Configuring Tomcat to support SSL and https

This exercise will guide you through the steps of setting up Tomcat to support SSL and https. See todays slides, and the following videos for a quick and easy to understand introduction Encryption and Certificates.

- Symmetric Key and Public Key Encryption (6-7 min): https://www.youtube.com/watch?v=ERp8420ucGs
- What are Certificates (15 min) : https://www.youtube.com/watch?v=LRMBZhdFjDI

We will add more details next week.

Generating a keystore

Open a command prompt and navigate to $JAVA_HOME/bin ($JAVA_HOME is the location of your Java-JDK[1])

Type: keytool -genkey -alias [youralias] -keyalg RSA -keystore [path]

Replace youralias and path with values of your own choice. On my laptop I did:

keytool -genkey -alias lam -keyalg RSA -keystore C:\Users\lam\keystore\lam-keystore

Make sure the directory given in path exists.

- When prompted for a password, provide one (and write it down)
- When prompted for your *first* and *last* name, provide the information or just press return
- When prompted for the name of "your organizational unit", type *ss-prog* or just press return
- When prompted for the name of "your organization?" type *cphbusiness*
- Just press return for *City, State, Country-code*
- Finally type "*yes*" to accept the values
- Press return to use the same pasword for your alias as for the keystore (the one you entered above)

This will create a self-signed certificate in the folder C:\users\lam\keystore with the name lam-keystore (for the example given above).

View the Certificate Details

You can list the certificate details using keytool as sketched below (replace path with your own):

keytool -list -keystore C:\Users\lam\keystore\lam-keystore

---

1 Probably C:\Program Files\Java\jdk1.8.???? on a Windows Computer

Configuring Tomcat to use SSL

Now that we have a functional keystore populated with a valid certificate, it's time to configure Tomcat to use SSL.

For that we need to configure Tomcat's SSL connectors, which are configured in Tomcat's configuration file *server.xml*.

The server.xml is located in the Servers folder in the project explorer view:

Locate the place in the file with content like this:

```
<!-- Define a SSL HTTP/1.1 Connector on port 8443
   This connector uses the NIO implementation that requires the JSSE
```

Now add a connector element as outlined below

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
        maxThreads="150" scheme="https" secure="true"
        clientAuth="false" sslProtocol="TLS"
        keystoreFile="C:\Users\lam\keystore\lam-keystore"
        keystorePass="testtest" />
```

Replace the value for *keystoreFile* with your own path and the value for *password* with the one you used when the keystore was generated.

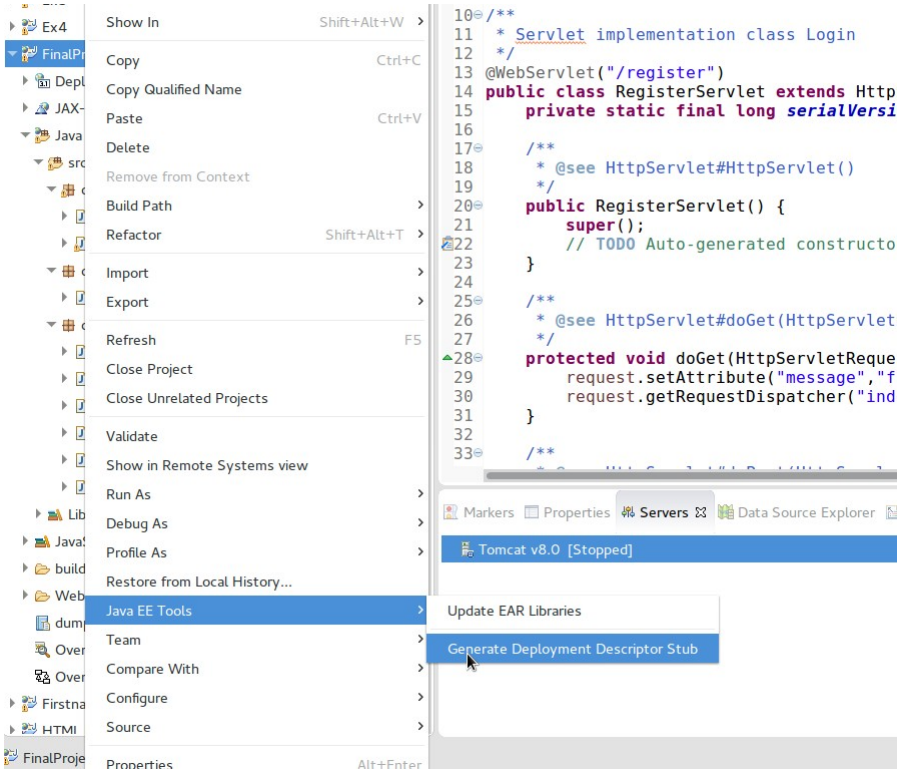
Configuring a Web-Project to use SSL

Now it is time to test whether our self-issued certificate is installed the right way in order for Tomcat to provide SSL connections.

1. Create a new dynamic-web-project. For this exercise we only need the default index.xml file
2. Add a new Deployment Descriptor file (web.xml) (look at the screenshot)
3. Open web.xml

4. Add the following in side the <web-app> side of the element:

```
<security-constraint>
```



```
    <web-resource-collection>
        <web-resource-name>Secure everything</web-resource-name>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <web-resource-collection>
        <web-resource-name>Another secured page</web-resource-name>
        <url-pattern>/login</url-pattern>
    </web-resource-collection>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

5. Stop and start your Tomcat
6. Verify that you can (you might have to do several clicks to convince your browser to access the site) create a secure https connection: