

# Nginx & Tomcat

## 一、Nginx

```
1 【nginx常用命令】
2 查看Nginx的版本号: nginx -v
3 启动Nginx: start nginx
4 快速停止或关闭Nginx: nginx -s stop
5 正常停止或关闭Nginx: nginx -s quit
6 配置文件修改重装载命令: nginx -s reload
7
8 【nginx结构目录说明】 https://www.cnblogs.com/liang-io/p/9340335.html
9 [root@www ~]# tree /application/nginx/
10 /application/nginx/
11 |-- client_body_temp
12 |-- conf                                # 这是Nginx所有配置文件的目录，极其重要
13 |   |-- fastcgi.conf                      # fastcgi相关参数的配置文件
14 |   |-- fastcgi.conf.default              # fastcgi.conf的原始备份
15 |   |-- fastcgi_params                   # fastcgi的参数文件
16 |   |-- fastcgi_params.default
17 |   |-- koi-utf
18 |   |-- koi-win
19 |   |-- mime.types                      # 媒体类型,
20 |   |-- mime.types.default
21 |   |-- nginx.conf                       # 这是Nginx默认的主配置文件
22 |   |-- nginx.conf.default
23 |   |-- scgi_params                     # scgi相关参数文件，一般用不到
24 |   |-- scgi_params.default
25 |   |-- uwsgi_params                    # uwsgi相关参数文件，一般用不到
26 |   |-- uwsgi_params.default
27 |   '-- win-utf
28 '-- fastcgi_temp                         # fastcgi临时数据目录
29 '-- html                                  # 这是编译安装时Nginx的默认站点目录，类似
30             Apache的默认站点htdocs目录
31 |   '-- 50x.html                         # 错误页面优雅替代显示文件，例如：出现502错误时会调用此页面
32     #      error_page  500$02503504  /50x.html;
33     '-- index.html                      # 默认的首页文件，首页文件名字是在nginx.conf中事先定义好的。
34 '-- logs
35   '-- access.log                      # 这是Nginx默认的日志路径，包括错误日志及访问日志
36   '-- error.log                       # 这是Nginx的错误日志文件，如果Nginx出现启动故障等问题，一定要看看这个错误日志
37   '-- nginx.pid                        # Nginx的pid文件，Nginx进程启动后，会把所有进程的ID号写到此文件
38 '-- proxy_temp
39 '-- sbin                               # 这是Nginx命令的目录，如Nginx的启动命令nginx
40   '-- nginx                            # Nginx的启动命令nginx
41 '-- scgi_temp
42 '-- uwsgi_temp                         # 临时目录
43
44 【以windows系统为例】
45 一、利用反向代理实现负载均衡
46 1. 下载 tomcat 和 nginx 包
47 2. 复制 tomcat 目录分别为 tomcat1 和 tomcat2 (启动之前 删除 CATALINA_HOME 配置, server.xml修改端口server.port和connector.port)
48 3. 用 eclipse 导出一个 简单应用 war 包, 命名 app.war
49 4. 将 app.war 放到 tomcat 的 webapp 下 (先清空所有文件)
50 5. localhost:8080/app 即可访问应用的首页
51 6. 修改 hosts 文件 (C:\Windows\System32\drivers\etc)，添加域名映射
52 127.0.0.1 router.com
53 127.0.0.1 web.com
54 127.0.0.1 static.com
55 7. web.com:8081/app 和 web.com:8082/app 访问首页
```

```

57 8. 在nginx下的conf下的nginx.conf
58 添加负载
59 http {
60     include      vhost/*.conf;
61     #gzip  on;
62     upstream web.com {
63         server web.com:8081 weight=1;
64         server web.com:8082 weight=1;
65     }
66 }
67
68 在nginx.conf同级目录添加vhost目录并添加www.conf文件，配置路由：
69 server {
70     listen 80;
71     server_name router.com;
72     location / {
73         proxy_pass http://web.com;
74         proxy_connect_timeout 500ms;
75     }
76 }
77 9. 在nginx目录下打开cmd, nginx -s reload 命令刷新配置(nginx之前已经是启动状态)
78 10. 访问 router.com/app 即可负载到不同的tomcat上
79
80 二、静态服务器
81 1. 创建静态资源目录 E:\test\static， 里面添加一些文本文件 app.txt 、 app.jpg 等
82 2. 在 vhost 下添加 static.conf 并添加配置
83 server {
84     listen 80;
85     server_name static.com;
86     location / {
87         root E:/test/static/;
88         index index.html;
89     }
90 }
91 3. nginx -s reload 命令刷新配置 或 重启nginx
92 4. 访问 static.com/app.jpg 即可访问该资源
93 5. app应用中<img src= "http://static.com/app.jpg" /> 即可引用该资源
94
95
96 【原理解释】
97 hosts地址映射
98 127.0.0.1 static.com
99 127.0.0.1 web.com
100 127.0.0.1 router.com
101 【static.com/app.jpg】 ==>nginx静态引用==> 【E:/test/static/app.jpg】
102 【router.com】 ==>nginx代理==> 【 web.com】 ==>nginx负载==>【web.com:8081 || web.com:8082】 ==>tomcat应用==> 【index.html】 =
103
104 【linux部署前端vue项目】
105 一、nginx安装见docker安装nginx
106 二、nginx.conf的配置
107 1. 将前端打的dist.zip包放入到html目录下，并通过unzip命令解压
108 2. 如前端项目访问接口配置已抽离出文件serverconfig.json，修改里面的后台接口服务访问地址
109 3. 由于nginx.conf中导入了include /etc/nginx/conf.d/*.conf，故修改conf.d下的default.conf即可
110 指定自定义的index.html
111 location / {
112     root /usr/share/nginx/html/dist/;           //原来为/usr/share/nginx/html/
113     index index.html index.htm;
114 }
115 4. 重新启动nginx镜像使配置生效

