Nginx变量机制允许跨模块访问

**概念**

**添加变量**

使用函数ngx\_http\_add\_variable添加变量

ngx\_http\_variable\_t \*ngx\_http\_add\_variable(

    ngx\_conf\_t \*cf,     // 配置

    ngx\_str\_t \*name,    // 变量名

    ngx\_uint\_t flags)   // 标志

标志：

#define NGX\_HTTP\_VAR\_CHANGEABLE   1     // 是否可以修改，如果为true，则我们可以直接修改”变量值”结构的data和len

#define NGX\_HTTP\_VAR\_NOCACHEABLE  2     // 是否允许缓存

#define NGX\_HTTP\_VAR\_INDEXED      4     // 是否索引

#define NGX\_HTTP\_VAR\_NOHASH       8     // 是否允许 hash

**变量**

使用ngx\_http\_variable\_t结构表示变量

struct ngx\_http\_variable\_s {

    // 变量名

    ngx\_str\_t                     name;

    // 设置变量值函数

    ngx\_http\_set\_variable\_pt      set\_handler;

    // 获取变量值函数

    ngx\_http\_get\_variable\_pt      get\_handler;

    // 设置或获取变量值函数的辅助参数

    uintptr\_t                     data;

    // 标志

    ngx\_uint\_t                    flags;

    ngx\_uint\_t                    index;

};

**获取变量值函数**

函数需要根据请求结构，填充变量值结构

typedef ngx\_int\_t (\*ngx\_http\_get\_variable\_pt) (

    // 请求结构

    ngx\_http\_request\_t \*r,

    // 变量值，函数需要填充该结构

    ngx\_http\_variable\_value\_t \*v,

    // 辅助参数

    uintptr\_t data);

**变量值**

typedef struct {

    // 数据的长度

    unsigned    len:28;

    // 数据释放有效

    unsigned    valid:1;

    // 数据是否不缓存

    unsigned    no\_cacheable:1;

    unsigned    not\_found:1;

unsigned    escape:1;

    // 变量的数据

    u\_char     \*data;

} ngx\_variable\_value\_t;

**根据变量名获取变量值**

使用函数ngx\_http\_get\_variable来根据变量名获取变量值，该函数会调用我们定义的get\_handler（获取变量值函数）生成变量值

ngx\_http\_variable\_value\_t \*ngx\_http\_get\_variable(

    ngx\_http\_request\_t \*r,  // 请求结构

    ngx\_str\_t \*name,        // 变量名

    ngx\_uint\_t key)         // 变量名对应的 key

**应用**

**模块配置解析前函数**

在模块配置解析前函数中添加变量

// 配置解析前调用函数

static ngx\_int\_t   mypreconfiguration(ngx\_conf\_t \*cf)

{

    ngx\_str\_t name = ngx\_string("myvar");

    auto name\_pt = ngx\_pcalloc(cf->pool, name.len);

    ngx\_copy(name\_pt, name.data, name.len);

    name.data = reinterpret\_cast<u\_char\*>(name\_pt);

    // 添加变量

    auto var = ngx\_http\_add\_variable(cf, &name, NGX\_HTTP\_VAR\_CHANGEABLE);

var->get\_handler = myGetVarFun;

    return NGX\_OK;

}

**在请求处理函数中使用变量**

// 请求处理函数

static ngx\_int\_t handler(ngx\_http\_request\_t \*r)

{

    auto cf = reinterpret\_cast<NdgTestConf\*>(ngx\_http\_get\_module\_loc\_conf(r, ndg\_test\_module));

    // 根据配置决定输出

    if(!cf->enabled){

        return NGX\_DECLINED;

}

    // 变量名称

    ngx\_str\_t name = ngx\_string("myvar");

    auto name\_pt = ngx\_pcalloc(r->pool, name.len);

    ngx\_copy(name\_pt, name.data, name.len);

name.data = reinterpret\_cast<u\_char\*>(name\_pt);

    // 这zz的key获取

auto key = ngx\_hash\_strlow(name.data, name.data, name.len);

    // 获取变量值

    auto varvalue = ngx\_http\_get\_variable(r, &name, key);

    // 控制台显示

    u\_char text[varvalue->len + 1];

    ngx\_copy(text, varvalue->data, varvalue->len);

    text[varvalue->len ] = '\0';

std::cout << text << std::endl;

    // 执行成功，但未作处理

    return NGX\_DECLINED;

}

**完整示例**

#include <iostream>

extern "C" {

#include <ngx\_http.h>

}

extern ngx\_module\_t ndg\_test\_module;

