APACHE

HTTP Server

COURSE MATERIALS

You can access the course materials via this link

http://goo.gl/LwyPUe

CONTENTS

- History
- Clients, servers and URLs
- Anatomy of request and response
- HTTP status codes
- Web servers
- Apache HTTP Server components
- Documentation explained
- Directives examples
- .htacess
- Modules
- Virtual hosts

MODULES

- You can always list the modules currently used by the server. The command below will display only the modules compiled into Apache.

```
$ httpd -l
```

- To list all modules, both static and shared

```
$ httpd  -m
```

- To enable module (Ubuntu only!)

```
$ sudo a2enmod rewrite
```

LoadModule

Description:	Links in the object file or library, and adds to the list of active modules
Syntax:	LoadModule module filename
Context:	server config
Override:	Extension
Status:	mod_so
Module:	Links in the object file or library, and adds to the list of active modules

The LoadModule directive links in the object file or library filename and adds the module structure named module to the list of active modules. Module is the name of the external variable of type module in the file, and is listed as the Module Identifier in the module documentation

IfModule

Description:	Encloses directives that are processed conditional on the presence or absence of a specific module
Syntax:	<IfModule [!]module-file|module-identifier> ... </IfModule>
Context:	server config, virtual host, directory, .htaccess
Override:	All
Status:	Core
Module:	core

The <IfModule test>...</IfModule> section is used to mark directives that are conditional on the presence of a specific module. The directives within an <IfModule> section are only processed if the test is true. If test is false, everything between the start and end markers is ignored.

MULTI-PROCESSING MODULES (MPMS)

- Apache 2.x supports pluggable concurrency models, called Multi-Processing Modules (MPMs). When building Apache, you must choose an MPM to use.
- The choice of MPM can affect the speed and scalability of the httpd:
  - The worker MPM
  - The event MPM
  - The prefork MPM

PREFORK MODULE

- This Multi-Processing Module (MPM) implements a **non-threaded**, pre-forking web server.
- Each server process may answer incoming requests, and a parent process manages the size of the server pool. It is appropriate for sites that need to avoid threading for compatibility with non-thread-safe libraries.
- It is also the best MPM for isolating each request, so that a problem with a single request will not affect any other.

PREFORK MODULE

```
<IfModule mpm_prefork_module>
    StartServers             5
    MinSpareServers          5
    MaxSpareServers         10
    MaxClients             150
</IfModule>
```

WORKER MODULE

- This Multi-Processing Module (MPM) implements a hybrid multi-process **multi-threaded** server.
- By using threads to serve requests, it is able to serve a large number of requests with fewer system resources than a process-based server.
- It retains much of the stability of a process-based server by keeping multiple processes available, each with many threads.

WORKER MODULE

```
<IfModule mpm_worker_module>
    StartServers          2
    MinSpareThreads      25
    MaxSpareThreads      75
    ThreadsPerChild      25
    MaxRequestWorkers    150
</IfModule>
```

EVENT MODULE

- The event MPM is threaded like the Worker MPM, but is designed to allow more requests to be served.
- Run-time configuration directives are identical to those provided by worker, with the only addition of the `AsyncRequestWorkerFactor`.
- KeepAlive may lead to situations where all workers are tied up and no worker thread is available to handle new work.
- To mitigate this problem, the event MPM does two things:
  - it limits the number of connections accepted per process, depending on the number of idle request workers;
  - if all workers are busy, it will close connections in keep-alive state even if the keep-alive timeout has not expired.

EVENT MODULE

```
<IfModule mpm_event_module>
    StartServers             2
    MinSpareThreads          25
    MaxSpareThreads          75
    ThreadsPerChild          25
    MaxRequestWorkers        150
    AsyncRequestWorkerFactor  2
</IfModule>
```

