# nginx rewrite执行顺序（二）

作者：jaz
原文链接：https://ld246.com/article/1453775512099
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

<h2>例题 1</h2>
配置：
error_log  logs/error.log info;
 
server {
        listen       9090;
        server_name  localhost;
        root html;
        rewrite_log on;
 
        rewrite "^/aaa\.html$"  /bbb.html;
 
        location  /ccc.html {
            rewrite "/ccc\.html$"  /eee.html;
        }
 
        location  /bbb.html {
            rewrite "/bbb\.html$" /ccc.html;
            rewrite "/ccc\.html$" /ddd.html;
        }  
}   
结果：
[root@web108 ~]# curl http://localhost:9090/aaa.html
ddd html file
[root@web108 ~]#
 
日志：
2011/08/08 10:05:41 [notice] 31592#0: *90 "^/aaa\.html$" matches "/aaa html" , client: 127.0.0.1, server: localhost, request: "GET /aaa.html HTTP/1.1", ost: "localhost:9090"
2011/08/08 10:05:41 [notice] 31592#0: *90 rewritten data: "/bbb.html"</ trong> , args: "", client: 127.0.0.1, server: localhost, request: "GET /aaa.html HTTP/1.1", ost: "localhost:9090"
2011/08/08 10:05:41 [notice] 31592#0: *90 "^/bbb\.html$" matches "/bb .html" , client: 127.0.0.1, server: localhost, request: "GET /aaa.html HTTP/1.1", host: "localhost:9090"
2011/08/08 10:05:41 [notice] 31592#0: *90 rewritten data: "/ccc.html"</s rong> , args: "", client: 127.0.0.1, server: localhost, request: "GET /aaa.html HTTP/1.1", h st: "localhost:9090"
2011/08/08 10:05:41 [notice] 31592#0: *90 "^/ccc\.html$" matches "/ccc. tml" , client: 127.0.0.1, server: localhost, request: "GET /aaa.html HTTP/1.1", h st: "localhost:9090"
2011/08/08 10:05:41 [notice] 31592#0: *90 rewritten data: "/ddd.html"</ trong> , args: "", client: 127.0.0.1, server: localhost, request: "GET /aaa.html HTTP/1.1", ost: "localhost:9090"
2011/08/08 10:05:41 [notice] 31592#0: *90 "^/aaa\.html$" does not mat h "/ddd.html" , client: 127.0.0.1, server: localhost, request: "GET /aaa.html HT P/1.1", host: "localhost:9090"
2011/08/08 10:05:41 [info] 31592#0: *90 client 127.0.0.1 closed keepalive connection</p

 
解释：
GET /aaa.html 请求，首先执行 server 级的 rewrite 指令，被重写为 /bbb.html ，然后匹配到 l
cation /bbb.html {} ，接着执行 location 级的 rewrite 指令，先重写为 /ccc.html ，再重写为 /ddd.
tml ；由于 URI 被 location 级的 rewrite 指令重写了，因此需要重新进行 location 的匹配，相当于
写后的 URI 被当做一个新的请求，会重新执行 server 级的 rewrite ，然后重新匹配 location ，日志&
dquo; 2011/08/08 10:05:41 [notice] 31592#0: *90 "^/aaa\.html$" does
ot match "/ddd.html" , client: 127.0.0.1, server: localhost, request: "GE
/aaa.html HTTP/1.1", host: "localhost:9090" ”体现了重新匹配 location 的流程。
 
<h2><a name="t3"></a>例题 2</h2>
配置：
error_log  logs/error.log info;
 
server {
        listen   &n
sp;  9090;
        server_name  localhost;
        root html;
        rewrite_log on;
 
         rewrite "^/aaa\.html$"  /bbb.h
ml;
           &
bsp;      rewrite "^/ccc\.html$"  /ddd.html;
        
         location  /bbb.html {
            rewrite "
/bbb\.html$" /ccc.html;
         }  
           &
bsp;       location  /ddd.html {
             rewrite
^/ddd\.html$" /eee.html;
         }
}   
结果：
[root@web108 ~]# curl http://localhost:9090/aaa.html
eee html file
[root@web108 ~]#
 
日志：
2011/08/08 10:21:00 [notice] 2218#0: *91 "^/aaa\.html$" matches "/aaa.
tml" , client: 127.0.0.1, server: localhost, request: "GET /aaa.html HTTP/1.1", h
st: "localhost:9090"
2011/08/08 10:21:00 [notice] 2218#0: *91 rewritten data: "/bbb.html"</st
ong> , args: "", client: 127.0.0.1, server: localhost, request: "GET /aaa.html HTTP/1.1", ho
t: "localhost:9090"
2011/08/08 10:21:00 [notice] 2218#0: *91 "^/ccc\.html$" does not
match "/bbb.html" , client: 127.0.0.1, server: localhost, request: "GET /
aa.html HTTP/1.1", host: "localhost:9090"
2011/08/08 10:21:00 [notice] 2218#0: *91 "^/bbb\.html$" matches "/bbb
html" , client: 127.0.0.1, server: localhost, request: "GET /aaa.html HTTP/1.1",

