Acme Madrugá – Https Configuration

What is https?

Hyper Text Transfer Protocol Secure (HTTPS) is the secure version of HTTP, the protocol over which data is sent between your browser and the website that you are connected to. The 'S' at the end of HTTPS stands for 'Secure'. It means all communications between your browser and the website are encrypted.

Creating the SSL certificate

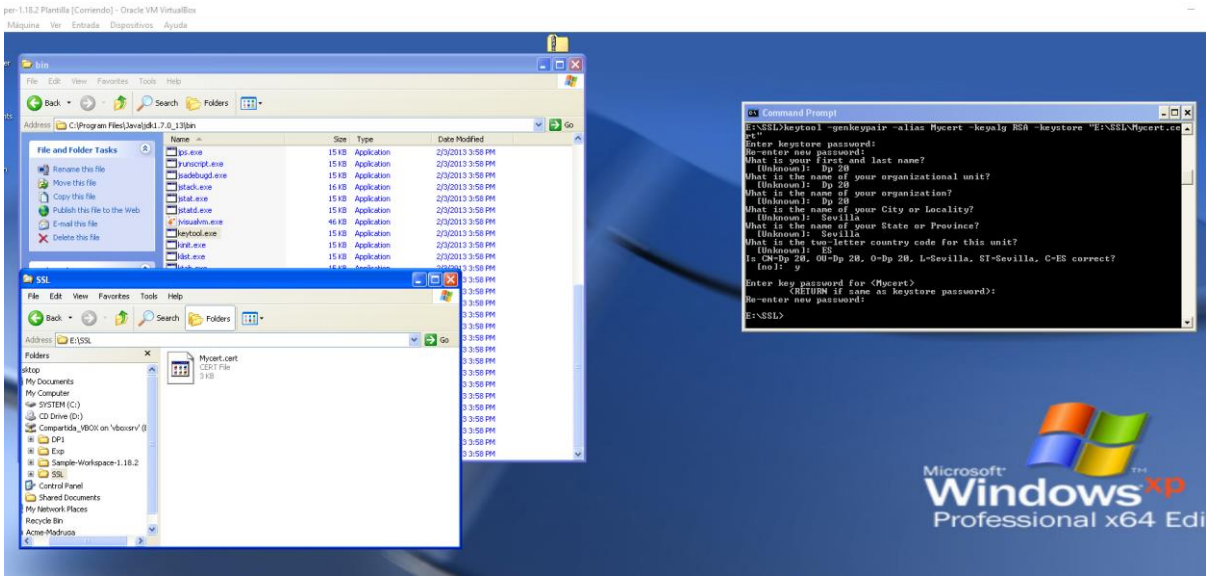
In first place we need to create a SSL certificate for our website, SSL Certificates are small data files that digitally bind a cryptographic key to an organization's details. When installed on a web server, it activates the padlock and the https protocol and allows secure connections from a web server to a browser. Typically, SSL is used to secure credit card transactions, data transfer and logins.

To create our certificate we are going to open a command prompt in the location where we are going to stablish the certificate and execute the next command:

keytool –genkeypair –alias NameOfTheCertificate –keyalg RSA –keystore "TheLocation\NmeOfTheCertificate.cert"

