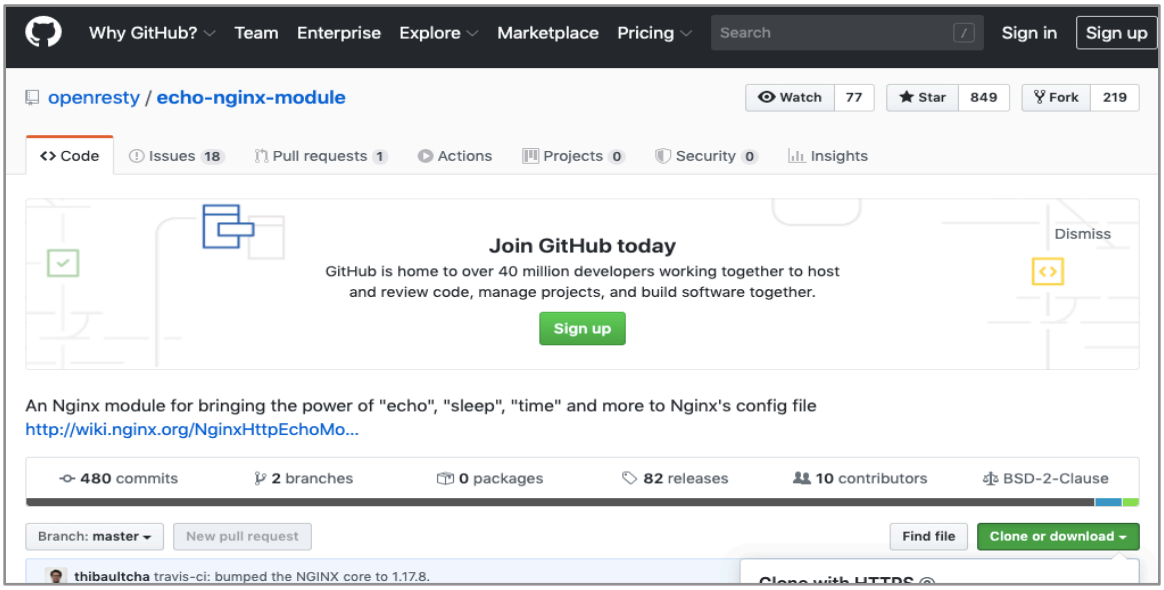
京峰教育 第十一次作業 – 0422 課程

作業 1：簡述 location 的常見規則優先級, 並逐個驗證

1.1、要驗證 location 常見規則優先級, 使用 echo 模塊較容易進行驗證, 由於 echo 模塊屬於第三方套件, 故需要進行額外新增, 先使用命令
git clone https://github.com/openresty/echo-nginx-module.git
複製到/usr/src/目錄下, 如圖一(a)、圖一(b)



圖一(a)、github 平台上 echo-nginx-module

```
[root@NGINX-170 src]# git clone https://github.com/openresty/echo-nginx-module.git
Cloning into 'echo-nginx-module'...
remote: Enumerating objects: 21, done.
remote: Counting objects: 100% (21/21), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 3018 (delta 10), reused 14 (delta 7), pack-reused 2997
Receiving objects: 100% (3018/3018), 1.15 MiB | 330.00 KiB/s, done.
Resolving deltas: 100% (1621/1621), done.
[root@NGINX-170 src]# ls
debug  echo-nginx-module  kernels  nginx-1.16.1  nginx-1.16.1.tar.gz
```

圖一(b)、git clone echo 模塊

1.2、重新預編譯 nginx, 新增 echo-nginx-module, 如圖二

```
[root@NGINX-170 nginx-1.16.1]# /usr/local/nginx/sbin/nginx -V
nginx version: JFWS/1.1
built by gcc 4.8.5 20150623 (Red Hat 4.8.5-39) (GCC)
configure arguments: --prefix=/usr/local/nginx --with-http_stub_status_module
[root@NGINX-170 nginx-1.16.1]# ./configure --prefix=/usr/local/nginx --with-http_stub_status_module --add-module=../echo-nginx-m
odule/
```

圖二、預編譯

1.3、編譯安裝, 如圖三

```
[root@NGINX-170 nginx-1.16.1]# make & make install
```

圖三、編譯及安裝

1.4、進行更新, 如圖四

```
[root@NGINX-170 nginx-1.16.1]# make upgrade
/usr/local/nginx/sbin/nginx -t
nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok
nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful
kill -USR2 `cat /usr/local/nginx/logs/nginx.pid`
sleep 1
test -f /usr/local/nginx/logs/nginx.pid.oldbin
kill -QUIT `cat /usr/local/nginx/logs/nginx.pid.oldbin`
[root@NGINX-170 nginx-1.16.1]# /usr/local/nginx/sbin/nginx -V
nginx version: JFWS/1.1
built by gcc 4.8.5 20150623 (Red Hat 4.8.5-39) (GCC)
configure arguments: --prefix=/usr/local/nginx --with-http_stub_status_module --add-module=../echo-nginx-module/
```

