APACHE
HTTP Server

Information Technology Institute

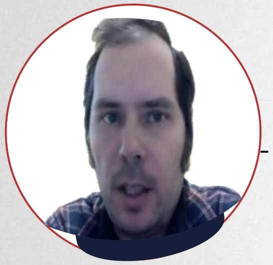
COURSE MATERIALS

You can access the course materials via this link

http://goo.gl/LwyPUe

CONTENTS

HISTORY



1994

Developed by Rob McCool at the National Center for Supercomputing Applications, University of Illinois

1995

Brian Behlendorf and Cliff Skolnick with others continued the development after Rob McCool had stopped the development

1999

Apache Software Foundation (ASF) formed to provide organizational, legal, and financial support for the Apache HTTP Server

“

Apache Software Foundation (ASF) is an American non-profit corporation to support Apache software projects, including the Apache HTTP Server. The ASF was formed from the Apache Group in June 1999

WEB ADDRESS

- Addresses on the Web are expressed with URLs - Uniform Resource Locators - which specify
  - a **protocol** (e.g. http),
  - a **servername** (e.g. www.apache.org),
  - a **URL-path** (e.g. /docs/current/getting-started.html), and
  - possibly a **query string** (e.g. ?arg=value) used to pass additional arguments to the server

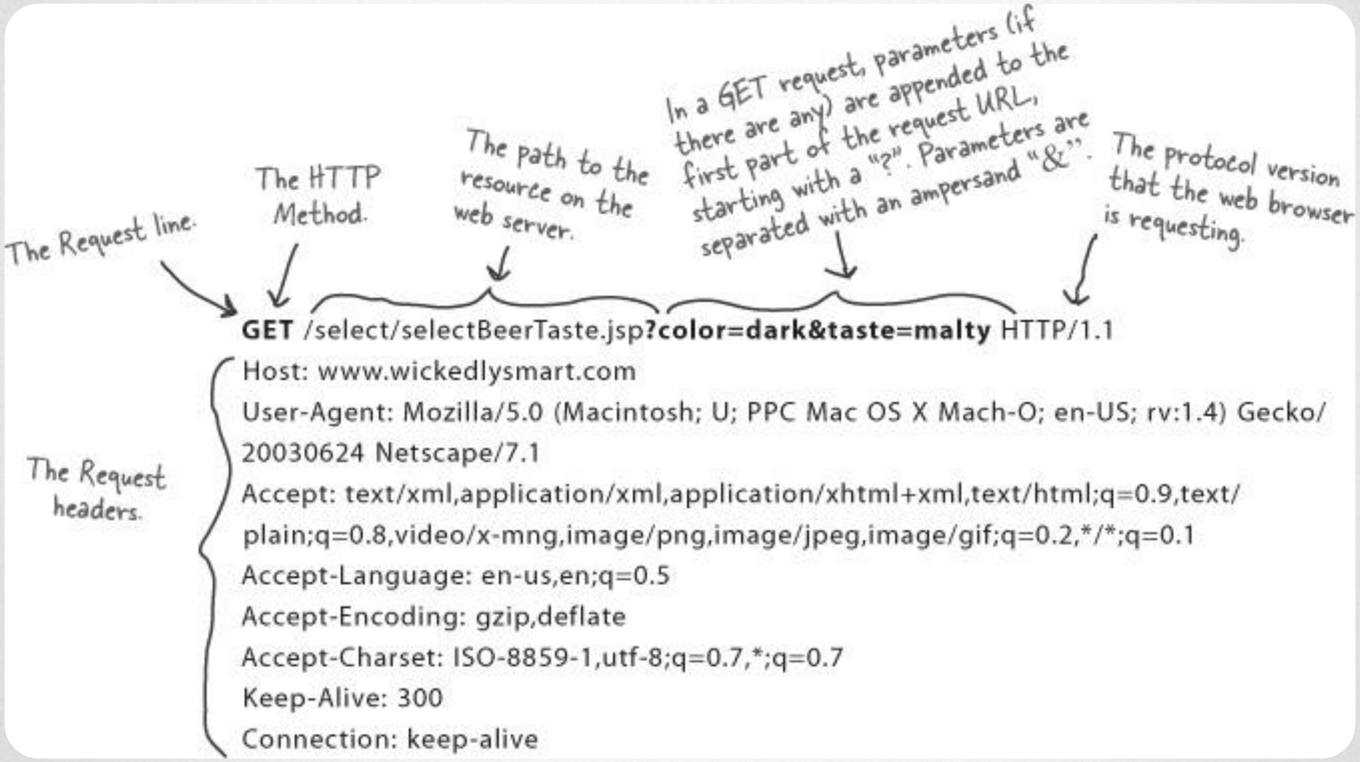
CLIENT/SERVER

- A **client** (e.g., a web browser) connects to a **server** (e.g., your Apache HTTP Server), with the specified protocol, and makes a **request** for a resource using the URL-path.
- The URL-path may represent any number of things on the server. It may be a file (like getting-started.html) or some kind of program file (like index.php).

RESPONSE

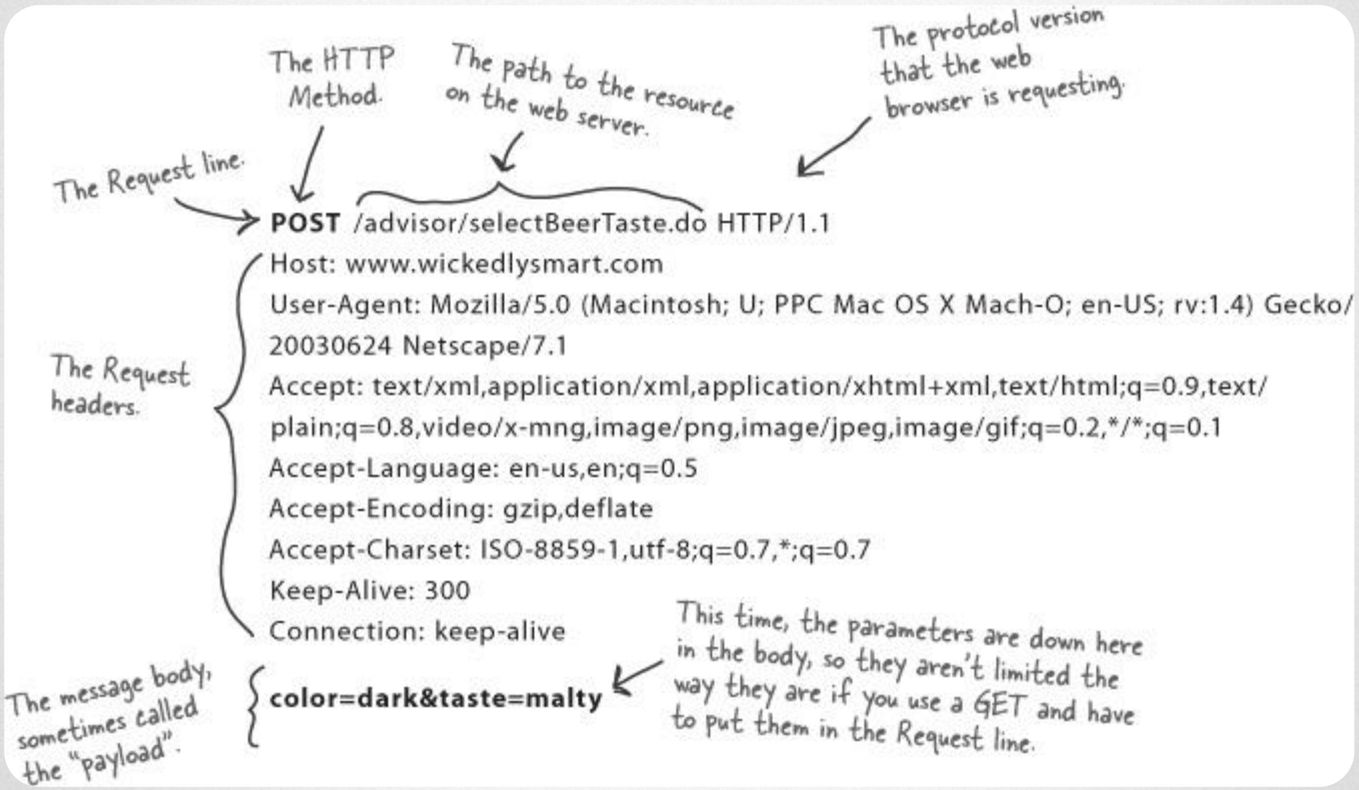
- The server will send a response consisting of
  - a status code and,
  - optionally, a response body.
- The status code indicates whether the request was successful, and, if not, what kind of error condition there was. This tells the client what it should do with the response.

