

## 第 8 章

# Nginx 模块开发

一些访问量非常大、业务逻辑简单的 Web 应用，如果采用 PHP 等解析型语言去处理，虽然可行，但是在并发能力、处理速度上将受到限制，耗费的系统资源也会较大，这就要求我们增加更多的服务器来处理这类应用。而采用 Nginx 模块来处理这类 Web 应用，在性能上将得到极大的提高，大大减少服务器的数量，并将在很大程度上节省服务器的运维成本。

要编写一个 Nginx 模块，你要熟悉 Nginx 的配置文件。Nginx 配置文件主要分成 4 部分：main（全局配置）、server（虚拟主机配置）、upstream（主要为反向代理、负载均衡相关配置）和 location（目录匹配配置），每部分包含若干个指令。main 部分的指令将影响其他所有部分；server 部分的指令主要用于指定虚拟主机域名、IP 和端口；upstream 的指令用于设置反向代理及后端服务器的负载均衡；location 部分用于匹配网页位置（例如，根目录“/”、“/images”，等等）。location 部分会继承 server 部分的指令，而 server 部分会继承 main 部分的指令；upstream 既不会继承指令也不会影响其他部分。它有自己的特殊指令，不需要在其他地方应用。

## 8.1 Nginx 模块概述

Nginx 的模块不能够像 Apache 那样动态添加，所有的模块都要预先编译进 Nginx 的二进制可执行文件中。

Nginx 模块有 3 种角色：

- (1) Handlers（处理模块）——用于处理 HTTP 请求并输出内容；
- (2) Filters（过滤模块）——用于过滤 Handler 输出的内容；
- (3) Load-balancers（负载均衡模块）——当有多于一台的后端服务器供选择时，选择一台后端服务器并将 HTTP 请求转发到该服务器。

当 Nginx 发送文件或转发请求到其他服务器时，可以用 Handlers 处理模块为其服务；当需要 Nginx 把输出压缩或在服务端加一些东西时，可以用 Filters 过滤模块；Nginx 的核心模块主要管理网络层和应用层协议，以及启动针对特定应用的一系列候选模块。

接下来，我们介绍 Nginx 模块的处理流程。

客户端发送 HTTP 请求到 Nginx 服务器→Nginx 基于配置文件中的位置选择一个合适的处理模块→负载均衡模块选择一台后端服务器（反向代理情况下）→处理模块进行处理并把输出缓冲放到第一个过滤模块上→第一个过滤模块处理后输出给第二个过滤模块→然后第二个过滤模块又到第三个过滤模块→第 N 个过滤模块→最后把处理结果发送给客户端。

模块相当于钩子，可以挂在 Nginx 的以下位置，在某些时段执行某些功能：

- (1) 当服务读配置文件之前；
- (2) 当读取在 location 和 server 部分或其他任何部分的每一个配置指令时；
- (3) 当 Nginx 初始化全局部分的配置时；
- (4) 当 Nginx 初始化主机部分（比如主机/端口）的配置时；
- (5) 当 Nginx 将全局部分的配置与主机部分的配置合并时；
- (6) 当 Nginx 初始化匹配位置部分配置时；
- (7) 当 Nginx 将其上层主机配置与位置部分配置合并时；
- (8) 当 Nginx 的主进程（master）开始时；
- (9) 当一个新的工作进程（worker）开始时；
- (10) 当一个工作进程退出时；
- (11) 当主进程退出时；
- (12) 处理 HTTP 请求时；
- (13) 过滤 HTTP 回复的头部时；
- (14) 过滤 HTTP 回复的主体时；

- (15) 选择一台后端服务器时;
- (16) 初始化到后端服务器的请求时;
- (17) 重新初始化到后端服务器的请求时;
- (18) 处理来自后端服务器的回复时;
- (19) 完成与后端服务器的交互时。

## 8.2 Nginx 模块编写实践

众所周知,“Hello World”几乎已经变成了所有程序语言的第一个范例。“Hello World”程序指的是只在计算机屏幕上输出“Hello World”(意思为“世界,你好!”)这行字符串的计算机程序。一般来说,这是每一种计算机编程语言中最基本、最简单的程序,通常亦是初学者所编写的第一个程序。那么,在这一节,就让我们动手编写一个在浏览器输出“Hello Word”的 Nginx 模块,在实践中了解如何编写 Nginx 模块。

### 8.2.1 Hello World 模块编写与安装

- (1) 执行以下命令创建一个目录,将在该目录内编写我们的 Nginx 模块:

```
mkdir -p /opt/nginx_hello_world  
cd /opt/nginx_hello_world
```

