对于SSL的支持，Shiro只是判断当前url是否需要SSL登录，如果需要自动重定向到https进行访问。

**首先生成数字证书，生成证书到D:\localhost.keystore**

使用JDK的keytool命令，生成证书（包含证书/公钥/私钥）到D:\localhost.keystore：

keytool -genkey -keystore "D:\localhost.keystore" -alias localhost -keyalg RSA

输入密钥库口令:

再次输入新口令:

您的名字与姓氏是什么?

  [Unknown]:  localhost

您的组织单位名称是什么?

  [Unknown]:  sishuok.com

您的组织名称是什么?

  [Unknown]:  sishuok.com

您所在的城市或区域名称是什么?

  [Unknown]:  beijing

您所在的省/市/自治区名称是什么?

  [Unknown]:  beijing

该单位的双字母国家/地区代码是什么?

  [Unknown]:  cn

CN=localhost, OU=sishuok.com, O=sishuok.com, L=beijing, ST=beijing, C=cn是否正确

?

  [否]:  y

输入 <localhost> 的密钥口令

        (如果和密钥库口令相同, 按回车):

再次输入新口令:

通过如上步骤，生成证书到D:\ localhost.keystore；

**然后设置tomcat下的server.xml**

此处使用了apache-tomcat-7.0.40版本，打开conf/server.xml，找到：

**Java代码  [收藏代码](javascript:void())**

1. <!--
2. <Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
3. maxThreads="150" scheme="https" secure="true"
4. clientAuth="false" sslProtocol="TLS" />
5. -->

替换为

**Java代码  [收藏代码](javascript:void())**

1. <Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
2. maxThreads="150" scheme="https" secure="true"
3. clientAuth="false" sslProtocol="TLS"
4. keystoreFile="D:\localhost.keystore" keystorePass="123456"/>

keystorePass就是生成keystore时设置的密码。

**添加SSL到配置文件（spring-shiro-web.xml）**

此处使用了和十三章一样的代码：

**Java代码  [收藏代码](javascript:void())**

1. <bean id="sslFilter" **class**="org.apache.shiro.web.filter.authz.SslFilter">
2. <property name="port" value="8443"/>
3. </bean>
4. <bean id="shiroFilter" **class**="org.apache.shiro.spring.web.ShiroFilterFactoryBean">
5. ……
6. <property name="filters">
7. <util:map>
8. <entry key="authc" value-ref="formAuthenticationFilter"/>
9. <entry key="ssl" value-ref="sslFilter"/>
10. </util:map>
11. </property>
12. <property name="filterChainDefinitions">
13. <value>
14. /login.jsp = ssl,authc
15. /logout = logout
16. /authenticated.jsp = authc
17. /\*\* = user
18. </value>
19. </property>
20. </bean>

SslFilter默认端口是443，此处使用了8443；“/login.jsp = ssl,authc”表示访问登录页面时需要走SSL。

**测试**

最后把shiro-example-chapter14打成war包（mvn:package），放到tomcat下的webapps中，启动服务器测试，如访问localhost:9080/chapter14/，会自动跳转到<https://localhost:8443/chapter14/login.jsp>。

如果使用Maven Jetty插件，可以直接如下插件配置：

**Java代码  [收藏代码](javascript:void())**

1. <plugin>
2. <groupId>org.mortbay.jetty</groupId>
3. <artifactId>jetty-maven-plugin</artifactId>
4. <version>8.1.8.v20121106</version>
5. <configuration>
6. <webAppConfig>
7. <contextPath>/${project.build.finalName}</contextPath>
8. </webAppConfig>
9. <connectors>
10. <connector implementation="org.eclipse.jetty.server.nio.SelectChannelConnector">
11. <port>8080</port>
12. </connector>
13. <connector implementation="org.eclipse.jetty.server.ssl.SslSocketConnector">
14. <port>8443</port>
15. <keystore>${project.basedir}/localhost.keystore</keystore>
16. <password>123456</password>
17. <keyPassword>123456</keyPassword>
18. </connector>
19. </connectors>
20. </configuration>
21. </plugin>