

Lab 10 Tomcat 中的 SSL 配置

随着万维网的发展，越来越多的应用开始搭建在 HTTP 协议的基础上。HTTP 协议是一种简单并且可靠的应用层协议，但是也有其不足之处，其中一点就是安全性问题。由于 HTTP 协议是明文传输，因此任何通过 HTTP 协议传输的数据理论上都可以被任何传输中间结点所捕获和解析。因此 SSL (Secure Socket Layer) 技术应运而生。在这次实验中，我们将会了解到 SSL 的意义、原理以及在 Tomcat 中部署 SSL 应用的配置方法。

1 预备知识

1.1 SSL 简介

Secure socket layer(SSL)协议最初由 Netscape 企业发展，现已成为网络用来鉴别网站和网页浏览者身份，以及在浏览器使用者及网页服务器之间进行加密通讯的全球化标准。由于 SSL 技术已建立到所有主要的浏览器和 WEB 服务器程序中，因此，仅需安装数字证书，或服务器证书就可以激活服务器功能了。

后来，随着 SSL 应用的推广，IETF(www.ietf.org)将 SSL 作了标准化，即 RFC2246,并将其称为 TLS (Transport Layer Security)，从技术上讲，TLS1.0 与 SSL3.0 的差别非常微小。

SSL 是一个介于 HTTP 协议与 TCP 之间的一个可选层，其位置大致如下：

```
-----  
| HTTP |  
-----  
| SSL |  
-----  
| TCP |  
-----  
| IP |  
-----
```

SSL 在 TCP 之上建立了一个加密通道，通过这一层的数据经过了加密，因此达到保密的效果。

1.2 SSL 与 HTTPS

安全超文本传输协议 (HTTPS 或 SHTTP) 是一种结合 HTTP 而设计的消息的安全通信协议。HTTPS 实际上应用了 Netscape 的完全套接字层 (SSL) 作为 HTTP 应用层的子层。HTTPS 使用端口 443, 而不是像 HTTP 那样使用端口 80 来和 TCP/IP 进行通信。

2 实验 1 在 Tomcat 中配置 SSL

实验目的：

- (1) 熟悉 SSL 的实现机制。
- (2) 学习 Tomcat 中简单配置 SSL 的方法。

实验任务：

通过配置 Tomcat, 使得浏览器可以对服务器进行单项验证, 并通过 Https 协议访问服务器中的内容。

实验环境：

- Java SDK : JDK 1.4.2
- 服务器 : Tomcat 5.1

实验交付物：

1. 服务器证书文件 : tomcat.keystore
2. Tomcat 的配置文件 : server.xml

实验步骤：

- (1) 打开 Tomcat 的 HTTPS 监听端口 (默认为关闭)。找到 Tomcat 安装目录下的 /conf/server.xml 文件 (如图 1-1 所示)。

