Security in Tomcat

Read below link

<https://tomcat.apache.org/tomcat-7.0-doc/realm-howto.html>

You can create new roles in conf\tomcat-users.xml

|  |
| --- |
| <role rolename="manager-gui"/>  <role rolename="admin"/>  <role rolename="tomcat"/>  <role rolename="manager-status"/>  <role rolename="manager-script"/>  <role rolename="manager-jmx"/>  <user password="bac4$tomcat" roles="admin,manager-gui,manager-status,manager-script,manager-jmx,tomcat" username="admin"/>  <user password="tomcat" roles="tomcat" username="tomcat"/> |

Added two lines for rolename tomcat

Run tomcat server

**FORM Based security**

Check below example

<http://localhost:8080/examples/jsp/security/protected/>

check web.xml placed under webapps\examples\WEB-INF

|  |
| --- |
| <security-constraint>  <display-name>Example Security Constraint</display-name>  <web-resource-collection>  <web-resource-name>Protected Area</web-resource-name>  <!-- Define the context-relative URL(s) to be protected -->  <url-pattern>/jsp/security/protected/\*</url-pattern>  <!-- If you list http methods, only those methods are protected -->  <http-method>DELETE</http-method>  <http-method>GET</http-method>  <http-method>POST</http-method>  <http-method>PUT</http-method>  </web-resource-collection>  <auth-constraint>  <!-- Anyone with one of the listed roles may access this area -->  <role-name>tomcat</role-name>  <role-name>role1</role-name>  </auth-constraint>  </security-constraint>  <!-- Default login configuration uses form-based authentication -->  <login-config>  <auth-method>FORM</auth-method>  <realm-name>Example Form-Based Authentication Area</realm-name>  <form-login-config>  <form-login-page>/jsp/security/protected/login.jsp</form-login-page>  <form-error-page>/jsp/security/protected/error.jsp</form-error-page>  </form-login-config>  </login-config>  <!-- Security roles referenced by this web application -->  <security-role>  <role-name>role1</role-name>  </security-role>  <security-role>  <role-name>tomcat</role-name>  </security-role> |



Only tomcat and role1 Roles can access jsp page and it uses FORM auth mehod

There are four types of auth methods

1. **BASIC** (User passwords are sent in simple base64 ENCODING (no encryption))
2. **DIGEST** (the authentication is performed by transmitting the password in an ENCRYPTED form which is much MORE SECURE than the simple base64 encoding used by Basic Authentication)
3. **CLIENT-CERT** (use HTTP over SSL , This mechanism requires the user to possess a Public Key Certificate (PKC))
4. **FORM** (The look and feel of the 'login screen' cannot be varied using the web browser's built-in authentication mechanisms. This specification introduces a required form based authentication mechanism which allows a Developer to CONTROL the LOOK and FEEL of the login screens.

The web application deployment descriptor contains entries for a login form and error page. The login form must contain fields for entering a username and a password. These fields must be named j\_username and j\_password, respectively.)

Example of form

<form method='post' action='j\_security\_check'>

<input type='text' name='j\_username'>

<input type='password' name='j\_password'>

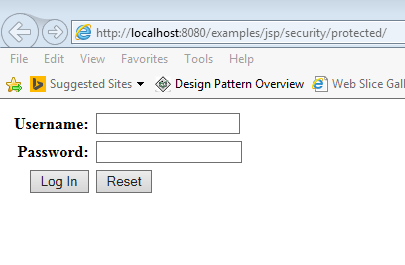
</form>

Check <http://java.boot.by/wcd-guide/ch05s03.html>

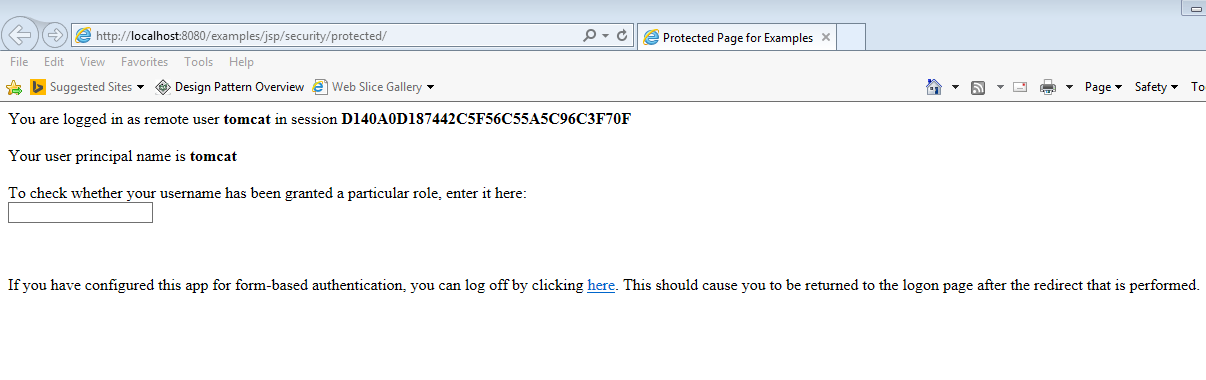
Check login.jsp page under

apache-tomcat-7.0.29\webapps\examples\jsp\security\protected

|  |
| --- |
| <html>  <head>  <title>Login Page for Examples</title>  <body bgcolor="white">  <form method="POST" action='<%= response.encodeURL("j\_security\_check") %>' >  <table border="0" cellspacing="5">  <tr>  <th align="right">Username:</th>  <td align="left"><input type="text" name="j\_username"></td>  </tr>  <tr>  <th align="right">Password:</th>  <td align="left"><input type="password" name="j\_password"></td>  </tr>  <tr>  <td align="right"><input type="submit" value="Log In"></td>  <td align="left"><input type="reset"></td>  </tr>  </table>  </form>  </body>  </html> |



Give **tomcat**/**tomcat** to login as we mentioned in tomcat-users.xml file



**JDBCRealm**

First create users and user\_roles table in one of the schema later we need to configure the same in server.xml

|  |
| --- |
| create table users (  user\_name varchar(15) not null primary key,  user\_pass varchar(15) not null  );  create table user\_roles (  user\_name varchar(15) not null,  role\_name varchar(15) not null,  primary key (user\_name, role\_name)  );  insert into users values ('shalaj','shalaj');  insert into user\_roles values ('shalaj','tomcat') |

To configure JDBCRealm, you will create a <Realm> element and nest it in your $CATALINA\_BASE/conf/server.xml file, as

|  |
| --- |
| <Realm className="org.apache.catalina.realm.JDBCRealm"  driverName="oracle.jdbc.driver.OracleDriver"  connectionURL="jdbc:oracle:thin:@t1crt1d4.vci.att.com:1524:arewesd4"  connectionName="DVLP0\_ODR\_OWNER"  connectionPassword="adxcw\_5z"  userTable="users" userNameCol="user\_name" userCredCol="user\_pass"  userRoleTable="user\_roles" roleNameCol="role\_name"/> |

Now again if we ran

<http://localhost:8080/examples/jsp/security/protected/>

by using shalaj/shalaj , it will be authenticated by jdbcrealm

**BASIC Authentication in tomcat**

User passwords are sent in simple base64 ENCODING (no encryption)),

It store the user name and password in header under the key Authorization,

It basically create username:password and encode it using base64 Encoding and store it in Authorization key

Authorization = Basic **1234jkafkjhjk!=**

We need to use Basic authentication only over https otherwise anyone can get username and password information. So the next question is why we encode it and answer is encoding is for eliminating any character which are not supported by http

<http://localhost:8080/test/jsp/security/protected/>

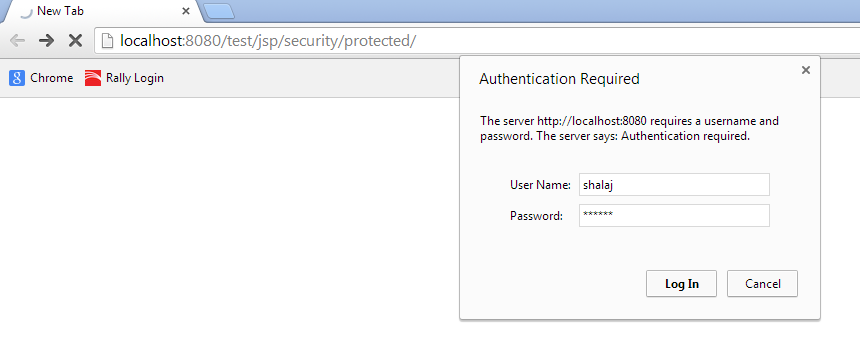
Web.xml



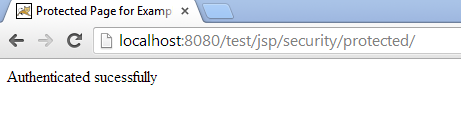
**Source code**

**Put test folder under webapps**

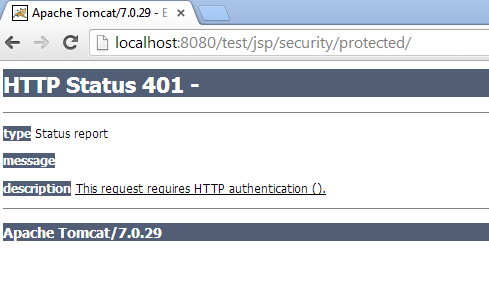
****



**Valid Password**



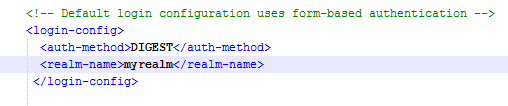
**Invalid password**



**DIGEST based authentication**

Just change auth-method in web.xml from **BASIC** to **DIGEST , BASIC** authentication did not encrypt the password it just decode the password anyone can crack it , DIGEST mechanism use key encryption so that it’s hard to decrypt the password

Now we will run same application with DIGEST based authentication, open web.xml and change auth method to digest also put **relam-name** element

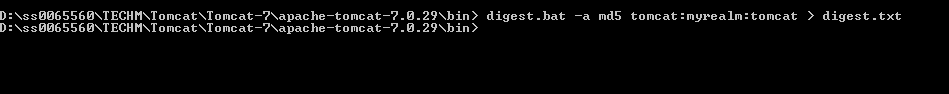


The value <realm-name> element is used to calculate the digested value of a cleartext password

First we need to encrypt the password for user tomcat, issue below command

**digest.bat -a md5 tomcat:myrealm:tomcat > digest.txt**

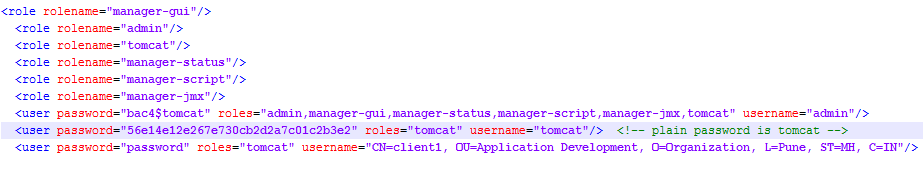
**digest.bat -a <algo name> <username> :< realm name> :< password>**

 **This will create digest.txt file under bin folder open this file, it contain following contents**

|  |
| --- |
| **tomcat:myrealm:tomcat:56e14e12e267e730cb2d2a7c01c2b3e2** |

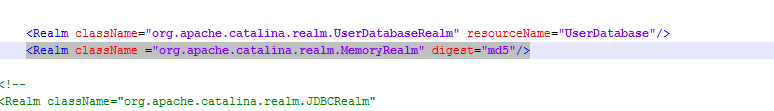
**Here 56e14e12e267e730cb2d2a7c01c2b3e2 is encrypted value for password, in our case it is tomcat**

**Now open conf/tomcat-users.xml and put this password for tomcat user**



**Put below entry in server.xml file**

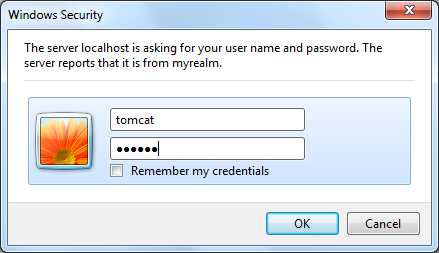
|  |
| --- |
| **<Realm className ="org.apache.catalina.realm.MemoryRealm" digest="md5"/>** |

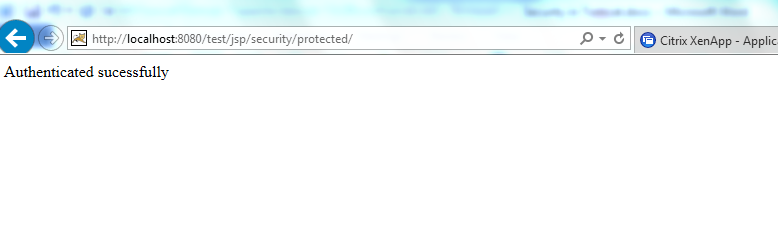


**Now restart tomcat and open below url**

[**http://localhost:8080/test/jsp/security/protected/**](http://localhost:8080/test/jsp/security/protected/)

