服务器证书安装配置指南（Tomcat）

一、关于 Tomcat 和 SSL

通常 Tomcat 作为独立 Web 服务器运行时才需要配置 SSL。当将 Tomcat 作为 Servlet/JSP 容器运行在另一个 Web 服务器（例如 Apache 或 Microsoft IIS）后端时，Tomcat 容器是不需要配置 SSL 的，通常需要配置主 Web 服务器（Apache 或 Microsoft IIS）处理来自用户的 SSL 连接请求,主 Web 服务器协商所有与 SSL 相关的功能，在解密这些请求后传递发往后端的 Tomcat 容器。同样，Tomcat 将返回明文响应，这些响应将在返回用户浏览器之前再进行加密。在这种环境中，Tomcat 知道主 Web 服务器和客户端之间的通信是通过安全连接进行的但它本身并不参与加密或解密,这就需要在主 Web 服务器配置 SSL。

二、Tomcat SSL 实现方式介绍

Tomcat 有两种 SSL 实现方式：
（1）、Java 运行时提供的 JSSE 实现（自 1.4 版本起）
（2）、APR 实现，默认使用 OpenSSL 引擎。

确切的配置细节取决于正在使用的实现方式。如果通过指定配置 Connector，protocol="HTTP/1.1"则 Tomcat 会自动选择。如果安装使用 APR 即已经安装了 Tomcat APR 库,那么它将使用 APR SSL 实现，否则它将使用 Java JSSE 实现。

下面的图表，显示了两个连接器的不同之处。

	Java Blocking Connector BIO	Java Nio Connector NIO	Java Nio2 Connector NIO2	APR/native Connector APR
Classname	Http11Protocol	Http11NioProtocol	Http11Nio2Protocol	Http11AprProtocol
Tomcat Version	3.x onwards	6.x onwards	8.x onwards	5.5.x onwards
Support Polling	NO	YES	YES	YES
Polling Size	N/A	maxConnections	maxConnections	maxConnections
Read Request Headers	Blocking	Non Blocking	Non Blocking	Blocking
Read Request Body	Blocking	Blocking	Blocking	Blocking

Write Response Headers and Body	Blocking	Blocking	Blocking	Blocking
Wait for next Request	Blocking	Non Blocking	Non Blocking	Non Blocking
SSL Support	Java SSL	Java SSL	Java SSL	OpenSSL
SSL Handshake	Blocking	Non blocking	Non blocking	Blocking
Max Connections	maxConnections	maxConnections	maxConnections	maxConnections

由于 SSL 支持的配置属性在 APR 与 JSSE 实现之间存在显着差异，因此建议避免自动选择，通常通过在 Connector 的 protocol 属性中指定一个类名来完成。

定义方法：
要定义 Java（JSSE）连接器，无论是否加载 APR 库，使用以下方法之一：

```
<!-- 定义 HTTP/1.1 Connector 端口 8443, JSSE NIO 生效 -->
<Connector protocol="org.apache.coyote.http11.Http11NioProtocol"
           port="8443" .../>
```

```
<!-- 定义 HTTP/1.1 Connector 端口 8443, JSSE NIO2 生效 -->
<Connector protocol="org.apache.coyote.http11.Http11Nio2Protocol"
           port="8443" .../>
```

```
<!-- 定义 HTTP/1.1 Connector 端口 8443, JSSE BIO 生效 -->
<Connector protocol="org.apache.coyote.http11.Http11Protocol"
           port="8443" .../>
```

要定义 APR 连接器（APR 库必须可用，如果没有必须安装)：

```
<!-- 定义 HTTP/1.1 Connector 端口 8443, APR 生效 -->
<Connector protocol="org.apache.coyote.http11.Http11AprProtocol"
           port="8443" .../>
```

如果您使用的是 APR，则可以选择为 OpenSSL 配置备用引擎。

```
<Listener className="org.apache.catalina.core.AprLifecycleListener"
          SSLEngine="someengine" SSLRandomSeed="somedevice" />
```

默认值为

```
<Listener className="org.apache.catalina.core.AprLifecycleListener"
```

```
SSLEngine="on" SSLRandomSeed="builtin" />
```

因此，要在 APR 下使用 SSL，请确保将 SSLEngine 属性设置为除 off 以外的值。默认值为 on，如果指定其他值，则必须是有效的引擎名称。

SSLRandomSeed 允许指定熵源。生产系统需要可靠的熵源，但是熵需要大量的时间来收集，因此测试系统可以不使用阻塞熵源，如“/dev/urandom”，这样 Tomcat 将启动的更快。