```

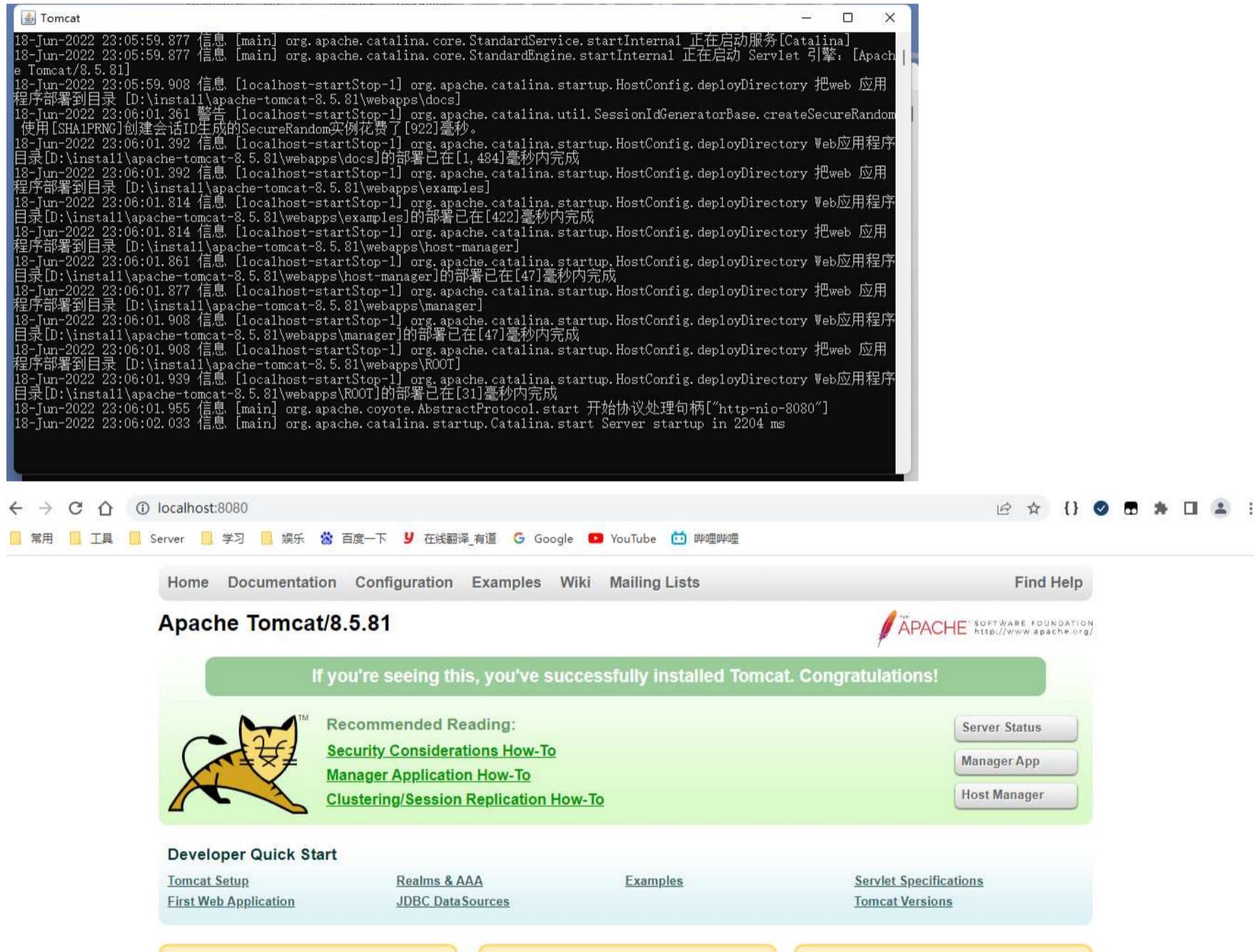
```
117
118 【docker安装Nginx】
119 1. 简单启动容器
120 docker run --name nginx -p 80:80 -d nginx
121
122 2. 查看容器目录，并创建挂载目录
123 // docker exec -it nginx bash
124 mkdir -p /home/docker/nginx/{conf,html}
125
126 3. 复制容器文件到挂载目录
127 docker cp nginx:/etc/nginx/nginx.conf /home/docker/nginx/conf;
128 docker cp nginx:/etc/nginx/conf.d /home/docker/nginx/conf;
129 docker cp nginx:/usr/share/nginx/html/50x.html /home/docker/nginx/html;
130 docker cp nginx:/usr/share/nginx/html/index.html /home/docker/nginx/html
131
132 4. 删除容器，并挂载新启动容器
133 // docker stop nginx; docker rm nginx
134 // html上面加 <meta charset="UTF-8"> 配合-e使用，解决页面中文乱码问题
135 docker run --name nginx --net host \
136 -e LANG=C.UTF-8 \
137 -e LC_ALL=C.UTF-8 \
138 -v /etc/localtime:/etc/localtime \
139 -v /home/docker/nginx/html:/usr/share/nginx/html \
140 -v /home/docker/nginx/conf/nginx.conf:/etc/nginx/nginx.conf \
141 -v /home/docker/nginx/conf/conf.d:/etc/nginx/conf.d \
142 -v /home/docker/nginx/logs:/var/log/nginx \
143 --restart always \
144 -d nginx:latest
145
146 5. 访问
147 http://10.207.0.167
148
149 6. 配置说明
150 [主配置 nginx.conf]
151 # 全局块(全局指令)
152 user nginx; # 用户组
153 worker_processes auto; # 自动生成工作进程
154 error_log /var/log/nginx/error.log notice; # 日志保存路径
155 pid /var/run/nginx.pid; # 进程保存路径
156 # events块(服务器或与用户的网络连接)
157 events {
158     worker_connections 1024; # 最大连接数，这是指一个子进程最大允许连1024个连接， 默认为512
159     accept_mutex on; # 设置网路连接序列化，防止惊群现象发生， 默认为off
160     on multi_accept on; # 设置一个进程是否同时接受多个网络连接， 默认为off
161     use epoll; # 事件驱动模型， select|poll|kqueue|epoll|resig|/dev/poll|eventport
162 }
163 # http块(可以嵌套多个server，配置代理，缓存，日志定义等绝大多数功能和第三方模块的配置)
164 http {
165     include /etc/nginx/mime.types; # 文件扩展名与文件类型映射表
166     default_type application/octet-stream; # 默认文件类型， 默认为text/plain
167     # 自定义日志格式
168     log_format main '$remote_addr - $remote_user [$time_local] "$request" '
169                 '$status $body_bytes_sent "$http_referer" '
170                 '"$http_user_agent" "$http_x_forwarded_for"';
171     access_log /var/log/nginx/access.log main; # off则取消服务日志
172     sendfile on; # 允许sendfile方式传输文件， 默认为off， 可以在http块， server块， location块
173     sendfile_max_chunk 100k; # 每个进程每次调用传输数量不能大于设定的值， 默认为0， 即不设上限。
174     #tcp_nopush on;
175     keepalive_timeout 65; # 连接超时时间， 默认为75s， 可以在http， server， location块
176     client_max_body_size 50M; # 上传大小限制
```

```
177 #gzip on;
178
179 # 路由配置
180 upstream myserver {
181     server 127.0.0.1:7878;
182     server 192.168.10.121:3333 backup; # 热备
183 }
184 error_page 404 https://www.baidu.com; # 错误页面配置
185
186 # server块
187 server {
188     keepalive_requests 120; # 单连接请求上限次数。
189     listen 80; # 监听端口
190     server_name abc.com; # 监听域名或ip
191     #rewrite ^(.*)$ https://${server_name}$1 permanent; # http请求重定向到https上
192     location / { # 请求的url过滤, 正则匹配, ~为区分大小写, ~*为不区分大小写。
