LIMERICK INSTITUTE OF TECHNOLOGY
SCHOOL OF SCIENCE, ENGINEERING & I.T.
Department of Information Technology

B.Sc. In Software Development. Year 4.
Distributed Object Based Systems.
Using SSL To Work With A Secure Connection.

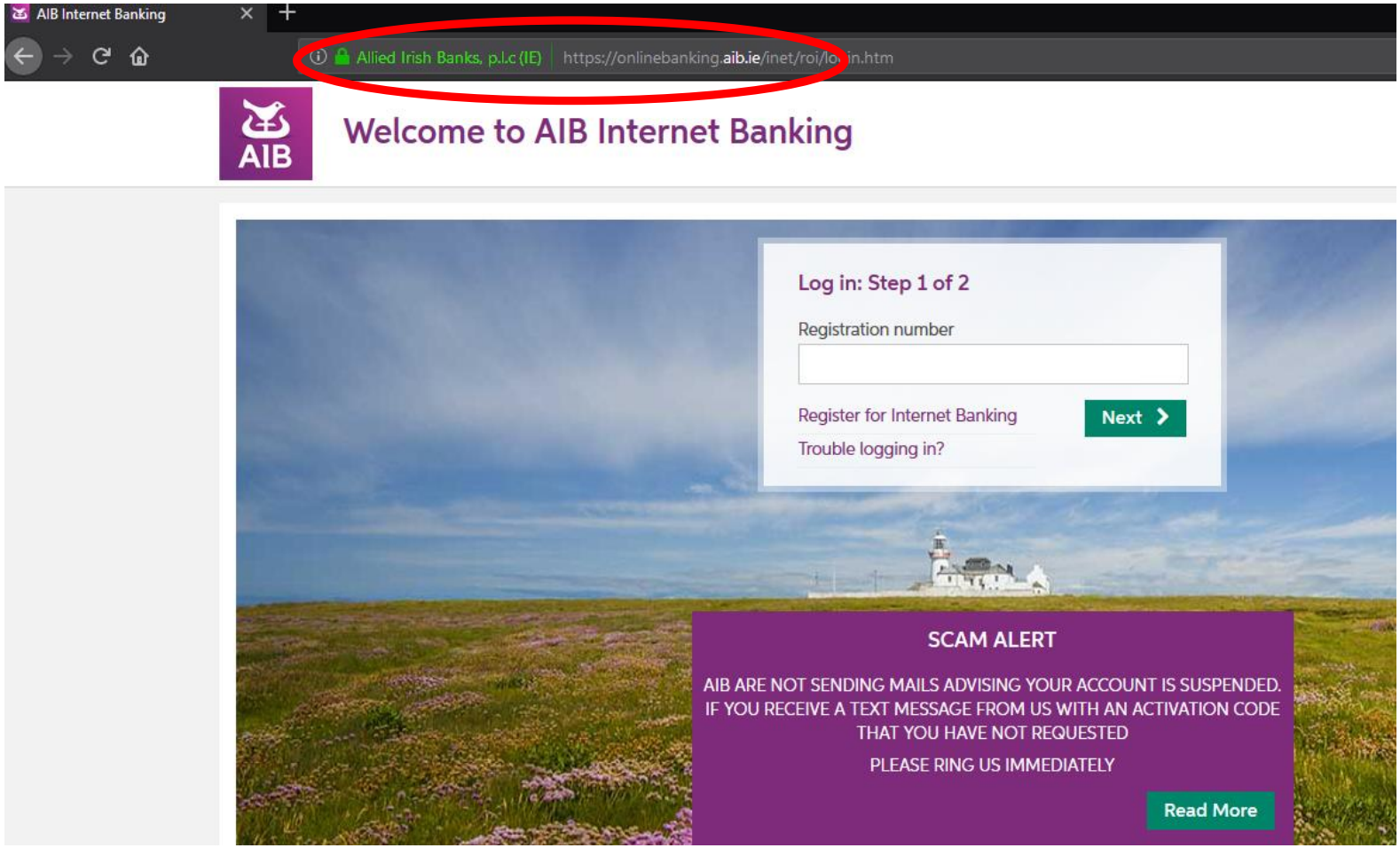
Introduction

- If your application workds with sensitive data such as credit-card numbers or passwords, you should use a secure connection when you send data between the client and sever.
  - Otherwise this data can be intercepted.
- SSL (Secure Sockets Layer) is a protocol that lets you transfer data securely.
- Two purposes of SSL.
  1. Verifying that you are talking directly to the server that you think you are talking to
  2. Ensuring that only the server can read what you send it and only you can read what it sends back

Introduction

- Even if someone were to intercept the message transmitted from the browser to the server, if they're encrypted they wont be able to make sense of any of the actual data you send.
  - They can roughly estimate how much data you're sending, but that's about it.
- There is now a trend amongst a number of websites to conduct conversations entirely using HTTPS.
  - HTTP traffic is vulnerable.
- SSL only protects data in transit from the browser -> server. It doesn't (obviously) protect your site from SQLi or XSS etc.