- (2) 开始创建 Nginx 模块所需的配置文件(名称为 config):

```
vi /opt/nginx_hello_world/config
```

然后输入以下内容并保存退出:

```
ngx_addon_name=ngx_http_hello_world_module  
HTTP_MODULES="$HTTP_MODULES ngx_http_hello_world_module"  
NGX_ADDON_SRCS="$NGX_ADDON_SRCS $ngx_addon_dir/ngx_http_hello_world_module.c"  
CORE_LIBS="$CORE_LIBS -lpcrc"
```

- (3) 创建 Nginx 模块的 C 程序文件(名称格式为“ngx\_http\_模块名称\_module.c”,在本示例中,文件名称为 ngx\_http\_hello\_world\_module.c:

```
vi /opt/nginx_hello_world/ngx_http_hello_world_module.c
```

然后输入如代码 8-1 所示内容并保存退出:

代码 8-1

```

#include <ngx_config.h>
#include <ngx_core.h>
#include <ngx_http.h>

static char *ngx_http_hello_world(ngx_conf_t *cf, ngx_command_t *cmd, void *conf);

static ngx_command_t ngx_http_hello_world_commands[] = {

    { ngx_string("hello_world"),
      NGX_HTTP_LOC_CONF|NGX_CONF_NOARGS,
      ngx_http_hello_world,
      0,
      0,
      NULL },

    ngx_null_command

};

static u_char ngx_hello_world[] = "hello world";

static ngx_http_module_t ngx_http_hello_world_module_ctx = {
    NULL,                          /* preconfiguration */
    NULL,                          /* postconfiguration */

    NULL,                          /* create main configuration */
    NULL,                          /* init main configuration */

    NULL,                          /* create server configuration */
    NULL,                          /* merge server configuration */

    NULL,                          /* create location configuration */
    NULL,                          /* merge location configuration */
};

ngx_module_t ngx_http_hello_world_module = {
    NGX_MODULE_V1,
    &ngx_http_hello_world_module_ctx, /* module context */
    ngx_http_hello_world_commands, /* module directives */
    NGX_HTTP_MODULE,                /* module type */
    NULL,                            /* init master */
    NULL,                            /* init module */
    NULL,                            /* init process */
    NULL,                            /* init thread */
    NULL,                            /* exit thread */
    NULL,                            /* exit process */
    NULL,                            /* exit master */
    NGX_MODULE_V1_PADDING
};

static ngx_int_t ngx_http_hello_world_handler(ngx_http_request_t *r)

```

---

```

{
    ngx_buf_t    *b;
    ngx_chain_t  out;

    r->headers_out.content_type.len = sizeof("text/plain") - 1;
    r->headers_out.content_type.data = (u_char *) "text/plain";

    b = ngx_palloc(r->pool, sizeof(ngx_buf_t));

    out.buf = b;
    out.next = NULL;

    b->pos = ngx_hello_world;
    b->last = ngx_hello_world + sizeof(ngx_hello_world);
    b->memory = 1;
    b->last_buf = 1;

    r->headers_out.status = NGX_HTTP_OK;
    r->headers_out.content_length_n = sizeof(ngx_hello_world);
    ngx_http_send_header(r);

    return ngx_http_output_filter(r, &out);
}

static char *ngx_http_hello_world(ngx_conf_t *cf, ngx_command_t *cmd, void *conf)
{
    ngx_http_core_loc_conf_t *clcf;

    clcf = ngx_http_conf_get_module_loc_conf(cf, ngx_http_core_module);
    clcf->handler = ngx_http_hello_world_handler;

    return NGX_CONF_OK;
}

```

---

(4) 下载 Nginx 源码包，并将 hello world 模块编译到其中，如代码 8-2 所示：

#### 代码 8-2

---

```

wget ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-7.9.tar.gz
tar zxvf pcre-7.9.tar.gz
cd pcre-7.9/
./configure
make && make install
cd ../

wget http://sysoev.ru/nginx/nginx-0.8.14.tar.gz
tar zxvf nginx-0.8.14.tar.gz
cd nginx-0.8.14/
./configure --prefix=/usr/local/nginx --add-module=/opt/nginx_hello_world
make
make install

```

---

(5) 配置 nginx.conf, 在 server 部分增加以下内容:

```
location = /hello {  
    hello_world;  
}
```

(6) 启动 Nginx, 用浏览器访问 `http://localhost/hello`, 就可以看到编写的 Nginx Hello World 模块输出的文字 “Hello World”。