193         root /var/www/abc.com; # 根目录
194         index index.html; # 设置默认页
195         proxy_pass http://myserver; # 请求转向myserver 定义的服务器列表
196         deny 127.0.0.1; # 拒绝的ip
197         allow 172.18.5.54; # 允许的ip
198     }
199 }
200
201 # HTTPS server
202 server {
203     listen 443 ssl;
204     server_name localhost; # 需要访问的域名, 这里也不用加https
205
206     # cd /data/nginx/cert 挂载到 /etc/nginx/cert
207     # 生成证书(公钥) openssl req -new -x509 -key nginx.key -out nginx.crt
208     # 生成私钥 openssl genrsa -out nginx.key
209     ssl on;
210     ssl_certificate /etc/nginx/cert/nginx.crt; # 指定证书位置
211     ssl_certificate_key /etc/nginx/cert/nginx.key; # 指定私钥位置
212     ssl_session_cache shared:SSL:1m;
213     ssl_session_timeout 5m;
214     ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE:ECDSA:HIGH:!NULL:!aNULL:!MD5:!ADH:!RC4;
215     ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
216     ssl_prefer_server_ciphers on;
217
218     location / {
219         proxy_pass http://需要访问的域名:8080/;
220     }
221 }
222
223 # 导入其他server块
224 include /etc/nginx/conf.d/*.conf;
225 }
226
227 [分配给 conf.d/default.conf]
228 # server块(配置虚拟主机的相关参数)
229 server {
230     listen 80;
231     listen [::]:80;
232     server_name localhost;
233
234     #access_log /var/log/nginx/host.access.log main; # 每个host对应日志路径
235
236     # location块(配置请求的路由, 以及各种页面的处理情况)
```

```
237 location / {
238     root /usr/share/nginx/html;
239     index index.html index.htm;
240 }
241
242 #error_page 404 /404.html;
243
244 # redirect server error pages to the static page /50x.html
245 #
246 error_page 500 502 503 504 /50x.html;
247
248 location = /50x.html {
249     root /usr/share/nginx/html;
250 }
251
252 # proxy the PHP scripts to Apache listening on 127.0.0.1:80
253 #
254 #location ~ \.php$ {
255 #    proxy_pass http://127.0.0.1;
256 #}
257
258 # 代理(示例)
259
260 location /test/ {
261     proxy_pass http://192.111.90.111:18901;
262     proxy_set_header Host $host:$proxy_port;
263     proxy_set_header X-Real-IP $remote_addr;
264     proxy_set_header REMOTE-HOST $remote_addr;
265     proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
266 }
267
268 # 负载均衡(示例)
269
270 location /app {
271     proxy_pass http://myserver;
272 }
273
274 upstream myserver{
275     ip_hash;
276
277     # 后端节点30秒内出现2次不可用情况，判定节点不可用。判定不可用后30秒内请求不会转发到此节点，直到30秒后重新检测节点健康情况
278     server 192.111.90.11:28095 weight=10 max_fails=2 fail_timeout=30s;
279     server 192.111.90.12:28095 weight=20 max_fails=2 fail_timeout=30s;
280 }
281
282
283
284
285
286
287
288 # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
289 #
290 #location ~ \.php$ {
291 #    root html;
292 #    fastcgi_pass 127.0.0.1:9000;
293 #    fastcgi_index index.php;
294 #    fastcgi_param SCRIPT_FILENAME /scripts$fastcgi_script_name;
295 #    include fastcgi_params;
296 #}
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394 }
```