**It will ask for username and password but this time it is having encrypted password**





**Please find attached code**

****

**CLIENT-CERT based authentication**

For that we need to have public certificate key installed on server

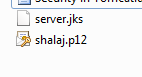
<http://stackoverflow.com/questions/1180397/tomcat-server-client-self-signed-ssl-certificate>

Create\_cert.bat file

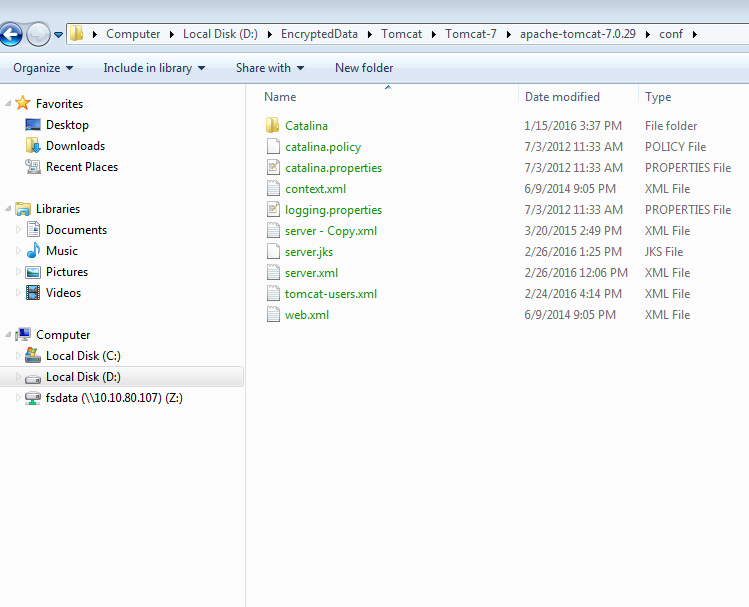
|  |
| --- |
| @echo off  if "%1" == "" goto usage  keytool -genkeypair -alias servercert -keyalg RSA -dname "CN=client1, OU=Application Development, O=Organization, L=Pune, ST=MH, C=IN" -keypass password -keystore server.jks -storepass password  keytool -genkeypair -alias %1 -keystore %1.p12 -storetype pkcs12 -keyalg RSA -dname "CN=%1,OU=Application Development, O=Organization, L=Pune, ST=MH, C=IN" -keypass password -storepass password  keytool -exportcert -alias %1 -file %1.cer -keystore %1.p12 -storetype pkcs12 -storepass password  keytool -importcert -keystore server.jks -alias %1 -file %1.cer -v -trustcacerts -noprompt -storepass password  keytool -list -v -keystore server.jks -storepass password  del %1.cer  goto end  :usage  echo Need user id as first argument: generate\_keystore [username]  goto end  :end  pause |

Create personal certificate and jks by using this script, note: all passwords value is **password**



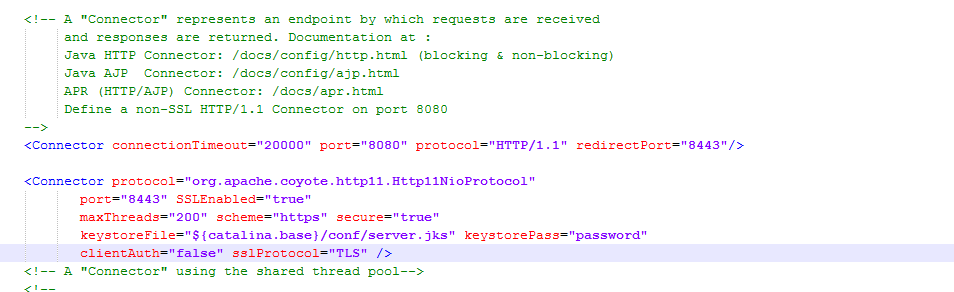


Put server.jks in conf folder



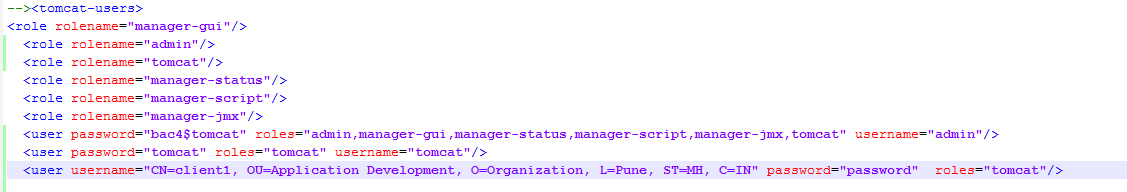
Replace connecter tag with below details

|  |
| --- |
| <Connector protocol="org.apache.coyote.http11.Http11NioProtocol"  port="8443" SSLEnabled="true"  maxThreads="200" scheme="https" secure="true"  keystoreFile="${catalina.base}/conf/server.jks" keystorePass="**password**"  clientAuth="false" sslProtocol="TLS" /> |



Added new user in tomcat-user.xml

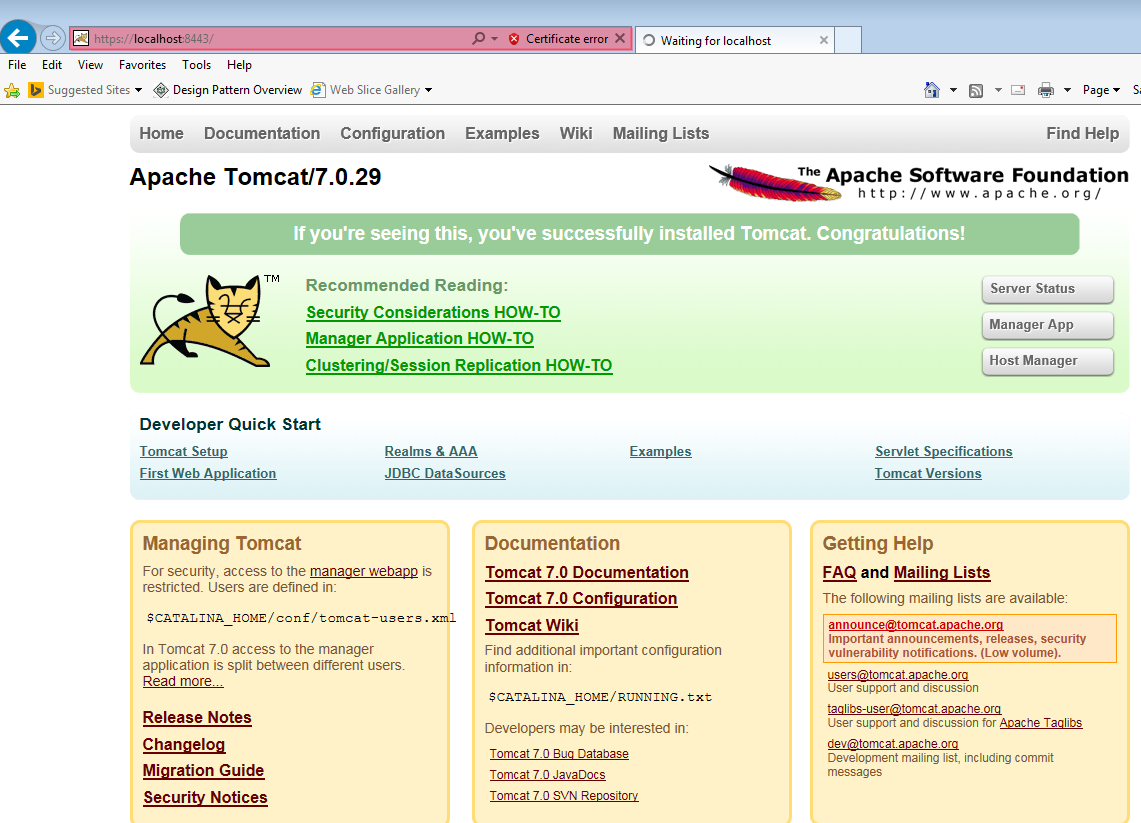
<user username="CN=client1, OU=Application Development, O=Organization, L=Pune, ST=MH, C=IN" password="password" roles="tomcat"/>



Now we can access tomcat over https

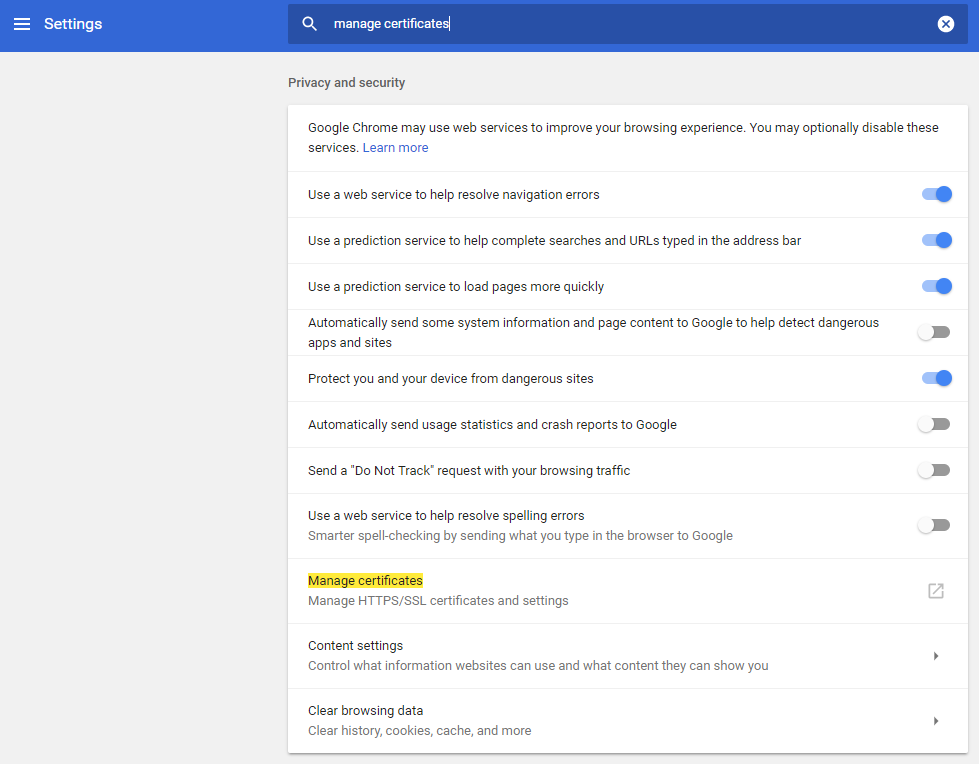
Use link below

<https://localhost:8443>

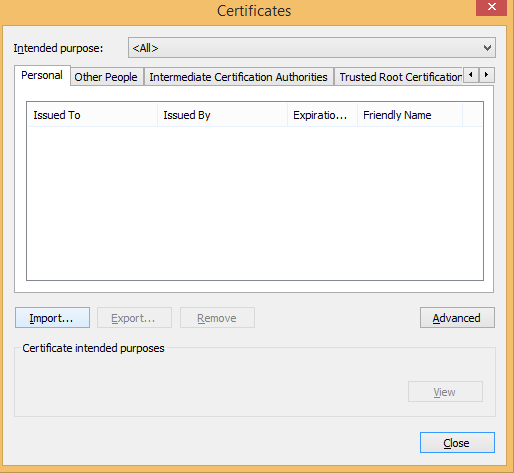


If you want to import client certificate for a particular client (let’s say chrome), so that tomcat can be access via client which have valid certificate we need to do below steps

