简易反向代理服务器

疑惑咨询：

简述：任务主要分为两大部分，第一个部分为解析符合nginx配置文件语法的配置文件，第二个部分为实现一个反向代理服务器，该反向代理服务器仅需实现两个功能：将请求转发到下游地址和静态文件服务器。在完成任务的时候你可以选择交叉进行，也可以一个一个完成之后将两个整合在一起。

关键词：反向代理、nginx配置文件解析、HTTP请求（HTTP基础结构）、nginx location规则

第一阶段 - 解析配置文件

概览：解析符合 Nginx 配置文件规则（简易规则）的文件。

nginx配置文件完整介绍：https://www.nginx.com/resources/wiki/start/topics/examples/full/

要求实现规则如下：

解析server规则

- [ ] listen 表示监听端口，如：80、8080 ★★★
- [ ] server_name 表示监听的请求的域名，如：baidu.com、localhost，只有请求的域名符合要求的时候才会继续处理 ★★★
- [ ] error_log 表示如果请求出现错误打印错误日志，后跟记录日志文件路径，如：请求404、500等则为错误，将错误记录到 *error.log* 中
- [ ] access_log 表示请求的日志记录，后跟记录日志文件路径，记录信息如下：请求时间、请求方法、请求路由(/api/v1/index?token=5d898323faa6ee003513d5b7)等

eg：

```
server {
    listen 80;
    server_name localhost;

    error_log  /mnt/log/nginx/localhost/error.log;
    access_log /mnt/log/nginx/localhost/access.log;
}
```

解析 location 路由规则 ★★★

注：可先实现 `location /uri` 之后继续非location规则，待完成度较高时再回头来完成其他location规则

语法 规则

模式	含义
location = /uri	= 表示精确匹配，只有完全匹配上才能生效
location ^~ /uri	^~ 开头对URL路径进行前缀匹配，并且在正则之前。
location ~ pattern	开头表示区分大小写的正则匹配
location ~* pattern	开头表示不区分大小写的正则匹配
location /uri ★★★	不带任何修饰符，也表示前缀匹配，但是在正则匹配之后
location /	通用匹配，任何未匹配到其它location的请求都会匹配到，相当于switch中的default

匹配顺序

多个 location 配置的情况下匹配顺序为：

- 首先精确匹配 `=`
- 其次前缀匹配 `^~`
- 其次是按文件中顺序的正则匹配
- 然后匹配不带任何修饰的前缀匹配。
- 最后是交给 `/` 通用匹配

当有匹配成功时候，停止匹配，按当前匹配规则处理请求。

eg:

```
server {
    listen 80;
    server_name localhost;

    location = / {
       # echo 表示打印
       echo "规则A";
    }
    location = /login {
       echo "规则B";
    }
    location ^~ /static/ {
       echo "规则C";
    }
    location ^~ /static/files {
        echo "规则X";
    }
    location ~ \.(gif|jpg|png|js|css)$ {
       echo "规则D";
    }
    location ~* \.png$ {
       echo "规则E";
    }
    location /img {
        echo "规则Y";
    }
```

```
    location / {
       echo "规则F";
    }

    error_log  /mnt/log/nginx/localhost/error.log;
    access_log /mnt/log/nginx/localhost/access.log;
}
```

其他规则

- [ ] # 表示注释
- [ ] proxy_pass 将请求转发到下游地址 ★★★
- [ ] proxy_set_header 设置转发到下游地址的请求Header(头部)
- [ ] root 表示读取文件根目录 ★★
- [ ] index 表示默认读取文件 ★★

配置文件示例

```
server {
    listen 80;
    server_name localhost;

    location ^~ /api/v1 {
        proxy_set_header Host $host;
        proxy_pass http://127.0.0.1:6550;
    }

    location /static {
        # root 表示文件根目录
        # 如：/static/img/test.png -> /mnt/var/www/localhost/img/test.png
        root /mnt/var/www/localhost;
        # index 表示默认文件
        # 如：/static -> /mnt/var/www/localhost/index.html
        index index.html;
    }

    error_log  /mnt/log/nginx/localhost/error.log;
    access_log /mnt/log/nginx/localhost/access.log;
}

server {
    listen 80;
    server_name localhost2;

    location /api/v1 {
        proxy_set_header Host $host;
        proxy_pass http://127.0.0.1:7000;
    }

    error_log  /mnt/log/nginx/localhost2/error.log;
    access_log /mnt/log/nginx/localhost2/access.log;
}
```