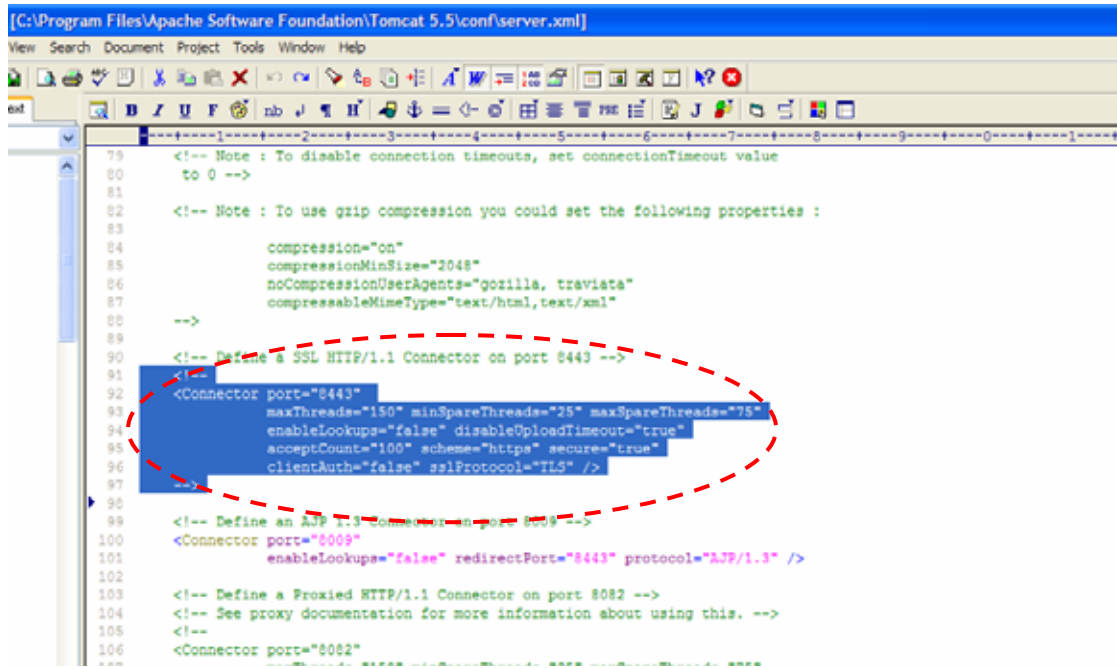


图 10-1

- (2) 注意图 10-1 中所选中的部分。该部分在 Tomcat 安装时被注释。现在将注释符号去掉,我们就可以打开 Tomcat 的 HTTPS 监听端口(默认为 8443 端口)。在这个 Connector 元素中,clientAuth="false"说明了客户端验证不需要,因此只是一个单向验证。如果需要进行双向验证,则需要将该值设为\。
- (3) 接下来我们要做的是获得证书(Certificate)。由于我们这个实验所要做的只是浏览器对服务器的单向认证,因此我们只需要获得服务器证书。获得证书的方法很多,现在我们使用 JRE 中自带的 keytool 来手动创建一个证书文件。Keytool 位于 JRE 安装目录中的/bin/目录下。如图 10-2 所示,keytool 是一个非常强大的密钥生成和管理工具。



```
C:\WINDOWS\system32\cmd.exe
C:\Program Files\Java\jre1.5.0\bin>keytool
keytool usage:

-certreq      [-v] [-protected]
               [-alias <alias>] [-sigalg <sigalg>]
               [-file <csr_file>] [-keypass <keypass>]
               [-keystore <keystore>] [-storepass <storepass>]
               [-storetype <storetype>] [-providerName <name>]
               [-providerClass <provider_class_name> [-providerArg <arg>]] ...

-delete       [-v] [-protected] -alias <alias>
               [-keystore <keystore>] [-storepass <storepass>]
               [-storetype <storetype>] [-providerName <name>]
               [-providerClass <provider_class_name> [-providerArg <arg>]] ...

-export       [-v] [-rfc] [-protected]
               [-alias <alias>] [-file <cert_file>]
               [-keystore <keystore>] [-storepass <storepass>]
               [-storetype <storetype>] [-providerName <name>]
               [-providerClass <provider_class_name> [-providerArg <arg>]] ...

-genkey       [-v] [-protected]
               [-alias <alias>]
               [-keyalg <keyalg>] [-keysize <keysize>]
               [-sigalg <sigalg>] [-dname <dname>]
               [-validity <valDays>] [-keypass <keypass>]
               [-keystore <keystore>] [-storepass <storepass>]
               [-storetype <storetype>] [-providerName <name>]
               [-providerClass <provider_class_name> [-providerArg <arg>]] ...

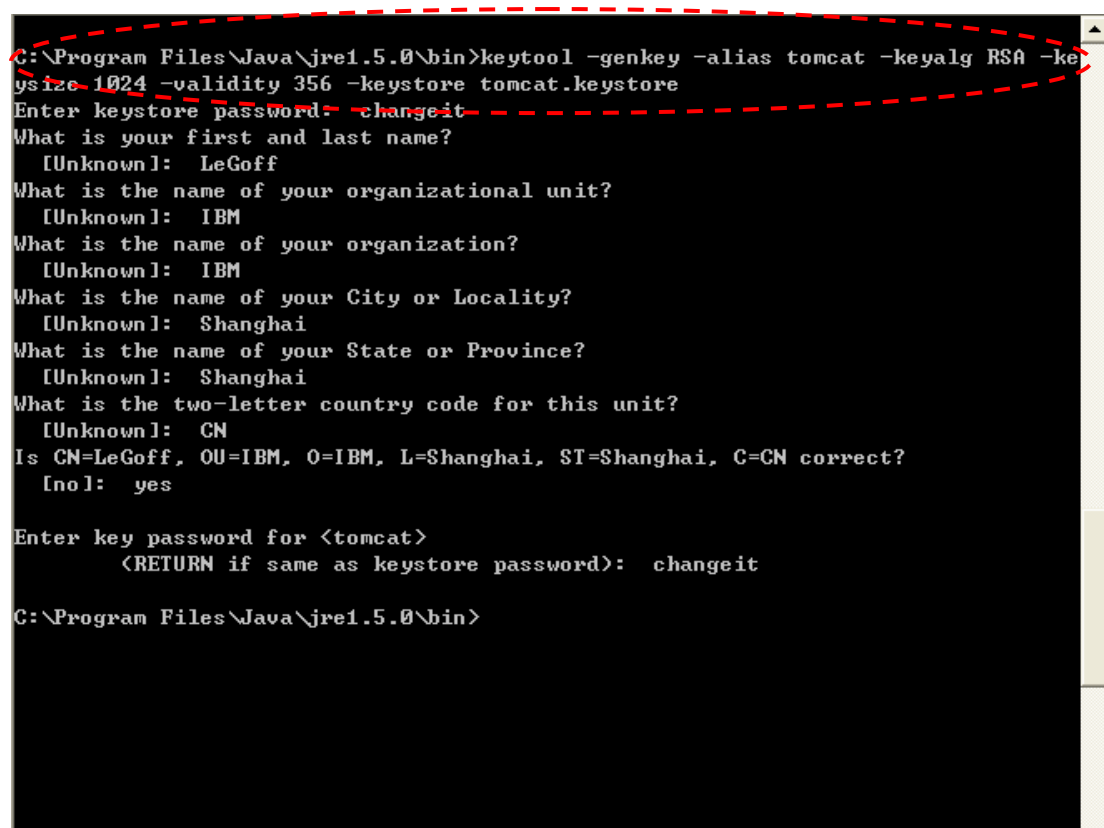
-help

-identitydb   [-v] [-protected]
               [-file <idb_file>]
               [-keystore <keystore>] [-storepass <storepass>]
               [-storetype <storetype>] [-providerName <name>]
               [-providerClass <provider_class_name> [-providerArg <arg>]] ...

-import       [-v] [-noprompt] [-trustcacerts] [-protected]
               [-alias <alias>]
               [-file <cert_file>] [-keypass <keypass>]
               [-keystore <keystore>] [-storepass <storepass>]
               [-storetype <storetype>] [-providerName <name>]
               [-providerClass <provider_class_name> [-providerArg <arg>]] ...
```

