way not possible by just looking at the raw logs.

How does it all work?!?

Well, to understand what you can analyze, you really should know what information is provided by your web server and how it gets there. At the very least, you should know how the HTTP (*HyperText Transport Protocol*) protocol works, and it's strengths and weaknesses. At it's simplest, a web server just sits there listening on the network for a web browser to make a request. Once a request is received, the server processes it and then sends something back to the requesting browser (and as explained above, the request is logged to a log file). These requests are typically for some URL, although there are other types of information a browser can request, such as server type, HTTP protocol versions supported, modification dates, etc., but those types are not as common. To visualize the interaction between server, browser and web pages, lets use an example to illustrate the information flow. Imagine a simple web page, '**mypage.html**', which is a HTML web page that contains two graphic images, '**myimage1.jpg**', and '**myimage2.jpg**'. A typical server/browser interaction might go something like this:

- The web browser asks for the URL **mypage.html**.
- The server sees the request and sends back the HTML page.
- The web browser notices that there are two inline graphic links in the HTML page, so it asks for the first one, **myimage1.jpg**.
- The server sees the request and sends back the graphic image.

Error Logs

- Apache can be configured to log error messages and debugging information, in addition to client requests.
  - In addition to errors generated by Apache itself, CGI errors can be logged. (e.g. **php_error_log**)
  - Each error log entry is prefixed by the time the error occurred and the client IP address or hostname, if available.
- As with HTTP request logging, you can log error information to a file or program.
  - On UNIX systems, you can also log to the **syslog daemon**.
  - On Windows, errors can be logged in the Windows event log and are then viewable via the **Windows Event Viewer**.
- Use the **ErrorLog** directive to define where you want your logs to go. (in httpd.conf)

ErrorLog Directive

Description:	Location where the server will log errors
Syntax:	`ErrorLog` *file-path*`|syslog[:[`*facility*`][:`*tag*`]]`
Default:	`ErrorLog logs/error_log (Unix) ErrorLog logs/error.log (Windows and OS/2)`
Context:	server config, virtual host
Status:	Core
Module:	core

The `ErrorLog` directive sets the name of the file to which the server will log any errors it encounters. If the *file-path* is not absolute then it is assumed to be relative to the `ServerRoot`.

```
ErrorLog "/var/log/httpd/error_log"
```

If the *file-path* begins with a pipe character "`|`" then it is assumed to be a command to spawn to handle the error log.

```
ErrorLog "|/usr/local/bin/httpd_errors"
```

See the notes on piped logs for more information.

Using `syslog` instead of a filename enables logging via syslogd(8) if the system supports it. The default is to use syslog facility `local7`, but you can override this by using the `syslog:`*facility* syntax where *facility* can be one of the names usually documented in syslog(1). The facility is effectively global, and if it is changed in individual virtual hosts, the final facility specified affects the entire server. Same rules apply for the syslog tag, which by default uses the Apache binary name, `httpd` in most cases. You can also override this by using the `syslog::`*tag* syntax.

```
ErrorLog syslog:user
ErrorLog syslog:user:httpd.srv1
ErrorLog syslog::httpd.srv2
```

Additional modules can provide their own ErrorLog providers. The syntax is similar to the `syslog` example above.

Logging Errors to a File

- A file argument indicates the path to the error log file. If the path is relative, it is assumed to be relative to the server root.

- By default, the error log file is located in the logs directory and is named **error_log** on UNIX and **error.log** on Windows.

- An example:

  ErrorLog logs/my_error_log

Logging Errors to a Program

- You can specify the path to a program, prefixed by a pipe |.
- Apache logs errors to the standard input of the program, and the program further processes them.
- An example:

  ErrorLog “|/usr/local/bin/someprogram”

Log Levels

- The error log contains reports for all errors of greater than a chosen severity.
- The control is in a configuration parameter; you can select ‘debug’, ‘info’, ‘notice’, ‘**warn**’, ‘error’, ‘crit’, ‘alert’ or ‘emerg’.
- Most sites run at ‘warn’ level.
- The log will include entries for missing files, access failures for authorization, problems with file permissions and errors in CGI programs.

The LogLevel Directive

- The error information provided by Apache has several degrees of importance.
- You can choose to log only important messages and disregard informational or trivial warning messages.
- The LogLevel directive takes an error-level argument.
- Only errors of that level of importance or higher are logged.