最后是在 $CATALINA_BASE/conf/server.xml 文件中配置 Connector，其中 $CATALINA_BASE 表示 Tomcat 实例的安装目录，比如/usr/local/apache-tomcat-7.0.94。<Connector> SSL 连接器的示例元素包含在 server.xml 中，随 Tomcat 一起安装的默认文件中。要配置使用 JSSE 的 SSL 连接器，需要删除注释并对其进行编辑，JSSE 配置的一个示例：

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector
               protocol="org.apache.coyote.http11.Http11NioProtocol"
               port="8443" maxThreads="200"
               scheme="https" secure="true" SSLEnabled="true"
               keystoreFile="${user.home}/keystore.jks" keystorePass="password"
               clientAuth="false" sslProtocol="TLS"/>
```

APR 连接器对许多 SSL 设置使用不同的属性，尤其是密钥和证书。APR 配置的一个示例：

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector
               protocol="org.apache.coyote.http11.Http11AprProtocol"
               port="8443" maxThreads="200"
               scheme="https" secure="true" SSLEnabled="true"
               SSLCertificateFile="${user.home}/server.pem"
               SSLCertificateKeyFile="${user.home}/server.key"
               SSLVerifyClient="optional" SSLProtocol="TLS"/>
```

确保正在使用的连接器的正确属性，BIO，NIO 和 NIO2 连接器使用 JSSE，而 APR 连接器使用 APR 库。

port 属性是 Tomcat 将侦听安全连接的 TCP/IP 端口号。可以将其更改为任何端口号（例如，https 通信的默认端口，即 443）。如果在此处更改端口号，则还应更改 redirectPort 为非 SSL 连接器上的属性指定的值。这允许 Tomcat 根据 Servlet 规范的要求，自动重定向尝试访问具有安全约束的页面的用户，该安全约束指定需要 SSL。

完成这些配置更改后，必须重新启动 Tomcat，才能够通过 SSL 访问 Tomcat 支持的任何

Web 应用程序。例如：https://localhost:8443/ 应该看到通常的 Tomcat 启动页面。

三、Linux 系统 Tomcat SSL 配置

说明：
操作系统： Centos 7.6 (64 位)
Tomcat 版本： 7.0.94
JDK 版本：1.7.0_181
Tomcat 安装路径：/usr/local/apache-tomcat-7.0.94

现在网络安全越来越重要，好多网站需要 SSL 支持，也要求客户通过 https 访问站点，但是由于 TLSv1.0、TLSv1.1 容易被黑客攻击，于是很多站点只提供 TLSv1.2 协议支持。TLSv1.2 从 JDK1.7 update96 以后的版本才开始支持，JDK1.8 是默认支持 TLSv1.2 协议的，所以要支持 TLSv1.2 协议确保你的 JDK 版本 >= 1.7.0_96.如果小于请先升级。

1、JDK1.7 升级 1.8 的方法：

（1）下载 jdk1.8 版本
下载： http://www.oracle.com/technetwork/java/javase/downloads/index.html

进入下载目录，执行：
解压（/usr/java/为我的安装目录，可以根据自己想要的目录进行安装）

```
#    tar -zxf jdk-8u171-linux-x64.tar.gz -C    /usr/java/
```

（2）更新

```
# update-alternatives --install /usr/bin/java java    /usr/java/jdk1.8.0_171/bin/java 1
# update-alternatives --install /usr/bin/javac javac    /usr/java/jdk1.8.0_171/bin/javac 1
# update-alternatives --config java
```

由于之前安装的是 1.7 版本，所以要进行选择，选择 1.8 版本

```
# update-alternatives --config javac
```

同样选择 1.8 版本

（3）验证

```
# java -version
# javac -version
```

2、https 配置（使用 JSSE 的 SSL 连接器）：

编辑 /usr/local/apache-tomcat-7.0.94/conf/server.xml
（你的 tomcat 安装目录可能非/usr/local/apache-tomcat-7.0.94/，用你的安装路径替代）

```
# vi /usr/local/apache-tomcat-7.0.94/conf/server.xml
```

定位到下面的内容：

```
    <!--
    <Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"
                maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
                clientAuth="false" sslProtocol="TLS" />
    -->
```