第二阶段 - 反向代理服务器

模拟请求发送工具：Postman

Hello World ★★★

注：强烈建议先完成此任务

具体要求：在浏览器输入 http://127.0.0.1:3031/ping 浏览器展示Hello World 文字

示例代码：

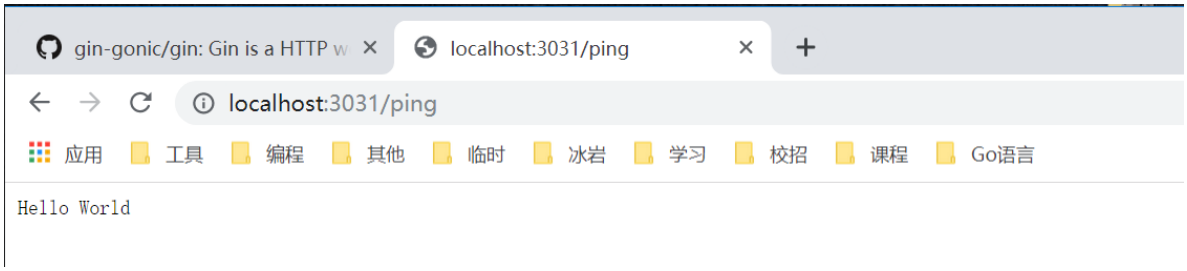
```
package main

import (
    "net/http"

    "github.com/gin-gonic/gin"
)

func main() {
    r := gin.Default()
    r.GET("/ping", func(c *gin.Context) {
        c.String(http.StatusOK, "Hello World")
    })
    _ = r.Run(":3031") // listen and serve on 0.0.0.0:3031
}
```

效果如下：



注：dns解析中 localhost -> 127.0.0.1（可查看本机hosts文件）

转发请求 ★★★

配置示例：

```
location /api/v1 {
    proxy_set_header Host $host;
    proxy_pass http://127.0.0.1:7000;
}
```

ps：前期可采用参数的形式或者用json形式配置文件直接使用json解析器解析即可

如：

config.json

```
{
    "location": [
        "/api/v1": {
            "proxy_pass": "http://127.0.0.1:7000",
            "proxy_set_header": {
                "Host": "test.com"
            }
        }
    ]
}
```

解析出参数之后，可根据配置将符合规则的请求转发到相应的下游地址。

对于上述配置解释如下：

假设请求如下：

```
Request URL: http://localhost/api/v1/healthy
Request Method: GET

Request Header:
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/77.0.3865.90 Safari/537.36
Accept: */*
...
```