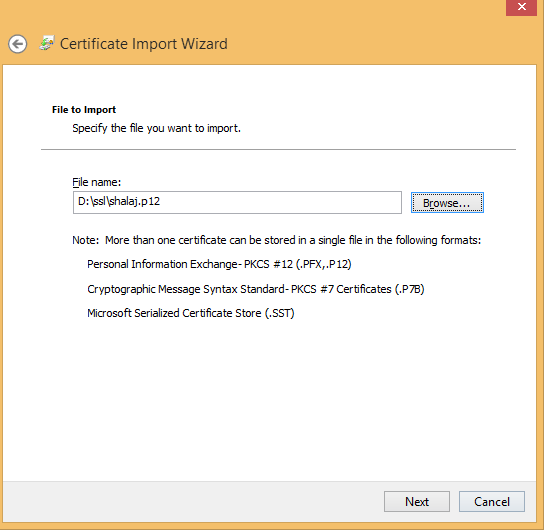
Go to chrome>>Settings



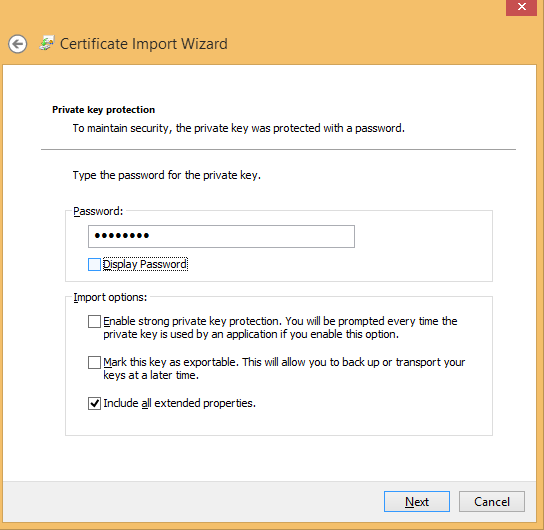
Click Manage certificates



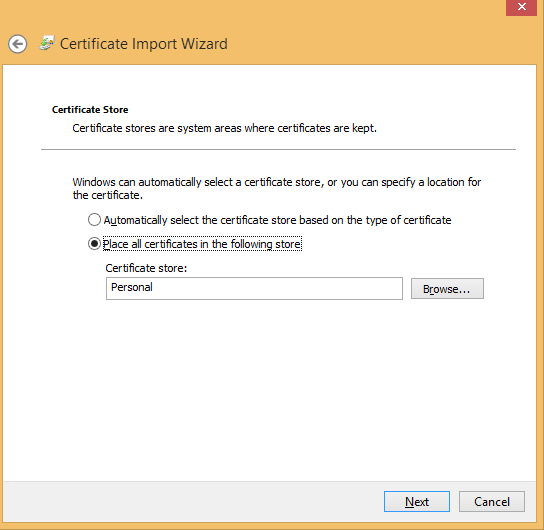
Click Import

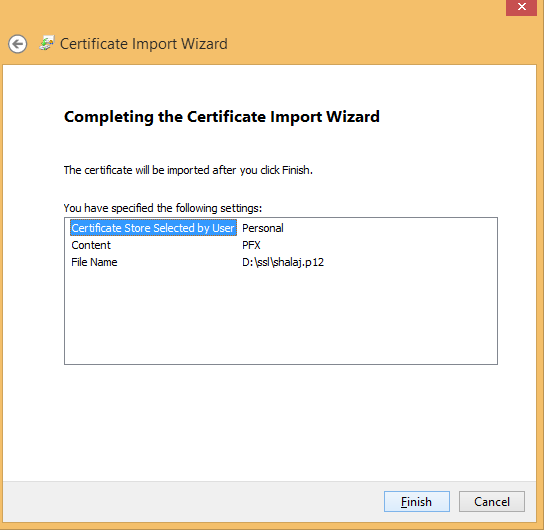


Select p12 file that we created before

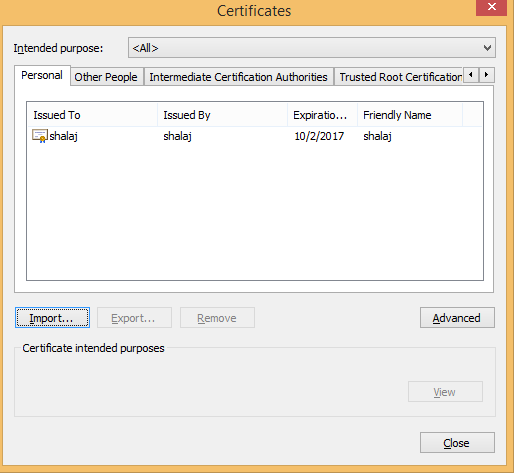


Enter password

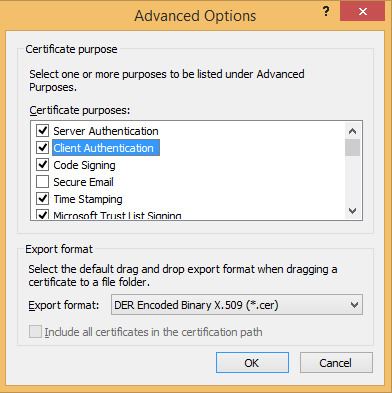




Click Finish



Click Advance



Select Client Authentication

Now make clientAuth to true so that only authorized client access that

|  |
| --- |
| <Connector protocol="org.apache.coyote.http11.Http11Protocol"  port="8443" SSLEnabled="true"  maxThreads="200" scheme="https" secure="true"  keystoreFile="${catalina.base}/conf/server.jks" keystorePass="password"  **clientAuth="true"** sslProtocol="TLS" ciphers="TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256,TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA,  TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA384,TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA,TLS\_ECDHE\_RSA\_WITH\_RC4\_128\_SHA,  TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256,TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA,TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA256,  TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA,SSL\_RSA\_WITH\_RC4\_128\_SHA"/> |

Yellow highlighted part is not required (this is required if you get issue like ERR\_SSL\_VERSION\_OR\_CIPHER\_MISMATCH)

Also comment below listener entry in server.xml

|  |
| --- |
| <!-- <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />--> |

Now you can access it from client which has a valid certificate

I have created one test application which uses BASIC authentication as well as SSL just check the below source code



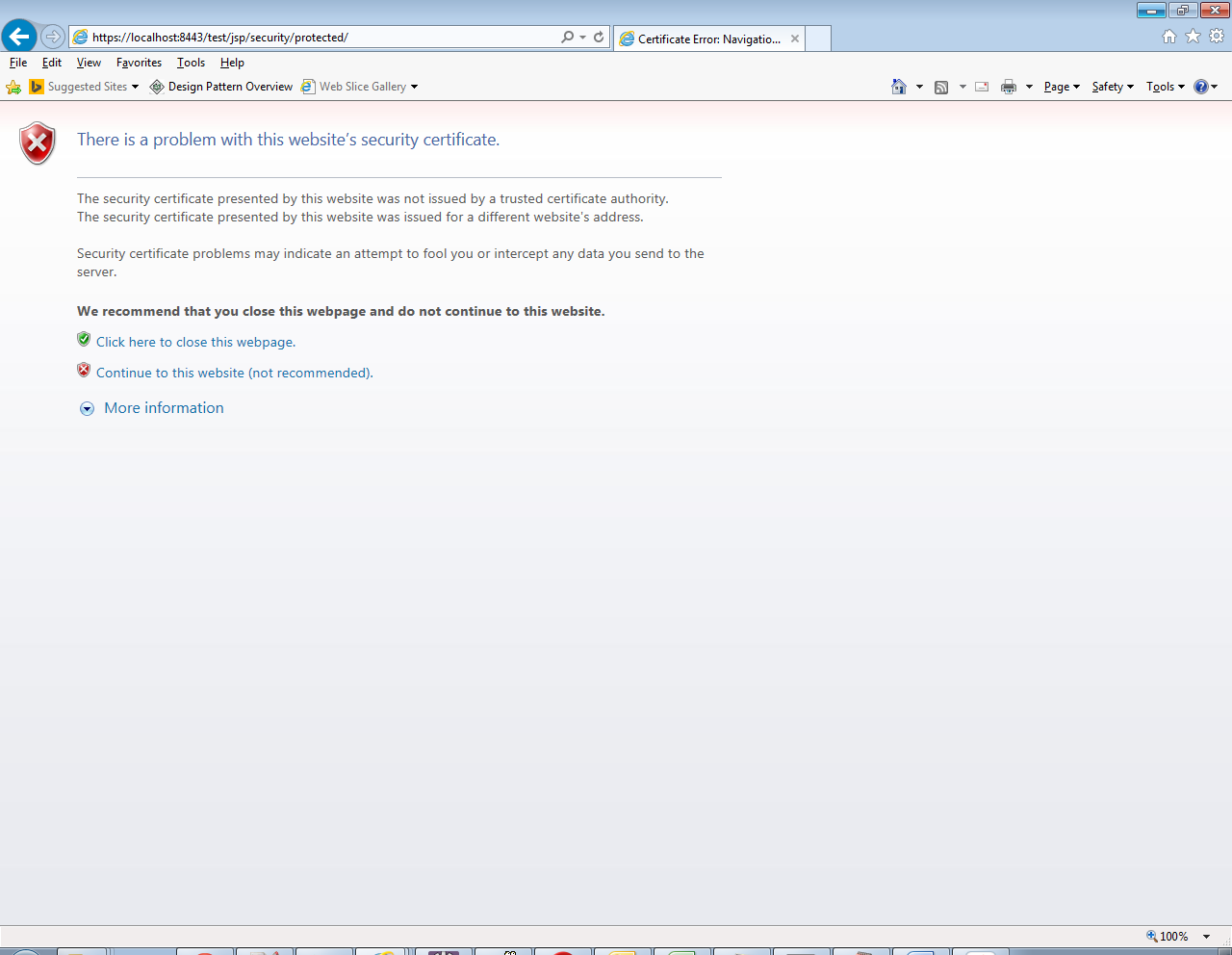
Web.xml

transport-guarantee value must be CONFIDENTIAL for https otherwise it should be NONE (default value)

|  |
| --- |
| <security-constraint>  <display-name>Example Security Constraint</display-name>  <web-resource-collection>  <web-resource-name>Protected Area</web-resource-name>  <!-- Define the context-relative URL(s) to be protected -->  <url-pattern>/jsp/security/protected/\*</url-pattern>  <!-- If you list http methods, only those methods are protected -->  <http-method>DELETE</http-method>  <http-method>GET</http-method>  <http-method>POST</http-method>  <http-method>PUT</http-method>  </web-resource-collection>  <user-data-constraint>  <transport-guarantee>CONFIDENTIAL</transport-guarantee>  </user-data-constraint>  <auth-constraint>  <!-- Anyone with one of the listed roles may access this area -->  <role-name>tomcat</role-name>  <role-name>role1</role-name>  </auth-constraint>    </security-constraint>  <!-- Default login configuration uses form-based authentication -->  <login-config>  <auth-method>BASIC</auth-method>  </login-config>  <!-- Security roles referenced by this web application -->  <security-role>  <role-name>role1</role-name>  </security-role>  <security-role>  <role-name>tomcat</role-name>  </security-role> |