取消注释符号 <!-- 和 -->,并修改为如下内容：

```
<Connector port="443" protocol="org.apache.coyote.http11.Http11NioProtocol"
maxThreads="150"
SSLEnabled="true"
scheme="https"
secure="true"
keystoreFile="/usr/local/apache-tomcat-7.0.94/keystore.jks"
keystorePass="password"
clientAuth="false"
sslProtocol="TLSv1.2"
ciphers="TLS_RSA_WITH_AES_128_GCM_SHA256,
                TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
                TLS_RSA_WITH_AES_128_CBC_SHA,
                TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
                TLS_RSA_WITH_AES_128_CBC_SHA256,
                TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,
                SSL_RSA_WITH_3DES_EDE_CBC_SHA,
                TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA"
 />
```

定位到下面的内容：

```
    <!-- Define an AJP 1.3 Connector on port 8009 -->
    <Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

修改为如下内容：

```
    <!-- Define an AJP 1.3 Connector on port 8009 -->
    <Connector port="8009" protocol="AJP/1.3" redirectPort="443" />
```

说明：
keystoreFile="/usr/local/apache-tomcat-7.0.94/keystore.jks" 是证书文件放置的具体位置
keystorePass="password" 是证书读取需要的密码

配置保存后，需要重新启动 tomcat 服务。

关闭 Tomcat 服务：

```
# /usr/local/apache-tomcat-7.0.94/bin/shutdown.sh
```

或者

```
# kill -9 PID
```

(PID 为 tomcat 的进程 ID 号可以用命令： ps -ef |grep java |grep -v grep 获得)

启动 Tomcat 服务：

```
# /usr/local/apache-tomcat-7.0.94/bin/startup.sh
```

查看 Tomcat 日志有无错误提示：

```
# tail -f /usr/local/apache-tomcat-7.0.94/logs/cataina.out
```

Tomcat 服务正常启动后可以通过浏览器访问 https://www.domain.com 看到正常显示的页面。

3、https 配置（使用 APR 的 SSL 连接器）

3.1、Tomcat 7.0.94 配置支持 APR

APR(Apache Portable Runtime)是一个高可移植库,它是 Apache HTTP Server 2.x 的核心.能更好地和其它本地 web 技术集成，总体上让 Java 更有效率作为一个高性能 web 服务器平台而不是简单作为后台容器. APR 是从操作系统级别来解决异步的 IO 问题,大幅度的提高性能 。

APR 的安装步骤：

首先安装：

```
# yum install libapr1.0-dev libssl-dev
# yum install apr-devel openssl-devel
```

下载安装 apr：

```
# wget http://apache.opencas.org//apr/apr-1.5.2.tar.gz
# tar -zxf   apr-1.5.2.tar.gz
# cd apr-1.5.2
# ./configure --prefix=/usr/local/apache2/apr
# make && make install
```

下载安装 apr-util：

```
# wget http://apache.opencas.org//apr/apr-util-1.5.4.tar.gz
# tar -zxf   apr-util-1.5.4.tar.gz
# cd apr-util-1.5.4
```

```
# ./configure --prefix=/usr/local/apache2/apr --with-apr=/usr/local/apache2/apr
# make && make install
```

下载安装 OpenSSL 1.0.2：

由于 centos 7 当前的 yum 库只有 1.0.1 版本 的 OpenSSL，所以需要手工安装 1.0.2 版本

```
# cd /usr/local/src
# wget https://www.openssl.org/source/openssl-1.0.2-latest.tar.gz
# tar -xzxf openssl-1.0.2-latest.tar.gz
# cd openssl-1.0.2g
# ./config --prefix=/usr/local/openssl -fPIC
// 注意需要加入 -fPIC 参数，否则后面在安装 tomcat native 组件会出错
// 注意：不要按照提示去运行 make depend
# make
# make install
# mv /usr/bin/openssl ~        //将旧的版本拷到其他位置做个备份
# ln -s /usr/local/openssl/bin/openssl /usr/bin/openssl

// 确认版本信息是 1.0.2
# openssl version
```

安装 tomcat-native：

在 Tomcat 的程序包中，在/usr/local/apache-tomcat-7.0.94/bin 目录下有 tomcat-native.tar.gz 解压执行：

```
# cd /usr/local/apache-tomcat-7.0.94/bin
#    tar -zxf tomcat-native.tar.gz
# cd tomcat-native-1.1.24-src/jni/native
```

首先执行 echo $JAVA_HOME，得出 JAVA 安装路径，再执行以下代码：

