Module Configuration

Rewrite Module

Perform URL rewriting. Allows to get rid URLs containing multiple para meters and transform to useful informatives new ones.

The mechanism consist of rewriting the URI of the client request after it is received and before serving the file. The rewritted URI is matched against the location blocks in order to find the configuration that should be applied to the request.

Rewriting is performed by `rewrite` directive using regular epressions.

Metacharacters

- ^ Beginning: Entity after must be found at the beginning
  - ^h matches hello, hh, h (anything beginning with h)
- $ End: e$
  - matching sample, e, file (anything ending with e)
- . (dot) Matches any character:
  - hell. matches hello, hellx, hell!. Not matches hell
- [ ] Set: Matches any character withing the specified set
  - [a - z] range
  - [abcd] set
  - [a-z0-9]
  - example hell[a-y123-]
- ^ Negate Set: Matches any character that is not withing the specified set
- | Alternation: Matches the entity placed either before or after |
  - hello|welcome matches hello, welcome, helloes, awelcome
- ( ) Grouping
- \ Escape: allows to scape special characters. Hello. matches Hello., Hello. How are You?, Hi! Hello...
  - not matching Hello,Hello!
-
  - 0 or More Times: The entity preceding * must be found 0 or more times
  - he*llo matches hllo,hello,heeeeeeeello
-
  - 1 or more times
- ? 0 or 1
- {x} x times
- {x,y} x to y times
- {x,} at least x times

Internal Requests

- External requests directly originate from the client. The URI is the nmatched agains possible location blocks
- Inetrnal requests are triggered by Nginx via specific directives.
  - error_page, index, rewrite, try_files, add_before_body, add_after_body directives creates internal requests

Internal Requests Types

- Internal redirects: Nginx redirects the client requests internally, the URI is changed.
- Sub-requests: Additional requests to generat content complementary to the main request.

error_page

error_page directive allow to define sv behavior when a specific error code occurs.

```
server {
    server_name website.com;
    error_page 403 /errors/forbidden.html;
    error_page 404 /errors/not_found.html;
}
```

Conditional Structure

```
server {
    if ($requests_method = POST) {
        ...
    }
}
```

Operators

- None: if ($string): is true if specified variable or data is not equial to an empty string or a string starting with 0 character.
- =, !=
- ~ case sensitive
- ~* case insensitive
- !~, !* negate matching
- -f !-f
    - if (-f $request_filename) tests the existence of a specified file
- -d, !-d similar -f but directory
- -e !-e, similar -f but file, directory or symbolic link
- -x !-x executable

Directives

rewrite

- server, location, if
- allow to rewrite the URI of the curent request
- `rewrite regexp replacement [flag]`
- regexp: the regular expression that the URI should match in order for the replacement to apply
- Flag:

- last
- break
- redirect

break

prevent rewrite directives

return

interrupts the processing of the request and returns the specified htt[status code or specified test

```
return code | text
```

set

initializes or redefines a variable. Note that some variables are reasd only.

```
set variable value
```

when a variable that has not yet been initalized nginx will issue log messages if declared `unitialized_variable_warn on;` before.

rewrite_log

if set to on, nginx will issue log messags for every operation performed by the rewrite engine at the notice error level.

<<<<<<< HEAD

Common rewrite rules

Basic rewrite rules to stisfy basics of dynamic websites that wish to beautify page links.
http://website.com/ was omitted at the beggining of URIs.

Search

- input: /search/some-search-keywords
- rewritten: /search.php?q=some-search-keywords
- `rewrite ^/search/(.*)$ /search.php?q=$1?;`

Multiple Parameters

- Input: /index.php/param1/param2/param3
- Rewritten: /index.php?p1=param1&p2=param2&p3=param3
-

Wiki-like

- /wiki/some_keyword
- /wiki/index.php?title=some_keyword
- rewrite ^/wiki/(*.)$ /wiki/index.php?title=$1?;

News Website Article

- /33526/us-economy
- /article.php?id=33526
- rewrite ^/([0-9]+)/.*$ /article.php?id=$1;

SSI Module

SSI (Server Side Includes) is a sort of server-side programming language interpreted by Nginx. Created in order to render web pages dynamically, from static .html files with client-side scripts to complex pages with server-processes indstrucitons.

