Nginx HTTP Framework

# 概述

暂无

# Configuration

## http_configuration.pngServer

Server代表一个Web（虚拟）服务器，由数据结构ngx\_http\_core\_srv\_conf\_t表示。

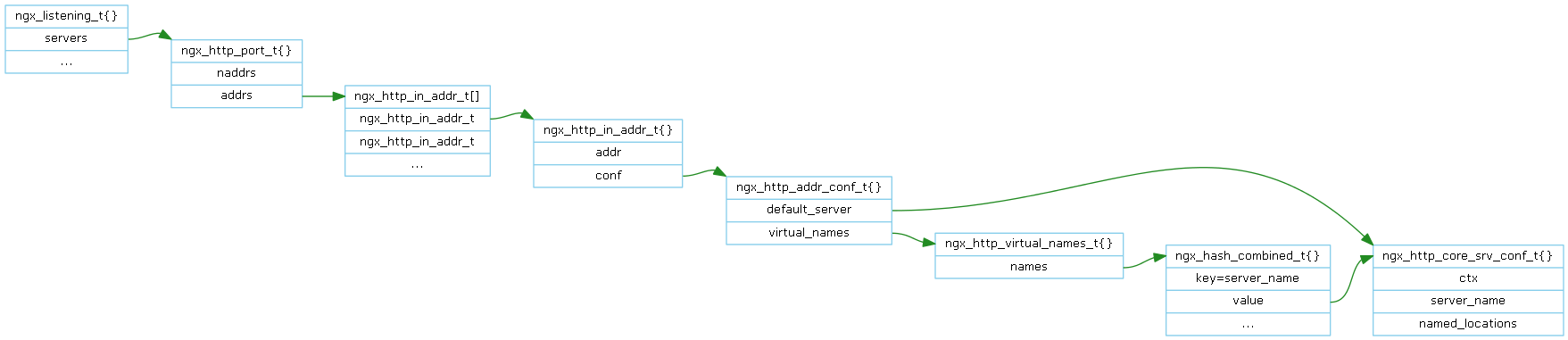
配置文件中一个server指令对应一个ngx\_http\_core\_srv\_conf\_t数组(每个HTTP模块有一个)。

所有HTTP Core模块的Server存放在ngx\_http\_core\_main\_conf\_t的servers数组中，通过它以及ngx\_http\_core\_srv\_conf\_t::ctx可以找到属于任何HTTP模块的任何Server。