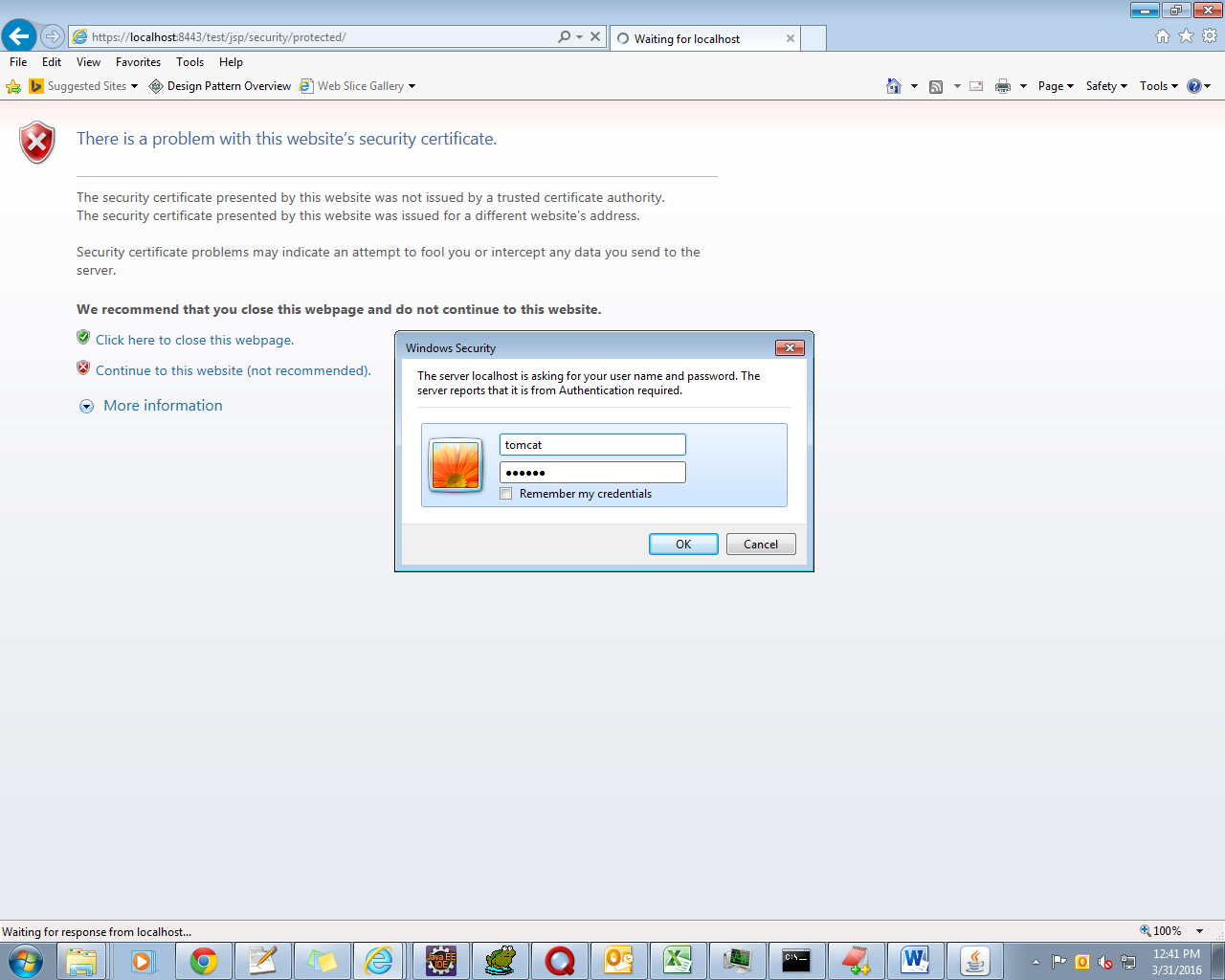
Now click on below link

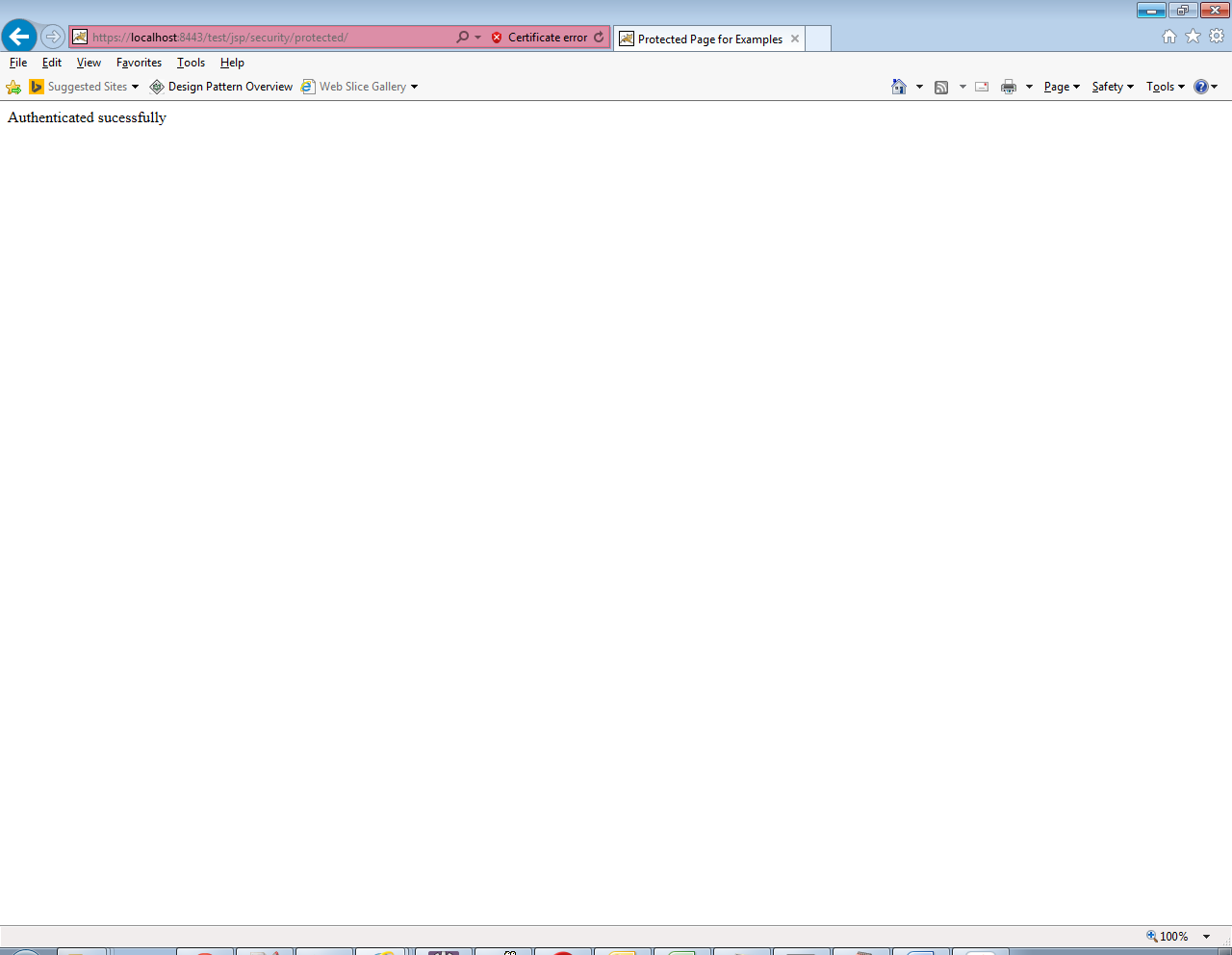
<https://localhost:8443/test/jsp/security/protected/>



Now click on Continue to this website

It will ask username/password since we used BASIC authentication





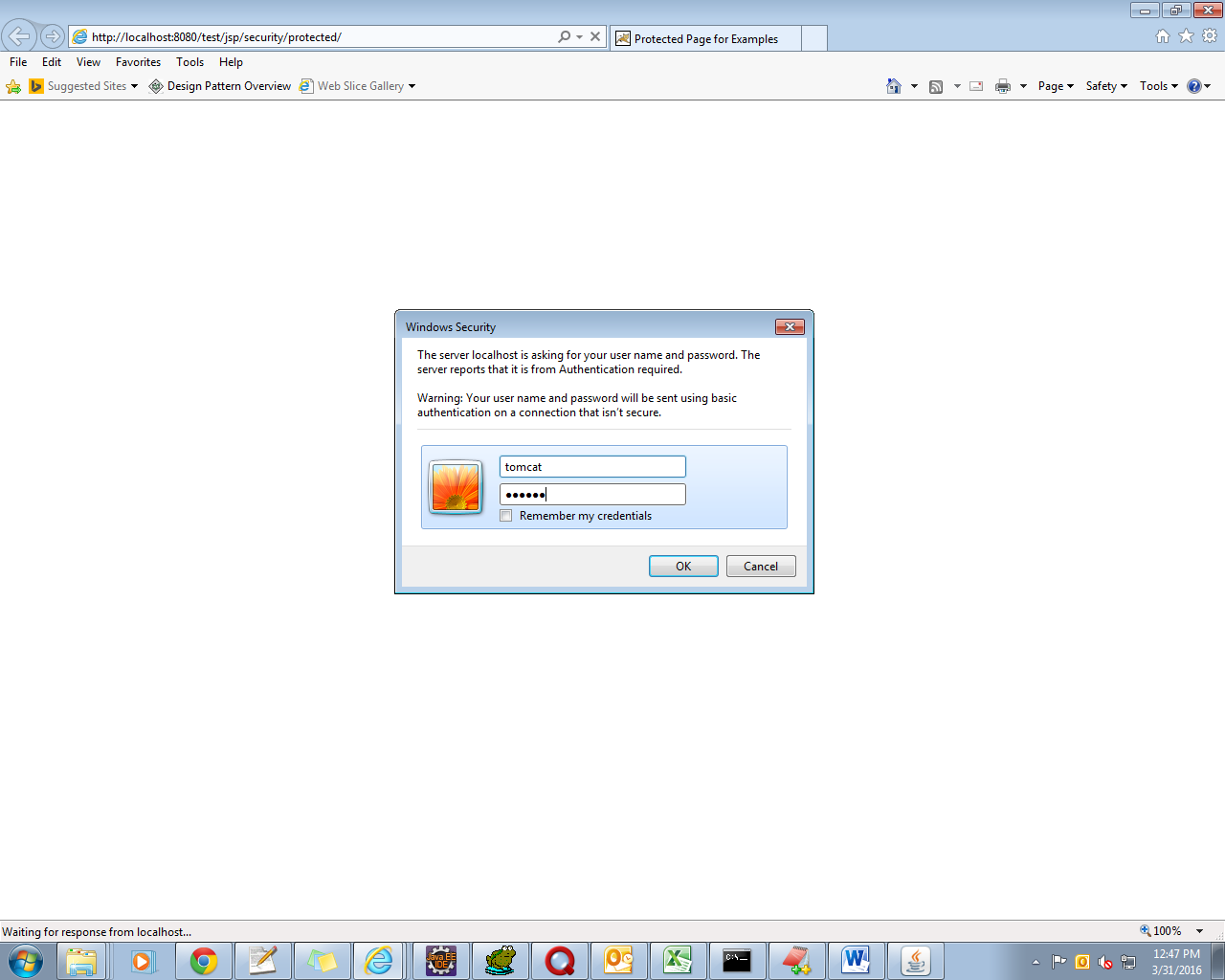
If you want to turn off the SSL, you don’t need to delete the code above from web.xml, simply change CONFIDENTIAL to NONE.

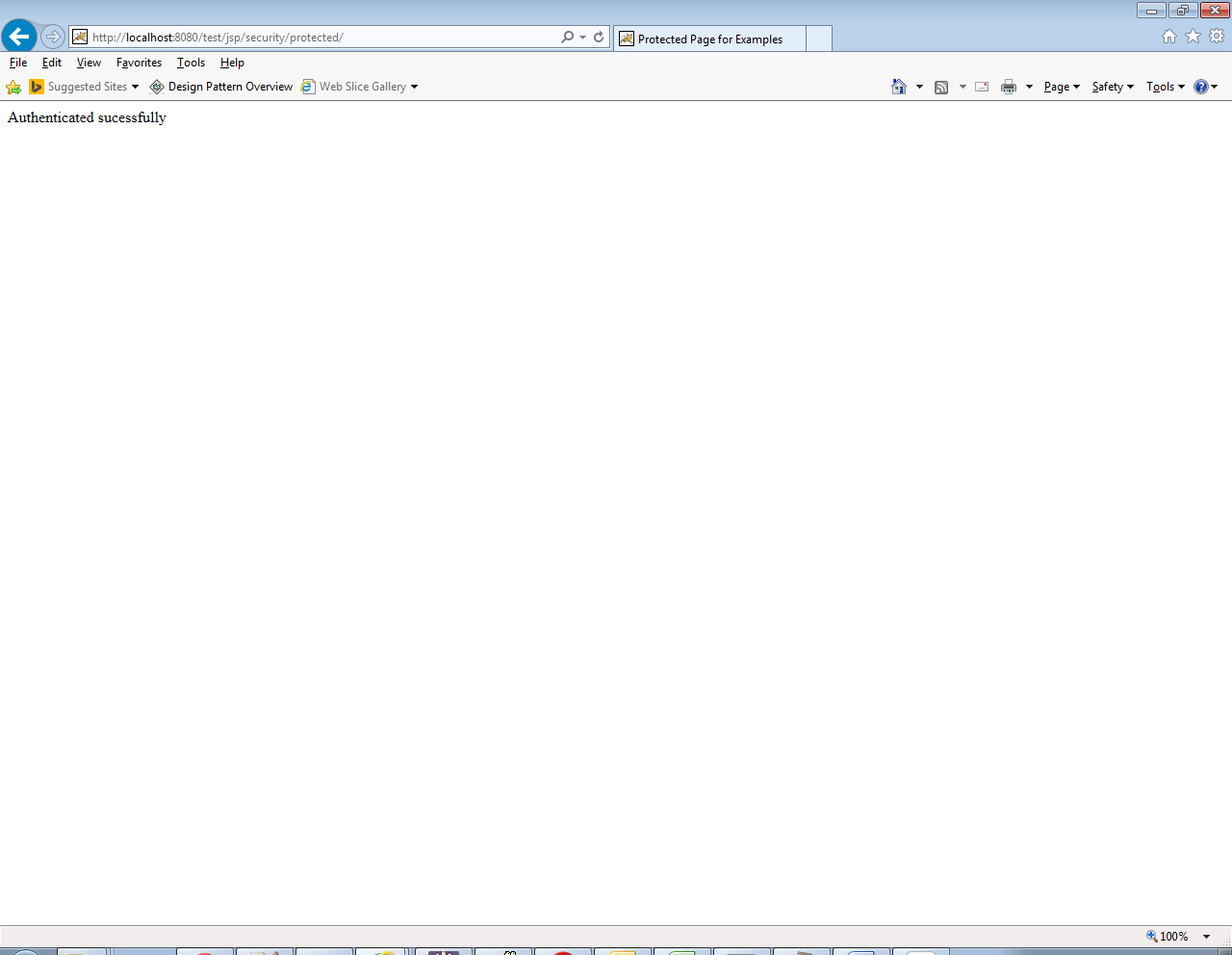
|  |
| --- |
| <security-constraint>  <display-name>Example Security Constraint</display-name>  <web-resource-collection>  <web-resource-name>Protected Area</web-resource-name>  <!-- Define the context-relative URL(s) to be protected -->  <url-pattern>/jsp/security/protected/\*</url-pattern>  <!-- If you list http methods, only those methods are protected -->  <http-method>DELETE</http-method>  <http-method>GET</http-method>  <http-method>POST</http-method>  <http-method>PUT</http-method>  </web-resource-collection>  <user-data-constraint>  <transport-guarantee>NONE</transport-guarantee>  </user-data-constraint>  <auth-constraint>  <!-- Anyone with one of the listed roles may access this area -->  <role-name>tomcat</role-name>  <role-name>role1</role-name>  </auth-constraint>    </security-constraint>  <!-- Default login configuration uses form-based authentication -->  <login-config>  <auth-method>BASIC</auth-method>  </login-config>  <!-- Security roles referenced by this web application -->  <security-role>  <role-name>role1</role-name>  </security-role>  <security-role>  <role-name>tomcat</role-name> |

Now you can access application over http without changing any configuration

Using below link

<http://localhost:8080/test/jsp/security/protected/>





**Configure Tomcat use SSL**

<https://www.mulesoft.com/tcat/tomcat-ssl>

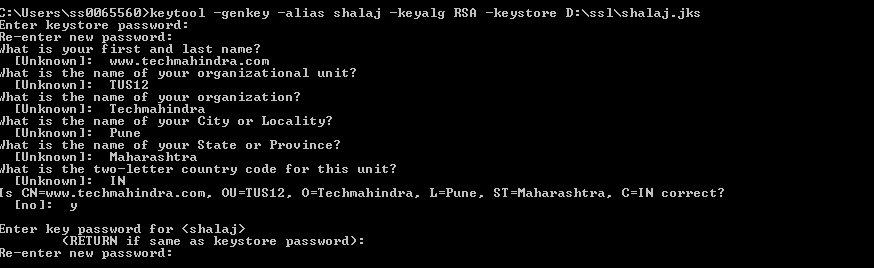
**Step 1 - Creating the Keystore**

Command

keytool -genkey -alias shalaj -keyalg RSA -keystore D:\ssl\shalaj.jks

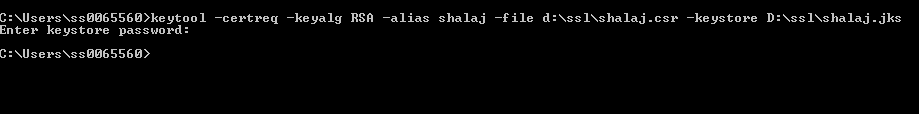
Make sure to keep password same (in my case password is shalaj)

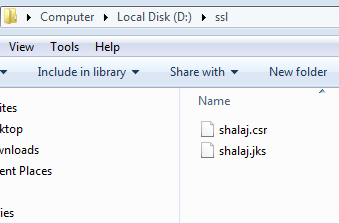
Note: put hostname in First name and last name like www.techmahindra.com



**Step 2 - Creating the Certificate Signing Request**

keytool -certreq -keyalg RSA -alias shalaj -file d:\ssl\shalaj.csr -keystore D:\ssl\shalaj.jks





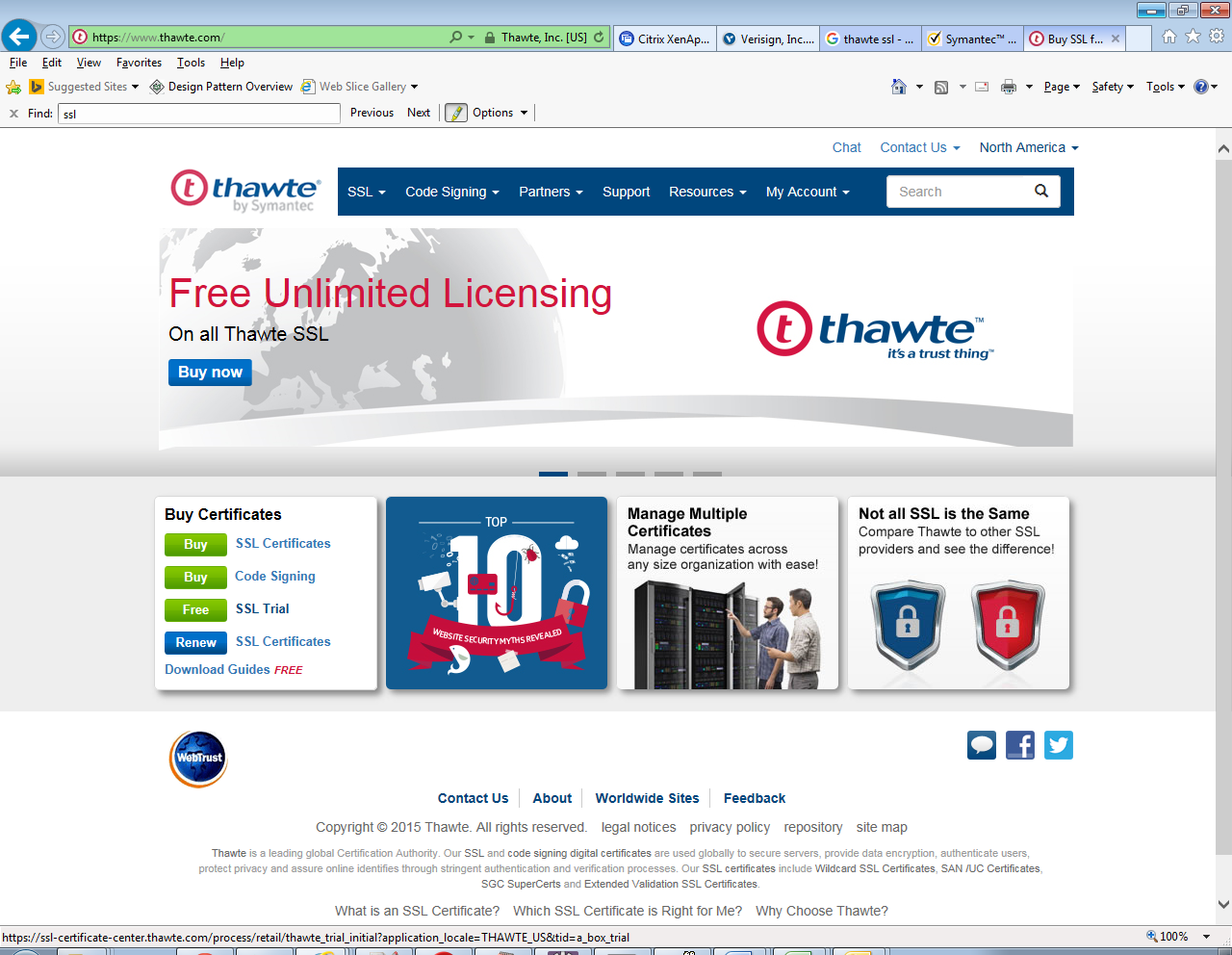
**Step 3 - Installing Your New Certificate**

In order to "anchor" your certificate's chain of trust, you have to download an additional certificate, called a "Root Certificate," from your CA, and then import both this certificate and your site's new certificate into your keystore. Your CA should provide information about obtaining a Root Certificate on their website.

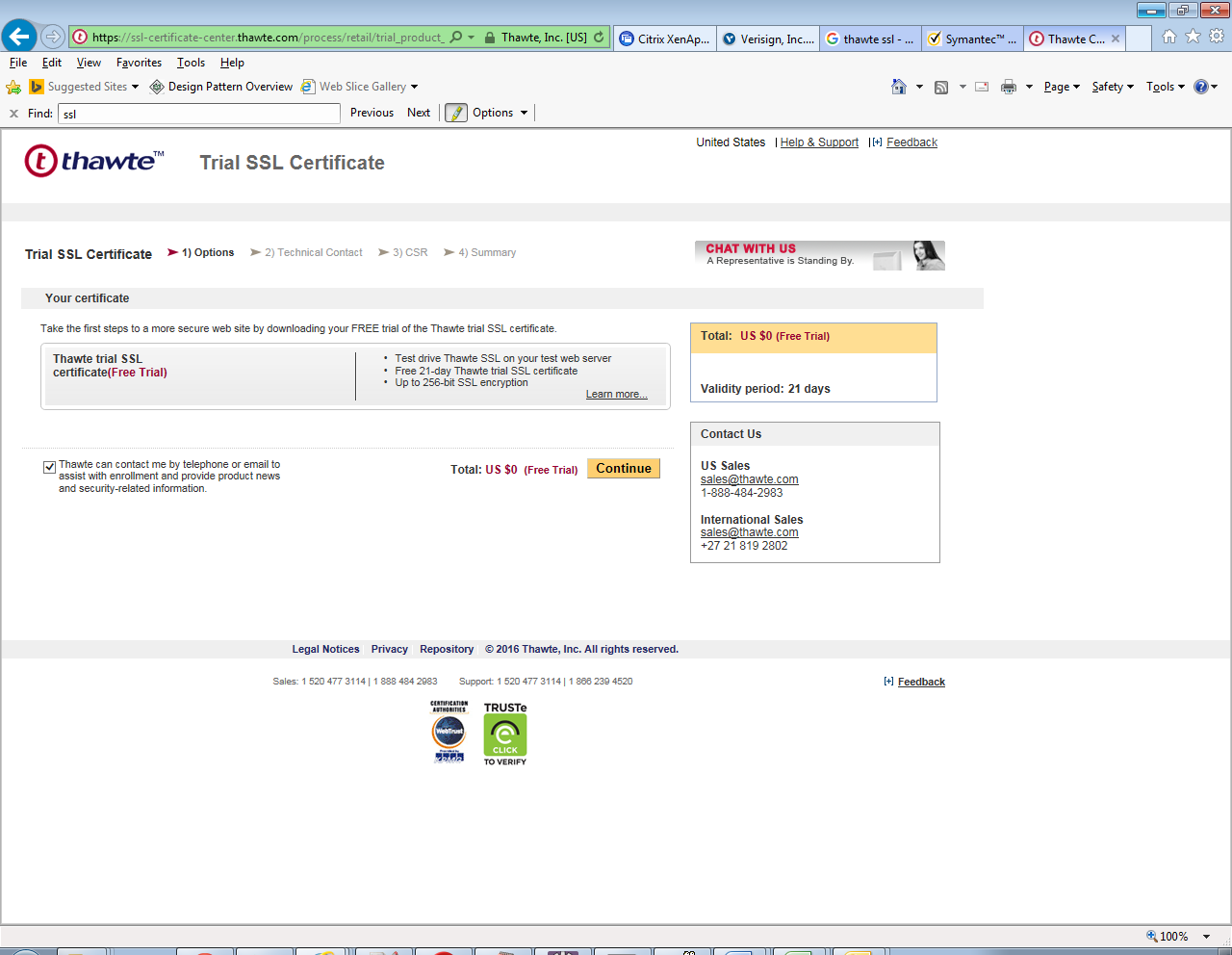
Once you've downloaded both your own Certificate and the Root certificate provided by your CA, import them into your keystore with the following commands