// 配置结构

struct NdgTestConf final

{

    // 是否启用

    ngx\_flag\_t enabled = NGX\_CONF\_UNSET;

};

// 创建配置结构函数

static void\* create(ngx\_conf\_t\* cf)

{

    auto conf = reinterpret\_cast<NdgTestConf\*>(ngx\_pcalloc(cf->pool, sizeof(NdgTestConf)));

    \*conf = NdgTestConf();

    return conf;

}

// 配置指令，描述了配置使用的指令，用于配置解析

static ngx\_command\_t ndg\_test\_cmds[] =

{

    {

        ngx\_string("myndg"),

        NGX\_HTTP\_LOC\_CONF | NGX\_CONF\_FLAG,

        ngx\_conf\_set\_flag\_slot,

        NGX\_HTTP\_LOC\_CONF\_OFFSET,

        offsetof(NdgTestConf, enabled),

        nullptr

    },

    ngx\_null\_command

};

// 请求处理函数

static ngx\_int\_t handler(ngx\_http\_request\_t \*r)

{

auto cf = reinterpret\_cast<NdgTestConf\*>(ngx\_http\_get\_module\_loc\_conf(r, ndg\_test\_module));

    // 根据配置决定输出

    if(!cf->enabled){

        return NGX\_DECLINED;

}

    // 变量名称

    ngx\_str\_t name = ngx\_string("myvar");

    auto name\_pt = ngx\_pcalloc(r->pool, name.len);

    ngx\_copy(name\_pt, name.data, name.len);

name.data = reinterpret\_cast<u\_char\*>(name\_pt);

    // 这zz的key获取

    auto key = ngx\_hash\_strlow(name.data, name.data, name.len);

    auto varvalue = ngx\_http\_get\_variable(r, &name, key);

    // 控制台显示

    u\_char text[varvalue->len + 1];

    ngx\_copy(text, varvalue->data, varvalue->len);

    text[varvalue->len ] = '\0';

std::cout << text << std::endl;

    // 执行成功，但未作处理

    return NGX\_DECLINED;

}

// 获取变量值函数

static ngx\_int\_t myGetVarFun(ngx\_http\_request\_t \*r, ngx\_http\_variable\_value\_t \*v, uintptr\_t data) {

        ngx\_str\_t value = ngx\_string("aaa");

        v->data = reinterpret\_cast<u\_char\*>(ngx\_pcalloc(r->pool, value.len));

        v->len = value.len;

        v->valid = true;

        v->not\_found = false;

        v->no\_cacheable = false;

        ngx\_copy(v->data, value.data, value.len);

        return NGX\_OK;

}

// 配置解析前调用函数

static ngx\_int\_t   mypreconfiguration(ngx\_conf\_t \*cf)

{

    ngx\_str\_t name = ngx\_string("myvar");

    auto name\_pt = ngx\_pcalloc(cf->pool, name.len);

    ngx\_copy(name\_pt, name.data, name.len);

name.data = reinterpret\_cast<u\_char\*>(name\_pt);

    // 添加变量

    auto var = ngx\_http\_add\_variable(cf, &name, NGX\_HTTP\_VAR\_CHANGEABLE);

var->get\_handler = myGetVarFun;

    return NGX\_OK;

}

// 初始化函数

static ngx\_int\_t init(ngx\_conf\_t\* cf)

{

    auto cmcf = reinterpret\_cast<ngx\_http\_core\_main\_conf\_t\*>(

        ngx\_http\_conf\_get\_module\_main\_conf(cf, ngx\_http\_core\_module)

);

    // 注册请求处理函数

    auto &arr = cmcf->phases[NGX\_HTTP\_REWRITE\_PHASE].handlers;

\*(reinterpret\_cast<ngx\_http\_handler\_pt\*>(ngx\_array\_push(&arr))) = handler;

    return NGX\_OK;

}

// 函数集合

static ngx\_http\_module\_t ndg\_test\_ctx = {

    mypreconfiguration,

    init,                   // 指定模块初始化函数

    nullptr,

    nullptr,

    nullptr,

    nullptr,

    create,             // 指定创建配置结构函数

    nullptr

};

// 模块

// ngx\_module\_t 用于描述我们模块

ngx\_module\_t ndg\_test\_module = {

    NGX\_MODULE\_V1,

    &ndg\_test\_ctx,                      // 模块函数集合

    ndg\_test\_cmds,                    // 配置指令

    NGX\_HTTP\_MODULE,

    nullptr,

    nullptr,

    nullptr,

    nullptr,

    nullptr,

    nullptr,

    nullptr,

    NGX\_MODULE\_V1\_PADDING

};