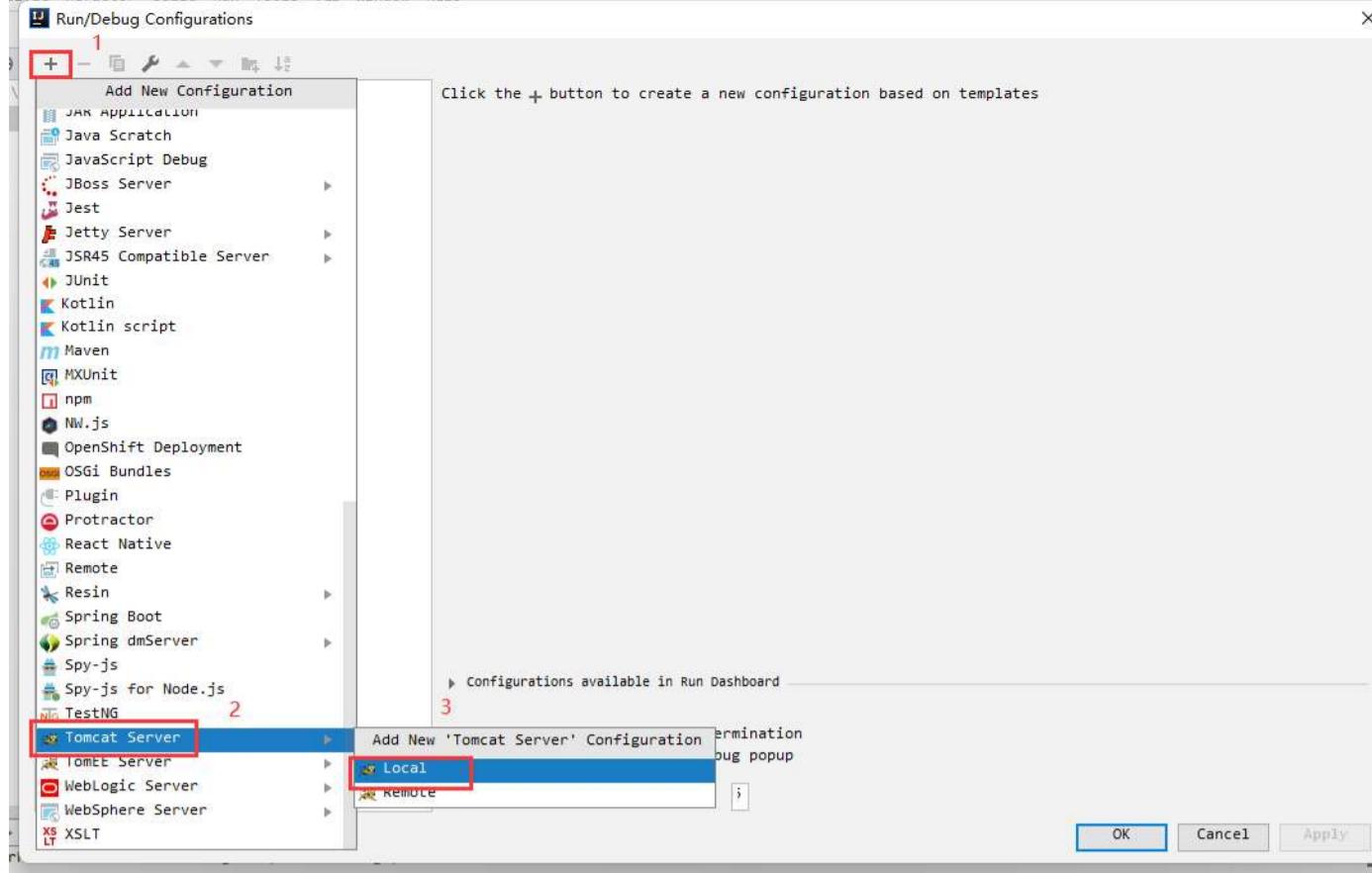
## 二、Tomcat

```
1 1. https://tomcat.apache.org/download-80.cgi 下载tomcat包，并配置环境变量
2 CATALINA_HOME = tomcat路径
3 %CATALINA_HOME%\bin
4
5 2. 验证
6 cmd进入dos窗口，startup 启动。localhost:8080 访问
7
8 3.cmd启动乱码问题
9 由于cmd默认编码是GBK，而Tomcat的默认编码是 UTF-8。打开Tomcat目录下的conf目录logging.properties文件设置
10 java.util.logging.ConsoleHandler.encoding = GBK
```