ANATOMY OF GET REQUEST


The Request line.
The HTTP Method.
The path to the resource on the web server.
In a GET request, parameters (if there are any) are appended to the first part of the request URL, starting with a "?". Parameters are separated with an ampersand "&".
The protocol version that the web browser is requesting.
The Request headers.

```
GET /select/selectBeerTaste.jsp?color=dark&taste=malty HTTP/1.1
Host: www.wickedlysmart.com
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en-US; rv:1.4) Gecko/
20030624 Netscape/7.1
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,video/x-mng,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

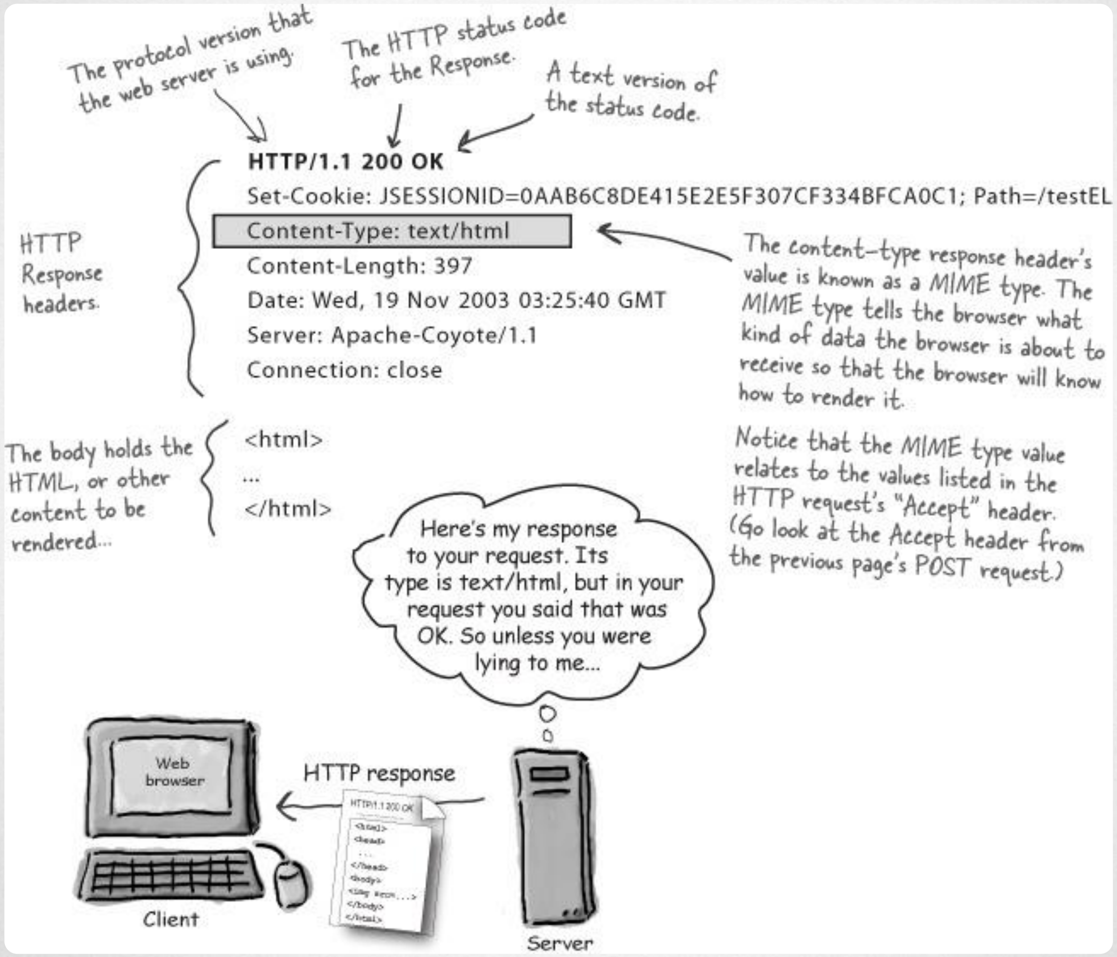
ANATOMY OF POST REQUEST


The HTTP Method.
The path to the resource on the web server.
The protocol version that the web browser is requesting.
The Request line.
The Request headers.
The message body, sometimes called the "payload".
This time, the parameters are down here in the body, so they aren't limited the way they are if you use a GET and have to put them in the Request line.