Module Directives

- ssi: Enables parsing files for SSI commands. Nginx only parses the files corresonding to the MIME types selected with the ssi_types directive. Syntax on/off, default off; `ssi on;`
- ssi_types: Define MIME types that should be eligible for SSI parsing. `ssi_types type1 [type2] [type3]` default text/html;
- ssi_client_errors: Enabling this option to silence Nginx error ssi messages.
- ssi_value_length: Maximun length accepted by Nginx in characters
- ssi_ignore_recycled_buffers
- ssi_min_file_chunk
- ssi_last_modified

=======

Common rewrite rules

SSI commands

The principle is simple. Disgn regular HTML pages and insert inside SSI commands that looks like regular HTML comments

```
<!--# command param1="value" param2="value2 -->
<!--# include file="header.html" -->
```

This command generates an HTTP sub-request to be processed by Nginx. The body of the response is inserted instead the command itself.

```
<!--# include virtual="/sources/header.php?id=123" -->
```

This sends a sub-request to the server. The directive wait="yes" is using to specify that Nginx should wait for the completion of the request before moving on to other includes.

If the result of the include is empty Nginx inserts 404 or 500 error page.

Variables

Nginx SSI module offers the option of working with variables. Displaying a variable can be done with echo command

```
<!-- var="variable_name" -->
```

- var: Name of the variable to display.
- default: A string to be displayed in case the var is empty
- encoding: encoding method for the string

the command set, insthead of echo, write the variable value

```
<!--# echo var="MYVAR"  --> (none)
<!--# set var=MYNAME" value="Gian" -->
<!--# set var="MYVAR" value="hello $MYNAME" -->
<!--# echo var="MYVAR"  --> (hello Gian)
```

```
<!-- if expr="expression1" -->
...
<!-- elif expr="expression2" -->
...
<!-- else -->
...
<!-- endif -->
```

```
<!--# if expr="variable = some_value" -->
<!--# if expr="variable = /pattern/" -->
```

Access and Logging Module

Allows to config the way visitors access your website and the way your server logs requests.

index

- http, server, location

```
index index.php index.html index.htm
```

Log

Controls the behavior of Nginx regarding the access logs. It is a key module for system administrators, allows to analyzing the runtime behavior of web apps.

access_log

- http, server, location, if, limit_except
- defines
  - acess log file path
  - format of entries in the access log
- `acess_log path [format [buffer=size] | off`
- off to disable access loging at the current level
- format argument corresponds to a tepmlate declared with log_format directive

log_format

- http, server, location
- template to be utilized by access_log directive
- `log_format template_name format_string;`
- default template is called **combined**
- default: `log_format combined '$remote_addr - $remote_user [$time_local] '"$request: $status $body_bytes_sent '"$http_referer" "$http_user_agent"';`
- `log_format simple '$remote_addr $request`

  open_log_file_cache

  - http, server, location
  - configure cache for log file descriptors

  Log Module Variables

  - connection
  - pipe
  - time_local
  - msec
  - request_time
  - status
  - bytes_sent

- body_bytes_sent
- apache_bytes_sent
- request_length

Limits and Restrictions

Regulates access to the coduments of your websites. Restricts the access and require users to:

- authenticate
- match a set of rules

Auth_basic module

enables basic authentication funcitonality. Username and password

```
location /admin/ {
  auth_basic "Admin control panel"; #Variables supported
  auth_basic_user_file access/password_file
}

auth_basic can be set to either off or a text message, referred to as
authentication challenge or realm. Is displayed by browsers in a
username/password box when client attemps to access.

auth_basic_user_file defines the path of the password file relative to the
directory of the configuration file.

```txt
password file syntax"
username:[{SCHEME}]password[:comment].

username: plain text user name
SCHEME: optionally. PAssword hashing method
  - PLAIN
  - SHA (SHA-1 Hashing)
  - SSHA SaltedSHA-1 hashing
password: password
comment: plaint text comment for own use
```

Access Module

- allow and deny directives
- let allow or deny access to a resource for a specific IP address or IP address range.
- `allow/deny IP | CIDR | unix: | all`
- unix represent domain sockets

```
location {
  allow 127.0.0.1;
```
```

```
  allow unix;
  deny all; #deny all other ip addresses
}
```

Limit Connections

Allows to define maximun number of simultaneous connections to the server for a specific zone.