## 8.2.2 Hello World 模块分析

(1) 在 `ngx_http_hello_world_module.c` 代码中, `ngx_command_t` 函数用于定义包含模块指令的静态数组 `ngx_http_hello_world_commands`, 如代码 8-3 所示。

代码 8-3

```
static char *ngx_http_hello_world(ngx_conf_t *cf, ngx_command_t *cmd, void *conf);  
  
static ngx_command_t ngx_http_hello_world_commands[] = {  
  
    { ngx_string("hello_world"),  
      NGX_HTTP_LOC_CONF|NGX_CONF_NOARGS,  
      ngx_http_hello_world,  
      0,  
      0,  
      NULL },  
  
    ngx_null_command  
};
```

在数组中, 第一项参数 `ngx_string("hello_world")` 为指令名称字符串, 不能含有空格, 数据类型是 `ngx_str_t`, 经常用来进行字符串实例化。

注意: `ngx_str_t` 结构体由包含有字符串的 `data` 成员和表示字符串长度的 `len` 成员组成。Nginx 用这个数据类型来存放字符串。

第二项参数用来设置指令在配置文件位置的哪一部分使用是合法的, 可选值如下, 多个选项以 “|” 隔开:

NGX\_HTTP\_MAIN\_CONF——指令出现在全局配置部分是合法的;

NGX\_HTTP\_SRV\_CONF——指令出现在 `server` 主机配置部分是合法的;

NGX\_HTTP\_LOC\_CONF——指令出现在 `location` 配置部分是合法的;

NGX\_HTTP\_UPS\_CONF——指令出现在 `upstream` 配置部分是合法的;

NGX\_CONF\_NOARGS——指令没有参数;

NGX\_CONF\_TAKE1——指令读入 1 个参数;

NGX\_CONF\_TAKE2——指令读入 2 个参数;

.....

NGX\_CONF\_TAKE7——指令读入 7 个参数;

NGX\_CONF\_FLAG——指令读入 1 个布尔型数据;

NGX\_CONF\_1MORE——指令至少读入 1 个参数;

NGX\_CONF\_2MORE——指令至少读入 2 个参数。

第三项参数 ngx\_http\_hello\_world 是一个回调函数。该回调函数有 3 个参数:

ngx\_conf\_t \*cf——指向 ngx\_conf\_t 结构体的指针, 包含从指令后面传过来的参数。

ngx\_command\_t \*cmd——指向当前 ngx\_command\_t 结构体的指针。

void \*conf——指向自定义模块配置结构体的指针。

第四项参数用于告诉 Nginx 是把要保留的值放在全局配置部分、server 主机配置部分还是 location 位置配置部分 (使用 NGX\_HTTP\_MAIN\_CONF\_OFFSET、NGX\_HTTP\_SRV\_CONF\_OFFSET、NGX\_HTTP\_LOC\_CONF\_OFFSET)。

第五项参数用于设置指令的值保存在结构体的哪个位置。第六项参数一般为 NULL。这样, 6 个参数都设置完毕, 此数组在读入 ngx\_null\_command 后停止。

(2) 接下来的 static u\_char ngx\_hello\_world[] = "hello world"; 语句, 则定义了用于输出给客户端浏览器的字符串内容。

(3) ngx\_http\_module\_t 用来定义结构体 ngx\_http\_hello\_world\_module\_ctx, 如代码 8-4 所示。

代码 8-4

```
static ngx_http_module_t ngx_http_hello_world_module_ctx = {
    NULL,                          /* preconfiguration */
    NULL,                          /* postconfiguration */

    NULL,                          /* create main configuration */
    NULL,                          /* init main configuration */

    NULL,                          /* create server configuration */
    NULL,                          /* merge server configuration */

    NULL,                          /* create location configuration */
    NULL,                          /* merge location configuration */
};
```

静态 `ngx_http_module_t` 结构体，包含很多函数引用，用来创建 3 个部分的配置和合并配置。一般结构体命名为 `ngx_http_<module name>_module_ctx`。这些函数引用包括：

- `preconfiguration`——在读入配置前调用；
- `postconfiguration`——在读入配置后调用；
- `create main configuration`——在创建全局部分配置时调用（比如，用来分配空间和设置默认值）；
- `init main configuration`——在初始化全局部分的配置时调用（比如，把原来的默认值用 `nginx.conf` 读到的值覆盖）；
- `create server configuration`——在创建虚拟主机部分的配置时调用；
- `merge server configuration`——与全局部分配置合并时调用；
- `create location configuration`——创建位置部分的配置时调用；
- `merge location configuration`——与主机部分配置合并时调用。