Introduction



Introduction

- With SSL the data is encrypted before it is transmitted between the browser and the server.
- Intercepting the data is useless. unless they can break the encryption code.
- TLS (Transport Layer Security) is another protocol that is used.
  - As a user its hard to tell whether you are using TLS or SSL.

Introduction

- Another advantage of using SSL/TSL is that you can determine if data has been tampered with during transit.
- You can also verify that a client or a server is who they claim to be.

How SSL Works

- To use SSL the client and server must provide authentication.
  - That way the client and server can accept or reject the secure connection.
- Before a secure connection is established, the server uses SSL server to authenticate itself.
  - It does this by providing a digital secure certificate to the browser.
  - If the browser deems that the certificate hasn’t come from a trusted source it informs the user
  - The user can decide in this case if they want to trust the certificate or not.
  - If the user chooses to accept the certificate a secure connection is established.

Sample Digital Certificate

Certificate Viewer: "onlinebanking.aib.ie"

General | Details

This certificate has been verified for the following uses:

- SSL Client Certificate
- SSL Server Certificate

Issued To

Common Name (CN)	onlinebanking.aib.ie
Organization (O)	Allied Irish Banks, p.l.c
Organizational Unit (OU)	Infra-Support
Serial Number	0F:B4:47:7F:C3:34:0E:88:4C:43:D6:86:A0:00:0F:91

Issued By

Common Name (CN)	DigiCert SHA2 Extended Validation Server CA
Organization (O)	DigiCert Inc
Organizational Unit (OU)	www.digicert.com

Period of Validity

Begins On	Wednesday 5 September 2018
Expires On	Saturday 23 November 2019

Fingerprints

SHA-256 Fingerprint	15:DD:39:C0:47:00:3D:CE:A0:F5:19:DD:99:E8:4C:7B:5B:8D:42:0A:FF:EB:CC:15:4B:24:FC:6B:06:17:73:EC
SHA1 Fingerprint	F7:59:41:D2:16:EC:C2:9F:7A:44:A6:AD:B7:F7:FF:55:8B:5A:F6:0D

Close

How SSL Works

- There are two types of certificates:

1. **Server Certificate:** Issued to trusted servers so client computers can connect to them using secure connections.

2. **Client Certificate:** Issued to trusted clients so server computers can confirm their identity.

How To Get a Certificate

- If you want to establish a secure connection with your clients, you must get a digital certificate from a Certification Authority (CA).
- Digital certificates aren’t free and the cost will depend on many factors such as the level of security.
  - You will need to decide on the level of strength (of encryption) you want your connection to support.

How To Get a Certificate

- In the early days of web programming, many web servers used certificates with 40-bit or 56-bit SSL strength.
  - At this strength its possible for a hackers to break the encryption code.
  - These strengths are appropriate for some sites.
- Today most browsers use 128-bit or higher SSL strength.
- Once you purchase a certificate you typically send it to your web host and they install it on your site.
- Once installed, clients can send data over a secure connection.

SSL Strengths

SSL strengths

Strength	Pros and Cons
40-bit	Most browsers support it, but it's relatively easy to crack the encryption code.
56-bit	It's thousands of times stronger than 40-bit strength and most browsers support it, but it's still possible to crack the encryption code.
128-bit	It's over a trillion times a trillion times stronger than 40-bit strength, which makes it extremely difficult to crack the encryption code, but it's more expensive.
256-bit	It's virtually impossible to crack the encryption code, but it's more expensive and not all browsers support it.

Configuring a Test Environment for SSL

- As already mentioned, to send information over a secure connection you need a digital certificate.
- Java allows you to create and install a self-signed digital certificate for free.
  - Since this certificate doesn’t come from a trusted source, it will cause a warning dialog to be displayed when you use it.
- The Java Secure Socket Extension (JSSE) is a collection of Java classes that let you use secure connections within your Java programs.
  - Without it, your applications won’t be able to connect to the server that transmits data over a secure connection.

Creating a self-signed certificate

- To create a self-signed certificate you need to create a *keystore* file.
- To do this you need to use the Java tool “keytool”, which can be run from the command window.