圖四、進行更新

1.5、location 是根據匹配的語法格式進行 URI 優先級的判斷

語法: location [= | ~ | ~* | ^~ | @] URI {

…

}

= : 精準匹配, 需要完全一模一樣的 URI 才會進行匹配

~ : 正則匹配, 符合正則的規則 URI 才會進行匹配

~* : 正則匹配, 忽略大小寫匹配 URI, 但必須要有實際的檔案, 否則會報錯

^~ : 前綴字符串匹配

根據優先級為下

=精準匹配 > ^~前綴字符匹配 > ~正則匹配 > 字符串

1.6、先測試 = > ^~ 優先級, 針對 echo URI 進行匹配, 新增配置如圖五(a), 測試結果驗證優先匹配 = 的精準匹配, 如圖五(b)

```
# [ = match ] - configuration A
location = /echo {
        echo "This is = /echo match!";
}

# [ ^~ match ] - configuration B
location ^~ /echo {
        echo "This is ^~ /echo match!";
}
```

圖五(a)、配置=及^~匹配

```
[root@NGINX-170 vhost]# curl http://10.99.128.170/echo
This is = /echo match!
```

圖五(b)、= 精準匹配優先級較^~為高

1.7、測試~正則匹配優先級較^~為低, 將=精準匹配配置區塊先行註釋(優先級最高), ^~於設定檔第二順位, 若~正則匹配優先級比^~為高, 則會優先打印出 This is ~ /echo match!, 配置如圖六(a)、實際打印出 This is ^~ /echo match!, 測試結果如圖六(b), 代表^~優先級> ~ 正則匹配優先級

```
# [ ~ match ] - configuration C
location ~ /echo {
        echo "This is ~ /echo match!";
}

# [ ^~ match ] - configuration B
location ^~ /echo {
        echo "This is ^~ /echo match!";
}

# [ = match ] - configuration A
location = /echo {
        echo "This is = /echo match!";
}
```

圖六(a)、配置正則匹配 configuration C

```
[root@NGINX-170 vhost]# curl http://10.99.128.170/echo
This is ^~ /echo match!
```

圖六(b)、^~優先級較正則為高

1.8、測試~正則匹配以及~*正則不區分大小寫比對優先級, 新增 configuration D 為~*正則不區分大小寫, 置於~正則匹配之上, 並註釋^~前綴匹配配置區塊, 如圖七(a), 若區分大小寫優先級較高, 則打印出 This is ~* /echo match! , 如圖七(b), 但實際~*正則匹配不區分大小寫及~正則匹配區分大小寫, 在配置檔對換位置, 測試結果如圖七(c), 則表示兩者優先級相同, 按照配置檔順序來決定優先級

```
        # [ ~* match ] - configuration D
        location ~* /echo {
                echo "This is ~* /echo match!";
        }

        # [ ~ match ] - configuration C
        location ~ /echo {
                echo "This is ~ /echo match!";
        }

#       # [ ^~ match ] - configuration B
#       location ^~ /echo {
#               echo "This is ^~ /echo match!";
#       }
```

圖七(a)、配置正則區分大小寫匹配~*

```
[root@NGINX-170 vhost]# curl http://10.99.128.170/echo
This is ~* /echo match!
```

圖七(b)、~*正則不匹配大小寫置於~正則匹配大小寫之上

```
[root@NGINX-170 vhost]# curl http://10.99.128.170/echo
This is ~ /echo match!
```

圖七(c)、~正則不匹配大小寫置於~*正則匹配大小寫之上

1.9、~正則匹配不區分大小寫, 則打印出 This is ~ /under-maintenance.png match!, 若是~*正則匹配區分大小寫, 則打印出 This is ~* /under-maintenance.png match!, 如圖八

```
[root@NGINX-170 vhost]# curl http://10.99.128.170/under-maintenance.png
This is ~ /under-maintenance.png match!
[root@NGINX-170 vhost]# curl http://10.99.128.170/under-mainTenance.png
This is ~* /under-maintenance.png match!
```