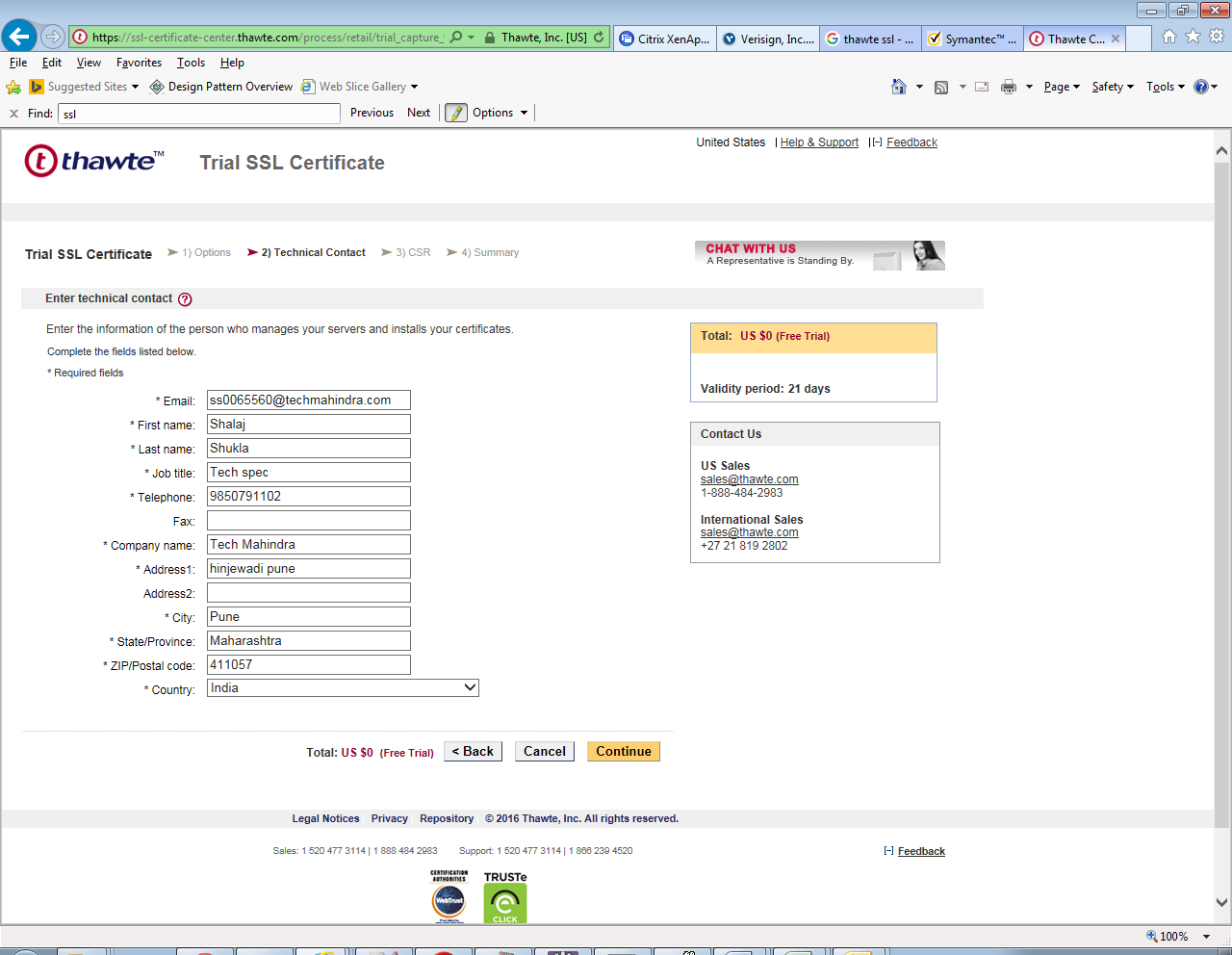
**THAWTE (Certificate Authority)**

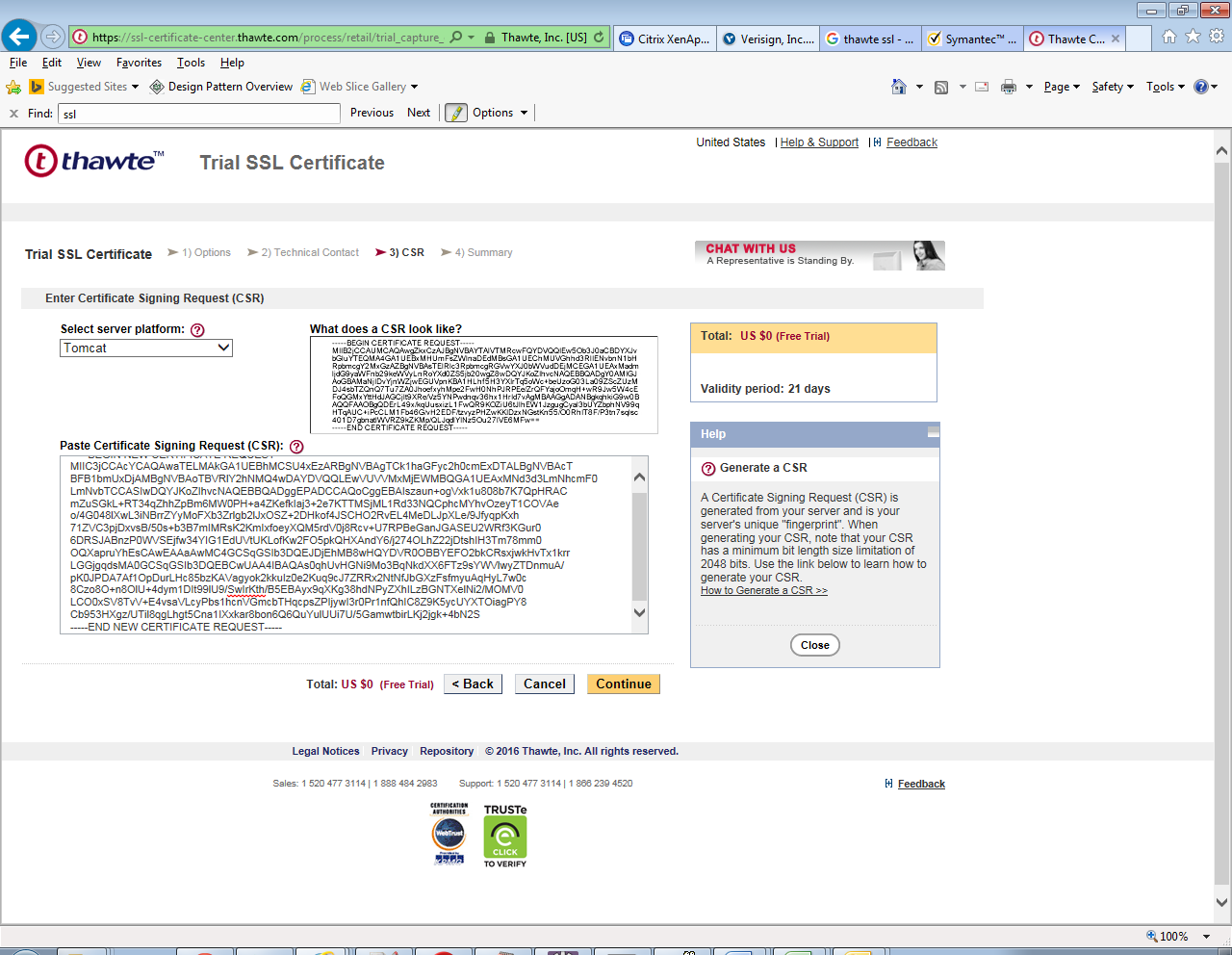
https://www.**thawte**.com

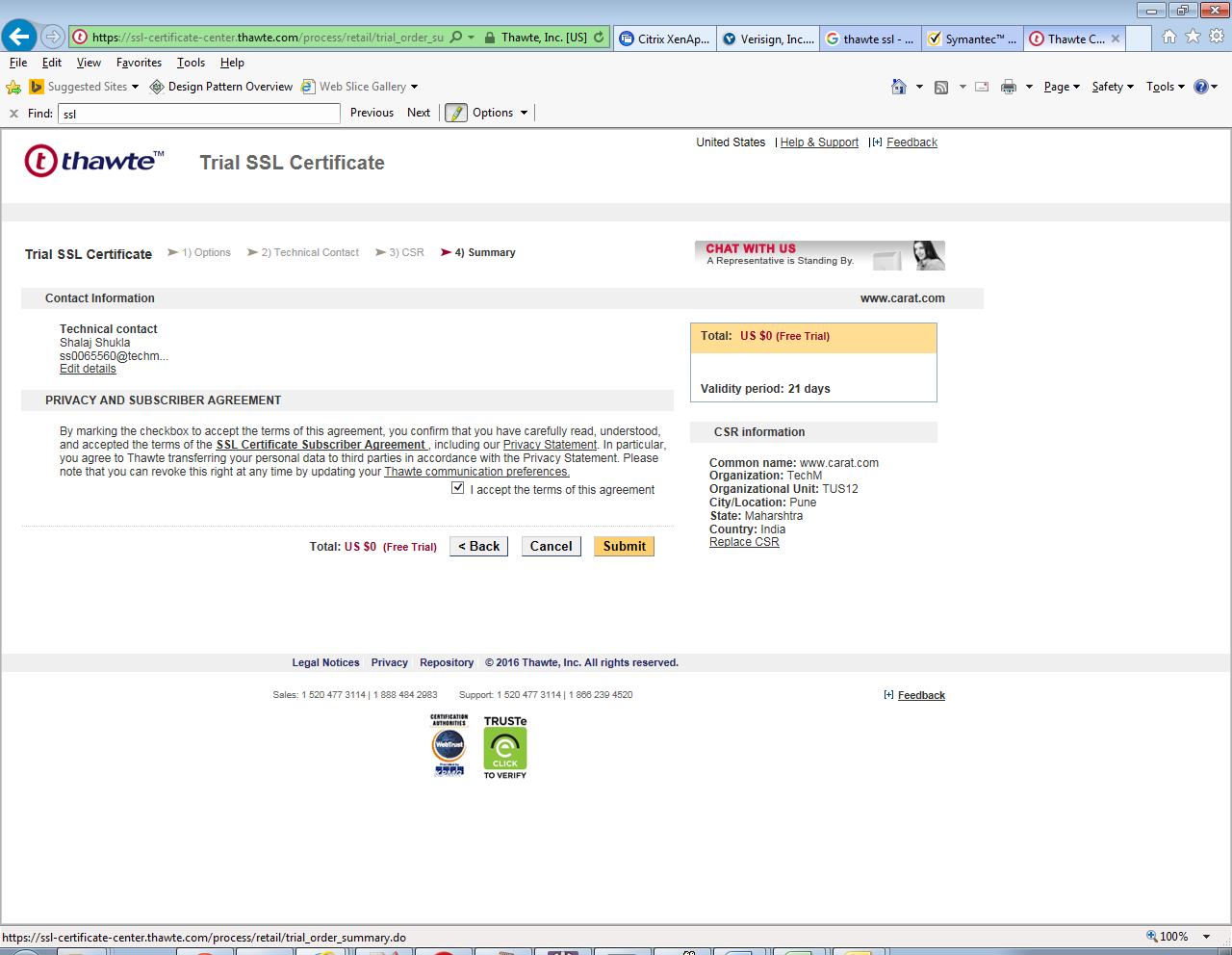


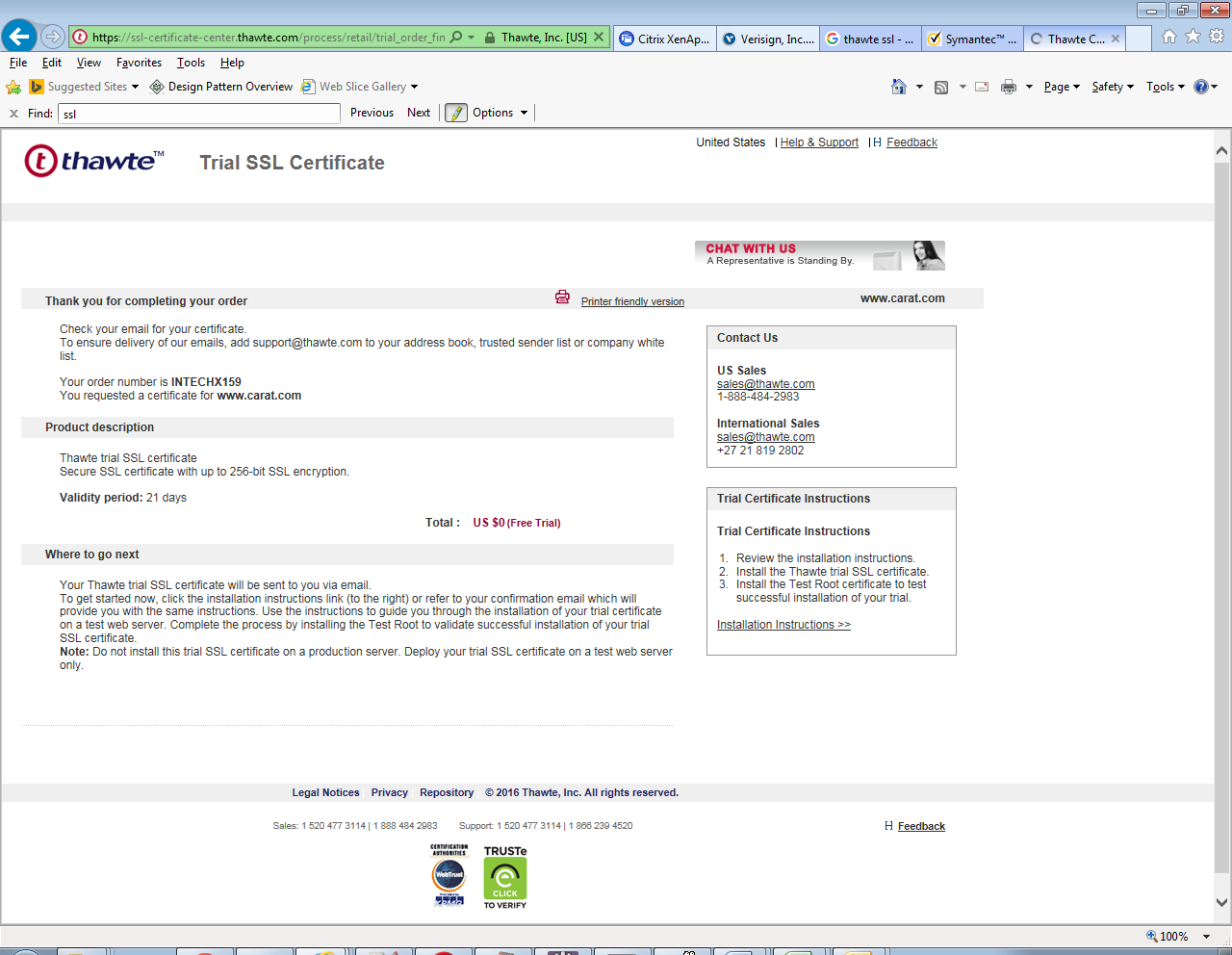
Click Free SSL Certificate



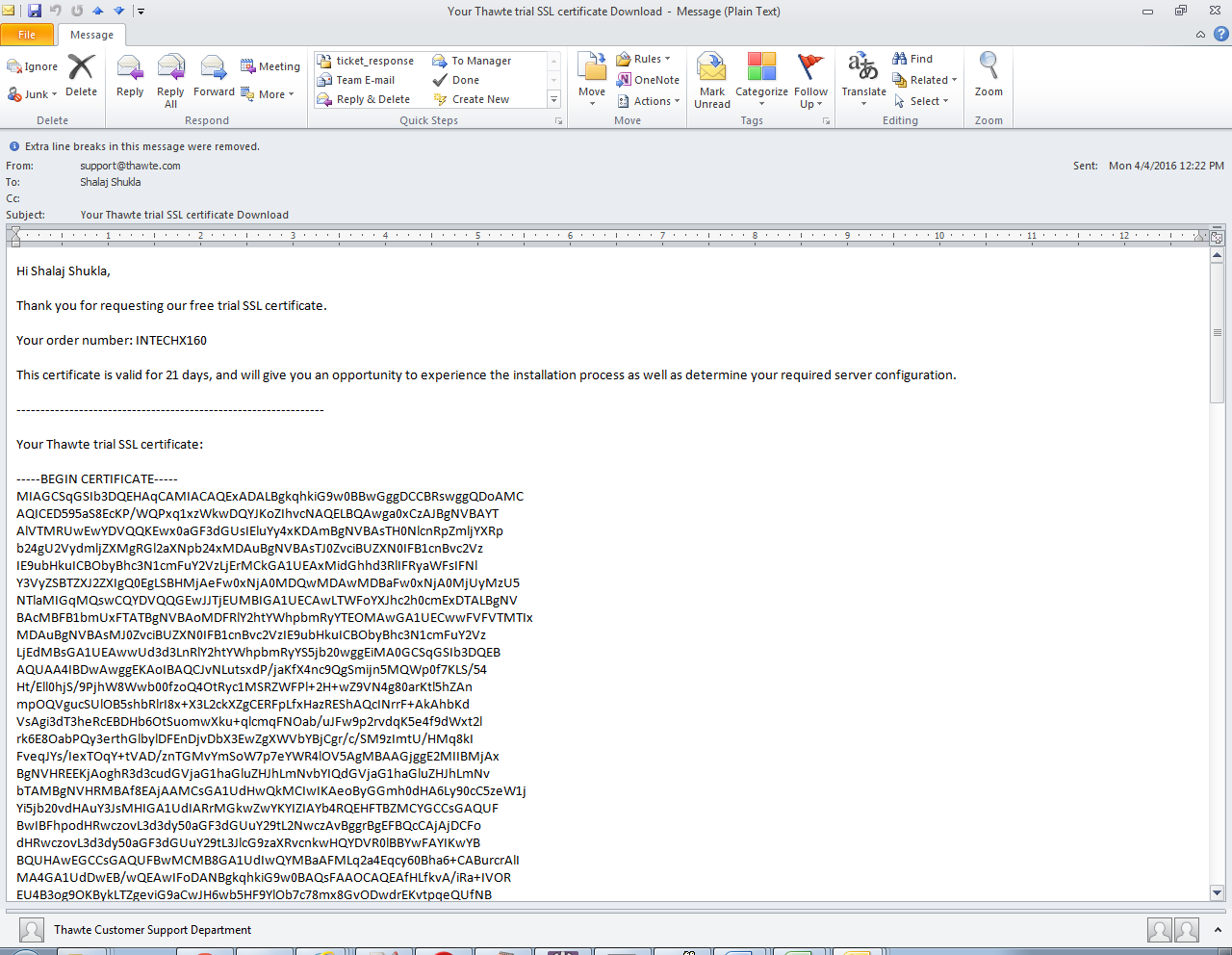








You will get an email on email id you have provided



|  |
| --- |
| Hi Shalaj Shukla,  Thank you for requesting our free trial SSL certificate.  Your order number: INTECHX160  This certificate is valid for 21 days, and will give you an opportunity to experience the installation process as well as determine your required server configuration.  ----------------------------------------------------------------  Your Thawte trial SSL certificate:  -----BEGIN CERTIFICATE-----  MIAGCSqGSIb3DQEHAqCAMIACAQExADALBgkqhkiG9w0BBwGggDCCBRswggQDoAMC  AQICED595aS8EcKP/WQPxq1xzWkwDQYJKoZIhvcNAQELBQAwga0xCzAJBgNVBAYT  AlVTMRUwEwYDVQQKEwx0aGF3dGUsIEluYy4xKDAmBgNVBAsTH0NlcnRpZmljYXRp  b24gU2VydmljZXMgRGl2aXNpb24xMDAuBgNVBAsTJ0ZvciBUZXN0IFB1cnBvc2Vz  IE9ubHkuICBObyBhc3N1cmFuY2VzLjErMCkGA1UEAxMidGhhd3RlIFRyaWFsIFNl  Y3VyZSBTZXJ2ZXIgQ0EgLSBHMjAeFw0xNjA0MDQwMDAwMDBaFw0xNjA0MjUyMzU5  NTlaMIGqMQswCQYDVQQGEwJJTjEUMBIGA1UECAwLTWFoYXJhc2h0cmExDTALBgNV  BAcMBFB1bmUxFTATBgNVBAoMDFRlY2htYWhpbmRyYTEOMAwGA1UECwwFVFVTMTIx  MDAuBgNVBAsMJ0ZvciBUZXN0IFB1cnBvc2VzIE9ubHkuICBObyBhc3N1cmFuY2Vz  LjEdMBsGA1UEAwwUd3d3LnRlY2htYWhpbmRyYS5jb20wggEiMA0GCSqGSIb3DQEB  AQUAA4IBDwAwggEKAoIBAQCJvNLutsxdP/jaKfX4nc9QgSmijn5MQWp0f7KLS/54  Ht/Ell0hjS/9PjhW8Wwb00fzoQ4OtRyc1MSRZWFPl+2H+wZ9VN4g80arKtl5hZAn  mpOQVgucSUlOB5shbRlrI8x+X3L2ckXZgCERFpLfxHazREShAQcINrrF+AkAhbKd  VsAgi3dT3heRcEBDHb6OtSuomwXku+qlcmqFNOab/uJFw9p2rvdqK5e4f9dWxt2l  rk6E8OabPQy3erthGlbylDFEnDjvDbX3EwZgXWVbYBjCgr/c/SM9zImtU/HMq8kI  FveqJYs/IexTOqY+tVAD/znTGMvYmSoW7p7eYWR4lOV5AgMBAAGjggE2MIIBMjAx  BgNVHREEKjAoghR3d3cudGVjaG1haGluZHJhLmNvbYIQdGVjaG1haGluZHJhLmNv  bTAMBgNVHRMBAf8EAjAAMCsGA1UdHwQkMCIwIKAeoByGGmh0dHA6Ly90cC5zeW1j  Yi5jb20vdHAuY3JsMHIGA1UdIARrMGkwZwYKYIZIAYb4RQEHFTBZMCYGCCsGAQUF  BwIBFhpodHRwczovL3d3dy50aGF3dGUuY29tL2NwczAvBggrBgEFBQcCAjAjDCFo  dHRwczovL3d3dy50aGF3dGUuY29tL3JlcG9zaXRvcnkwHQYDVR0lBBYwFAYIKwYB  BQUHAwEGCCsGAQUFBwMCMB8GA1UdIwQYMBaAFMLq2a4Eqcy60Bha6+CABurcrAlI  MA4GA1UdDwEB/wQEAwIFoDANBgkqhkiG9w0BAQsFAAOCAQEAfHLfkvA/iRa+IVOR  EU4B3og9OKBykLTZgeviG9aCwJH6wb5HF9YlOb7c78mx8GvODwdrEKvtpqeQUfNB  M4cCZoilDiT1HCdQ/c5tDwvP4nplRK8eBy8LZ2+//WX3oIBCNsX4jy8yWxS42bMW  r+Vtaza9x9ZoM9yIJAM+mkea4+kFXvCIXW7qoZxepwEd1BcBlWNeXD4tQeSms+Ti  LSRBguNCmMfkCRWkNwKhWHii6cD8JzZOm0JWYm/XzMbQW3obANlx2IUSqQvrAiBJ  MJ0fOLAfE2IV+qRmhJsv9hzLfjRdh8AQ3AVMXxuZiDVCoTsoROVVmVVGG5DTQMDs  d6BrVzCCBI8wggN3oAMCAQICEHR4ahOp2n68snSbRQTQwJEwDQYJKoZIhvcNAQEL  BQAwga0xCzAJBgNVBAYTAlVTMRUwEwYDVQQKEwx0aGF3dGUsIEluYy4xKDAmBgNV  BAsTH0NlcnRpZmljYXRpb24gU2VydmljZXMgRGl2aXNpb24xMDAuBgNVBAsTJ0Zv  ciBUZXN0IFB1cnBvc2VzIE9ubHkuICBObyBhc3N1cmFuY2VzLjErMCkGA1UEAxMi  dGhhd3RlIFRyaWFsIFNlY3VyZSBTZXJ2ZXIgUm9vdCBDQTAeFw0xNDEyMDUwMDAw  MDBaFw0yNDEyMDQyMzU5NTlaMIGtMQswCQYDVQQGEwJVUzEVMBMGA1UEChMMdGhh  d3RlLCBJbmMuMSgwJgYDVQQLEx9DZXJ0aWZpY2F0aW9uIFNlcnZpY2VzIERpdmlz  aW9uMTAwLgYDVQQLEydGb3IgVGVzdCBQdXJwb3NlcyBPbmx5LiAgTm8gYXNzdXJh  bmNlcy4xKzApBgNVBAMTInRoYXd0ZSBUcmlhbCBTZWN1cmUgU2VydmVyIENBIC0g  RzIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCsYTNXjrydbz2Zg1jn  6jVty8oHHG0qchnHZlSA9PwdqbcvMj6XQM1B7au1FgICaUo6g7pumM3HbgmXM/i5  