- An example:

```
# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
#
LogLevel warn
```

LogLevel Directive

Description:	Controls the verbosity of the ErrorLog
Syntax:	`LogLevel [module:]level [module:level] ...`
Default:	`LogLevel warn`
Context:	server config, virtual host, directory
Status:	Core
Module:	core
Compatibility:	Per-module and per-directory configuration is available in Apache HTTP Server 2.3.6 and later

`LogLevel` adjusts the verbosity of the messages recorded in the error logs (see `ErrorLog` directive). The following *level*s are available, in order of decreasing significance:

Level	Description	Example
`emerg`	Emergencies - system is unusable.	"Child cannot open lock file. Exiting"
`alert`	Action must be taken immediately.	"getpwuid: couldn't determine user name from uid"
`crit`	Critical Conditions.	"socket: Failed to get a socket, exiting child"
`error`	Error conditions.	"Premature end of script headers"
`warn`	Warning conditions.	"child process 1234 did not exit, sending another SIGHUP"
`notice`	Normal but significant condition.	"httpd: caught SIGBUS, attempting to dump core in ..."
`info`	Informational.	"Server seems busy, (you may need to increase StartServers, or Min/MaxSpareServers)..."
`debug`	Debug-level messages	"Opening config file ..."
`trace1`	Trace messages	"proxy: FTP: control connection complete"
`trace2`	Trace messages	"proxy: CONNECT: sending the CONNECT request to the remote proxy"
`trace3`	Trace messages	"openssl: Handshake: start"
`trace4`	Trace messages	"read from buffered SSL brigade, mode 0, 17 bytes"
`trace5`	Trace messages	"map lookup FAILED: map=rewritemap key=keyname"
`trace6`	Trace messages	"cache lookup FAILED, forcing new map lookup"
`trace7`	Trace messages, dumping large amounts of data	"| 0000: 02 23 44 30 13 40 ac 34 df 3d bf 9a 19 49 39 15 |"
`trace8`	Trace messages, dumping large amounts of data	"| 0000: 02 23 44 30 13 40 ac 34 df 3d bf 9a 19 49 39 15 |"

When a particular level is specified, messages from all other levels of higher significance will be reported as well. *E.g.*, when `LogLevel info` is specified,

Access Controls

- The Apache server allows you to provide selective access to resources using restrictions on a client's address, through a requirement for a password, or by a combination of both these methods.
  - Typically, different policies are applied to resources in different directories, but you can have additional global constraints
    - For example, it is possible to specify that clients may never access a file whose name starts with '.ht'
    - Such names are commonly used for Apache password files and some configuration files.

Access Controls

- Controls on resources can be defined either:
  - In the main **httpd.conf** runtime configuration file, or
  - In **.htaccess** files located in the directories holding the resources (or holding the subdirectories with resources).
- Generally, it is best to centralize all controls in the main **httpd.conf** file.
  - There are two problems with **.htaccess** files.
    - They do add to the work that a web server must perform.
    - The second problem is that these .htaccess files may reduce the security of your web site.
  - This is particularly likely to occur if you allow individual users to maintain files in their private directories and further allow them to specify their own access controls.

Access Controls

- Basic controls (in **mod_access,** now renamed as **mod_authz_host**) allow some restrictions based on the IP address or domain name included in the request.
- The controls allow you to specify that:
  - A resource is generally available.
  - Access to the resource is prohibited for clients with addresses that fall in a specified range of IP addresses (or a specified domain), but access is permitted from everywhere else.
  - Access is prohibited except for clients whose IP addresses fall in a specified range or whose domain matches a specified domain.

Access Controls

- Controls are defined in the **httpd.conf** file using **Directory**, **DirectoryMatch**, **Files** directives.
  - These **directives** have the general form:

    <Directory *resource-location*>

    control options

    </Directory>

  - A Directory directive will have the full pathname of the directory to which the controls apply.
  - A DirectoryMatch, or FilesMatch directive, can use simple **regular expressions** to identify the resources.

Access Controls (DEPRECATED)

- The control options related to access are **Order**, **Allow** and **Deny**.
  - **Allow:** used to specify the IP range, or domain name, for those clients who are permitted access to a resource.
  - **Deny:** identifies those excluded.
  - **Order** specifies how the checks are to apply:
    - If the order is **Deny, Allow** then the resource by default is accessible; the client is checked against the Deny constraint and, if matched, will be blocked unless the client also matches the subsequent more specific Allow constraint.
    - If the order is **Allow, Deny** then the resource is by default inaccessible; if the client matches the Allow constraint access will be permitted provided the client is not caught by a more closely targeted Deny constraint.