REWRITE MODULE

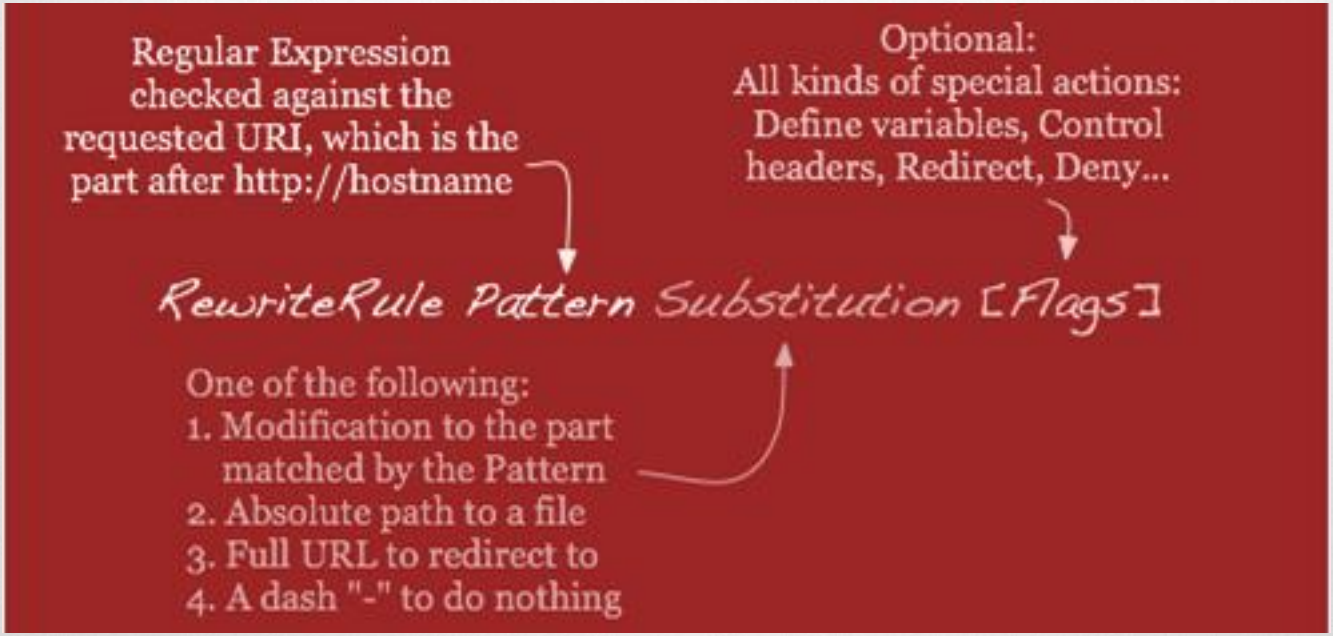
- The Apache module `mod_rewrite` is a very powerful and sophisticated module which provides a way to do URL manipulations.
- With it, you can do nearly all types of URL rewriting that you may need. It is, however, somewhat complex, and may be intimidating to the beginner.
- You need to know:
  - How to construct regular expression:
  - What is the `RewriteRule`?
  - What is the `RewriteCond`?

REGEX VOCABULARY

Character	Meaning
.	Matches any single character
+	Repeats the previous match one or more times
*	Repeats the previous match zero or more times.
?	Makes the match optional.
^	Called an anchor, matches the beginning of the string
$	The other anchor, this matches the end of the string.
( )	Groups several characters into a single unit, and captures a match for use in a backreference.
[ ]	A character class - matches one of the characters
[^]	Negative character class - matches any character not specified
!	can be used before a regular expression to negate it

RewriteRule BASICS

- A `RewriteRule` consists of **three** arguments separated by spaces. The arguments are
  1. **Pattern**: which incoming URLs should be affected by the rule;
  2. **Substitution**: where should the matching requests be sent;
  3. **[flags]:** options affecting the rewritten request.



RewriteRule BASICS

- The Substitution can itself be one of **three** things:
- **A full filesystem path to a resource**

```
RewriteRule "^/games" "/usr/local/games/web"
```

This maps a request to an arbitrary location on your filesystem, much like the Alias directive.

- **A web-path to a resource**

```
RewriteRule "^/foo$" "/bar"
```

If `DocumentRoot` is set to `/var/www/html/`, then this directive would map requests for http://example.com/foo to the path `/var/www/html/bar`.

RewriteRule BASICS

- **An absolute URL**

```
RewriteRule "^/product/view$"
"http://site2.example.com/seeproduct.html"
[R]
```

This tells the client to make a new request for the specified URL.

- The Substitution can also contain back-references to parts of the incoming URL-path matched by the Pattern. Consider the following:

```
RewriteRule "^/product/(.*)/view$"
"/var/web/productdb/$1"
```

Rewrite FLAGS

- The **behavior** of a `RewriteRule` can be modified by the application of one or more **flags** to the end of the rule.
- [NC] flag causes the `RewriteRule` to be matched in a case-insensitive manner.

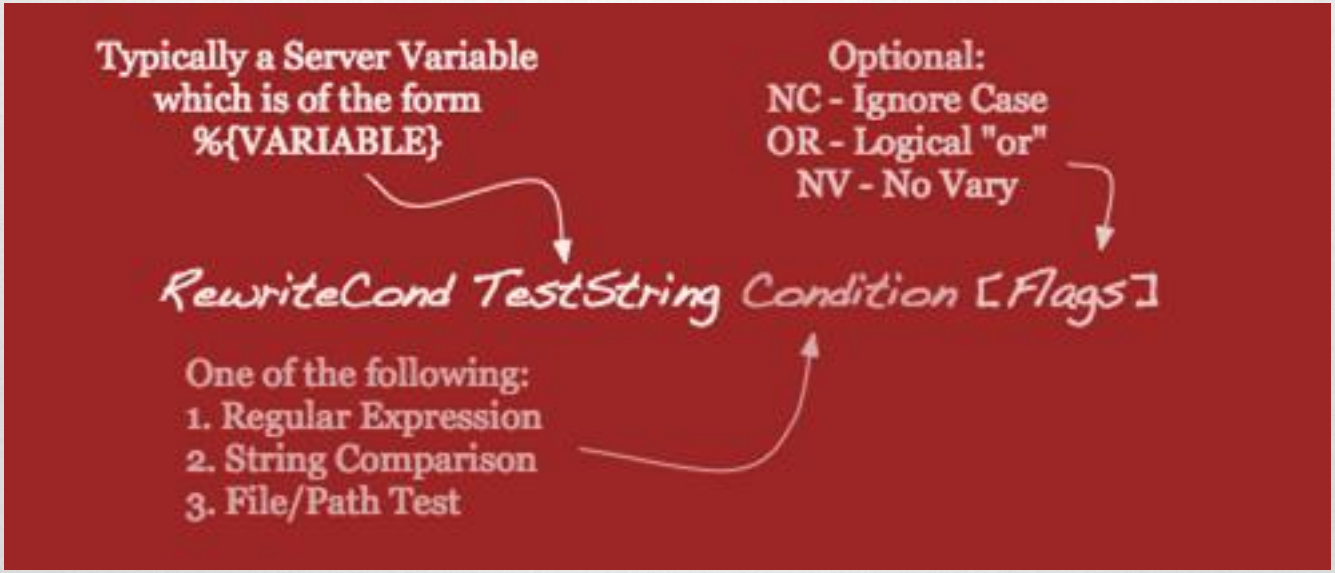
```
RewriteRule "^puppy.html" "smalldog.html" [NC]
```

- [R] flag causes a HTTP redirect to be issued to the browser. If a fully-qualified URL is specified (that is, including http://servername/) then a redirect will be issued to that location. Otherwise, the current protocol, servername, and port number will be used to generate the URL sent with the redirect
- For more flags, see

http://httpd.apache.org/docs/2.4/rewrite/flags.html

Rewrite CONDITIONS

- One or more `RewriteCond` directives can be used to restrict the types of requests that will be subject to the following `RewriteRule`.
  - The first argument is a variable describing a characteristic of the request,
  - the second argument is a regex that must match the variable,
  - and a third optional argument is a list of flags that modify how the match is evaluated.



Rewrite CONDITIONS EXAMPLES

- For example, to send all requests from a particular IP range to a different server, you could use:

```
RewriteCond "%{REMOTE_ADDR}" "^10\.2\."
RewriteRule "(.*)" "http://intranet.example.com$1"
```

- When more than one `RewriteCond` is specified, **they must all match** for the `RewriteRule` to be applied.
- For example, to deny requests that contain the word "hack" in their query string, unless they also contain a cookie containing the word "go", you could use:

```
RewriteCond "%{QUERY_STRING}" "hack"
RewriteCond "%{HTTP_COOKIE}" "!go"
RewriteRule "." "-" [F]
```

Rewrite CONDITIONS EXAMPLES

- Matches in the regular expressions contained in the `RewriteConds` can be used as part of the Substitution in the `RewriteRule` using the variables `%1, %2`, etc.
- For example, this will direct the request to a different directory depending on the hostname used to access the site:

```
RewriteCond "%{HTTP_HOST}" "(.*)"
RewriteRule "^/(.*)" "/sites/%1/$1"
```

If the request was for http://example.com/foo/bar, then %1 would contain example.com and $1 would contain foo/bar.

VIRTUAL HOST

- The term Virtual Host refers to the practice of running more than one web site (such as company1.example.com and company2.example.com) on a single machine.
- Virtual hosts can be "IP-based", meaning that you have a different IP address for every web site, or "name-based", meaning that you have multiple names running on each IP address.
- The fact that they are running on the same physical server is not apparent to the end user.

IP-BASED

- the server **must have a different IP address/port combination for each IP-based virtual host**.

```
<VirtualHost 172.20.30.40:80>
    ServerAdmin webmaster@www1.example.com
    DocumentRoot "/www/vhosts/www1"
    ServerName www1.example.com
    ErrorLog "/www/logs/www1/error_log"
    CustomLog "/www/logs/www1/access_log" combined
</VirtualHost>
<VirtualHost 172.20.30.50:80>
DocumentRoot "/www/vhosts/www2"
    ServerName www2.example.org
    ErrorLog "/www/logs/www2/error_log"
    CustomLog "/www/logs/www2/access_log" combined
</VirtualHost>
```

NAME-BASED

```
<VirtualHost *:80>
    # This first-listed virtual host is also the
default for *:80
    ServerName www.example.com
    ServerAlias example.com
    DocumentRoot "/www/domain"
</VirtualHost>

<VirtualHost *:80>
    ServerName other.example.com
    DocumentRoot "/www/otherdomain"
</VirtualHost>
```

VIRTUAL HOST

- Any request that doesn't match an existing `<VirtualHost>` is handled by **the global server configuration**, regardless of the hostname or ServerName.
- To be accessible by more than one name. use `ServerAlias` directive, placed inside the `<VirtualHost>` section.

```
ServerAlias example.com *.example.com
```