9e0WIVVhoputeEThCkmBX28uYkYimHzUq9ag4YnrAkCqp721LLTtIJwLMdERcGeG  kpVgfOLixCCiWmAyBZOuJPJV1oGRELfrob70bHEz14VbkX4/NpQXTQkwy1Xx9Omz  z2zjcrCNUxcssTqgxizHh3AYr/Qf/veex+x/PzEzPNQdOovWWizEbCJvm0h8Etxt  j9N9yX/QLc4ss9waUXRAtHPPgyONFqxr20h3SsepHNojmK+2DINTn1XGGpUxGNkv  IHzFAgMBAAGjgagwgaUwEgYDVR0TAQH/BAgwBgEB/wIBADAOBgNVHQ8BAf8EBAMC  AQYwHQYDVR0OBBYEFMLq2a4Eqcy60Bha6+CABurcrAlIMD8GA1UdHwQ4MDYwNKAy  oDCGLmh0dHA6Ly9jcmwudGhhd3RlLmNvbS90aGF3dGVUcmlhbFNTTFJvb3RDQS5j  cmwwHwYDVR0jBBgwFoAUBUJohgPpyWXBJ7PZm9QP93/1BUAwDQYJKoZIhvcNAQEL  BQADggEBADwN2bXOe0XCLh/9I7n66m+Zo1GyCsb+ODh9l4iEktegFLQQDdfDw7NC  jpAxtMbAb01UN4QhetDyv6D3szl240GFFLXoFS9IARYzikPY5NjJLGRraacFYzD0  JmjUmPLzaazbloeB1YDL/cZzZ/qfhT+Udf6v1CcnzGcPjaetBX76A5S5zRgh2fKX  LqmXM96u3xY5FD9EYyRme2dReg2xUoLt+yUiqToFW+UxrNkBVEtc7yximc2D+sb1  Fx4Q3Ki03PjWOuxZv8F8Cuhwg8rMJKgUpBfkgRo8pe8yU/KM/cRZyV7keNV470Qb  22hr5l2kUrYGG5Q1/1MPVrZewMW1fN0AADEAAAAAAAAA  -----END CERTIFICATE-----  ----------------------------------------------------------------  Important installation instructions  For the trial SSL certificate to provide the same user experience as a trusted certificate, the Thawte Trial Secure Server Intermediate CA and the Thawte Test CA Root certificate must also be installed:    Thawte Trial Secure Server Intermediate CA:  -----BEGIN CERTIFICATE-----  MIIEijCCA3KgAwIBAgIQO6EebHiOSuFcciShhv7n1TANBgkqhkiG9w0BAQUFADCB  rTELMAkGA1UEBhMCVVMxFTATBgNVBAoTDHRoYXd0ZSwgSW5jLjEoMCYGA1UECxMf  Q2VydGlmaWNhdGlvbiBTZXJ2aWNlcyBEaXZpc2lvbjEwMC4GA1UECxMnRm9yIFRl  c3QgUHVycG9zZXMgT25seS4gIE5vIGFzc3VyYW5jZXMuMSswKQYDVQQDEyJ0aGF3  dGUgVHJpYWwgU2VjdXJlIFNlcnZlciBSb290IENBMB4XDTEwMDIwNDAwMDAwMFoX  DTIwMDIwMzIzNTk1OVowgagxCzAJBgNVBAYTAlVTMRUwEwYDVQQKEwxUaGF3dGUs  IEluYy4xKDAmBgNVBAsTH0NlcnRpZmljYXRpb24gU2VydmljZXMgRGl2aXNpb24x  MDAuBgNVBAsTJ0ZvciBUZXN0IFB1cnBvc2VzIE9ubHkuICBObyBhc3N1cmFuY2Vz  LjEmMCQGA1UEAxMdVGhhd3RlIFRyaWFsIFNlY3VyZSBTZXJ2ZXIgQ0EwggEiMA0G  CSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDWxZ4vCS+9h4gW7CnTwPhEFueWeH8I  FM/+n870zbwOV13YZQ4pYgxQYqqKwMXA+6l8fkv5n7zIAnluoaa/NfMb9yNGXBGU  8c/CLDOdBlrC2ZGnwjuxgsR2gK2Mvqq9PF3Z16SkHphbD6NLoJ/6O5C6JKebBgYj  UlumU4DF92wIZEFNBOsGOVd1IeU/wzJ/kQY8jU6JaydOPImie2OlfTVbMdSKP6GB  1OJ+s8Tn6LrMocUb2lwYebnV1IEPTsWQTgPz3dYWsshUtunStECIhPmVcfMARJ5L  Rfbwa/Pn3H9YSnjSV3T5z2BUuzDSqxJHqjJ6psatdheRql0accQPknFTAgMBAAGj  gagwgaUwEgYDVR0TAQH/BAgwBgEB/wIBADA/BgNVHR8EODA2MDSgMqAwhi5odHRw  Oi8vY3JsLnRoYXd0ZS5jb20vdGhhd3RlVHJpYWxTU0xSb290Q0EuY3JsMA4GA1Ud  DwEB/wQEAwIBBjAdBgNVHQ4EFgQUKWy1Nf0D1kj7BO86n6sVTgr0TVAwHwYDVR0j  BBgwFoAUBUJohgPpyWXBJ7PZm9QP93/1BUAwDQYJKoZIhvcNAQEFBQADggEBAEiZ  692P7wsPEJonArD7jKMHE9s6y1FRXzzDP7ahGZ7OQgJdrqRP8vYDo/1O/tAQQ3W9  jfWb3vTZUMFMpzLHyiVi+gmK05TOkMLXDvtPLW1WBO0VxZH3Q49C2k9eBFSqHmkh  y/7nay7BMnyFhWZKwtR/P2octog6fZRWxuXBxDmsjq2OiNotmXZqqEzN54gE+yWo  Yqy17Y09GQFjXBeqPhSjfrismdWGqQRTT8M6di1kxb+t7O1Xd+492on2DMtJev3X  4lpuhl7mcdSxNYbdxWol6PF/6B2nJUcvb3DYn5zNF9871Kes7raMLkjViKzVt8By  MjpGgXwjtW+WMNzutfM=  -----END CERTIFICATE-----  Thawte Test CA Root certificate:  -----BEGIN CERTIFICATE-----  MIIEVwYJKoZIhvcNAQcCoIIESDCCBEQCAQExADALBgkqhkiG9w0BBwGgggQsMIIE  KDCCAxCgAwIBAgIQP1MpAnGSsgnuvzehial42DANBgkqhkiG9w0BAQUFADCBrTEL  MAkGA1UEBhMCVVMxFTATBgNVBAoTDHRoYXd0ZSwgSW5jLjEoMCYGA1UECxMfQ2Vy  dGlmaWNhdGlvbiBTZXJ2aWNlcyBEaXZpc2lvbjEwMC4GA1UECxMnRm9yIFRlc3Qg  UHVycG9zZXMgT25seS4gIE5vIGFzc3VyYW5jZXMuMSswKQYDVQQDEyJ0aGF3dGUg  