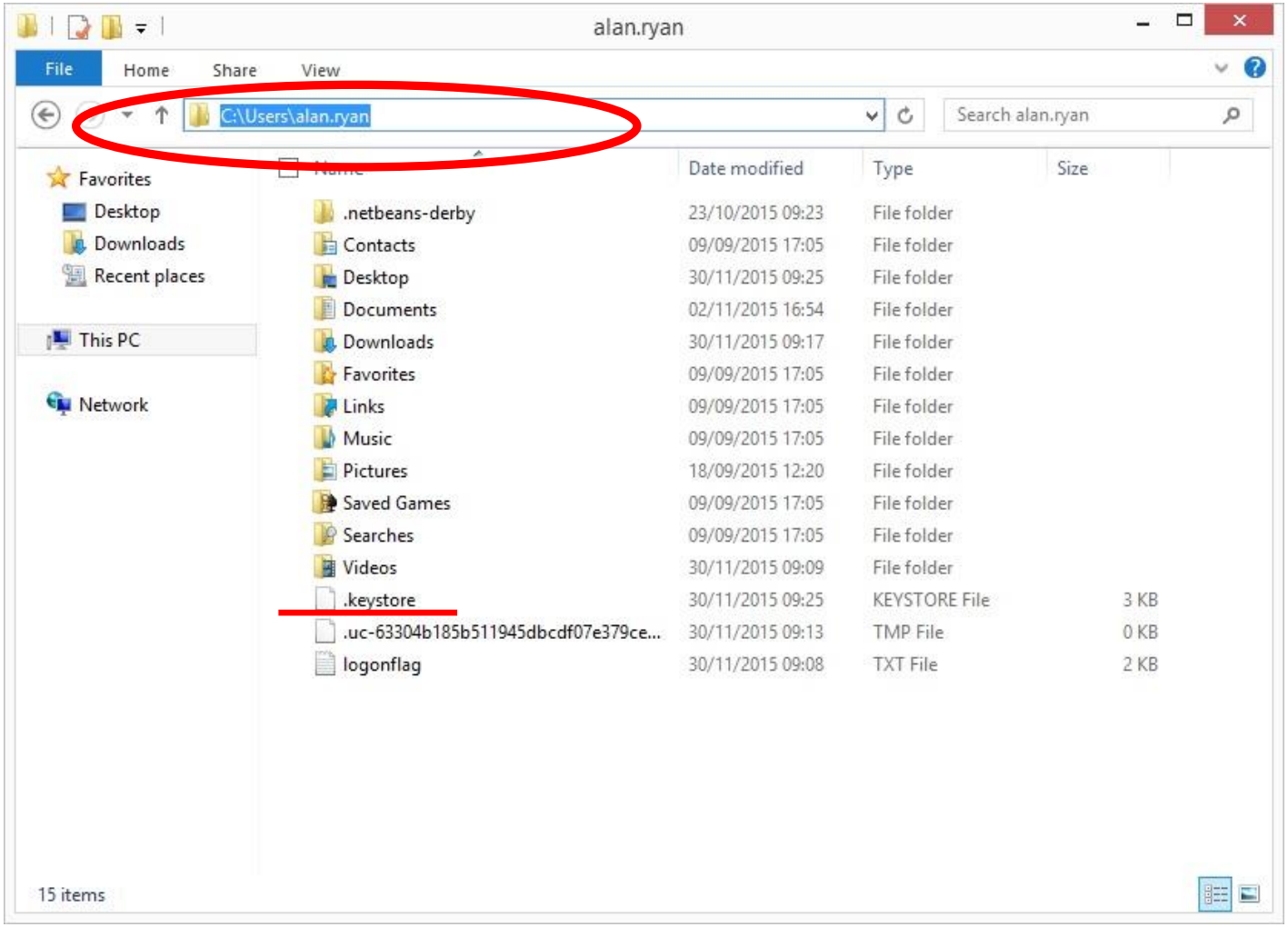
Command Prompt

```
C:\Program Files\Java\jdk1.8.0_11\bin>keytool -genKey -alias tomcat -keyalg RSA
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  Alan Ryan
What is the name of your organizational unit?
  [Unknown]:  LIT
What is the name of your organization?
  [Unknown]:  LIT
What is the name of your City or Locality?
  [Unknown]:  Limerick
What is the name of your State or Province?
  [Unknown]:  Munster
What is the two-letter country code for this unit?
  [Unknown]:  LK
Is CN=Alan Ryan, OU=LIT, O=LIT, L=Limerick, ST=Munster, C=LK correct?
  [no]:  y

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

C:\Program Files\Java\jdk1.8.0_11\bin>
```