Access Controls (DEPRECATED)

- The following examples illustrate constraints applied to the contents of directories and their subdirectories. They assume that your Apache is installed in /local/apache. (C:\xampp for XAMPP for Windows)
  - The first example:

```
<Directory "/local/apache/htdocs/onCampus">
    Order deny, allow
    Deny from all
    Allow from .bigcampus.edu
</Directory>
```

  - It defines a restricted subdirectory that is only to be accessed by people who are logged into the domain **bigcampus.edu**.

Access Controls (DEPRECATED)

- The second example:

```
<Directory /local/apache/htdocs/notForTheFrench>
    Order allow, deny
    Allow from all
    Deny from .fr
</Directory>
```

- This would be appropriate if you had a resource that was for some reason not to be available to clients in France.

Access Controls (DEPRECATED)

- The standard **httpd.conf** file contains an example of a FilesMatch directive:

```
# The following lines prevent .htaccess and .htpasswd files from being
# viewed by Web clients.
#
<FilesMatch "^\.ht">
    Order allow,deny
    Deny from all
    Satisfy All
</FilesMatch>
```

- This has a regular expression match that defines all files that start with the sequence .ht.
- Access to these files is globally prohibited.

The Satisfy Directive

- This directive is only useful if access to a particular area is being restricted by ***both*** **username/password** ***and*** **client host address.**
- The parameter can be either **All** or **Any**.
  - In this case the default behavior (All) is to require that the client passes the address access restriction and enters a valid username and password.
  - With the Any option the client will be granted access if they either pass the host restriction or enter a valid username and password.
  - This can be used to password-protect an area, but also to let clients from particular addresses in without prompting for a password.

Access Controls

- Incorporation of any of **mod_auth**, **mod_auth_dbd** or **mod_auth_dbm** modules into your Apache allows you to utilize HTTP authentication.
  - These modules differ only with respect to how name and password data are stored.
    - The _dbd and _dbm modules use various versions of the dbd/dbm simple database package that is available for Linux/Unix.
    - The basic mode_auth module works with text files defining your users and their passwords, and also any user-groups that you wish to have. (The passwords in the password file are held in encrypted form.)
  - The text files are simpler; but if you are likely to have hundreds of users, you should use one of the db packages to avoid performance problems with large text files.

Access Controls (DEPRECATED)

- An example of a Directory directive specifying an authorization control is:

```
<Directory /local/apache/htdocs/notices>
    AuthName  "Private Departmental Notices"
    AuthType  Basic
    AuthUserFile  /local/apache/pwrds/.htpasswds
    AuthGroupFile  /local/apache/pwrds/.htgroups
    Require  valid-user
</Directory>
```

- **Group files** are simple text files; each line in the file defines a group and its members:

```
BridgePlayers: anne david carol phillip peter jon james
```

Access Controls (DEPRECATED)

- Authorization and IP/domain restrictions can be combined:

```
<Directory /local/apache/htdocs/DevelopMent/hotstuff>
    Order deny, allow
    Deny from all
    Allow from 130.130
    AuthName ...
    ...
    Require group staff
    Satisfy all
</Directory>
```

Access control by host

If you wish to restrict access to portions of your site based on the host address of your visitors, this is most easily done using `mod_authz_host`.

The `Require` provides a variety of different ways to allow or deny access to resources. In conjunction with the `RequireAll`, `RequireAny`, and `RequireNone` directives, these requirements may be combined in arbitrarily complex ways, to enforce whatever your access policy happens to be.

> The `Allow`, `Deny`, and `Order` directives, provided by `mod_access_compat`, are deprecated and will go away in a future version. You should avoid using them, and avoid outdated tutorials recommending their use.

The usage of these directives is:

```
Require host address
Require ip ip.address
```

In the first form, *address* is a fully qualified domain name (or a partial domain name); you may provide multiple addresses or domain names, if desired.

In the second form, *ip.address* is an IP address, a partial IP address, a network/netmask pair, or a network/nnn CIDR specification. Either IPv4 or IPv6 addresses may be used.

See the mod_authz_host documentation for further examples of this syntax.

You can insert `not` to negate a particular requirement. Note, that since a `not` is a negation of a value, it cannot be used by itself to allow or deny a request, as *not true* does not constitute *false*. Thus, to deny a visit using a negation, the block must have one element that evaluates as true or false. For example, if you have someone spamming your message board, and you want to keep them out, you could do the following:

```
<RequireAll>
    Require all granted
    Require not ip 10.252.46.165
</RequireAll>
```

Visitors coming from that address (`10.252.46.165`) will not be able to see the content covered by this directive. If, instead, you have a machine name, rather than an IP address, you can use that.

```
Require not host host.example.com
```

AllowOverride Directive

Description:	Types of directives that are allowed in `.htaccess` files
Syntax:	`AllowOverride All|None|directive-type [directive-type] ...`
Default:	`AllowOverride None (2.3.9 and later), AllowOverride All (2.3.8 and earlier)`
Context:	directory
Status:	Core
Module:	core

When the server finds an `.htaccess` file (as specified by `AccessFileName`), it needs to know which directives declared in that file can override earlier configuration directives.

