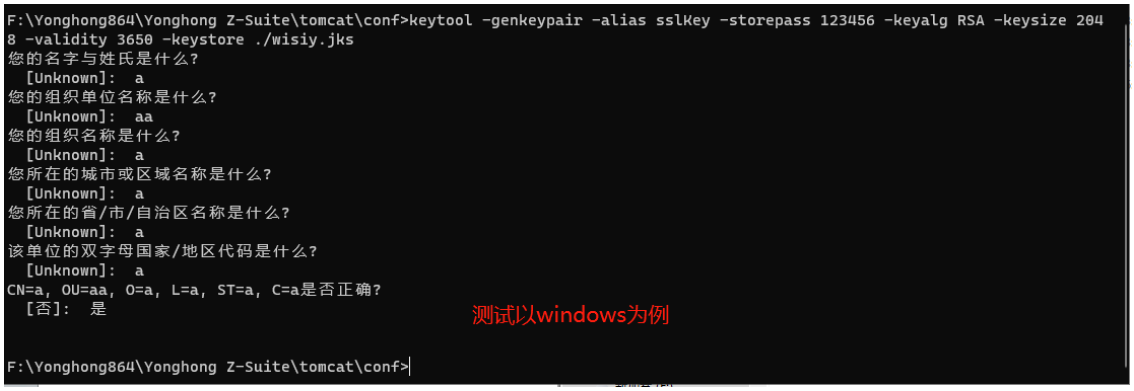
# 产品关于HTTPS的配置

> Tomcat版本：apache-tomcat-9.0.44
>
> > 测试时，Tomcat使用的证书是JDK中的keytool工具生成的自签名证书
>
> JDK版本：JDK-11.0.10
>
> Nginx版本：1.19.9
>
> > 测试时，Nginx使用的证书由OpenSSL生成，需要先安装OpenSSL，Linux使用yum安装(Redhat)或apt安装(Ubantu)，windows安装后续说明。

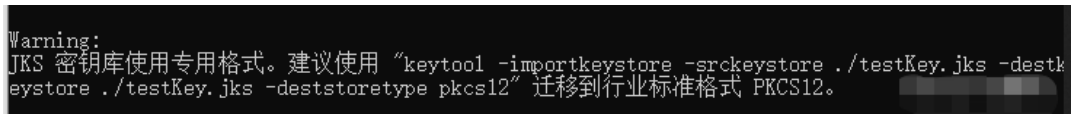
## Tomcat配置https

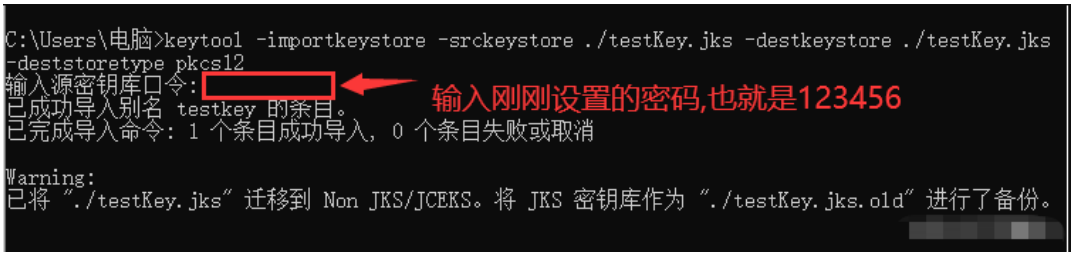
1. 首先确保本地有jdk的环境，可以使用java -version来查看，如果配置了环境变量，那就直接在产品安装目录/tomcat/conf目录下打开cmd，Linux环境下就进入该目录，之后运行以下命令：