Creating a self-signed certificate

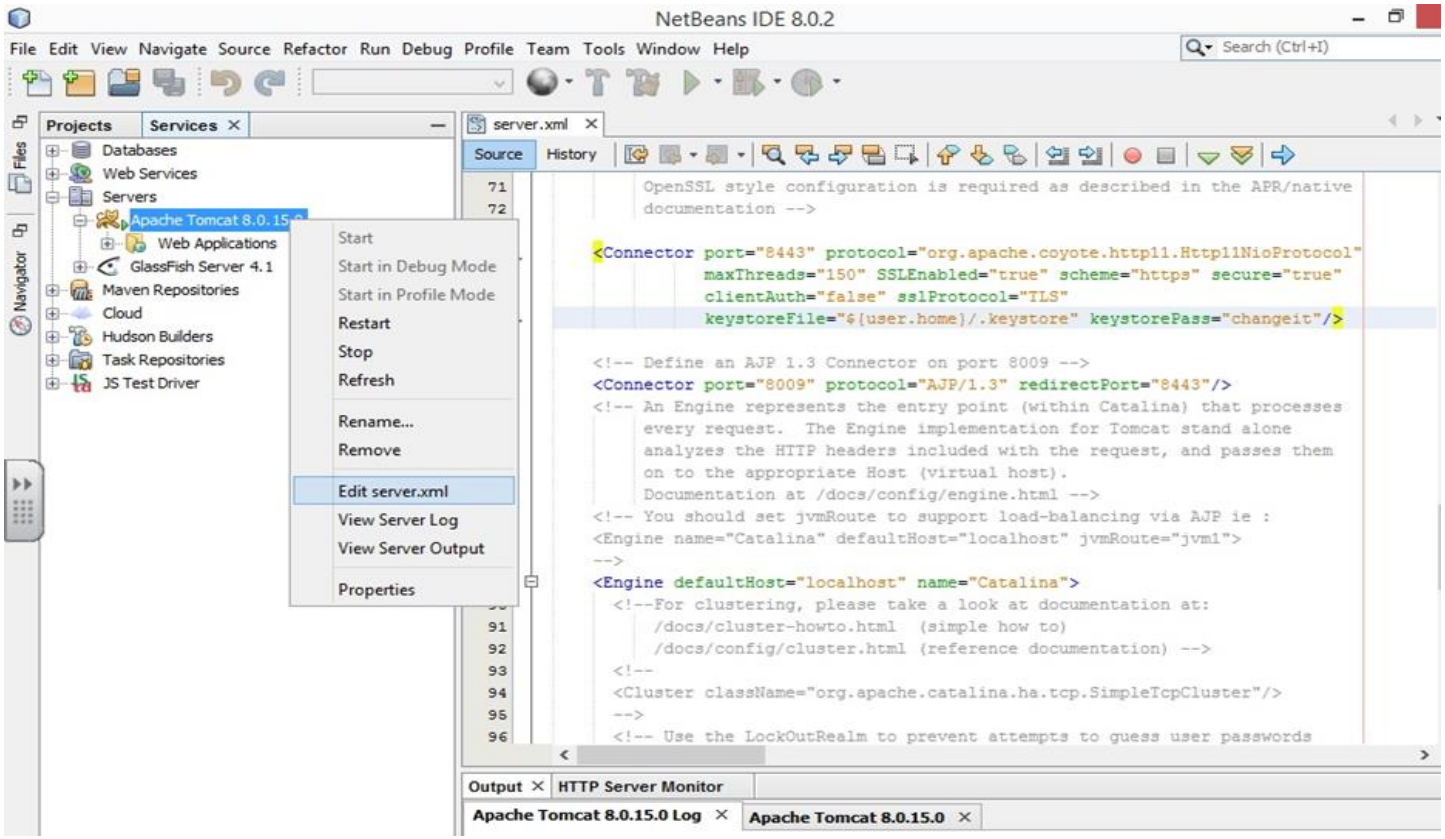
- Once the keytool program runs you are prompted to enter some information about yourself and your organisation.
- When prompted to enter a password be sure to enter "*changeit*".
- When prompted to enter a password for Tomcat press enter to use the same password as that used for the certificate.
- When this process is finished a keystore file with a *.keystore* extension is created in your home directory.

Creating a self-signed certificate



Creating a self-signed certificate

- Once the keystore file is created you need to edit the *server.xml* file in Tomcat.
- The quickest way to do this is through Netbeans.