那么反向代理服务器接收到这个请求之后将请求转发给`http://127.0.0.1:7000`，转发之后得到response（回应）之后再将其回应给请求方。

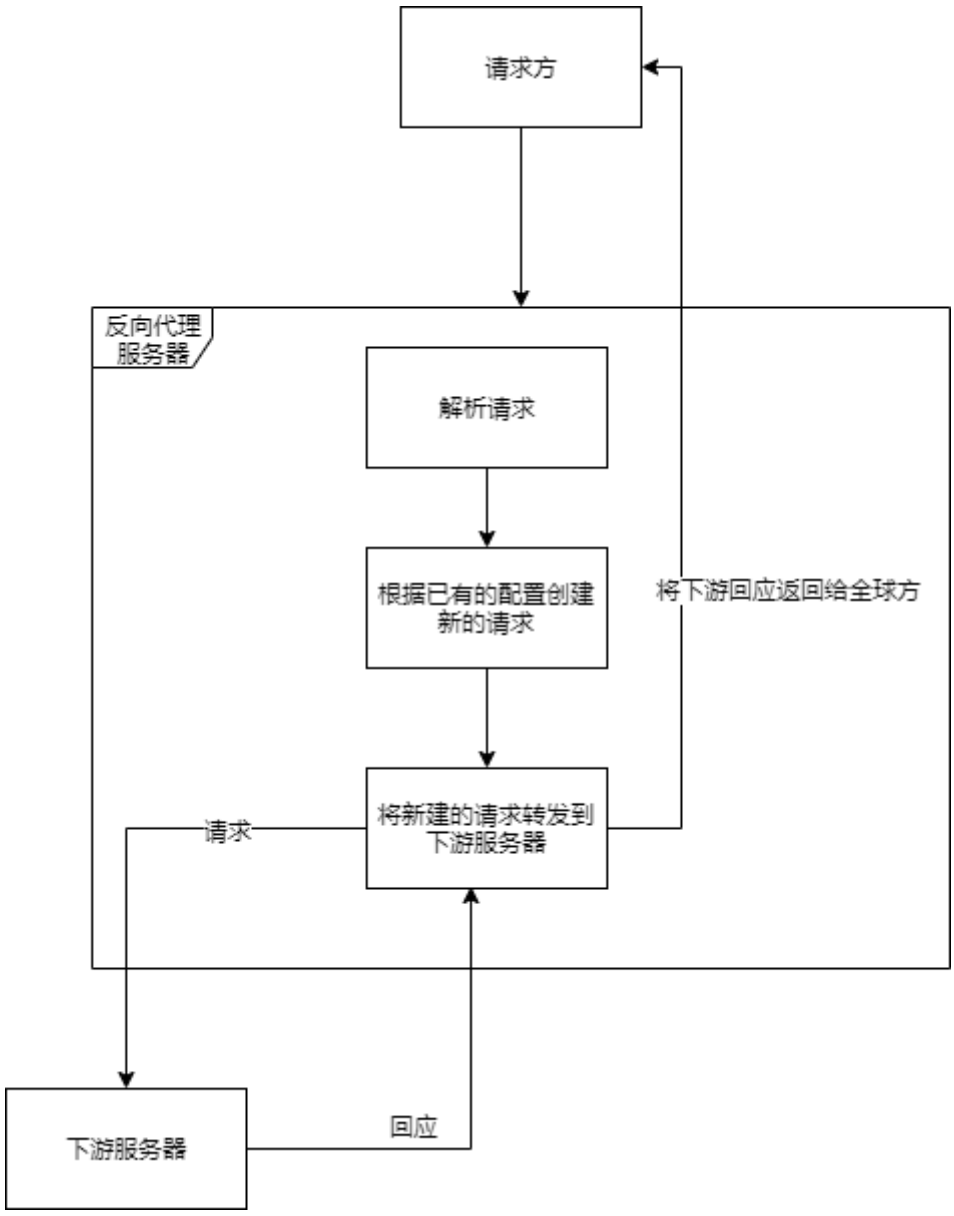
进行转发的请求如下：

```
Request URL: http://127.0.0.1:7000/api/v1/healthy
Request Method: GET

Request Header:
Host: test.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/77.0.3865.90 Safari/537.36
Accept: */*
...
```

注意上述请求中**Request URL**和**Host**与原始请求不同

示意图如下：



静态代理服务器 ★★

配置示例：

```
    location /static {
        # root 表示文件根目录
        # 如：/static/img/test.png -> /mnt/var/www/localhost/img/test.png
        root /mnt/var/www/localhost;
        # index 表示默认文件
        # 如：/static -> /mnt/var/www/localhost/index.html
        index index.html;
    }
```

ps：前期可采用参数的形式或者用 json 形式配置文件直接使用 json 解析器解析即可

如：

config.json

```
{
    "location": [
        "/static": {
            "root": "/mnt/var/www/localhost",
            "index": "index.html"
        }
    ]
}
```

假设请求文件为：http://localhost/static/img/test.png

那么依据规则，我们将提取服务器文件系统中路径为 `/mnt/var/www/localhost/img/test.png` 的文件，并返回给请求方。

注：这里只要求读取静态文件，并返回给请求方即可。

第三阶段 - 整合 ★★

将第一阶段和第二阶段所做的东西整合起来，即需要达到要求如下：

- [ ] 启动时根据路径解析配置文件，并进行语法错误检查
- [ ] 根据配置文件，提取必要参数启动反向代理服务
- [ ] 反向代理服务器处理请求并完成反向代理

参考资料

如何编写一个HTTP反向代理服务器

静态文件服务器实战 （文章到读取静态文件部分）

HTTP

Nginx 新手起步

nginx.conf Full Example

JSON 教程