```
POST /advisor/selectBeerTaste.do HTTP/1.1
Host: www.wickedlysmart.com
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en-US; rv:1.4) Gecko/
20030624 Netscape/7.1
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,video/x-mng,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive

color=dark&taste=malty
```

ANATOMY OF RESPONSE



```
HTTP/1.1 200 OK
Set-Cookie: JSESSIONID=0AAB6C8DE415E2E5F307CF334BFCA0C1; Path=/testEL
Content-Type: text/html
Content-Length: 397
Date: Wed, 19 Nov 2003 03:25:40 GMT
Server: Apache-Coyote/1.1
Connection: close

<html>
...
</html>
```

The protocol version that the web server is using.
The HTTP status code for the Response.
A text version of the status code.
HTTP Response headers.
The content-type response header's value is known as a MIME type. The MIME type tells the browser what kind of data the browser is about to receive so that the browser will know how to render it.
Notice that the MIME type value relates to the values listed in the HTTP request's "Accept" header. (Go look at the Accept header from the previous page's POST request.)
The body holds the HTML, or other content to be rendered...
Here's my response to your request. Its type is text/html, but in your request you said that was OK. So unless you were lying to me...
Web browser
HTTP response
HTTP/1.1 200 OK
Client
Server

HTTP STATUS CODES

- 1xx Informational
- 2xx Success
- 3xx Redirection
- 4xx Client Error
- 5xx Server Error

HTTP STATUS CODES EXAMPLES

- 200 OK
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 500 Internal Server Error

See full list at:

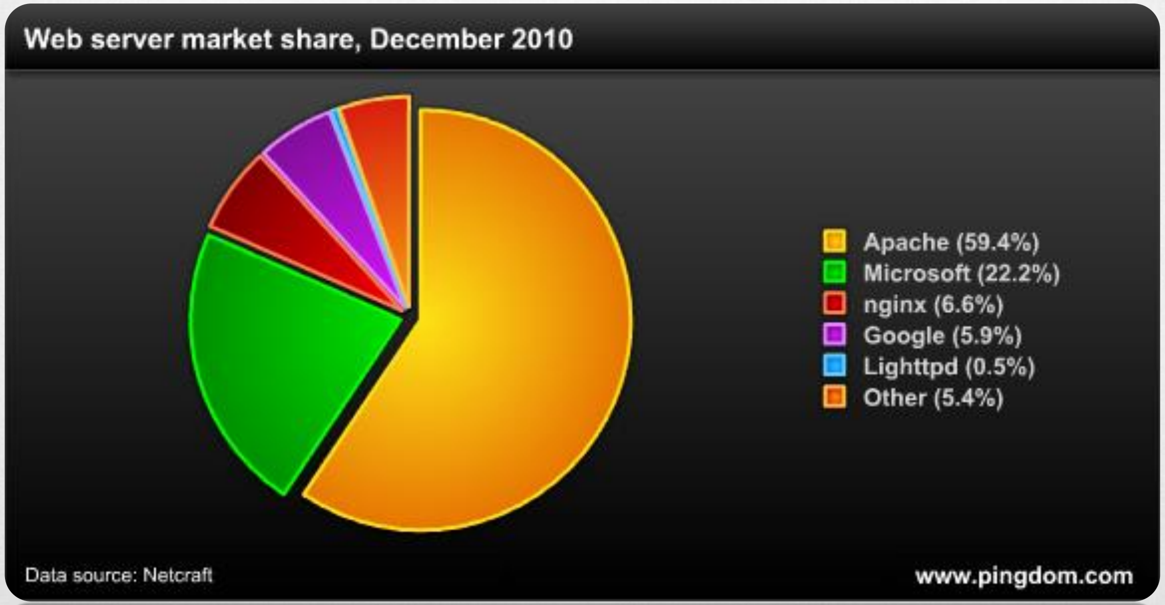
http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

WHAT IS A WEB SERVER?

- This is a computer that's sole purpose is to distribute information that is hosted within its hard drives.
- Depending on the information, it is accessed and distributed differently.