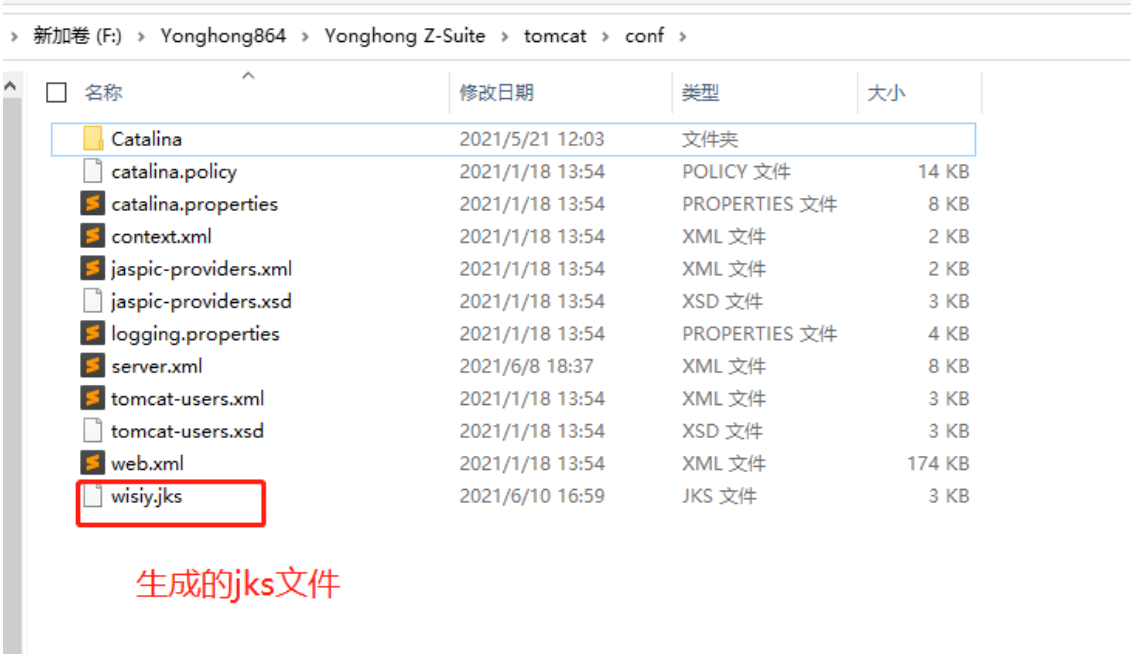
```
keytool -genkeypair -alias sslKey -storepass 123456 -keyalg RSA -keysize 2048 -validity 3650 -keystore ./wisiy.jks
/*
-alias testKey：证书项的名字，必填项
-keyalg RSA：证书签名算法，tomcat建议RSA
-storepass 123456：密钥库密码，也就是等下要生成的test.keystore的访问密码，妥善保管
-validity 3650：证书有效期，3650天，即10年
-keystore ./wisiy.jks：要生成的文件的位置，./test.keystore表示存储在当前目录下
*/
```




   - 如果有提示如下：按照信息运行里面的代码即可，这个过程会使用到刚刚的密码






> PS：也是可以使用OpenSSL来生成这个证书和密钥文件的，步骤如下：
>
> 1. 先用OpenSSL生成一个证书链文件(.crt)和密钥文件(.key)
> 2. 导出p12文件，参考如下：先参考文章中关于OpenSSL生成证书的内容，再执行以下内容

```
openssl pkcs12 -export -in server.crt -inkey server.key -out server.p12 -name server
```

```
keytool -importkeystore -srckeystore server.p12 -srcstoretype PKCS12 -destkeystore server.jks
```

2. 配置tomcat的https信息

> 注意事项：由于tomcat不能默认支持cer证书文件和key证书密钥文件，需要提前通过这两个文件生成tomcat可识别的p12等文件，详情参考链接：OpenSSL 把cer证书链以及key文件生成keystore

- 如果没有在产品安装目录/tomcat/conf目录下生成jks文件，那就事先将jks密钥库文件copy到此目录下
- 然后修改server.xml配置文件，找到https的Connector连接器，作如下修改

```
<Connector port="8090" protocol="HTTP/1.1"
           connectionTimeout="20000"          这里是http的连接器，如果不需要http连接可以注释掉
           redirectPort="8443" />
<!-- A "Connector" using the shared thread pool-->
<!--
<Connector executor="tomcatThreadPool"
           port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
-->
<!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443
     This connector uses the NIO implementation. The default
     SSLImplementation will depend on the presence of the APR/native
     library and the useOpenSSL attribute of the
     AprLifecycleListener.
     Either JSSE or OpenSSL style configuration may be used regardless of
     the SSLImplementation selected. JSSE style configuration is used below.
-->
这里是https的连接器，在文件中原来是被注释掉的，可以去掉注释或者手动填写
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
           maxThreads="150" SSLEnabled="true">
    <SSLHostConfig>
        <Certificate certificateKeystoreFile="conf/tomcatHttpsKeys.jks"   这个是密钥库jks文件
                     certificateKeystorePassword="123456"
                     type="RSA" />                                        这个填写前文中的密码
    </SSLHostConfig>
</Connector>
```

> 修改https连接器的端口号(如果URL不想带端口号,则修改成https的默认端口443即可,切记不能设置为80,否则无法访问,因为80是http的默认端口)、秘钥库文件路径,及配置秘钥的口令(就是生成秘钥文件时设置的口令)，pfx格式是同样的设置方式,只需修改证书路径和密码即可。

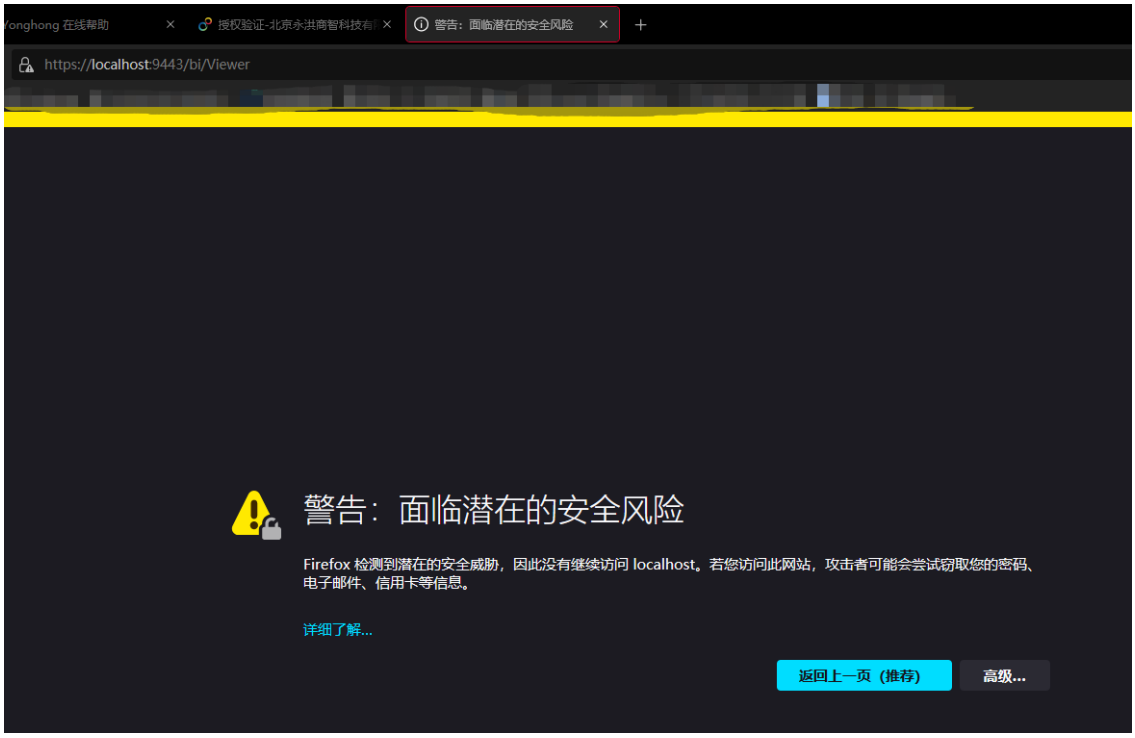
- 如果想要强制所有访问都通过https访问需要在：tomcat/conf/web.xml文件中加上以下内容

