Nginx HTTP Parse

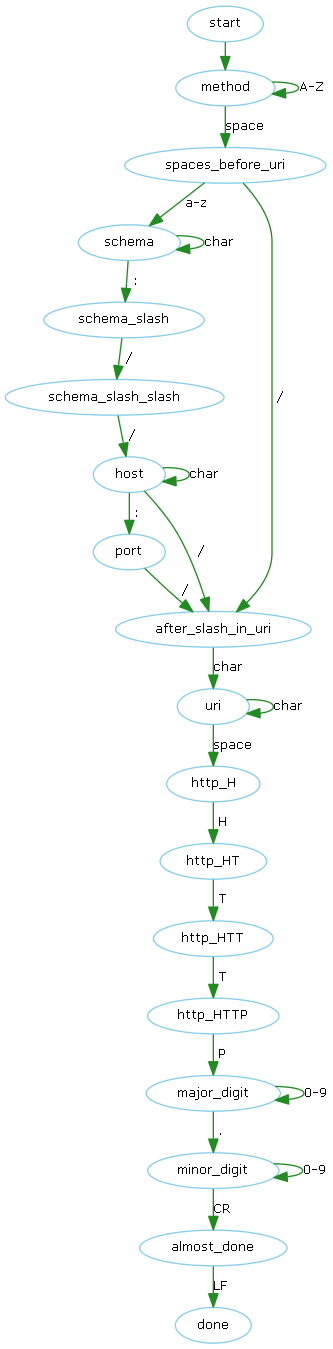
# 概述

# 解析Request-Line

RFC 2616

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

GET http://www.w3.org/pub/index.html HTTP/1.1



# 管理Request-Header

## http_handle_request_header.pngngx\_table\_elt\_t

该数据结构记录一个HTTP Header（Header Name、Header Value）。

* key：Header Name。比如：Host、User-Agent
* value：Header Value。比如：127.0.0.1、Chrome

## ngx\_http\_headers\_in\_t

该数据结构保存HTTP Request的所有HTTP Header信息。它是ngx\_request\_t的一个字段。

* headers：ngx\_table\_elt\_t链表，保存所有HTTP Header。
* host：“Host”头的ngx\_table\_elt\_t。
* user\_agent：“User-Agent”头的ngx\_table\_elt\_t。
* msie：是否是Microsoft Internet Explorer。根据“User-Agent”头信息，计算而得。
* chrome：是否是Google Chrome Browser。根据“User-Agent”头信息，计算而得。
* safari：是否是Apple Safari Brower。根据“User-Agent”头信息，计算而得。

## ngx\_http\_header\_t

该数据结构描述如何处理HTTP Request中的HTTP Header。

* name：Header名称，比如：Host、User-Agent、Content-Length。
* offset：ngx\_http\_headers\_in\_t结构中相应字段的偏移。比如：offsetof(ngx\_http\_headers\_in\_t, user\_agent)。
* handler：处理函数。

## ngx\_http\_headers\_in

这是一个全局ngx\_http\_header\_t数组变量，表示Nginx支持的HTTP Header集合。

为了性能，Nginx将这个数组用HASH表重新组织了一次，由ngx\_http\_core\_main\_conf\_t的headers\_in\_hash管理。

# 处理User-Agent

现在以处理User-Agent这个HTTP Header为例，进行分析。

1. ngx\_http\_parse\_header\_line解析到一个HTTP Header：“User-Agent： Chrome 12.10.24”
2. ngx\_http\_headers\_in\_t:headers加入一个ngx\_table\_elt\_t{“User-Agent”, “Chrome 12.10.24”}
3. 以“User-Agent”为key，从ngx\_http\_core\_main\_conf\_t:headers\_in\_hash中找到ngx\_http\_header\_t
4. 调用ngx\_http\_header\_t的处理函数handler ngx\_http\_process\_user\_agent()
5. ngx\_http\_headers\_in\_t::user\_agent 🡪 ngx\_table\_elt\_t{“User-Agent”, “Chrome 12.10.24”}
6. msie = 0
7. chrome = 1
8. safari = 0

# Prototype

暂无

# Latest revision

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