ost: "localhost:9090"
2011/08/08 10:21:00 [notice] 2218#0: *91 rewritten data: "/ccc.html"</st
ong> , args: "", client: 127.0.0.1, server: localhost, request: "GET /aaa.html HTTP/1.1", ho
t: "localhost:9090"
2011/08/08 10:21:00 [notice] 2218#0: *91 "^/aaa\.html$" does not
match "/ccc.html" , client: 127.0.0.1, server: localhost, request: "GET /a
a.html HTTP/1.1", host: "localhost:9090"
2011/08/08 10:21:00 [notice] 2218#0: *91 "^/ccc\.html$" matches
/ccc.html" , client: 127.0.0.1, server: localhost, request: "GET /aaa.html
HTTP/1.1", host: "localhost:9090"
2011/08/08 10:21:00 [notice] 2218#0: *91 rewritten data: "/ddd.ht
l" , args: "", client: 127.0.0.1, server: localhost, request: "GET /aaa.html
HTTP/1.1", host: "localhost:9090"
2011/08/08 10:21:00 [notice] 2218#0: *91 "^/ddd\.html$" matches "/ddd
html" , client: 127.0.0.1, server: localhost, request: "GET /aaa.html HTT
/1.1", host: "localhost:9090"
2011/08/08 10:21:00 [notice] 2218#0: *91 rewritten data: "/eee.htm
" , args: "", client: 127.0.0.1, server: localhost, request: "GET /aaa.html
TTP/1.1", host: "localhost:9090"
2011/08/08 10:21:00 [notice] 2218#0: *91 "^/aaa\.html$" does not
match "/eee.html" , client: 127.0.0.1, server: localhost, request: "GET /
aa.html HTTP/1.1", host: "localhost:9090"
2011/08/08 10:21:00 [notice] 2218#0: *91 "^/ccc\.html$" does not
match "/eee.html" , client: 127.0.0.1, server: localhost, request: "GET /
aa.html HTTP/1.1", host: "localhost:9090"
2011/08/08 10:21:00 [info] 2218#0: *91 client 127.0.0.1 closed keepalive connection
 
解释：
第一次迭代 location 匹配
GET /aaa.html，首先执行 server 级的重写，“ rewrite "^/aaa\.html$"  /bbb.h
ml ”把 /aaa.html 重写为 /bbb.html，但 /bbb.html 没匹配上“ rewrite "^/ccc\.htm
$"  /ddd.html ”，最终保留 /bbb.html； 接着，匹配 location /bbb.html {}，执行 l
cation 级的 rewrite 指令，把 /bbb.html 重写为 /ccc.html，由于 URI 被 location 级 rewrite 重写
因此需要重新迭代 location 匹配。
 
第二次迭代 location 匹配
对于第一次迭代结果 /ccc.html，首先依然是执行 server 级的 rewrite 指令，“ rewrite
^/aaa\.html$"  /bbb.html; ”跟 /ccc.html 不匹配，但“ rewrite "^/ccc\.html
"  /ddd.html; ”把 /ccc.html 重写为 /ddd.html； server 级 rewrite 执行完后，接着
ocation 匹配， /ddd.html 匹配到 location /ddd.html {}，执行 location 级的 rewrite 指令，把 /d
d.html 重写为 /eee.html 。同样由于 URI 被 location 级的 rewrite 指令重写，于是需要重新迭代 lo
ation 匹配。
 
第三次迭代 location 匹配
对于第二次迭代结果 /eee.html，首先依然执行 server 级的 rewrite 指令，“ rewrite "^
aaa\.html$"  /bbb.html; ”和“ rewrite "^/ccc\.html$"  /ddd.html; &r
quo;，只不过它们都没匹配上 /eee.html，接着 /eee.html 进行 location 匹配，也没有，最终结果是
/eee.html，返回“ eee html file ”页面。
 
 
最后说明下，如果把上述配置修改成server级rewrite和location的编辑顺序调整：
server {<br />        listen  &n
sp;    9090;<br />        serve
_name  localhost;<br />        root html;<

r />       rewrite_log on;<br /><br />             <br />       location  /bbb.html {<br />           rewrite "^/bb\.html$" /ccc.html;<br />       }   <br />        location  /ddd.html {<br />nbsp;           rewrite "^/ddd\html$" /eee.html;<br />       }
 
        rewrite "^/aaa\.html$"  /bbb.hml;<br />        rewrite "^/ccc\.html$"  /ddd.htm;<br />}
结果是不会受影响的，也就是说location匹配迭代总是先执行server级rewrite，再进行location配，再执行location级的rewrite，如果URI因location级rewrite指令重写，则需要进行下一次迭代。总的迭代次数不超过10次，否则nginx报500错误。
 
简单伪代码描述下rewrite执行过程：
 
boolean match_finish = false;<br />int match_count = 0;<br />while(!match_finish &amp& match_count < 10) {<br />        match_count++;<br />    （1）按编辑顺序执行server级的rewrite指令；<br /> &nsp;  （2）按重写后的URI匹配location；<br />    （3）<br /> nbsp;      String uri_before_location = uri;<br />  &nbp;     按编辑顺序执行location级的rewrite指令；<br />       String uri_after_location = rewrite(uri);<br />    &nsp;   if(uri_before_location != uri_after_location) {<br />    &bsp;       match_finish = false;     &bsp;      <br />        }else {<br />            match_finish = true;<br />        }<br />     nbsp;  if(location rewrite has last flag) {<br />     &nbp;     continue;//表示不执行后面的rewrite，直接进入下一次迭代<br />        }<br />        if(location rewrite has break flag) {<br />         &nbs;  break;//表示不执行后面的rewrite，并退出循环迭代<br />     nbsp;  }<br />}<br />if(match_count <= 10) {<br />    return HTP_200;<br />} else {<br />    return HTTP_500;<br />}