Only available in <Directory> sections

`AllowOverride` is valid only in `<Directory>` sections specified without regular expressions, not in `<Location>`, `<DirectoryMatch>` or `<Files>` sections.

When this directive is set to `None` and `AllowOverrideList` is set to `None` .htaccess, files are completely ignored. In this case, the server will not even attempt to read `.htaccess` files in the filesystem.

When this directive is set to `All`, then any directive which has the .htaccess Context is allowed in `.htaccess` files.

The *directive-type* can be one of the following groupings of directives.

AuthConfig

Allow use of the authorization directives (`AuthDBMGroupFile`, `AuthDBMUserFile`, `AuthGroupFile`, `AuthName`, `AuthType`, `AuthUserFile`, `Require`, *etc.*).

FileInfo

Allow use of the directives controlling document types (`ErrorDocument`, `ForceType`, `LanguagePriority`, `SetHandler`, `SetInputFilter`, `SetOutputFilter`, and `mod_mime` Add* and Remove* directives), document meta data (`Header`, `RequestHeader`, `SetEnvIf`, `SetEnvIfNoCase`, `BrowserMatch`, `CookieExpires`, `CookieDomain`, `CookieStyle`, `CookieTracking`, `CookieName`), `mod_rewrite` directives (`RewriteEngine`, `RewriteOptions`, `RewriteBase`, `RewriteCond`, `RewriteRule`), `mod_alias` directives (`Redirect`, `RedirectTemp`, `RedirectPermanent`, `RedirectMatch`), and `Action` from `mod_actions`.

Indexes

Allow use of the directives controlling directory indexing (`AddDescription`, `AddIcon`, `AddIconByEncoding`, `AddIconByType`, `DefaultIcon`, `DirectoryIndex`, `FancyIndexing`, `HeaderName`, `IndexIgnore`, `IndexOptions`, `ReadmeName`, *etc.*).

Options Directive

Description:	Configures what features are available in a particular directory
Syntax:	`Options [+|-]option [[+|-]option] ...`
Default:	`Options FollowSymlinks`
Context:	server config, virtual host, directory, .htaccess
Override:	Options
Status:	Core
Module:	core
Compatibility:	The default was changed from All to FollowSymlinks in 2.3.11

The `Options` directive controls which server features are available in a particular directory.

option can be set to `None`, in which case none of the extra features are enabled, or one or more of the following:

All
: All options except for `MultiViews`.

ExecCGI
: Execution of CGI scripts using `mod_cgi` is permitted.

FollowSymLinks
: The server will follow symbolic links in this directory. This is the default setting.

> Even though the server follows the symlink it does *not* change the pathname used to match against `<Directory>` sections.
>
> The `FollowSymLinks` and `SymLinksIfOwnerMatch` `Options` work only in `<Directory>` sections or `.htaccess` files.
>
> Omitting this option should not be considered a security restriction, since symlink testing is subject to race conditions that make it circumventable.

Includes
: Server-side includes provided by `mod_include` are permitted.

IncludesNOEXEC
: Server-side includes are permitted, but the `#exec cmd` and `#exec cgi` are disabled. It is still possible to `#include virtual` CGI scripts from `ScriptAlias`ed directories.

Indexes

The Apache Software Foundation
http://www.apache.org/

HTTP COMPONENTS

HttpCore Overview

HttpCore is a set of low level HTTP transport components that can be used to build custom client and server side HTTP services with a minimal footprint. HttpCore supports two I/O models: blocking I/O model based on the classic Java I/O and non-blocking, event driven I/O model based on Java NIO.

Documentation

1. HttpCore Tutorial (HTML / PDF)
2. Some examples of HttpCore components in action can be found here
3. Project reports
   - HttpCore
   - HttpCore NIO

Standards Compliance

HttpCore components strive to conform to the following specifications endorsed by the Internet Engineering Task Force (IETF) and the internet at large:

- RFC 1945 - Hypertext Transfer Protocol -- HTTP/1.0
- RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1

HttpCore Tutorial

Next

HttpCore Tutorial

Oleg Kalnichevski

4.3.1