```
# echo $JAVA_HOME
#       ./configure       --prefix=/usr/local/apache2/apr       --with-apr=/usr/local/apache2/apr
--with-java-home=/usr/java/jdk1.8.0_171 --with-ssl=yes
# make && make install
```

修改$TOMCAT_HOME/bin/catalina.sh 文件，添加如下代码：

CATALINA_OPTS='-Djava.library.path=/usr/local/apache2/apr/lib'

修改$TOMCAT_HOME/conf/server.xml

```
# vi /usr/local/apache-tomcat-7.0.94/conf/server.xml
```

```
<Connector port="8080"
           protocol="org.apache.coyote.http11.Http11AprProtocol"
```

```
                ...
</Connector>
```

重启 tomcat 服务,查看日志如果出现以下代码，则已经正确配置好了 APR:

```
 org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-apr-8080"]
org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-apr-8010"]
org.apache.catalina.startup.Catalina start
```

3.2、编辑 server.xml 配置 https

```
#    vi /usr/local/apache-tomcat-7.0.94/conf/server.xml
```

定位到下面的内容：

```
    <!--
    <Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"
                maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
                clientAuth="false" sslProtocol="TLS" />
    -->
```

取消注释符号 <!-- 和 -->,并修改为如下内容：

```
<Connector port="443" protocol="org.apache.coyote.http11.Http11aprProtocol"
maxThreads="150"
SSLEnabled="true"
scheme="https"
secure="true"
keystoreFile="/usr/local/apache-tomcat-7.0.94/keystore.jks"
keystorePass="password"
clientAuth="false"
sslProtocol="TLSv1.2"
ciphers="TLS_RSA_WITH_AES_128_GCM_SHA256,
                TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
                TLS_RSA_WITH_AES_128_CBC_SHA,
                TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
                TLS_RSA_WITH_AES_128_CBC_SHA256,
                TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,
                SSL_RSA_WITH_3DES_EDE_CBC_SHA,
                TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA"
 />
```

定位到下面的内容：

```
    <!-- Define an AJP 1.3 Connector on port 8009 -->
    <Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

修改为如下内容：

```
    <!-- Define an AJP 1.3 Connector on port 8009 -->
    <Connector port="8009" protocol="AJP/1.3" redirectPort="443" />
```

添加监听器 AprLifecycleListener：

```
  <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on"
SSLRandomSeed="builtin" userAprConnector="true" />
```

Tomcat 服务正常启动后可以通过浏览器访问 https://www.domain.com 看到正常显示的页面。

4、HTTP 自动跳转到 HTTPS

编辑 web.xml,在文件末尾添加如下配置,然后保存并重启 Tomcat 服务。

```
# vi /usr/local/apache-tomcat-7.0.75/conf/web.xml
```

```
<security-constraint>
    <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>
```

5、如果你的 Tomcat 是通过 yum 方式安装的，APR 依赖库可以用下面的命令安装：

```
#    yum -y install openssl
#    yum -y install apr
#    yum -y install apr-util
#    yum -y install tomcat-native
```

四、Windows 系统 https 配置

说明：
操作系统： Windows server 2008 R2 (64 位)
Tomcat 版本： 7.0.94
JDK 版本：1.7.0_181
Tomcat 安装路径：D:\Program Files\apache-tomcat-7.0.94\

1、使用 JSSE 的 SSL 连接器：

上传自己的数字证书到 Tomcat 安装下,并用任意一个编辑器（如 Sublime Text 之类）打开这个 server.xml 文件（在 D:\Program Files\apache-tomcat-7.0.94\conf 目录下）

定位到下面的内容：

```
<!--
<Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"
                maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
                clientAuth="false" sslProtocol="TLS" />
-->
```

取消注释符号 <!-- 和 -->,并修改为如下内容：

```
<Connector port="443" protocol="org.apache.coyote.http11.Http11NioProtocol"
maxThreads="150"
SSLEnabled="true"
scheme="https"
secure="true"
keystoreFile="D:\Program Files\apache-tomcat-7.0.94\keystore.jks"
keystorePass="password"
clientAuth="false"
sslProtocol="TLSv1.2"
ciphers="TLS_RSA_WITH_AES_128_GCM_SHA256,
                TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
                TLS_RSA_WITH_AES_128_CBC_SHA,
                TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
                TLS_RSA_WITH_AES_128_CBC_SHA256,
                TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,
                SSL_RSA_WITH_3DES_EDE_CBC_SHA,
                TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA"
 />
```