图 10-2

- (4) 现在，我们输入图 10-3 中所显示的命令并填入相应的证书属性便可以在命令行所在目录下生成一个证书文件。注意：密钥仓库的默认密码为 changeit，而最后我们要输入的证书密码必须与密钥仓库密码相一致；-alias 制定了证书的别名；-keyalg 指定了证书所选用的废堆成加密算法，这里为 RSA；-keysize 指定了密钥长度，这里为 1024 位长；-validity 指定了证书的有效期限，这里为 365 天；-keystore 制定了生成的证书文件名。



```
C:\Program Files\Java\jre1.5.0\bin>keytool -genkey -alias tomcat -keyalg RSA -keystore tomcat.keystore -keysize 1024 -validity 356 -keystore tomcat.keystore
Enter keystore password: changeit
What is your first and last name?
  [Unknown]: LeGoff
What is the name of your organizational unit?
  [Unknown]: IBM
What is the name of your organization?
  [Unknown]: IBM
What is the name of your City or Locality?
  [Unknown]: Shanghai
What is the name of your State or Province?
  [Unknown]: Shanghai
What is the two-letter country code for this unit?
  [Unknown]: CN
Is CN=LeGoff, OU=IBM, O=IBM, L=Shanghai, ST=Shanghai, C=CN correct?
  [no]: yes

Enter key password for <tomcat>
  (RETURN if same as keystore password): changeit

C:\Program Files\Java\jre1.5.0\bin>
```

图 10-3

- (5) 现在我们将生成出来的 tomcat.keystore 复制到 tomcat 的安装目录下。然后继续编辑 server.xml 文件。在 (1) 中所提到的 Connect 元素中添加属性 keystoreFile="tomcat.keystore", 表示证书文件的路径(可以是 tomcat 的相对路径, 也可以是绝对路径); 同时, 如果证书文件的密码不是默认的 changeit, 则需要再添加一个属性 password, 其值为证书文件的密码。重新启动 tomcat, 并通过 8443 端口进行访问, 我们便可以看到浏览器提示用户验证证书的可信性。点击 View Certificate 这个按钮, 我们就可以查看证书的详细信息。