VHJpYWwgU2VjdXJlIFNlcnZlciBSb290IENBMB4XDTA5MTAwOTAwMDAwMFoXDTI5  MTAwODIzNTk1OVowga0xCzAJBgNVBAYTAlVTMRUwEwYDVQQKEwx0aGF3dGUsIElu  Yy4xKDAmBgNVBAsTH0NlcnRpZmljYXRpb24gU2VydmljZXMgRGl2aXNpb24xMDAu  BgNVBAsTJ0ZvciBUZXN0IFB1cnBvc2VzIE9ubHkuICBObyBhc3N1cmFuY2VzLjEr  MCkGA1UEAxMidGhhd3RlIFRyaWFsIFNlY3VyZSBTZXJ2ZXIgUm9vdCBDQTCCASIw  DQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAML5kYGJjOCgOr6QShUH2ruIqOdS  YQJP+P/Rvm94Xkdn9X/ob8AfnLUaXAEByfLEaAlzVKgt2dqlg6KzMWv1Gui2i9rd  WXFdIJcfVruRC76RRup0SBW+KJAnwGTuv69fLtifb+3fhPcioe5bT/0i6/A4NEri  p1QmYWlt0G2m+jpNg1/bvNtvZuA1was/cpKUKwIWsx0jWbNhQjKKvEGuMzGnFImp  hg+Tu8LYVFevno9Z0+sk9OXugngBDykCPZeOFIvWl7VNasSRuNUL6W3DqKlUQIiO  YtHeLNur18z1sf2rq4iB5pAzySYqxyFNM1o8eoGFLXkt/kdZ74uW64MzTCsCAwEA  AaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMCAQYwHQYDVR0OBBYE  FAVCaIYD6cllwSez2ZvUD/d/9QVAMA0GCSqGSIb3DQEBBQUAA4IBAQC4PptVQCvj  jNW1WG+cq8pbv+IwnH7TC11VG3SNKEOp/3DIAR2yjPGUJy5+oDyeVxU9qWanO4uM  NaXMiGPDVsULlZwVOQAF6pqWXeZcL4ErcC8XzcrPufbKy2nP/8VUb7y7dA9YM6qc  25j29J2YTw9FNgQDVOvkCK+8SpTLVImSGYWE9/+qWXUff6cD9cw5nHPxnCo67ozm  k+K8FVK1mvA22IrH0MGEdyXhxNwbxeL/oOr7koALOv8lDR2IJqLZMwoMMG7dP64P  AQwPtTPBJr03ysFL61qDrYVRSXcE8bM2ar5KVXfIwxZuK5FOf8bMSp1DqKKLyOd9  BFgQ0GxeXdXAMQA=  -----END CERTIFICATE-----  ----------------------------------------------------------------  Installing your certificate  Installation instructions for a range of web server solutions are available on our support site here:  <https://search.thawte.com/support/ssl-digital-certificates/index?page=content&id=SO7137>  Remember, install your trial certificate on test or development servers only. The trial SSL certificate is intended for testing purposes only.  ----------------------------------------------------------------  Your Thawte advisor will be in touch during the next few days but if you need any immediate assistance please feel free to contact us by calling, sending an e-mail or  making use of our live chat facility below.  Thank you for your interest in thawte!  The Thawte Team  Email: [sales@thawte.com](mailto:sales@thawte.com)  Tel USA: +1 888 484 2983  Tel Int: +27 21 937 8902  Chat: <http://www.thawte.com/popups/chat/chat_retail.html>  ----------------------------------------------------------------  © 2009 thawte, Inc. All rights reserved  Read Our Privacy Policy:  <https://www.thawte.com/legal/privacy.html>  To opt out of future non-service e-mails, please visit:  <https://www.thawte.com/compref/> or mail your request including your e-mail address to:  thawte, Inc. - Attention: Subscriber Services, 487 East Middlefield Road, Mountain View, CA 94043, USA |

<https://products.thawte.com/orders/orderinformation/authentication.do>

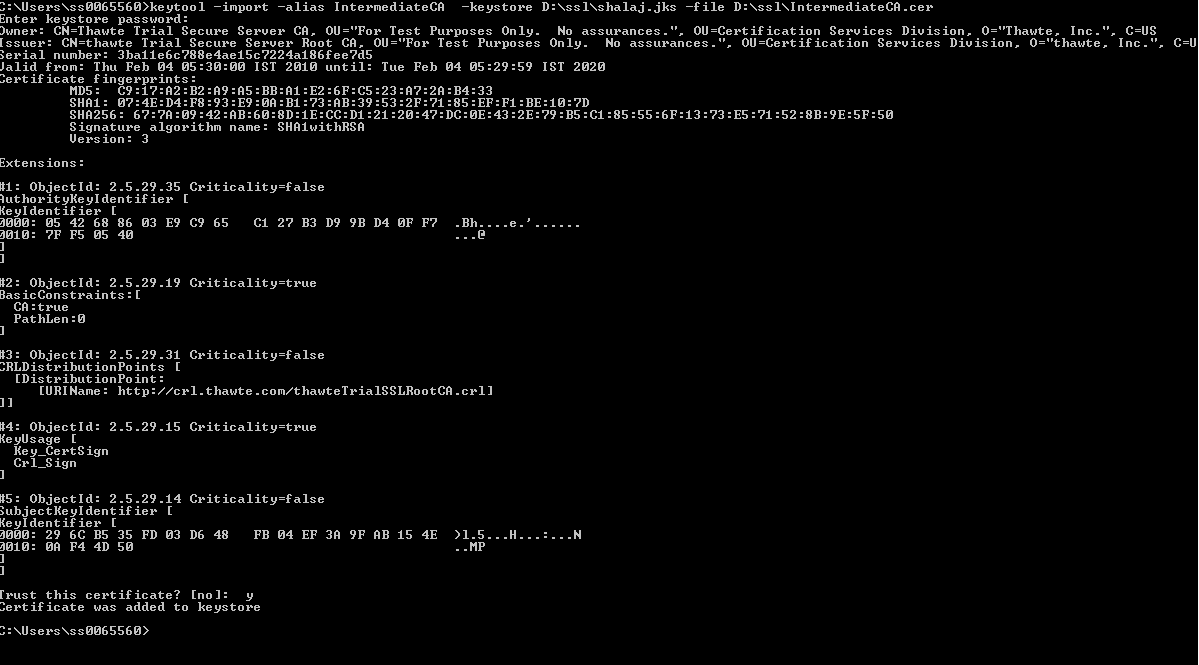
We are now having Root certificate and intermediate certificate

Now create two files namely IntermediateCA.cer and RootCA.cer in D:/ssl and copy paste the contenets that you received in mail

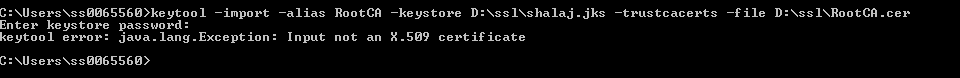
1. Import intermediate CA in your key store file

Issue following command

keytool -import -alias IntermediateCA -keystore D:\ssl\shalaj.jks -file D:\ssl\IntermediateCA.cer



Getting error while importing root certificate



Thwate give us separate link to download x.509 certificate but it need login information which is not available

Did with Symantec see attached doc