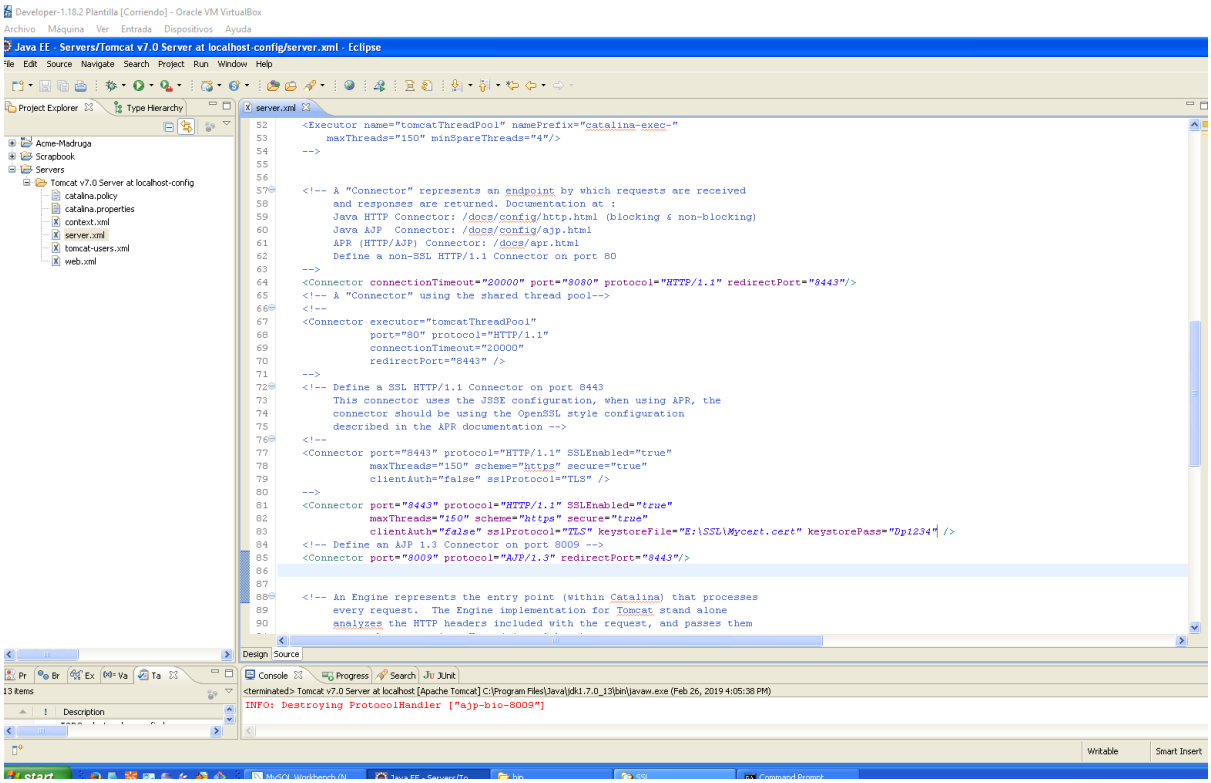
It will ask for a password, in our case we decided to put Dp1234, we will use this password in next steps configuring the server.

Make sure you have the keytool.exe in your java folder.



Configuring Tomcat

Configuring Tomcat is really easy, go to your server folder an open server.xml



We are going to define a new Connector tag with the next information:

```