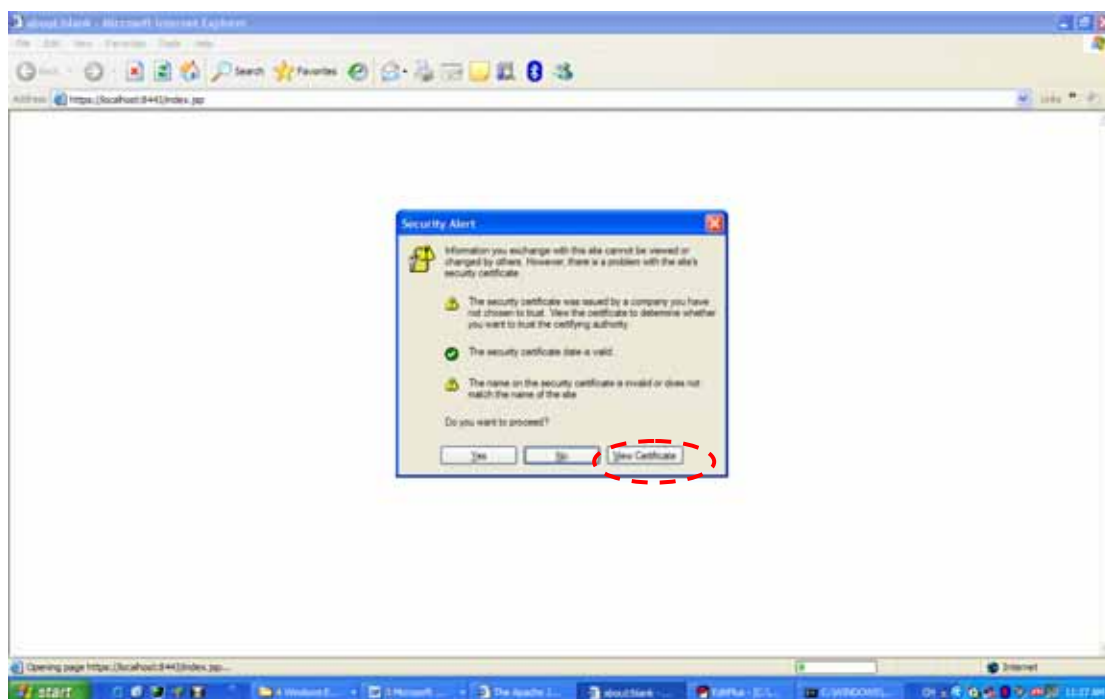


图 10-4

(6) 如图 10-5 种所示,在完成了证书可信性的验证之后,我们就可以正常访问网站了。注意,此时浏览器中的地址栏内容已经发生了改变,我们使用的是在 8443 端口上的 HTTPS 协议。

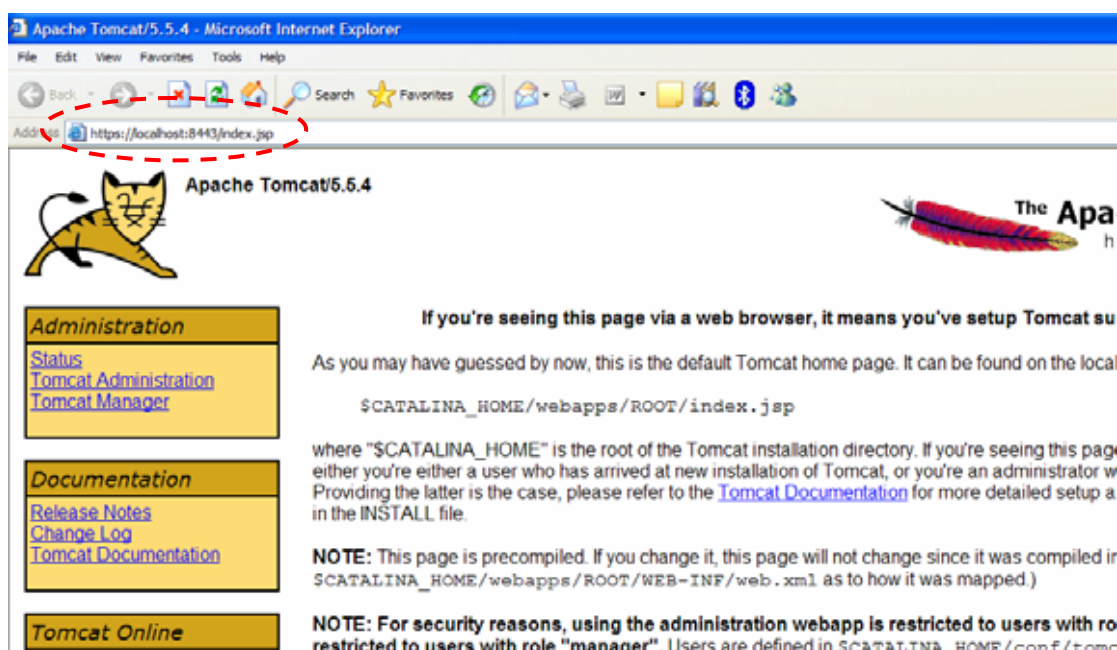


图 10-5