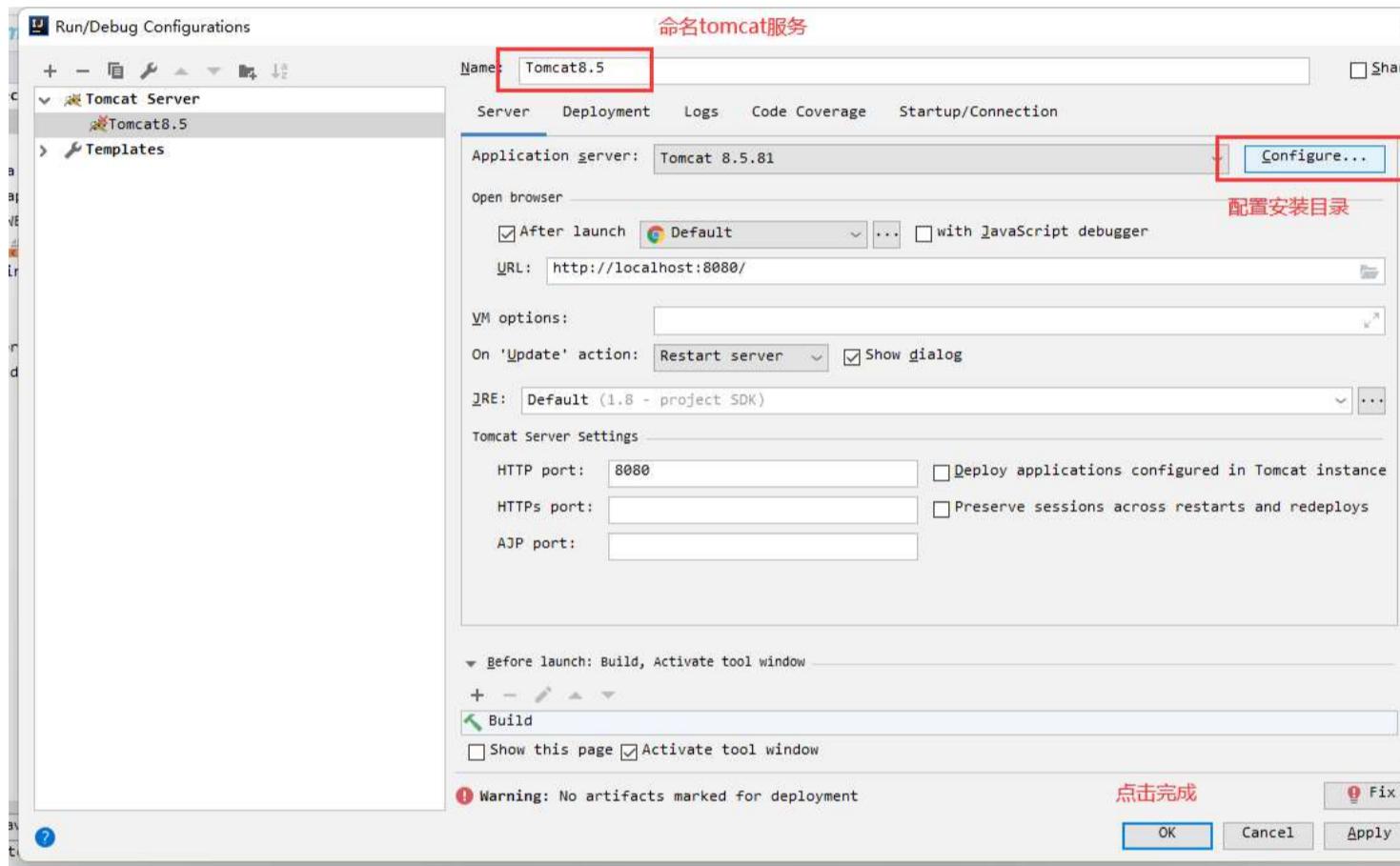


## 三、IDEA配置Tomcat及启动项目

1.idea工具栏 - Add Configuration - add



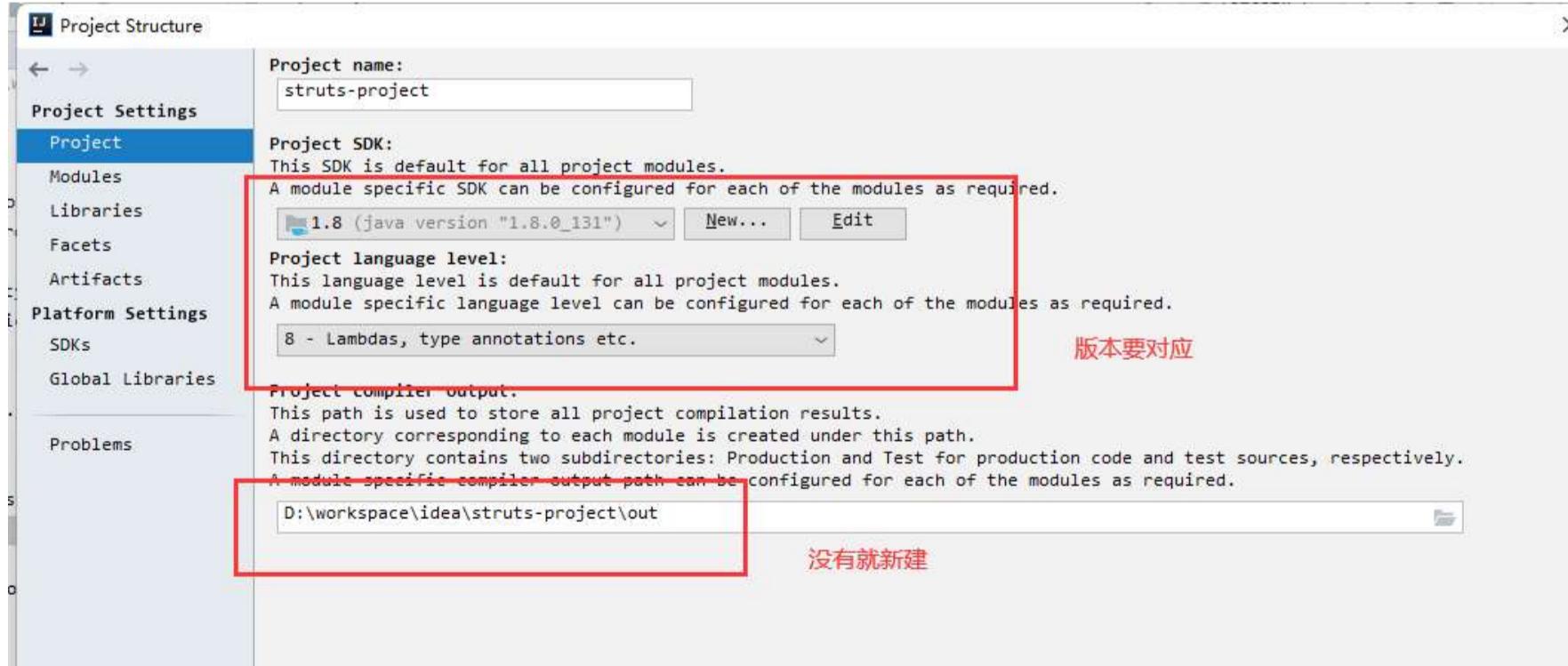