<Connector SSLEnabled="true" clientAuth="false" keystoreFile="E:\SSL\Mycert.cert"
keystorePass="Dp1234" maxThreads="150" port="8443" protocol="HTTP/1.1"
scheme="https" secure="true" sslProtocol="TLS"/>
```

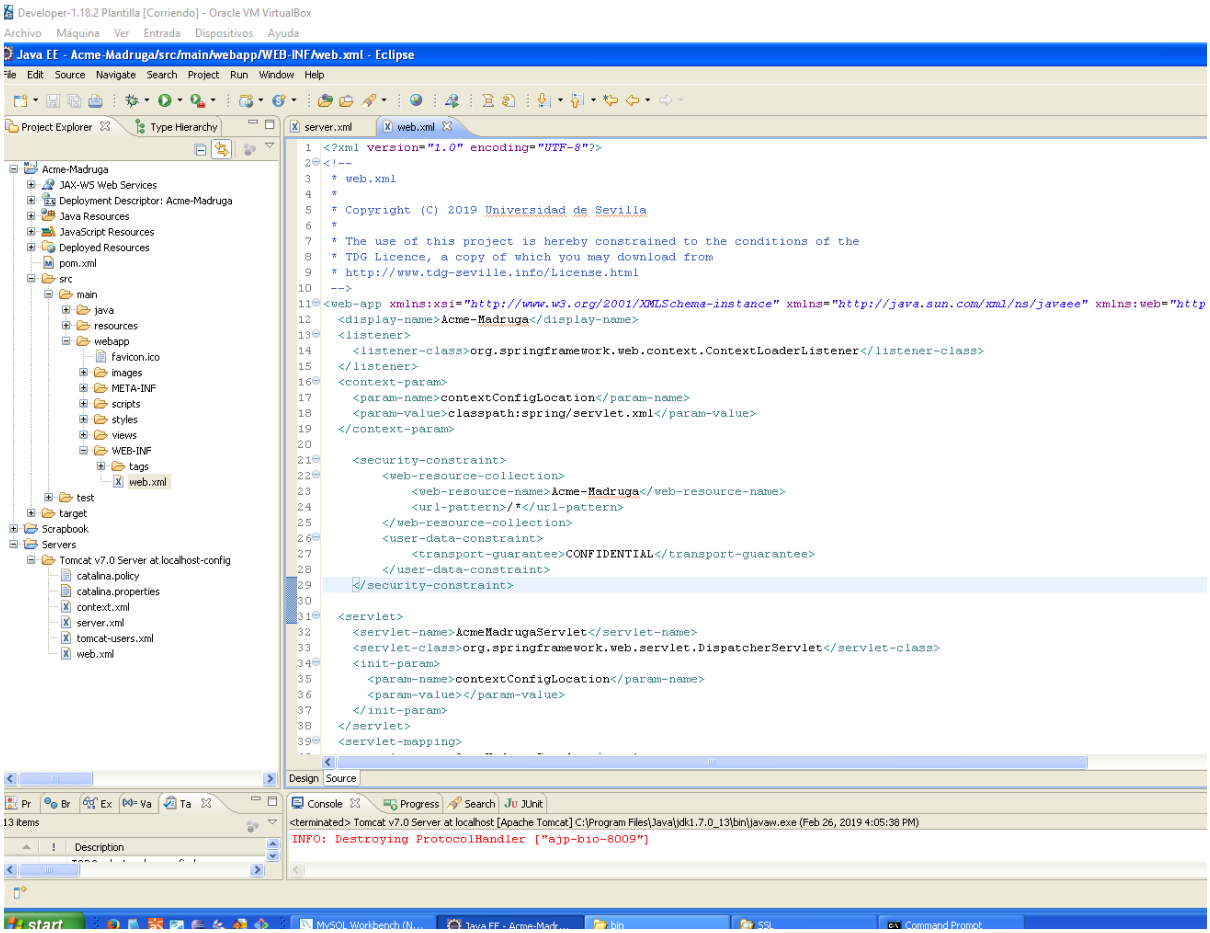
This tag already exist in the file but is commented, you just need to add two new params: keystoreFile and keystorePass, in keystoreFile put your certificate location, in keystorePass put the password you used to create the certificate.

Configuring our project

To configure our project go to the webapp folder and locate the web.xml file.

We are going to create a new security tag, in this tag we will indicate the name of our project (Acme-Madruga), the url pattern (put “/*” to use https in all our website, not just in the login process), and the transport guarantee constraint “Confidential” to tell the server that the use of SSL is required.

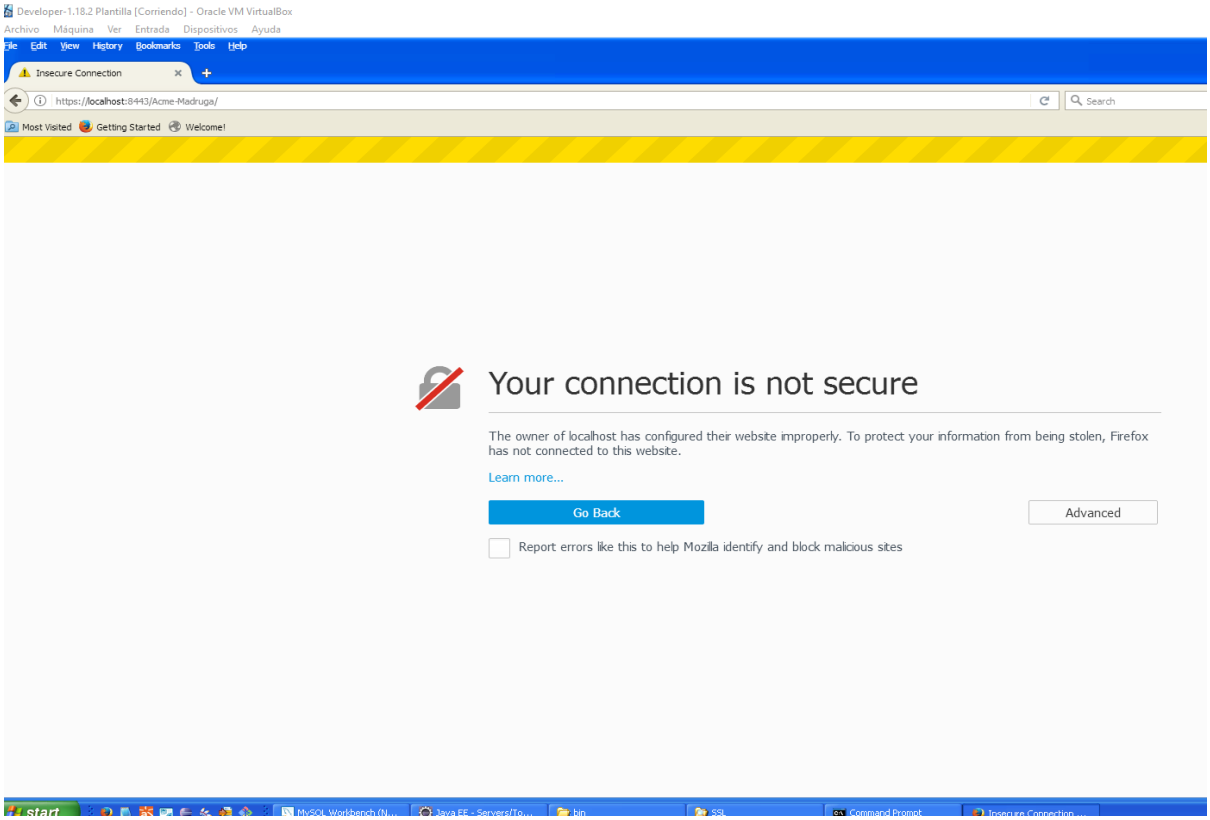
```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Acme-Madruga</web-resource-name>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```



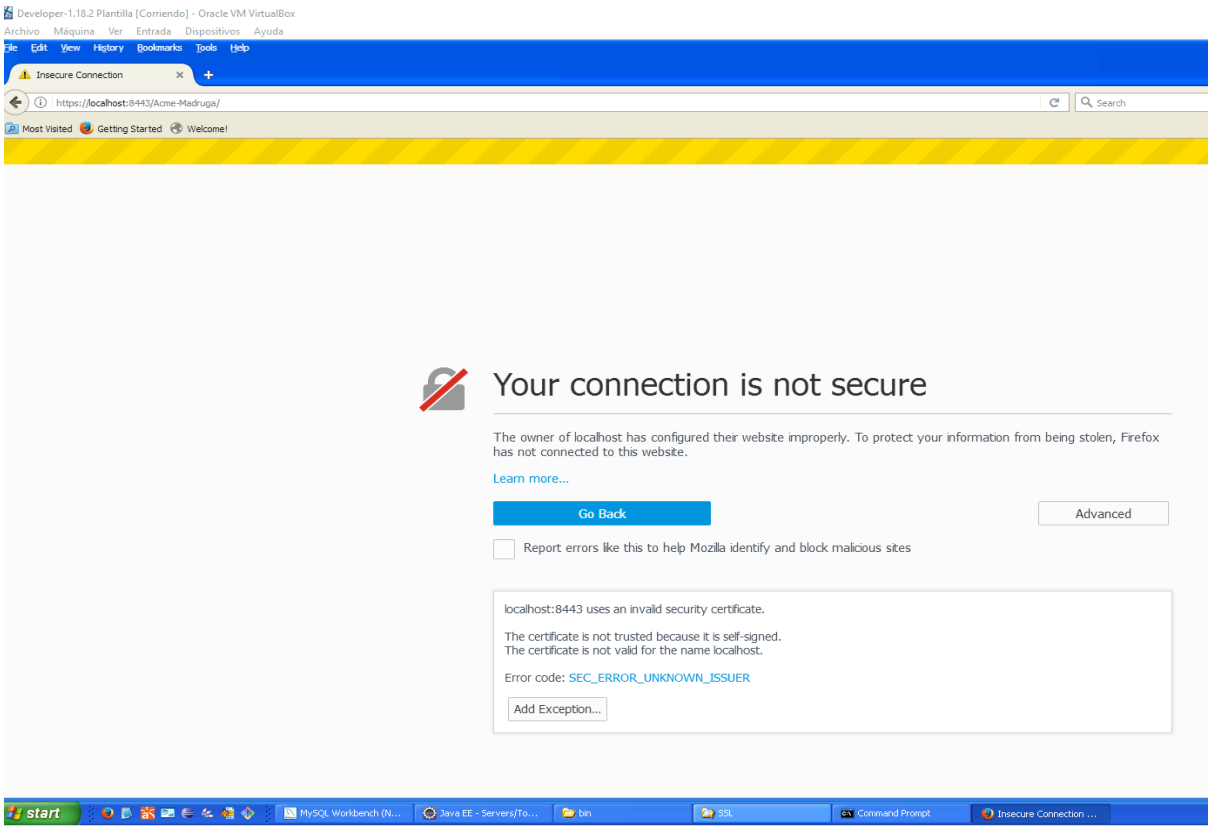
That was the last step, start the server and open your browser.

Checking the results

Now that our project is ready, go to http://localhost:8080/Acme-Madruga/ and it will appear something like this:



If we see this page, we succeeded configuring https for our application, but since we created our own certificate, the browser will warn us that is probably an insecure page.



Add the exception to the browser and confirm the security exception.