- limit_conn_zone $variable zone=name:size;
- $variable is the mechanism to differentiate one client from another. Typically $binary_remote_Addr
- name is an aribtrary name given to the zone
- size: max size to the table storing session states

```
# Examples
limit_conn_zone $binary_remote_addr zone=myzone:10m;
limit_conn zone_name connection_limit;
```

```
location /downloads/ {
  limit_conn myzone 1;
}
```

Limit request

allows to limit the number of requests for a defined zone.

```
limit_req_zone $variable zone=name:max_memory_size rate=rate;
```

the parameters are identical to limit connection except for the traillling rate, expressd in requests per second or per minute r/s, r/60s.

```
limit_req zone=name burst=burst [nodelay];
```

burst parameters defines maximun possible brsts of requests.

Auth Request

auth_requests allows to deny acccess to a resource based on the result of a sub-request. Nginx calls the URI to specify via auth_request directive. If sub request returns a 2xx response code access is allowed.

```
location /downloads/ {
  # if the script below returns a 200 status code
  # the download is authorized
```

```
    auth_request /autorization.php
  }
```

the auth_request_set allows to set a variable after the sub-request is executed.

Content and encoding

Provides functionalities having an effect of the contents served to the client.

Empty GIF

Serves a 1x1 transparent GIF image from the memory.

```
location = /empty.gif {
  empty_gif;
}
```

MP4

Enable useful funtionality when serving a MP4 file. It parses a special argument `start` which indicates the offset of the section that the client wishes to download or pseudo-stream. The uri to access the file is video.mp4?start=XXX.

To utilize this feautre insert the `mp4 directive` in the location of your choice.

```
location ~* /.mp4 {
  mp4;
}
```

Addition

Allows to add content before or after the body of the HTTP response.

- add_before_body file_uri;
- add_adter_body file_uri;

Substitution

Allows to search and replace text directly from the response body. `sub_filter searched_Text replacement_text;`.

Gzip filter

Allows to compress the response body with gzip before sending to the client.

```

```

```
gzip_buffers amount size;
gzip_comp_level 1;
gzip_disable;
gzip_http_version 1.1;
gzip_min_length 0;
gzip_proxied off;
gzip_types mime_type1 [mime_type2..];
gzip_vary off;
gzip_window MAX_WBITS;
gzip_hash MAX_MEM_LEVEL;
postpone_gzipping 0; # minimum data threshold to be reached before comp
gzip_no_buffer off;
```

Image filter

provides image processing functionalities through the GD Graphics Library (gdlib). Works on location block that filter image files only, such as

```
location ~* \.(png|jpg|gif)$ {...}.
```

- image_filter
  - test, size, resize width height, crop width height, rotate.
  - image_filter resize 200 100;
- image_filter_buffed
- image_filter_jpeg_quality
- image_filter_transaprency
- image_filter_sharpen
- image_filter_interlace

About visitors

Provides extra functionality that heps you find oyt more information about the visitors by parsing client requets headers for browser name and version.

Browser

The browser module parses user-agent HTTP header of the client request in order to establish values for the variables that can be emplyed later in the config

- $modern_browser: if client browser is identified as being a modern web browser/
- $ancient_browser
- $msie: is set to 1 if client is using Internet Explorer

```
modern_browser opera 10.0;
```

Map

Map allows to create values depending on a variable

```
map $uri $variable {
  /page.html 0;
  /contact.html 1;
  /index.html 2;
  default 0;
}
rewrite ^ /index.php?page=$variable
```

can only be inserted within the `http` block. The last instruction rewrites the URL accordnly.