这些函数参数不同，依赖于它们的功能。代码 8-5 是这个结构体的定义，摘自 `http/ngx_http_config.h`，你可以看到属性各不相同的回调函数：

代码 8-5

---

```
typedef struct {
    ngx_int_t    (*preconfiguration)(ngx_conf_t *cf);
    ngx_int_t    (*postconfiguration)(ngx_conf_t *cf);
    void         (*create_main_conf)(ngx_conf_t *cf);
    char         (*init_main_conf)(ngx_conf_t *cf, void *conf);
    void         (*create_srv_conf)(ngx_conf_t *cf);
    char         (*merge_srv_conf)(ngx_conf_t *cf, void *prev, void *conf);
    void         (*create_loc_conf)(ngx_conf_t *cf);
    char         (*merge_loc_conf)(ngx_conf_t *cf, void *prev, void *conf);
}

```

---

如果不用某些函数，可以设定为 `NULL`，Nginx 会剔除它。

(4) `ngx_module_t` 定义结构体 `ngx_http_hello_world_module`，如代码 8-6 所示。

代码 8-6

---

```
ngx_module_t ngx_http_hello_world_module = {
    NGX_MODULE_V1,
    &ngx_http_hello_world_module_ctx, /* module context */
    ngx_http_hello_world_commands, /* module directives */
    NGX_HTTP_MODULE, /* module type */
    NULL, /* init master */
    NULL, /* init module */
    NULL, /* init process */

```

---



---

```

NULL,                /* init thread */
NULL,                /* exit thread */
NULL,                /* exit process */
NULL,                /* exit master */
NGX_MODULE_V1_PADDING
};

```

---

这个结构体变量命名为 `ngx_http_<module name>_module`。它包含有模块的主要内容和指令的执行部分，也有一些回调函数（退出线程、退出进程，等等）。这个模块的定义是把数据处理关联到特定模块的关键。

(5) 回调函数 `ngx_http_hello_world`，分为两步，第一步获得这个 Location 位置配置的“核心”结构体，然后为它分配一个处理函数 `ngx_http_hello_world_handler`，如代码 8-7 所示。

#### 代码 8-7

---

```

static char *ngx_http_hello_world(ngx_conf_t *cf, ngx_command_t *cmd, void *conf)
{
    ngx_http_core_loc_conf_t *clcf;

    clcf = ngx_http_conf_get_module_loc_conf(cf, ngx_http_core_module);
    clcf->handler = ngx_http_hello_world_handler;

    return NGX_CONF_OK;
}

```

---

(6) 处理函数 `ngx_http_hello_world_handler`，也是 Hello World 模块的核心部分。参数 `ngx_http_request_t *r`，可以让我们访问到客户端的头部和不久要发送的回复头部，包含两个成员：`r->headers_in` 和 `r->headers_out`。

`b->pos = ngx_hello_world`；即为要输出的内容，`ngx_hello_world` 的内容就是之前通过 `static u_char ngx_hello_world[] = "hello world"`；语句定义的用于输出给客户端浏览器的字符串“hello world”，如代码 8-8 所示。

#### 代码 8-8

---

```

static ngx_int_t ngx_http_hello_world_handler(ngx_http_request_t *r)
{
    ngx_buf_t    *b;
    ngx_chain_t  out;

    r->headers_out.content_type.len = sizeof("text/plain") - 1;
    r->headers_out.content_type.data = (u_char *) "text/plain";

    b = ngx_palloc(r->pool, sizeof(ngx_buf_t));

    out.buf = b;
    out.next = NULL;
}

```

---



```
b->pos = ngx_hello_world;
b->last = ngx_hello_world + sizeof(ngx_hello_world);
b->memory = 1;
b->last_buf = 1;

r->headers_out.status = NGX_HTTP_OK;
r->headers_out.content_length_n = sizeof(ngx_hello_world);
ngx_http_send_header(r);

return ngx_http_output_filter(r, &out);
}
```

通过以上内容，相信您已经掌握了如何编写一个简单的 Nginx 模块。Nginx 的模块开发涉及的内容非常广，编写更为复杂的 Nginx 模块，不是本书的重点所在。如果您感兴趣，可以参阅 Evan Miller 编写的《Evan Miller's Guide to Nginx Module Development》(<http://www.evanmiller.org/nginx-modules-guide.html>) 获取更多的信息。