> 文件中原本的`<login-config>`需要注释掉

```
<login-config>
    <!-- Authorization setting for SSL -->
   <auth-method>CLIENT-CERT</auth-method>
    <realm-name>Client Cert Users-onlyArea</realm-name>
</login-config>
<security-constraint>
    <!-- Authorization setting for SSL -->
    <web-resource-collection >
        <web-resource-name>SSL</web-resource-name>
       <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <user-data-constraint>
       <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

3. 验证

   第一次进入系统时会提示安全信息，点击高级，忽略即可进入页面

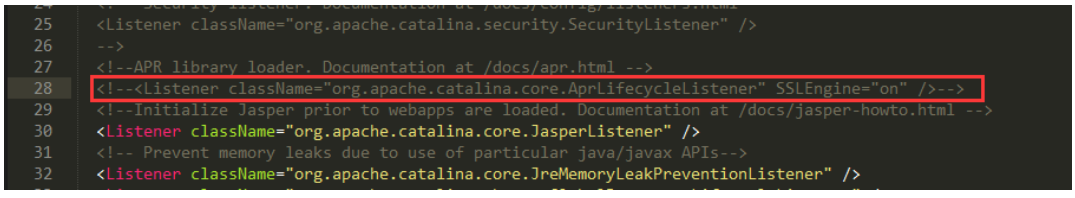



> 访问到这样的页面时，证明https已经配置成功了，此时的地址需要写全地址（加上/Viewer）否则会返回下面这样的错误
>
> 
> 

如果启动日志有报错，查看一下报错的关键字，一般有两种：

1. 报错：Failed to initialized connector

   这种，请注释tomcat/conf/server.xml文件中的SSlEngine配置内容




2. 报错关键字中提示ARP错误（这个错误很少见，一般是配置修改的地方不对才会出现）

   这种请修改tomcat/conf/server.xml关于protocol的内容，原本如果是Http11AprProtocol，修改为Http11NioProtocol

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
           maxThreads="150" SSLEnabled="true" >
    <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
    <SSLHostConfig>
        <Certificate certificateKeystoreFile="conf/tomcatHttpsKeys.jks"
                     certificateKeystorePassword="123456"
                     type="RSA" />
    </SSLHostConfig>
</Connector>
```

## Nginx配置https

> 关于Nginx的安装此处不做解释，有兴趣请参考Nginx配置文章。此处着重讲解如何利用OpenSSL生成自签名的证书以及如何配置Nginx的https和做跨协议转发。

1. 安装自签名的证书

```
# 1.安装 openssl
[root@localhost ~]$ yum install mod_ssl openssl
# 2：切换至nginx目录创建ssl文件夹（也可以在任何地方创建ssl文件夹，看自己的选择）
[root@localhost ~]$ cd /usr/nginx/nginx-1.19.9/conf/
[root@localhost conf]$ mkdir ssl
[root@localhost conf]$ cd ssl/
[root@localhost ssl]$
# 3：生成2048位秘钥
[root@localhost ssl]$ openssl genrsa -out server.key 2048
# 4：生成证书签名请求（CSR），这里需要填写许多信息，按照上面的提示进行填写即可，如果你想随便填写，那也是可以的。
[root@localhost ssl]$ openssl req -new -key server.key -out server.csr
# 5：生成类型为X509的自签名证书。有效期设置3650天，即有效期为10年。
[root@localhost ssl]$ openssl x509 -req -days 3650 -in server.csr -signkey server.key -out server.crt
```

此处配置安装完成后会生成三个文件：server.crt,server.csr,server.key，我们将会用到crt和key文件。

```
[root@instance-1dhk06tx conf]# pwd
/usr/nginx/nginx-1.19.9/conf
[root@instance-1dhk06tx conf]# tree
.
├── fastcgi.conf
├── fastcgi_params
├── koi-utf
├── koi-win
├── mime.types
├── nginx.conf
├── scgi_params
├── server.xml
├── ssl
│   ├── server.crt
│   ├── server.csr
│   └── server.key
├── uwsgi_params
└── win-utf

1 directory, 13 files
[root@instance-1dhk06tx conf]#
```