Geo

Provide functionality that affects a variable based on the client data. The syntax is slightly different in that you are allowed to specify IPv4 and IPv6 address ranges.

```
geo $variable {
  default unknown;
  127.0.0.1 local;
  123.12.3.0/24 uk;
  92.43.0.0/16 fr;
}
```

directives

- delete
- default
- include
- proxy: defines a subnet of trusted addresses
- proxy_recursive
- ranges `127.0.0.1-127.0.0.255 LOCAL`

GeoIP

Similar to Geo. Provides accurate graphic similarities with the previous one. uses MaxMind GeoIP binary databases.DOwnload the databases files from the MaxMind website and place them in Nginx directory.

```
geoip_country country.dat; # country information db
geoip_city city.dat; # city informatino db
geoip_org geoiporg.dat; # ISP/organization db
```

- $geoip_country_code (two letters c code)
- $geoip_country_name

- $geoip_region, $geoip_city, $geoip_postal_code, $geoip_city_continet_code, $geoip_latitude, $geoip_region_name, $geoip_org.

USerID filter

Assigns an identifier to the clients by issuing cookies. The identifier cab ne accessed from thevariables $uid_got and $uid_set

- userid: enables or disables issuing and logging of cookies.
  - default off.
  - on, v1, log, off
- userid_service: definfes the IP address of the server issuing the cookie
  - userid_service ip;
  - default: IP of the server
- userid_name: name assigned to the cookie
  - userid_name name;
  - default: user identifier
- userid_domain: domain assigned to the cookie
  - default: none
  - `userid_domain domain;`
- userid_path
  - default /
- userid_expires
  - default: no expiration date
  - `userid_expires date | max;`
- userid_p3p

Split Clients

resource-efficient way to split the visitor base into subgroups based on the percentages that specified. Nginx hashes a value privded (visitor IP, address, cookie data, query arguments, etc) and decides a group.

```
split_clients "$remote_addr" $variable {
  50% "group1";
  50% "group2";
}
location ~ /.php$ {
  set $args "${query_string}&group=${variable}";
}
```

SSL and Security

Secure HTTP functionalities through the SSL module, but also offers an extra module called Secure Link to protect the website and visitors.

SSL

Works on http and server contexts.

ssl

- Ena HTTPS for the specified server
- Equivalent to listen 443 or listen port ssl
- default `ssl off`

other directives

- `ssl_certificate file_path #PEM certificate`
- `ssl_certificate_key file_path`
- `ssl_client_certificate file_path`
- `ssl_crl` orders nginx to load Certificate Revocation List file
- `ssl_dhparam file_path`
- `ssl_protocols [SSLv2] [SSLv3] [TLSv1] [TLSv1.1] [TLSv1.2];`
- `ssl_ciphers` default ALL
- ssl_prefer_Server_ciphers
- ssl_verify_client
- ssl_verify_depth
- ssl_session_cache
- ssl_session_timeout
- ssl_password_phrase
- ssl_buffer_size
- ssl_session_tickets
- ssl_session_ticket_key
- ssl_trusted_certificate

Variables

- $ssl_cipher
- $ssl_client_serial
- $ssl_client_s_dn
- $ssl_protocol
- $ssl_client_cert
- $ssl_client_verify: set to success if client cert was successfully verified
- $ssl_session_id

Setting up SSL certificate

Ensure to already have the following elements

- A .key file generated with the following command: `openssl genrsa -out secure.website.comkey 1024`
- a .csr file generated with the following commmand ```openssl req -new -key secure.website.com.key -out secure.website.com.csr
- Website certificate file as issued by Certificate Authority as secure.website.com.crt.
- CA certificate (for example gd_bundle.crt purchased from goDaddy)

1. Merge website certificate and CA certificate

```
cat seucre.website.com.crt gd_bundle.crt > combined.crt
```

```
server {
  listen 443;
  server_name secure.website.com;
  ssl_certificate /path/to/combined.crt;
  ssl_certificate_key /path/to/secure.website.com.key;
}
```

SSL Stapling

SSL Stapling, also called OCSP (Online certificate Status Protocol Stapling), is a technique that allows clients to easily connect and resume sessions to an SSL/TLS server without having to contact the Certificate Authority, thus reducing the SSL negotiation time.

Enabling SSL stapling should thus speed up the communication between server and visitors.

```
ssl_stapling on;
ssl_stapling_verify on;
ssl_trusted_certificate filename; (extension sohuld be .pem)
```

```
#optional directives
ssl_stapling_file
ssl_stapling_responder
```