定位到下面的内容：

```
<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

修改为如下内容：

```
<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector port="8009" protocol="AJP/1.3" redirectPort="443" />
```

说明：

keystoreFile="D:\Program Files\apache-tomcat-7.0.94\keystore.jks" 是证书文件放置的具体位置

keystorePass="password" 是证书读取需要的密码

配置保存后，需要重新启动 tomcat 服务。

关闭 Tomcat 服务：

```
D:\Program Files\apache-tomcat-7.0.94\bin\shutdown.bat
```

或者

```
taskkill /F /IM PID
```

(PID 为 tomcat 的进程 ID 号可以用命令： netstat -ano | findstr 80 获得)

启动 Tomcat 服务：

```
D:\Program Files\apache-tomcat-7.0.94\bin\startup.bat
```

查看 Tomcat 日志有无错误提示：

```
tail -f D:\Program Files\apache-tomcat-7.0.94\logs\cataina.out
```

说明： tail 命令需要去微软站点下载安装 Windows Resource kit 工具包。

Tomcat 服务正常启动后可以通过浏览器访问 https://www.domain.com 看到正常显示的页面。

2、使用 APR 的 SSL 连接器：

Windows 下 APR APR 的 SSL 连接器配置需要 APR 库、OpenSSL 库以及 Tomcat 使用的 JRI 包装器（libtcnative）的支持。在 https://tomcat.apache.org/download-native.cgi 可以下载需要的文件包，（文件里面有 32 位和 64 位的，根据情况拷贝相应的 dll）。

Binaries for Microsoft Windows build with OpenSSL 1.0.2q

- Native 1.2.21 Windows Binaries zip (recommended)
  - [PGP], [SHA512]
- Native 1.2.21 Windows OCSP-enabled Binaries zip
  - [PGP], [SHA512]

Binaries for Microsoft Windows build with OpenSSL 1.1.1a

- Native 1.2.21 Windows Binaries zip (recommended)
  - [PGP], [SHA512]
- Native 1.2.21 Windows OCSP-enabled Binaries zip
  - [PGP], [SHA512]

压缩包文件包含了 openssl.exe、tcnative-1.dll，其中 APR 和 libtcnative-1 共享动态链接库。下载后解压文件后拷贝 openssl.exe、tcnative-1.dll 到你的 'JDK 安装目录'/bin/下。
本示例为 D:\Program Files\Java\jdk1.7.0_181\bin 下。

用任意编辑器编辑$TOMCAT_HOME\conf\server.xml

本示例： D:\Program Files\apache-tomcat-7.0.94\conf\server.xml

将

```
<Connector port="8080" protocol="HTTP/1.1"
                connectionTimeout="20000"
                redirectPort="8443" />
```

修改为

```
<Connector port="8080" protocol="org.apache.coyote.http11.Http11AprProtocol"
                connectionTimeout="20000"
                redirectPort="8443" />
```

重启 Tomcat 可以在日志中看到：

```
org.apache.coyote.http11.Http11AprProtocol init
Initializing ProtocolHandler ["http-apr-8080"]
```

表示 APR 启动成功。

用任意编辑器编辑$TOMCAT_HOME\conf\server.xml 定位到下面的内容：

```
<!--
<Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"
                maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
                clientAuth="false" sslProtocol="TLS" />
-->
```

取消注释符号 <!-- 和 -->,并修改为如下内容：

```
<Connector port="443" protocol="org.apache.coyote.http11.Http11aprProtocol"
```

```
maxThreads="150"
SSLEnabled="true"
scheme="https"
secure="true"
keystoreFile="D:\Program Files\apache-tomcat-7.0.94\keystore.jks"
keystorePass="password"
clientAuth="false"
sslProtocol="TLSv1.2"
ciphers="TLS_RSA_WITH_AES_128_GCM_SHA256,
                TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
                TLS_RSA_WITH_AES_128_CBC_SHA,
                TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
                TLS_RSA_WITH_AES_128_CBC_SHA256,
                TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,
                SSL_RSA_WITH_3DES_EDE_CBC_SHA,
                TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA"
 />
```

重启 Tomcat 看日志输出可以看到如下信息：