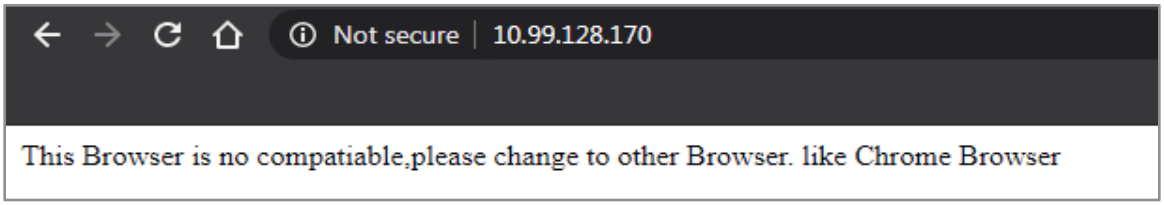
圖八、區分大小寫

作業 2：實現不同終端訪問不同域名的跳轉, 其中如果是 IE 瀏覽器訪問, 則提示此瀏覽器不兼容, 建議更換 chrome 瀏覽器繼續訪問

2.1、在/usr/local/nginx/conf/vhost/v1.jfedu.net.conf 下使用 if 判斷是否為 IE 瀏覽器(9|11), 主要判斷$http_user_agent 是否匹配 MISE|Trident, 如果匹配就 rewrite 到 ie.html 頁面顯示 This Browser is no compatiable,please change to other Browser. lik Chrome Browser, 如下圖九(a), 圖九(b)

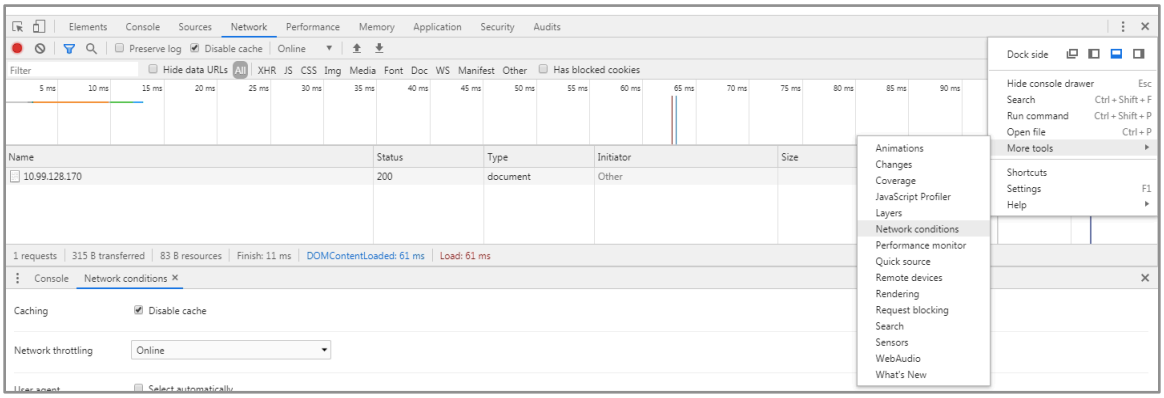
```
if ( $http_user_agent ~* (MSIE|Trident)) {
        rewrite  ^/(.*)$ /ie.html break;
        #return 403;
}
```

圖九、匹配 IE 瀏覽器 MSIE|Trident

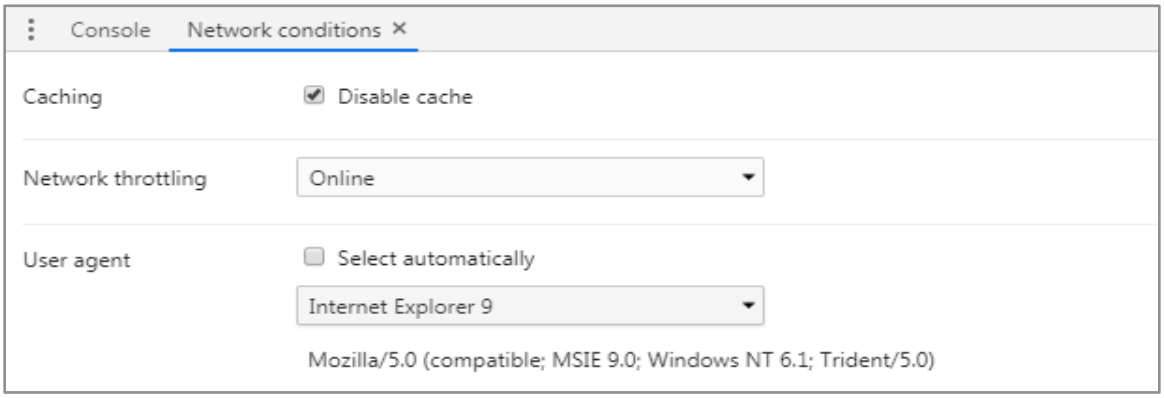


圖九、顯示瀏覽器不兼容

2.2、在瀏覽器端可以使用開發者工具(developer tools)進行模擬 IE 瀏覽器, 如圖十(a)、圖十(b)



圖十(a)、選擇 Network condition



圖十(b)、IE9 瀏覽器