当收到一个HTTP请求时，已知的是IP地址、端口号和URL，要根据这三个信息找到Server，

于是就创建了另一个管理Server的数据结构，由ngx\_http\_core\_main\_conf\_t的 ports管理。

为了更快地找到Server，Nginx对它作了进一步的处理：

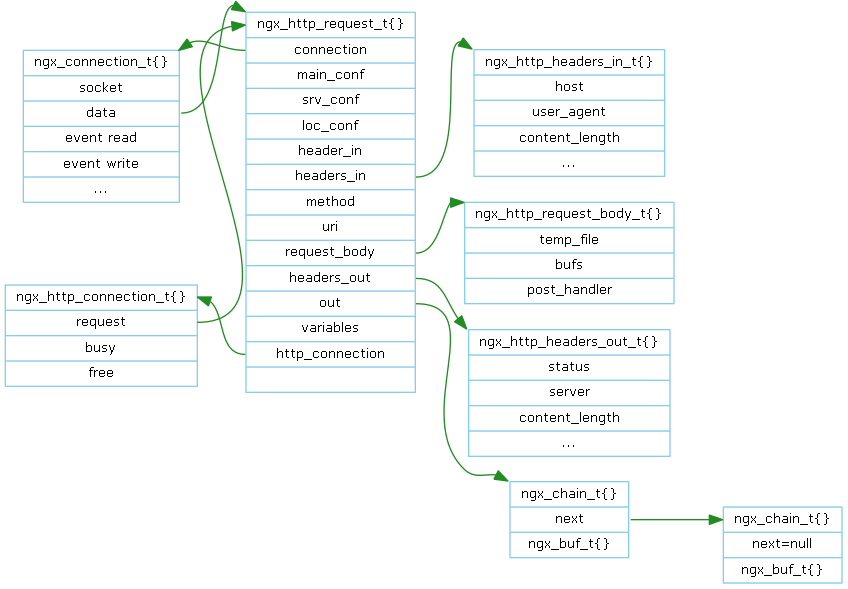


## Location

Location被所属的Server管理，字符串匹配方式的Location以树管理，正则表示式的Location以数组管理。

# Request

Request代表一个HTTP请求。



# Phase Handler

Nginx对Request的处理划分为11个阶段(Phase)，各个HTTP模块向各阶段注册回调函数来实现模块功能。

* POST\_READ

读取Request body数据。

* SERVER\_REWRITE

Server请求地址重写。

* FIND\_CONFIG

Server已经在处理Request Header时找到，现在从Server中查找Location。

* REWRITE

Location请求地址重写。

* POST\_REWRITE

请求地址重写后。

* PREACCESS

预权限检查。

* ACCESS

权限检查。

* POST\_ACCESS

后权限检查。

* TRY\_FILES

TODO

* CONTENT

响应内容产生。

* LOG\_PHASE

写日志。

## http_phase_handler.pngphases

各个HTTP模块在postconfiguration回调函数中，将回调处理函数ngx\_http\_handler\_pt，加入由phases域管理的二维数组中。

Nginx遍历此二维数组，即可完成请求处理。为了达到更好的性能，由phase\_engine重新组织了二维数组中的回调处理函数。

## phase\_engine

遍历一维数组优于遍历二维数组。

ngx\_http\_init\_phase\_handlers()将二维数组的函数，依次归并为一个数组，并用next指示出下一阶段开始的index。

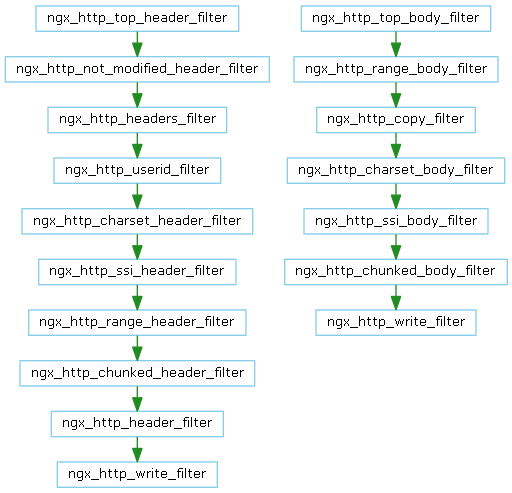
Nginx依次调用此一维数组的回调函数，当所属的阶段需中断处理，则通过next找到下一阶段的回调函数继续处理。

ngx\_http\_request\_t的phase\_handler是此一维数组的index，表示正在处理的回调函数。

当资源没有就绪时，遍历过程暂时退出，一旦就绪，在事件回调函数中从phase\_handler所指的位置断续遍历。

# Filter Chain

共有二个过滤器串：Header和Body。



## Header Filter

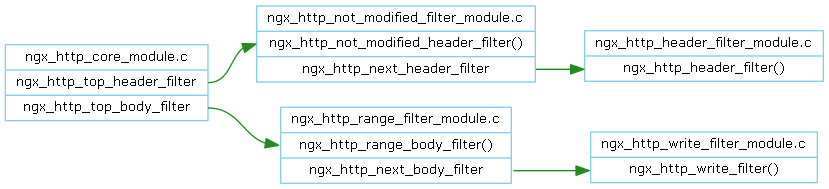
Header过滤器串的作用是对Response Header作检查、修改，最后由ngx\_http\_header\_filter

将ngx\_http\_request\_t:headers\_out组成字符串并由ngx\_http\_write\_filter加入发送队列ngx\_http\_request\_t::out。

## Body Filter

Body过滤器串的作用是对Response Body作检查、修改，最后由ngx\_http\_write\_filter()加入发送队列ngx\_http\_request\_t::out并将Response发送出去。