```
oar.apache.catalina.core.AprLifecyclelistener initializeSSL
OpenSSL successfully initialized
org.apache.coyote.http11.Http11AprProtocol init
initializing ProtocolHandler ["http-apr-8080"]
```

则 https 配置成功，可以用浏览器访问 https://www.domain.com.

3、HTTP 自动跳转到 HTTPS：

编辑 web.xml,在文件末尾添加如下配置,然后保存并重启 Tomcat 服务。

```
D:\Program Files\apache-tomcat-7.0.94\conf\web.xml
```

```
<security-constraint>
    <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>
```

五、TOMCAT 配置多域名多 SSL 证书

如果你的网站有多个域名对应同一个项目，编辑 server.xml 可以参考下面关键代码设置即可。

```
<Connector port="443" protocol="org.apache.coyote.http11.Http11NioProtocol" defaultSSLHostConfigName="www.AAA.com"
        maxThreads="150" SSLEnabled="true" >
        <SSLHostConfig hostName="www.AAA.com">
          <Certificate certificateKeystoreFile="tomcat.home/AAA.jks" certificateKeystorePassword="passwordAAA" type="RSA" />
        </SSLHostConfig>
        <SSLHostConfig hostName="www.BBB.com">
          <Certificate certificateKeystoreFile="tomcat.home/BBB.jks" certificateKeystorePassword="passwordBBB" type="RSA" />
        </SSLHostConfig>
</Connector>
```

定位到：

```
<Host name="localhost"  appBase="webapps"  unpackWARs="true" autoDeploy="true">
```

修改为：

```
<Host name="www.AAA.com" appBase="webapps" unpackWARs="false" autoDeploy="true">
                <Alias>www.AAA.com</Alias>
                <Alias>www.BBB.com</Alias>
```

关键点是 <Alias>。

六、Nginx 反向代理 Tomcat 配置 HTTPS

一般 Nginx 反向代理 Tomcat 启用 HTTPS 支持的时候，可以在 Nginx 和 Tomcat 两边同时配置 SSL 支持。也可以只在 Nginx 上启用了 HTTPS，客户的浏览器和 Nginx 之间走的 HTTPS ，而 Nginx 和 Tomcat 之间通过 proxy_pass 走的是普通 HTTP 连接。

配置示例如下：

```
（Nginx 端口 80/443，Tomcat 的端口 8080）
```

Nginx 配置：

```
upstream tomcat {
        server 172.16.0.100:8080 fail_timeout=0;
}

# HTTPS server
server {
        listen          443 ssl;
        server_name www.domain.com;

        ssl_certificate         /etc/nginx/server.pem;
        ssl_certificate_key /etc/nginx/server.key;
        ssl_session_cache shared:SSL:10m;
        ssl_session_timeout 10m;

        ssl_ciphers EECDH+AESGCM:EDH+AESGCM;

        ssl_protocols TLSv1.2;
        ssl_prefer_server_ciphers on;

        location / {
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
                proxy_set_header X-Forwarded-Proto https;
                proxy_redirect off;
                proxy_connect_timeout          240;
                proxy_send_timeout              240;
                proxy_read_timeout              240;
                # note, there is not SSL here! plain HTTP is used
                proxy_pass http://tomcat;
        }
}
```

proxy_set_header X-Forwarded-Proto https; 是关键配置。

Tomcat 配置：

```
<?xml version='1.0' encoding='utf-8'?>
<Server port="8005" shutdown="SHUTDOWN">
  <Service name="Catalina">
    <Connector port="8080" protocol="HTTP/1.1"
                    connectionTimeout="20000"
                    redirectPort="443"
                    proxyPort="443"/>
```

```
    <Engine name="Catalina" defaultHost="localhost">
      <Host name="localhost"    appBase="webapps"
            unpackWARs="true" autoDeploy="true">
            <Valve className="org.apache.catalina.valves.RemoteIpValve"
                  remoteIpHeader="x-forwarded-for"
                  remoteIpProxiesHeader="x-forwarded-by"
                  protocolHeader="x-forwarded-proto"
            />
            <Context path="" docBase="webapp" reloadable="false"/>
      </Host>
    </Engine>
  </Service>
</Server>
```

注意必须有 proxyPort="443"，当然 redirectPort 也必须是 443。同时 <Value> 段的配置也非常重要，否则在 Tomcat 中的应用在读取 getScheme() 方法以及在 web.xml 中配置的一些安全策略会不起作用。