Creating a self-signed certificate

- You then need to uncomment the following element in the *server.xml* file (which is stored in the *conf* folder of your Tomcat installation).

The Connector element for an SSL connection

```
<!-- Define a SSL HTTP/1.1 Connector on port 8443 -->
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
           SSLEnabled="true" maxThreads="150" scheme="https" secure="true"
           clientAuth="false" sslProtocol="TLS"
           keystoreFile="${user.home}/.keystore" keystorePass="changeit"/>
```

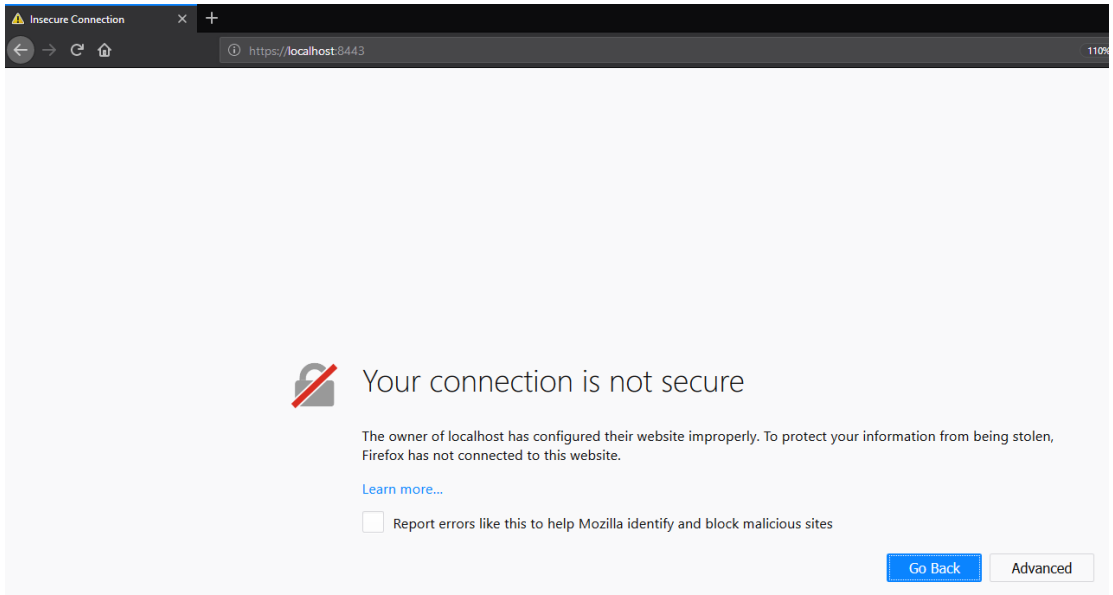
- You then need to add, the two highlighted attributes.

Testing a local SSL Certificate

- Once configured, restart (or start) Tomcat and launch your browser and try enter the following URL:

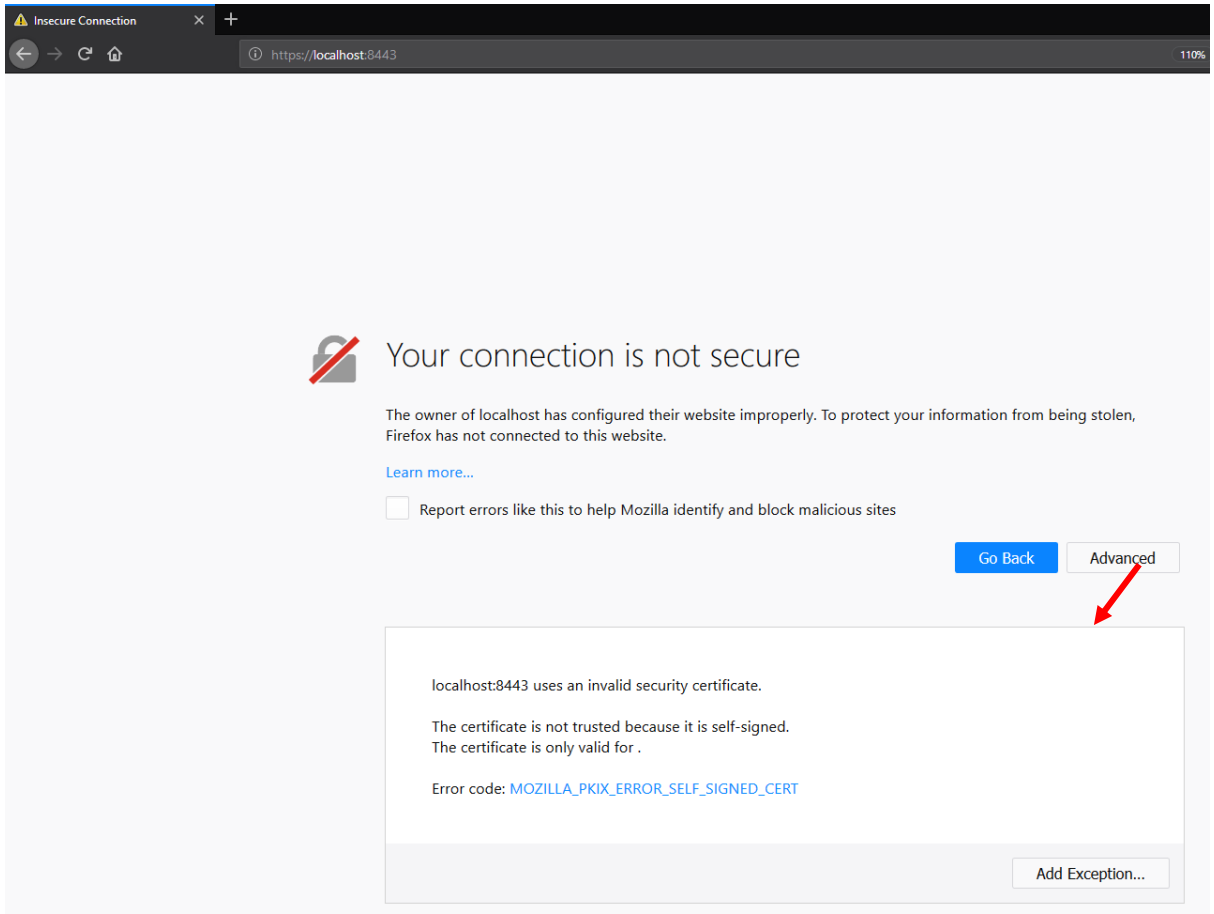
  https://localhost:8443/

- In Firefox this will bring up the following page initially.

