OpenResty+LuaJIT 高并发

web 服务实践教程知识点

第五章：Openresty 中基于 Nginx 的 Lua 模块

第一节：Openresty 中 Lua 的基本输入和输出

1. 获取 uri 参数

- ngx.req.get_uri_args()：获取get参数
- ngx.req.get_post_args() 获取post参数
  前提是开启ngx.req.read_body()或者设置
  lua_need_request_body on;

2. 获取请求内容：
   - ngx.req.get_body_data()：获取post内容
     前提是开启ngx.req.read_body()或者设置
     lua_need_request_body on;

3. 输出内容
   - 在 OpenResty 中调用 ngx.say 或 ngx.print 即可。
     这两者都是输出响应体，区别是 ngx.say 会对输出响应体多输出一个 \n 。

第二节：LuaCjson 模块

1. LuaCjson 模块的使用
   - 引入 cjson 模块，使用 require（‘cjon’）
   - cJson 模块的异常处理

     如果需要在 Lua 中处理错误，必须使用函数 pcall（protected call）来包装需要执行的代码。 pcall 接收一个函数和要传递给后者的参数，并执行，执行结果：有错误、无错误；返回值 true 或者或 false，errorinfo。pcall 以一种"保护模式"来调用第一个参数，因此 pcall 可以捕获函数执行中的任何错误

   - 空 table 编码为 array 还是 object

     默认 cjson 模块把空数组编码输出为 object，

     加上 cjson.encode_empty_table_as_object(false)就强制输出成 array 了

   - cjson 模块不能在 windows 中使用，windows 中可以使用 dkjson

第三节：LuaRestyRedis 模块

1. Lua 中冒号调用和点调用函数的区别

- 定义的时候冒号默认接收self参数
- 调用的时候冒号默认传递调用者自己为参数
- 而句号要显示传递或接收self参数

2. LuaRestyRedis 模块的使用

- 引入 reidis 模块

```
local redis = require "resty.redis"
```

- 连接判断

```
local red = redis:new()
local ok, err = red:connect("127.0.0.1", 6379)
if not ok then
    ngx.say("failed to connect: ", err)
    return
end
```

- 从 redis 中获取值以及设置值

```
ngx.say(red:get( “a” ))
red:set( “b” ,” test redis” )
```

第四节：OpenResty 中 Lua 发起 http 请求

1. OpenResty 中的子查询
   - 调用方式：ngx.location.capture 或者 ngx.location.capture_multi（并发）
   - 优势：子查询 Nginx 子请求是一种非常强有力的方式，它可以发起非阻塞的内部请求访问目标 location。**需要注意的是，子请求只是模拟 HTTP 接口的形式， 没有 额外的 HTTP/TCP 流量，也 没有 IPC (进程间通信) 调用。**

2. OpenResty 中发起外部 http 请求

   参考下面示例，利用 proxy_pass 完成 HTTP 接口访问的成熟配置+调用方法。

```
http {
    server {
        listen       80;
        location /test {
            content_by_lua_block {
                local res = ngx.location.capture('/send_out_req',
                    {
                        method = ngx.HTTP_POST,
                        body = args.data
                    }
                )
                ngx.say(res.status)
                ngx.say(res.body)
            }
        }
        location /send_out_req {
```

```
                internal;
                proxy_pass http://115.29.166.132:8080;
            }
        }
}
```

重点说明：

- 接口访问通过 ngx.location.capture 的子查询方式发起；
- 由于 ngx.location.capture 方式只能是 nginx 自身的子查询，需要借助 proxy_pass 发出 HTTP 连接信号；