2. 修改nginx的conf/nginx.conf文件，配置https

```
    # HTTPS server
    server {
        listen       443 ssl;
        # 如果有域名，请将这里改为域名
        server_name  106.13.87.104;
        # 这个配置是强制所有请求都通过https，此处只是表名可以强制，配置的时候不配置在这里，而是直接配置在http端口的server下
        #rewrite ^(.*) https://$server_name$1 permanent;
        # 设置证书，绝对路径最好
        ssl_certificate      ssl/server.crt;
        # 设置密钥，绝对路径最好
        ssl_certificate_key  ssl/server.key;
        # 缓存大小
        ssl_session_cache    shared:SSL:1m;
        # session过期时间s,m,h
        ssl_session_timeout  5Om;
        # 指定用于保护服务器通信的算法,!aNULL和!MD5指定不允许使用这些密码进行通信
        ssl_ciphers  HIGH:!aNULL:!MD5;
        ssl_prefer_server_ciphers  on;
        # 如果没有定义SSLv2、SSLv3、TLSv1、TLSv1.1、TLSv1.2那么就调用接口SSL_CTX_set_options 进入Openssl来设置协议,这里可以不配置，默认会开启TLSv1 TLSv1.1 TLSv1.2
        ssl_protocols SSLv2 SSLv3 TLSv1 TLSv1.1 TLSv1.2;

        location / {
            # 以下三个配置，如果是需要转发到http的端口上则必须要配置。如果是转发到https上则建议配置
            proxy_set_header Host   $host;
            proxy_set_header X-Real-IP  $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            # 如要使用移动端，还可以配置websocket，见nginx配置文档，此处略
            ####
            # 解决跨域问题，这里一般配合产品tomcat共同使用，保证session和cookie不丢失，一般跨协议转发才会使用，或者资源路径发生了变化时使用
            proxy_cookie_path / "/; httponly; secure; SameSite=None";
            # 转发到端口，如有多个地址也可配置负载均衡
            proxy_pass http://118.195.136.137:8090;
            }
        }
```

> 到这里，https的配置已经基本结束，唯一需要注意的是，如果只是单单tomcat配置了https或者单是nginx配置了https，可能会出现跨域问题，最好的是tomcat和nginx都配置好SameSite为none

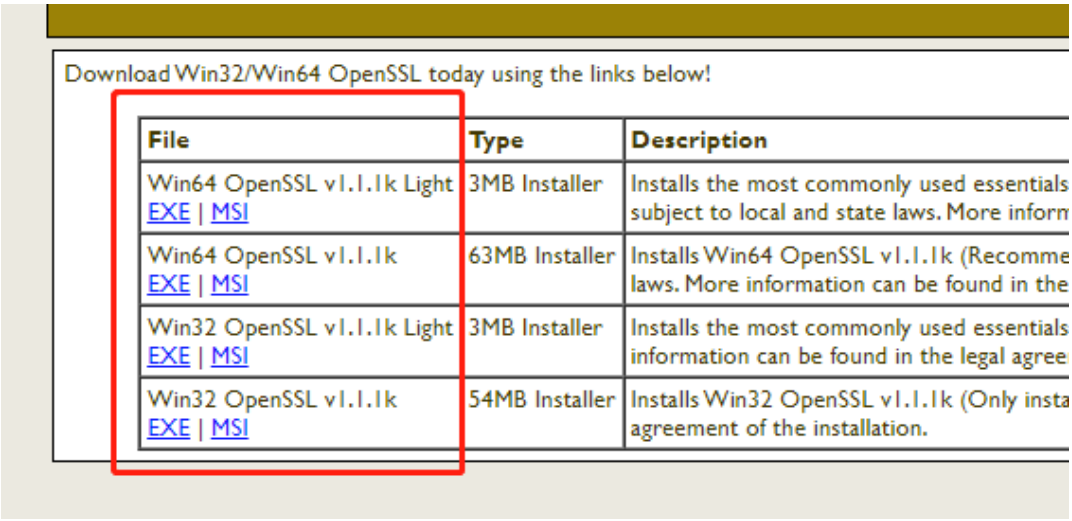
- Tomcat配置sameSite在安装目录/tomcat/conf/context.xml文件中添加：
  `<CookieProcessor sameSiteCookies="none" />`

```
<Context>

    <!-- Default set of monitored resources. If one of these changes, the    -->
    <!-- web application will be reloaded.                                   -->
    <WatchedResource>WEB-INF/web.xml</WatchedResource>
    <WatchedResource>WEB-INF/tomcat-web.xml</WatchedResource>
    <WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>
    <CookieProcessor sameSiteCookies="none" />

    <!-- Uncomment this to disable session persistence across Tomcat restarts -->

    <Manager pathname="" />
</Context>
```

- Nginx配置sameSite在nginx/conf/nginx.conf文件中相对应的位置添加：
  `proxy_cookie_path / "/; httponly; secure; SameSite=None";`

```
21          location / {
22
23              #proxy_pass http://118.195.136.137:8090;
24              #proxy_set_header Host $host:$server_port;
25              proxy_set_header Host   $host;
26              proxy_set_header X-Real-IP  $remote_addr;
27              proxy_set_header X-Forwarded-For  $proxy_add_x_forwarded_for;
28              #proxy_set_header X-Forwarded-Proto  $scheme;
29              #proxy_set_header X-Scheme $scheme;
30              #proxy_set_header X-SSL-Protocol $ssl_protocol;
31              #proxy_set_header X-HTTPS-Protocol $ssl_protocol;
32              #proxy_set_header X-Nginx-Proxy true;
33              #proxy_set_header Connection "";
34              ##proxy_set_header X-Forwarded-Proto https;
35              #proxy_redirect  off;
36              proxy_cookie_path / "/; httponly; secure; SameSite=None";
37              proxy_pass http://118.195.136.137:8090;
38
39              #root   html;
40              #index  index.html index.htm;
41          }
42      }
43
44  }
45
```

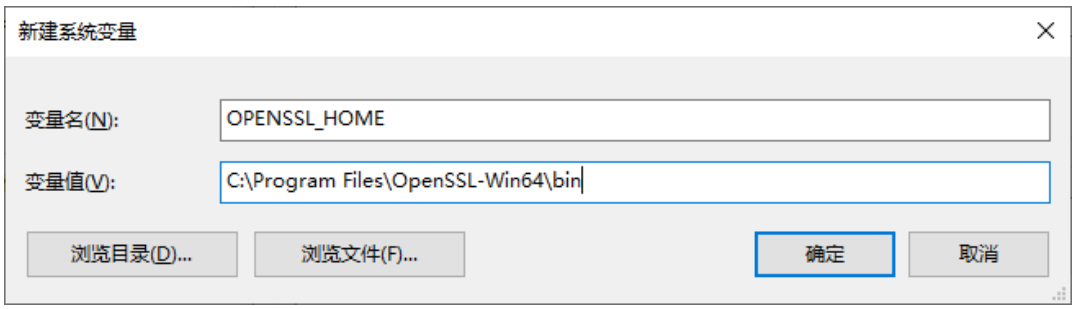
3. 在windows上使用OpenSSL当然也是可以的。只是步骤稍显繁琐。
   - 先下载一个OpenSSL的windows安装包