- A detailed example would be standard Web Pages that are accessed via the Internet protocol HTTP on port 80 and distributed back in the same fashion which are stored on a web server.

WHAT IS A WEB SERVER?

- Any computer can be turned into a Web server by installing server software and connecting the machine to the Internet.
- There are many Web server software applications.



APACHE VS IIS

- Apache is free while IIS is packaged with Windows.
- IIS only runs on Windows while Apache can run on almost any OS including UNIX, Apple's OS X, and on most Linux Distributions.
- IIS has a dedicated staff to answer most problems while support for Apache comes from the community itself.
- IIS is optimized for Windows because they are from the same company.
- The Windows OS is prone to security risks.

CONFIGURATION FILES

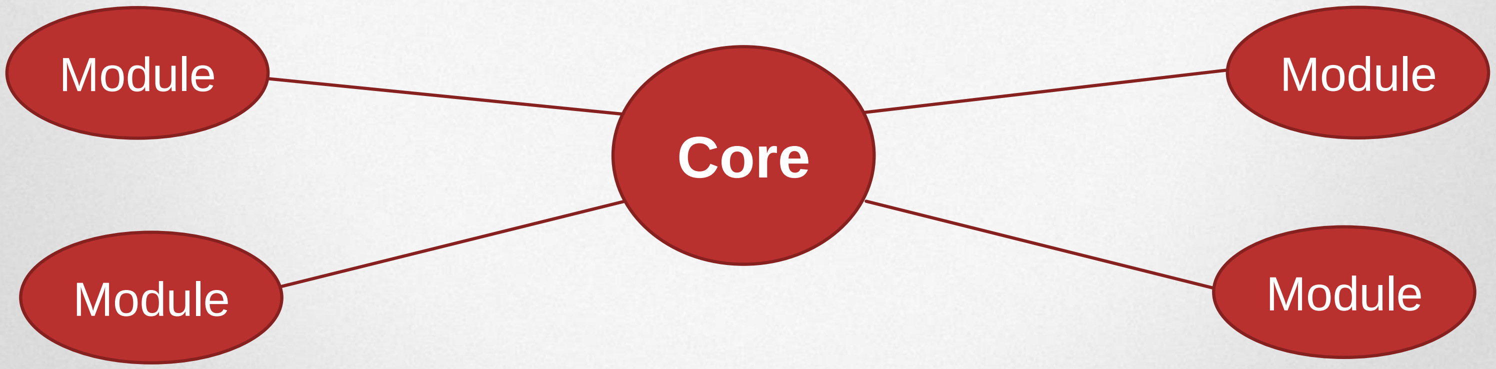
- The Apache HTTP Server is configured via simple text files.
- The default configuration file is usually called `httpd.conf`.
- The configuration is frequently broken into multiple smaller files, for ease of management. These files are loaded via the Include directive.

DIRECTIVES

- The server is configured by placing configuration directives in configuration files.
- A **directive** is a keyword followed by one or more arguments that set its value.
- In addition to the main configuration files, certain directives may go in `.htaccess` files located in the content directories. `.htaccess` files are primarily for people who do not have access to the main server configuration file(s)

MODULES

- Apache has always accommodated a wide variety of environments through its modular design.
- This design allows the web-master to choose which features will be included in the server by selecting which modules to load either at compile-time or at run-time.



INSTALLATION

- For Ubuntu/Debian :

```
$ Sudo apt-get update
$ Sudo apt-get install tasksel
$ Sudo tasksel install lamp-server
```

- For CentOS/Red Hat Distros:

```
# yum install httpd
# service httpd start
```

MAIN CONFIGURATION FILE

- Ubuntu:

  `/etc/apache2/apache2.conf`

- CentOS:

  `/etc/httpd/conf/httpd.conf`

PUBLIC WEB FILES / LOG FILES

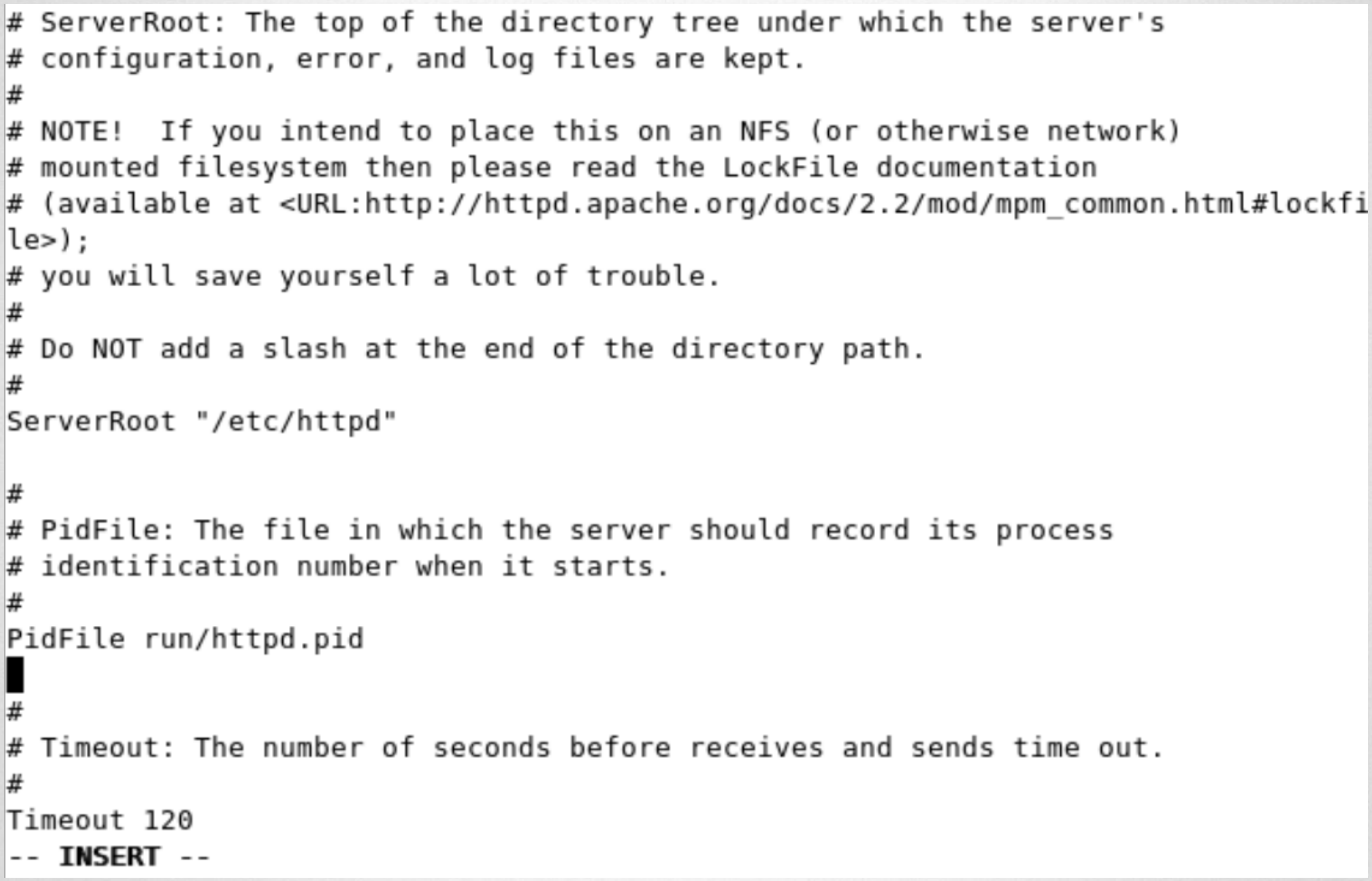
- Ubuntu:

```
/var/www/html
/var/log/apache2
```

- CentOS:

```
/var/www/html
/var/log/httpd
```

MAIN CONFIGURATION FILE



```
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
# NOTE!  If you intend to place this on an NFS (or otherwise network)
# mounted filesystem then please read the LockFile documentation
# (available at <URL:http://httpd.apache.org/docs/2.2/mod/mpm_common.html#lockfi
le>);
# you will save yourself a lot of trouble.
#
# Do NOT add a slash at the end of the directory path.
#
ServerRoot "/etc/httpd"


#
# PidFile: The file in which the server should record its process
# identification number when it starts.
#
PidFile run/httpd.pid


#
# Timeout: The number of seconds before receives and sends time out.
#
Timeout 120
-- INSERT --
```

DIRECTIVES EXAPMLES

- `ServerRoot` directive

The path to the server's configuration

- `PidFile` directive

The process identification number for the httpd registered at starting the server

- `ServerName` directive

This is where you declare the name of your website

- `DocumentRoot` directive

This is where your web documents (html files, images etc) should be located.

- `Listen` directive

The Listen directive instructs Apache httpd to listen to **only specific IP addresses** or **ports**

DIRECTIVES DESCRIPTION

Description:	A brief description of the purpose of the directive.
Syntax:	This indicates the format of the directive as it would appear in a configuration file
Status:	This indicates how tightly bound into the Apache Web server the directive: **Core**: means it is part of the innermost portions of the Apache Web server, and is always available. **MPM**: is provided by a Multi-Processing Module. **Base**: supported by one of the standard Apache modules which is compiled into the server by default, and is therefore normally available. **Extension**: is provided by one of the modules included with the Apache server kit, but the module isn't normally compiled into the server.
Module:	This quite simply lists the name of the source module which defines the directive.

DIRECTIVES DESCRIPTION

Context:	This indicates where in the server's configuration files the directive is legal. **Server config**: This means that the directive may be used in the server configuration files (e.g., httpd.conf), but not within any <VirtualHost> or <Directory> containers. It is not allowed in .htaccess files at all. **Virtual host** :This context means that the directive may appear inside <VirtualHost> containers in the server configuration files. **Directory** :A directive marked as being valid in this context may be used inside <Directory>, <Location>, <Files>, and <Proxy> containers in the server configuration files, subject to the restrictions outlined in Configuration Sections. **.htaccess**: If a directive is valid in this context, it means that it can appear inside per-directory .htaccess files. It may not be processed, though depending upon the overrides currently active.

AllowOverride

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.

KeepAlive

Description:	Enables HTTP persistent connections
Syntax:	KeepAlive On|Off
Default:	KeepAlive On
Context:	server config, virtual host
Status:	Core
Module:	core

The Keep-Alive extension to HTTP/1.0 and the persistent connection feature of HTTP/1.1 provide long-lived HTTP sessions which allow multiple requests to be sent over the same TCP connection. In some cases this has been shown to result in an almost 50% speedup in latency times for HTML documents with many images. To enable Keep-Alive connections, set KeepAlive On.

MaxKeepAliveRequests

Description:	Number of requests allowed on a persistent connection
Syntax:	MaxKeepAliveRequests number
Default:	MaxKeepAliveRequests 100
Context:	server config, virtual host
Status:	Core
Module:	core

The MaxKeepAliveRequests directive limits the number of requests allowed per connection when **KeepAlive** is **on**. If it is set to 0, unlimited requests will be allowed. We recommend that this setting be kept to a high value for maximum server performance.

ErrorLog

Description:	Location where the server will log errors
Syntax:	ErrorLog file-path|syslog[:facility]
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

Directory

Description:	Enclose a group of directives that apply only to the named file-system directory, sub-directories, and their contents.
Syntax:	<Directory directory-path> ... </Directory>
Context:	server config, virtual host
Status:	Core
Module:	core

<Directory> and </Directory> are used to enclose a group of directives that will apply only to the named directory, sub-directories of that directory, and the files within the respective directories. Any directive that is allowed in a directory context may be used.

DirectoryMatch

Description:	Enclose directives that apply to the contents of file-system directories matching a regular expression.