## 2. 配置tomcat目录



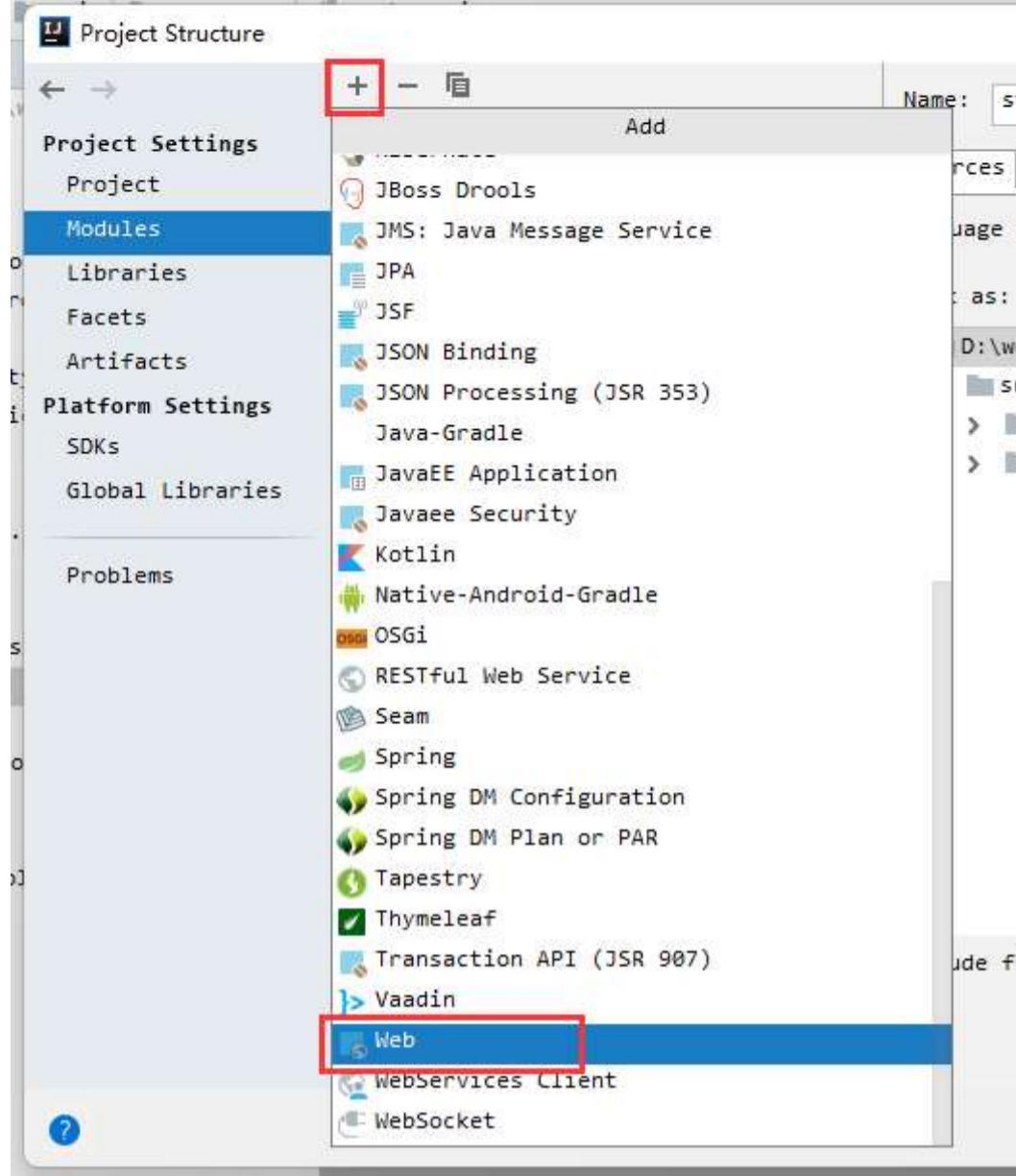
## 3. 项目配置

### File - Project Structure

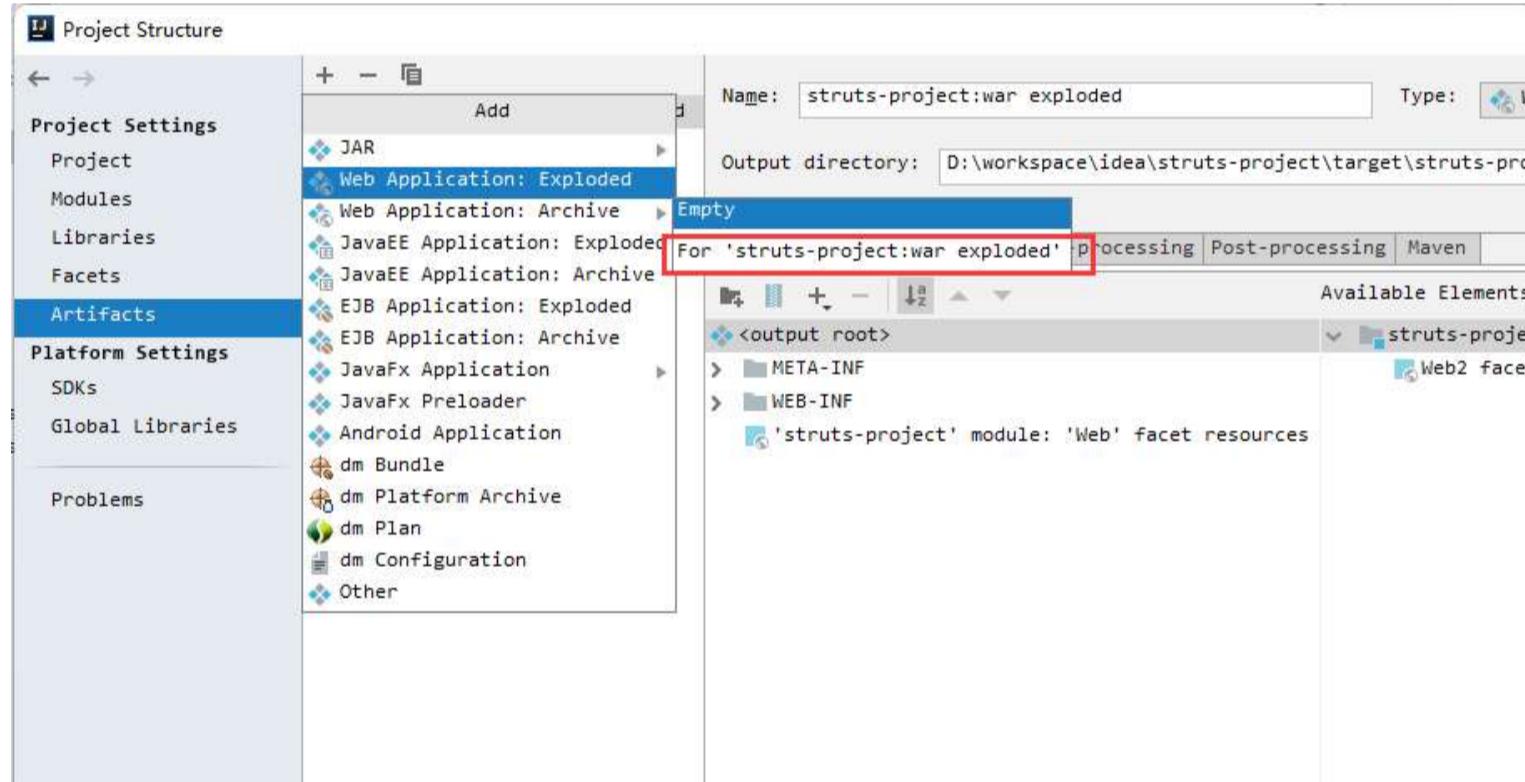
#### - Project