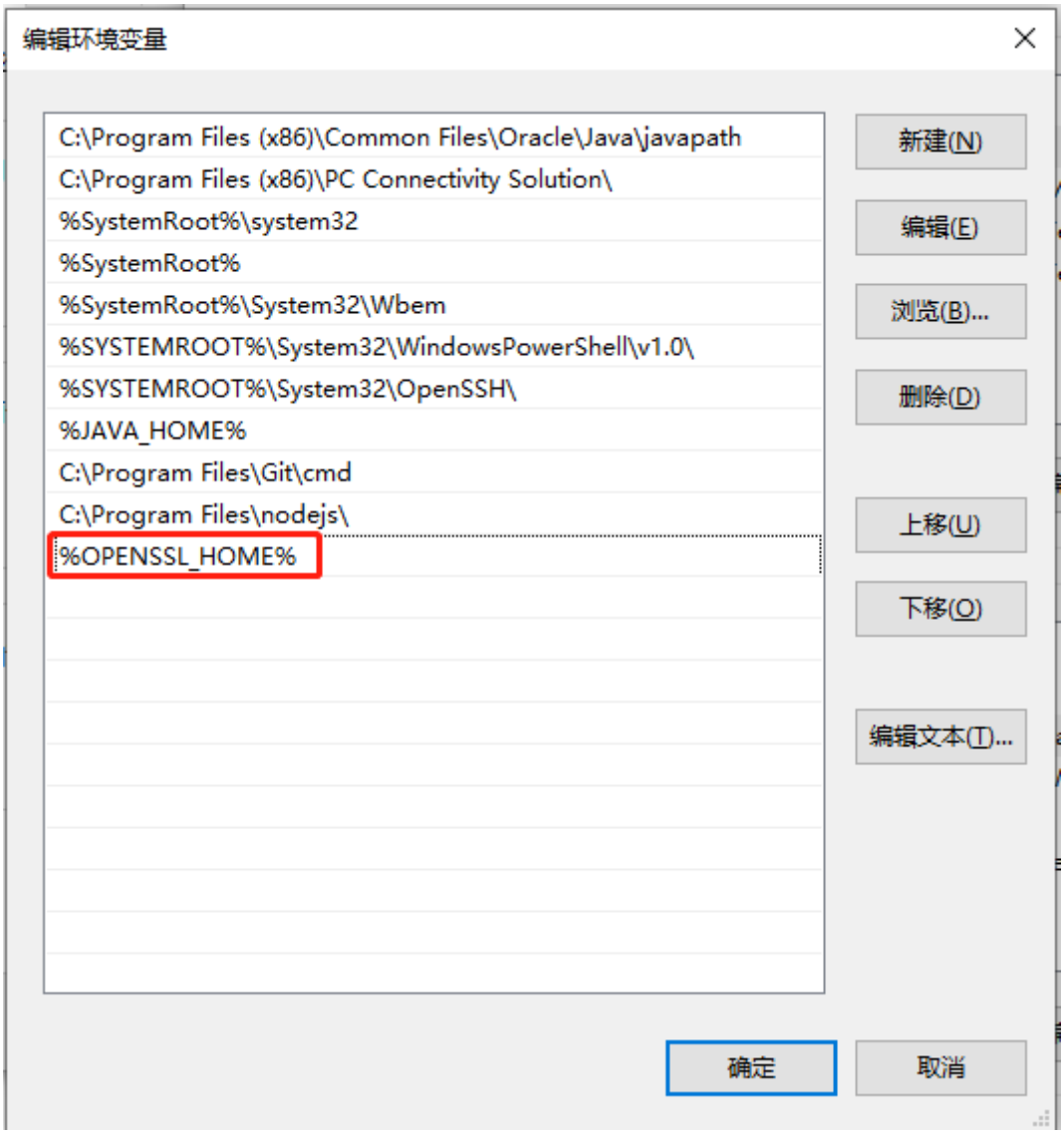

Download Win32/Win64 OpenSSL today using the links below!

| File | Type | Description |
|---|---|---|
| Win64 OpenSSL v1.1.1k Light EXE \| MSI | 3MB Installer | Installs the most commonly used essentials subject to local and state laws. More inform |
| Win64 OpenSSL v1.1.1k EXE \| MSI | 63MB Installer | Installs Win64 OpenSSL v1.1.1k (Recomme laws. More information can be found in the |
| Win32 OpenSSL v1.1.1k Light EXE \| MSI | 3MB Installer | Installs the most commonly used essentials information can be found in the legal agree |
| Win32 OpenSSL v1.1.1k EXE \| MSI | 54MB Installer | Installs Win32 OpenSSL v1.1.1k (Only insta agreement of the installation. |

   一般选择64位的任意一个，一路狂点下一步就行了，有条件的话记得赞助$10给作者，不捐也行。

   - 配置环境变量，老生常谈，这里简单贴一下配置OPENSSL_HOME




- 安装成功，可以愉快地使用OpenSSL了


> 其他注意事项：jdk和openssl本地生成的自签名证书，不受浏览器信任，但是可以提升信息传输的安全等级和解决跨域问题，非常不建议在公网的域名和环境下使用，内网或者本地可以使用，但是浏览器会弹出安全提示。需要机构认证的安全证书可以自行购买，或者到有关厂商去申请一个免费的SSL证书，谨记。