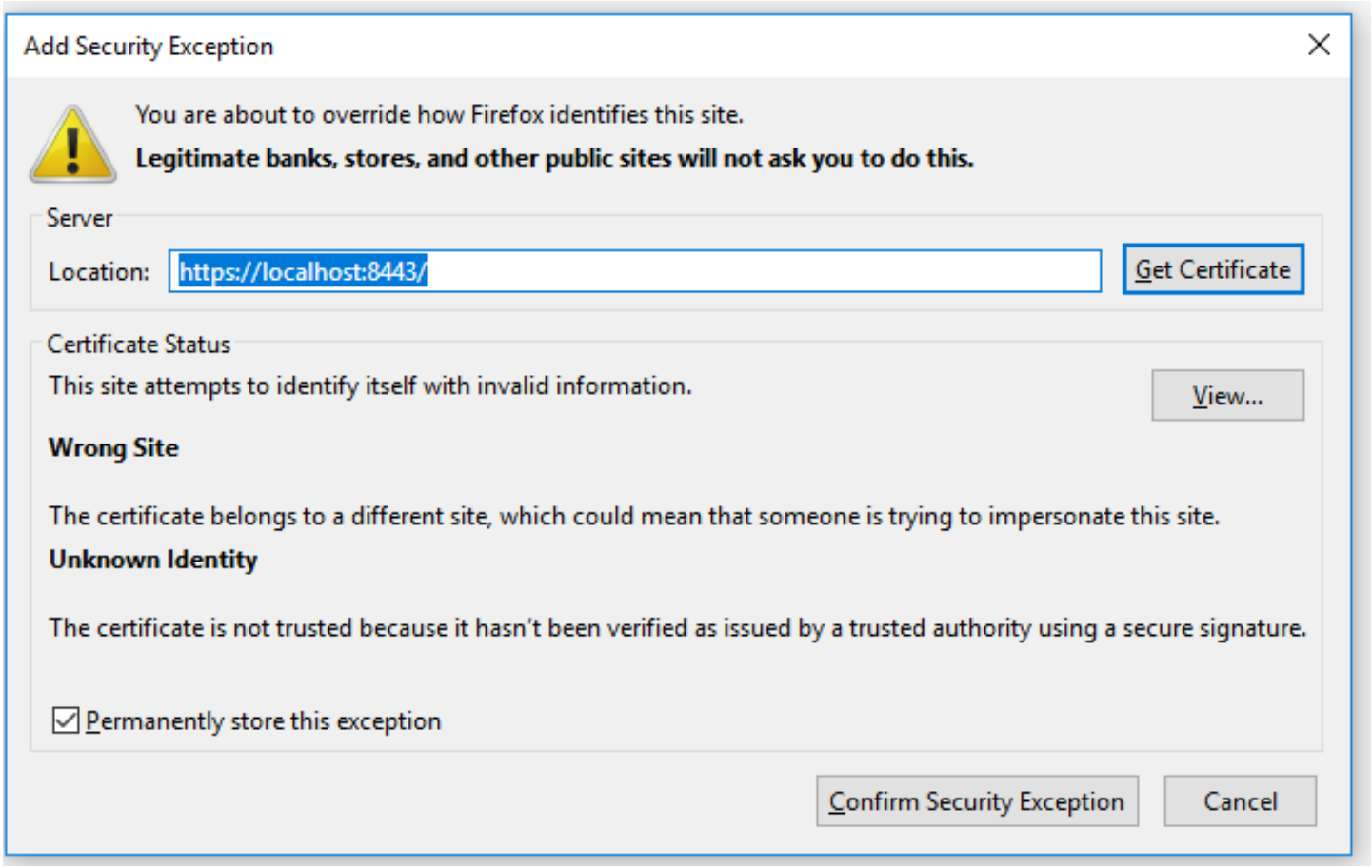


Testing a local SSL Certificate

- You will then be asked to add an exception for your unsigned certificate.