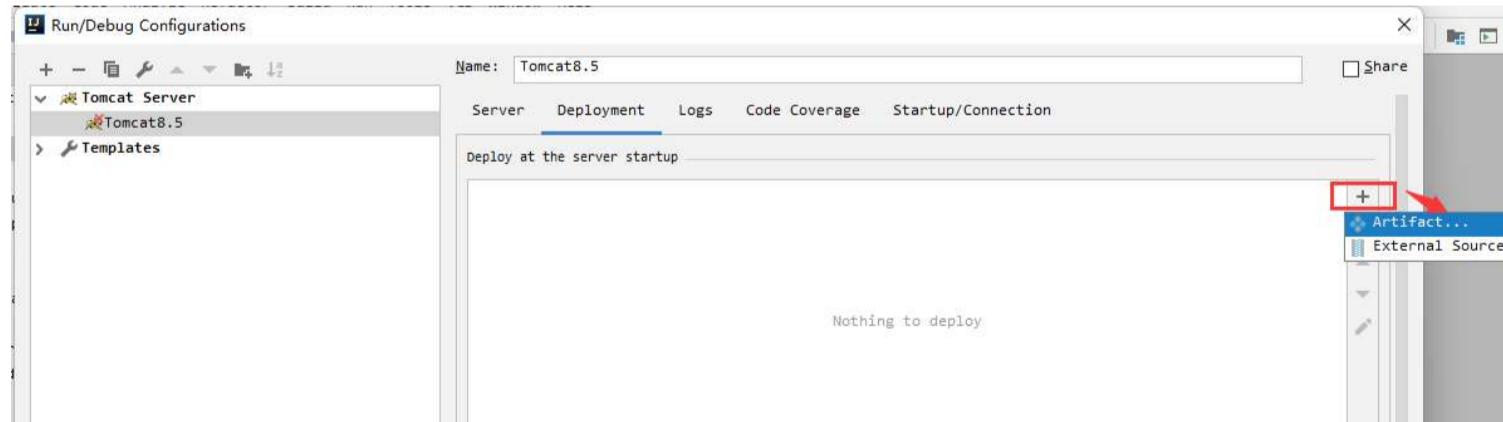
- Modules      path指定项目webapp目录下的web.xml

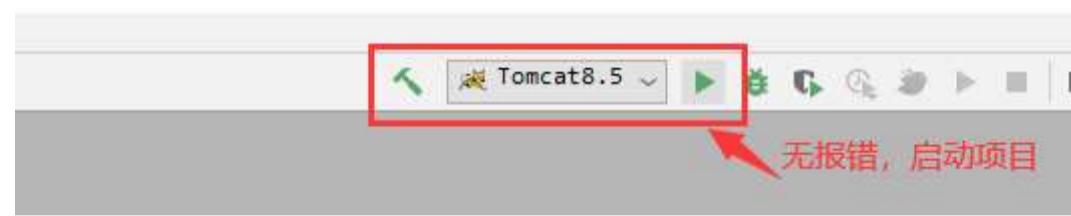
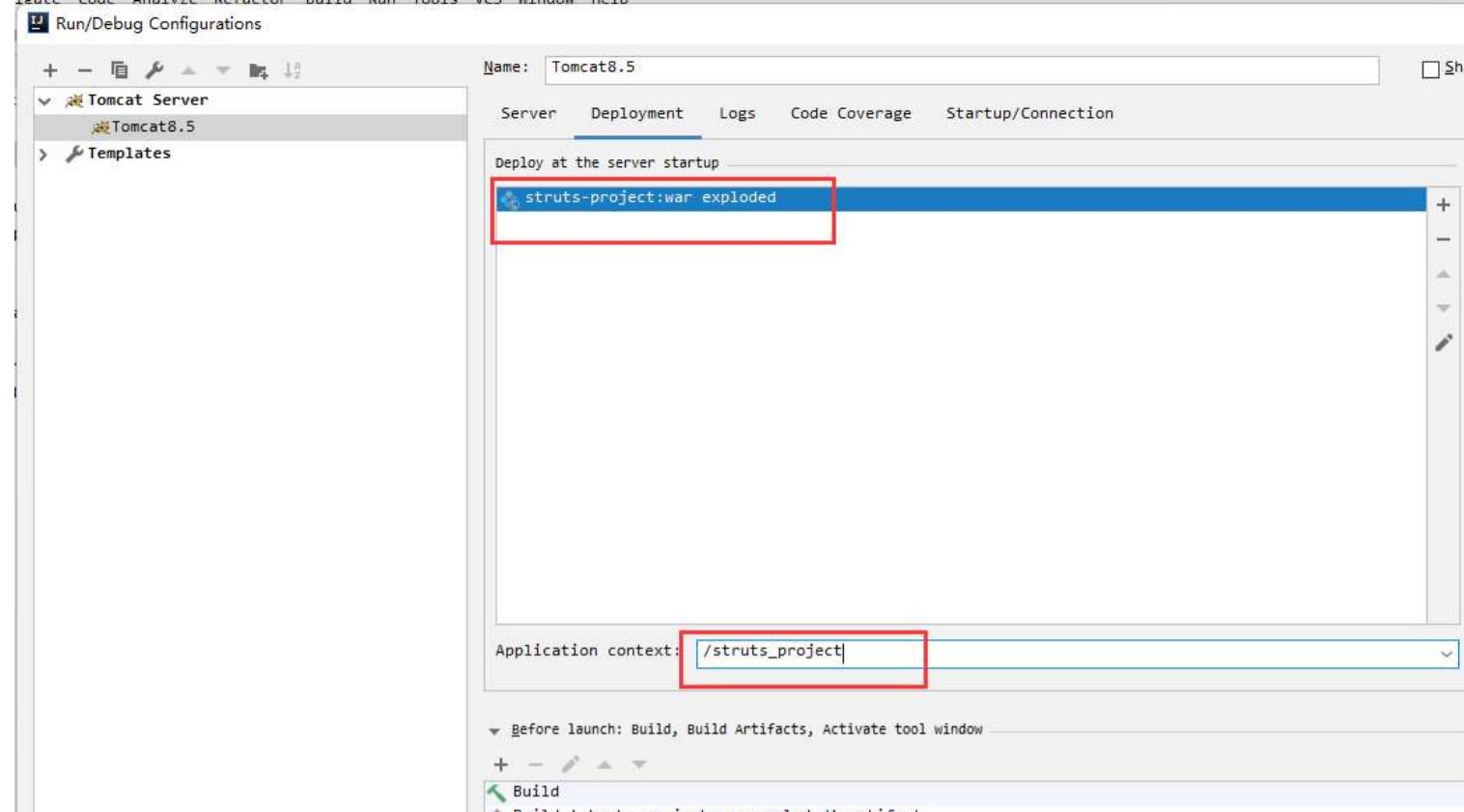