## 组织与运行

在源文件ngx\_http\_core\_module.c中有二个全局函数指针变量 ngx\_http\_top\_header\_filter和ngx\_http\_top\_body\_filter。

ngx\_http\_top\_header\_filter指向源文件ngx\_http\_not\_modified\_filter\_module.c的函数ngx\_http\_not\_modified\_header\_filter()。

在源文件ngx\_http\_not\_modified\_filter\_module.c有函数指针ngx\_http\_next\_header\_filter。

ngx\_http\_next\_header\_filter指向源文件ngx\_http\_header\_filter\_module.c的函数ngx\_header\_header\_filter()。

ngx\_header\_header\_filter()作为最后一个Header过滤器，所以源文件ngx\_http\_header\_filter\_module.c中没有ngx\_http\_next\_header\_filter。

上述过滤器串的组织是由各个过滤器模块的postconfiguration回调函数建立的。

过滤时，首先ngx\_http\_send\_header()调用由ngx\_http\_top\_header\_filter指向的ngx\_http\_not\_modified\_header\_filter()。

ngx\_http\_not\_modified\_header\_filter()做完自身的过滤操作后，调用由ngx\_http\_next\_header\_filter指向的ngx\_header\_header\_filter()。

ngx\_header\_header\_filter()做完过滤操作后过滤完成。

Body过滤器串，同理。

# 情境分析

下面以 wget http://localhost/index.html为例，作情境分析

## 接受Connection

在事件回调函数ngx\_event\_accept()中接受连接，得到新的Socket。

## 创建Connection

创建ngx\_connection\_t，并将读事件加入监视。

## 创建Request

在ngx\_http\_init\_request()中，创建ngx\_http\_request\_t和 ngx\_http\_connection\_t。

## 查找虚拟Server集合

根据ip address、port求得ngx\_http\_virtual\_names\_t。

## 处理Request Line

在读事件回调函数ngx\_http\_process\_request\_line()中，初始化解析Request-Line的状态机，从socket读入数据到内存缓冲区（ngx\_http\_request\_t::header\_in）。

用缓冲区的数据驱动状态机，直到结束，Request-Line处理完毕。

设置ngx\_http\_request\_t中method、http\_version信息。

## 查找虚拟Server

ngx\_http\_find\_virtual\_server()根据URI，求得虚拟Server。

## 处理Request Headers

在读事件回调函数ngx\_http\_process\_request\_headers()中，从socket读入数据到header\_in，并解析各个Header到headers\_in。

## 注销读事件

ngx\_http\_process\_request()中将读事件从监视列表中删除。Request Body的读取和处理由阶段处理的第一阶段POST\_READ处理。

## 运行阶段处理

ngx\_http\_handler()中将写事件回调函数设为ngx\_http\_core\_run\_phases()，并调用ngx\_http\_core\_run\_phases()遍历运行阶段处理回调函数。

如果因为资源不就绪，遍历中断。当资源就绪，可在写事件的回调函数中继续遍历。

### post read

无为。

### server rewrite

无为。

### find config

查找Location

### rewrite

无为。

### post rewrite

无为。

### pre access

无为。

### access

无为。

### post access

无为。

### try files

无为。

### content

ngx\_http\_static\_module在此阶段注册了回调函数ngx\_http\_static\_handler()。调用ngx\_http\_static\_handler()：

#### path

根据URI和Location的root，求得文件系统的path（路径）。

#### file

得到文件的属性，例如文件大少、最后修改时间等。并写入Request的headers\_out。

#### headers

调用ngx\_http\_sender\_header()，名为发送实为过滤Headers，并组成字符串并加入Request的out。

#### body

生成文件缓冲区ngx\_buf\_t，也挂入Request的out。调用ngx\_http\_output\_filter()发送Response。

### log phase

略

## 结束Request

ngx\_http\_finalize\_request()

ngx\_http\_close\_request()

## 结束Connection

ngx\_http\_finalize\_connection()

# Prototype

暂无

# Latest revision

https://github.com/lingjf/nginx\_analyse/blob/master/doc/