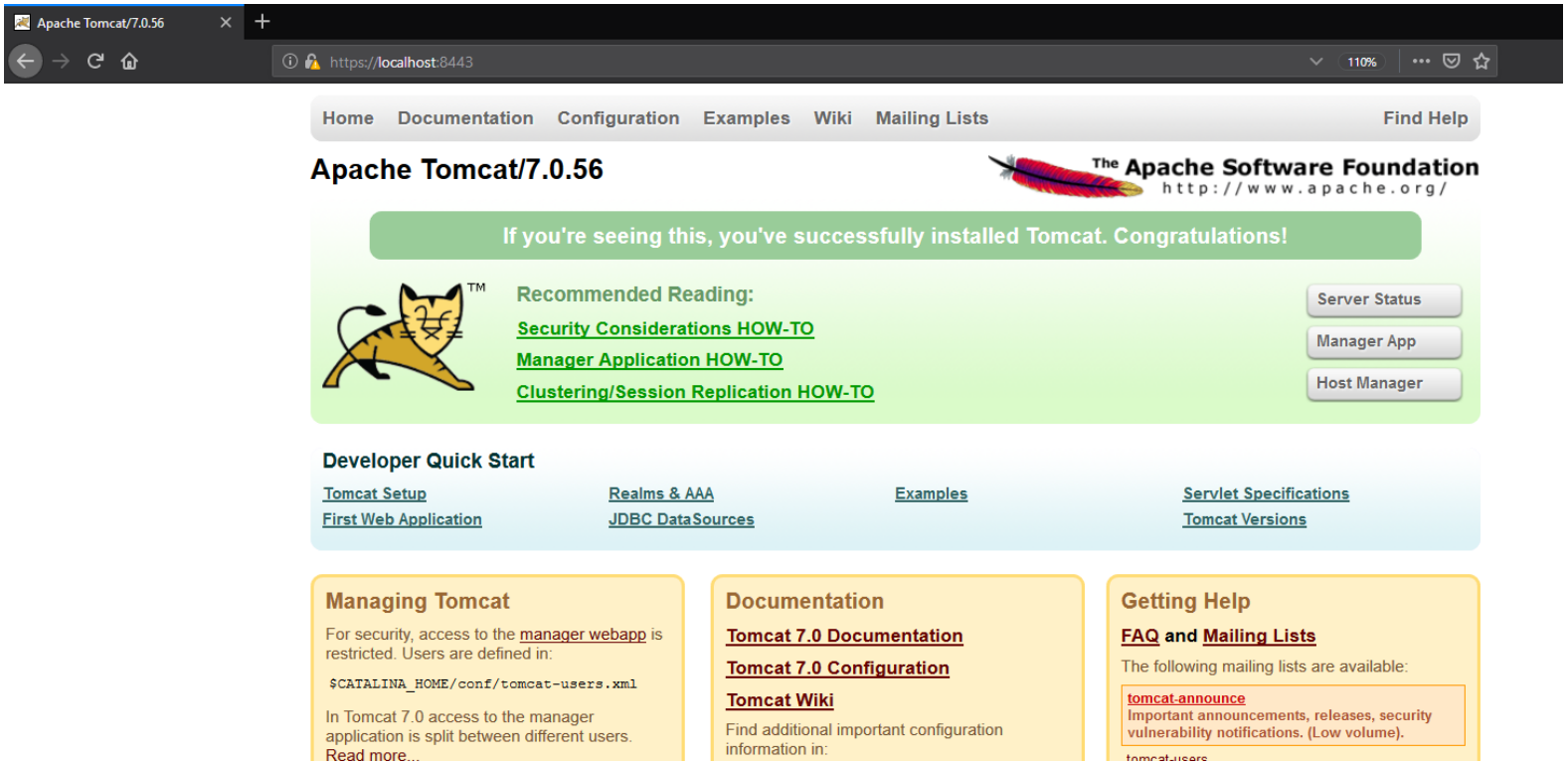


Testing a local SSL Certificate



Testing a local SSL Certificate

- Click on the "I understand the risks" option will bring up the test page for tomcat (using SSL).

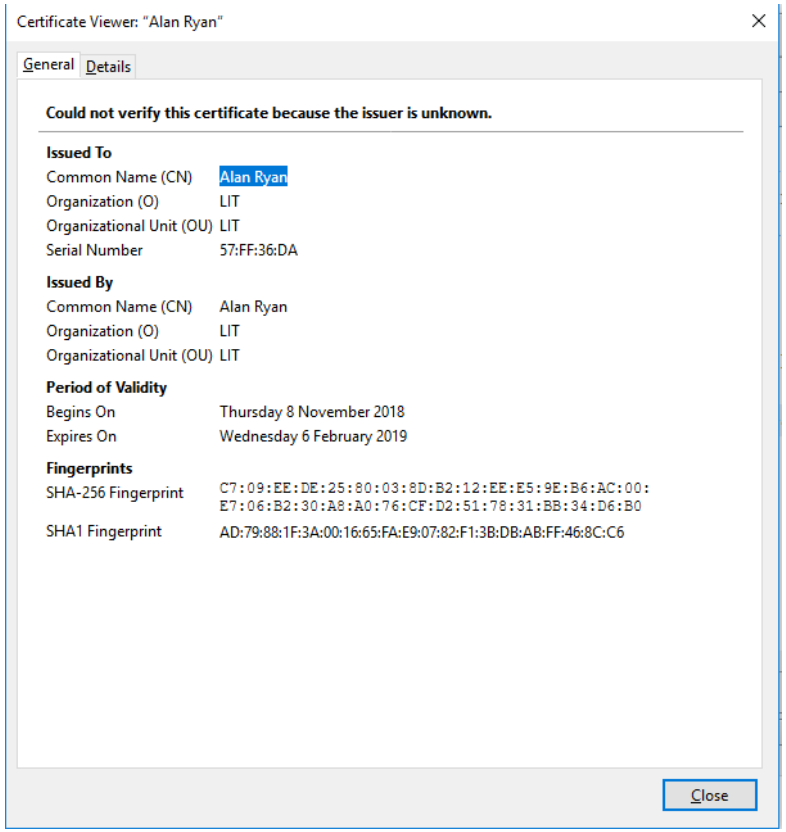


Testing a local SSL Certificate

- When a secure connection is requested (using a URL that starts with HTTPS) the server authenticates itself by sending its secure certificate to the browser.
- Then, if the certificate doesn’t come from a certification authority that’s registered with the browser the browser displays a warning.
  - A self-signed certificate doesn’t come from a trusted source.

Testing a local SSL Certificate

- Your self signed certificate will look like the following.



Common Problems When Configuring the Local SSL Connection

Problem 1

Problem: Tomcat can't find the keystore file. When you start Tomcat, it throws a java.io.FileNotFoundException.

Solution: Make sure the .keystore file is located in your home directory, which varies from system to system. For Windows, the home directory is C:\Users*user.name*.

Problem 2

Problem: The keystore password and key passwords that you used to create the keystore file don't match. When you start Tomcat, it displays a java.io.FileNotFoundException that says, "keystore was tampered with" or "password was incorrect."

Solution: Delete the old keystore file and create a new keystore file.

Where can you get a “proper” certificate?

www.symantec.com/ssl-sem-page
www.godaddy.com/ssl
www.globalsign.com
www.startcom.org
https://letsencrypt.org/ (free and backed by Mozilla)

References

Murach, J., (2014) *Murachs Java Servlets JSP*, 3rd edn. Mike Murach and Associates, Inc.

http://docs.oracle.com/javaee/6/tutorial/doc/

http://tomcat.apache.org/tomcat-8.0-doc/