Syntax:	<DirectoryMatch regex> ... </DirectoryMatch>
Context:	server config, virtual host
Status:	Core
Module:	core

<DirectoryMatch> and </DirectoryMatch> are used to enclose a group of directives which will apply only to the named directory (and the files within), the same as <Directory>. However, it takes as an argument a regular expression

Files

Description:	Contains directives that apply to matched filenames
Syntax:	<Files filename> ... </Files>
Context:	server config, virtual host, directory, .htaccess
Override:	All
Status:	Core
Module:	core

The <Files> directive limits the scope of the enclosed directives by filename. It is comparable to the <Directory> and <Location> directives. It should be matched with a </Files> directive. The directives given within this section will be applied to any object with a basename (last component of filename) matching the specified filename

FilesMatch

Description:	Contains directives that apply to regular-expression matched filenames
Syntax:	<FilesMatch regex> ... </FilesMatch>
Context:	server config, virtual host, directory, .htaccess
Override:	All
Status:	Core
Module:	core

The <FilesMatch> directive limits the scope of the enclosed directives by filename, just as the <Files> directive does. However, it accepts a regular expression.

Allow

Description:	Controls which hosts can access an area of the server
Syntax:	Allow from all|host|env=[!]env-variable [host|env=[!]env-variable] ...
Context:	directory, .htaccess
Override:	Limit
Status:	Extension
Module:	mod_access_compat

The Allow directive affects which hosts can access an area of the server. Access can be controlled by hostname, IP address, IP address range, or by other characteristics of the client request captured in environment variables.

Deny

Description:	Controls which hosts are denied access to the server
Syntax:	Deny from all|host|env=[!]env-variable [host|env=[!]env-variable] ...
Context:	directory, .htaccess
Override:	Limit
Status:	Extension
Module:	mod_access_compat

This directive allows access to the server to be restricted based on hostname, IP address, or environment variables. The arguments for the Deny directive are identical to the arguments for the Allow directive.

Order

Description:	Controls the default access state and the order in which Allow and Deny are evaluated.
Syntax:	Order ordering
Context:	directory, .htaccess
Override:	Limit
Status:	Extension
Module:	mod_access_compat

The Order directive, along with the Allow and Deny directives, controls a three-pass access control system. The first pass processes either all Allow or all Deny directives, as specified by the Order directive. The second pass parses the rest of the directives (Deny or Allow). The third pass applies to all requests which do not match either of the first two.

CacheEnable

Description:	Enable caching of specified URLs using a specified storage manager
Syntax:	CacheEnable cache_type [url-string]
Context:	server config, virtual host, directory, .htaccess
Override:	Extension
Status:	mod_cache
Module:	Enable caching of specified URLs using a specified storage manager

The CacheEnable directive instructs mod_cache to cache urls at or below url-string. The cache storage manager is specified with the cache_type argument. The CacheEnable directive can alternatively be placed inside either <Location> or <LocationMatch> sections to indicate the content is cacheable.

.htaccess

- `.htaccess` stands for *hypertext access*. This is the default name of the Apache **directory-level configuration file**.
- One of the most common uses is to require user authentication in order to serve certain web pages.
- Also you can use it to define rewrite rules and rewrite conditions.
- You need to configure apache using `AllowOverride` directive to allow/deny the usage of `.htaccess`

.htaccess FOR AUTHENTICATION

```
#.htaccess content
AuthType Basic
AuthName "Restricted web page"
AuthUserFile "/var/www/.htpasswd"
require valid-user
```

- To create .htpasswd use this command

```
htpasswd -c .htpasswd username
```

.htaccess FOR AUTHENTICATION

- **AuthType Basic** defines the type of authentication.
- **Basic** means there is no encryption and the password hash is sent as clear text. This is one of the major reasons why .htaccess cannot be considered for protection of confidential user data.
- **"Restricted web page"** is a window title string. When someone tries to access an .htaccess-protected page, a username & password window will pop in the web browser. This window will bear a title - this is the AuthName. It can be anything you like.
- **AuthUserFile** /var/www/.htpasswd defines the path to a file where user credentials are stored. This file does not exist, but we will create it soon.
- **require valid-user** indicates only successful authentication attempts will result in the loading of the page.