#### - Artifacts

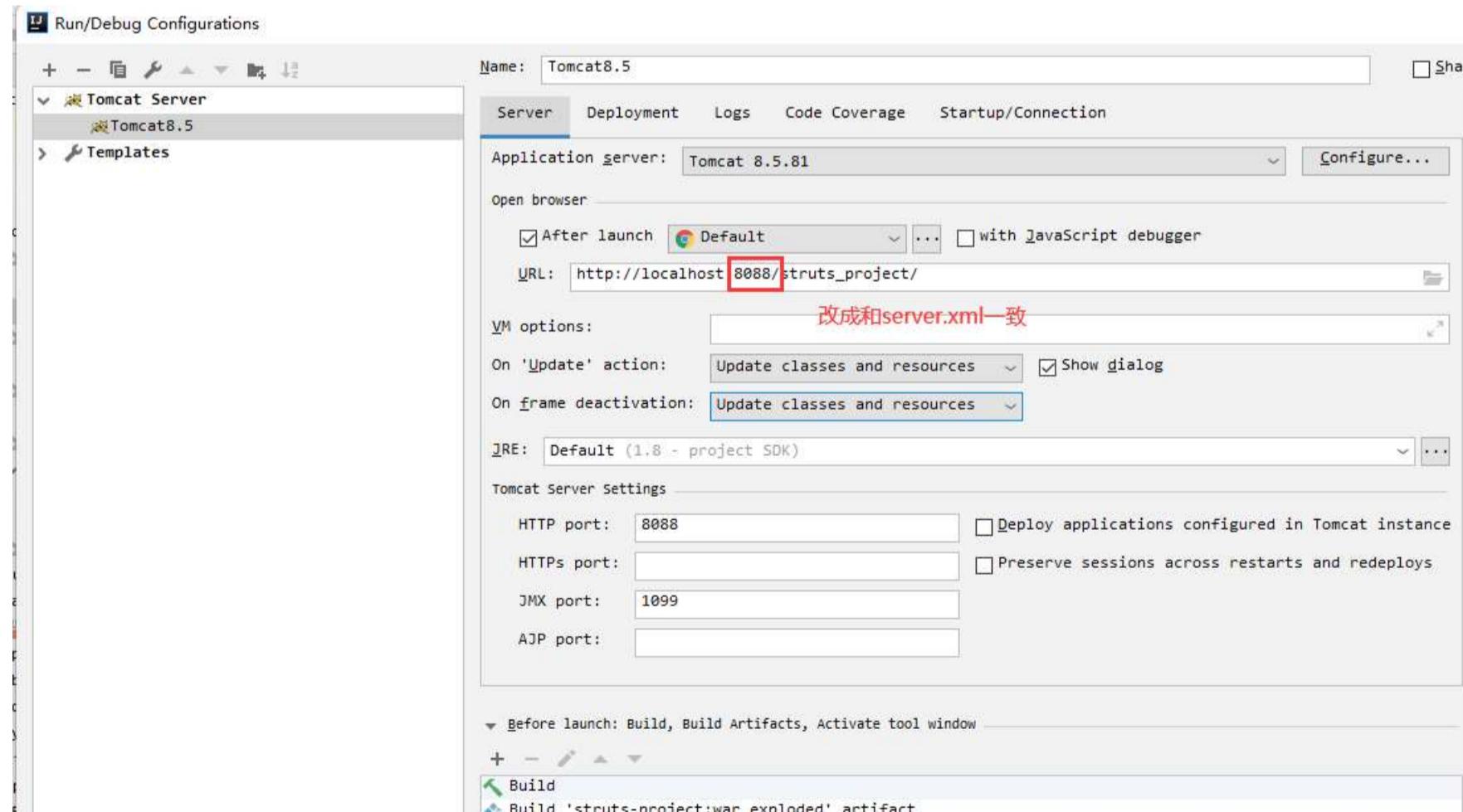


#### 4.tomcat绑定项目





```
61
62  <!-- A "Connector" represents an endpoint by which requests are received
63  and responses are returned. Documentation at :
64      Java HTTP Connector: /docs/config/http.html
65      Java AJP  Connector: /docs/config/ajp.html
66      APR (HTTP/AJP) Connector: /docs/apr.html
67      Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
68  -->
69  <Connector port="8088" protocol="HTTP/1.1"
70          connectionTimeout="20000"
71          redirectPort="8443" />
72  <!-- A "Connector" using the shared thread pool -->
73  <!--
74  <Connector executor="tomcatThreadPool"
75          port="8080" protocol="HTTP/1.1"
76          connectionTimeout="20000"
77          redirectPort="8443" />
78  -->
79  <!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443 -->
```



```
talina.startup.VersionLoggerListener.log 命令行参数: -Dcatalina.base=C:\Users\Administrator\IntelliJIdea2019.1\system\tomcat\Unnamed_struts-project
talina.startup.VersionLoggerListener.log 命令行参数: -Dcatalina.home=D:\install\apache-tomcat-8.5.81
talina.startup.VersionLoggerListener.log 命令行参数: -Djava.io.tmpdir=D:\install\apache-tomcat-8.5.81\temp
talina.core.AprLifecycleListener.lifecycleEvent 使用APR版本[1.7.0]加载了基于APR的Apache Tomcat本机库[1.2.33].
talina.core.AprLifecycleListener.lifecycleEvent APR功能: IPv6[true], sendfile[true], accept filters>false, random>true, UDS {[4]}.
talina.core.AprLifecycleListener.lifecycleEvent APR/OpenSSL配置: useAprConnector>false, useOpenSSL>true
talina.core.AprLifecycleListener.initializeSSL OpenSSL成功初始化 [OpenSSL 1.1.10 3 May 2022]
yote.AbstractProtocol.init 初始化协议处理器 ["http-nio-8088"]
talina.startup.Catalina.load Initialization processed in 1036 ms
talina.core.StandardService.startInternal 正在启动服务[Catalina]
talina.core.StandardEngine.startInternal 正在启动 Servlet 引擎: [Apache Tomcat/8.5.81]
yote.AbstractProtocol.start 开始协议处理句柄["http-nio-8088"]
talina.startup.Catalina.start Server startup in 133 ms

no available Artifact is being deployed, please wait
```

