Configure Tomcat to Use SSL

> This page is as quick notes and needs to be rewritten as an article.

Put the .bin file in the webapps directory of Tomcat. This avoids the need to provide an absolute path. For examples with our example version of Tomcat we would be using,

```
/opt/apache/tomcat.0/webapps
```

Here we would be making some changes to the server.xml file inside tomcat to tell it to use the keystore which was created in the earlier step for configuring SSL. Open the file server.xml which can be found as:

<CATALINA_HOME>/conf/server.xml

Now you have to modify it. Find the Connector element which has port=”8443 and uncomment it if already not done. Add the following lines to the Connector directive,

```
keystoreFile="./webapps/mywebservices.bin"
keystorePass="Password123"
```

The resulting directive should look like this,

```
<Connector port="8443
maxThreads="150 minSpareThreads="25 maxSpareThreads="75
enableLookups="true" disableUploadTimeout="true"
acceptCount="100 debug="0 scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS"
keystoreFile="./webapps/mywebservices.bin"
keystorePass="Password123" />
```

You can start Tomcat and check the SSL works by using a browser and going to the url, https://\[Server-IP\]:8443/

Tomcat still runs in normal mode too, i.e on port **8080** with **http**. Depending on your configuration this may not be desireable for your application. If so, you will want to continue to the next step to configure your web application to only work on 8443.

Configuring Web Application to Only Accept SSL Connections

Take an application which has already been deployed **successfully** in Tomcat and first access it through http and https to see if it works fine. If yes, then open the **web.xml** of that application and just add the following XML fragment **before** web-app ends i.e **</web-app>**

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

Explanation of the fragment is beyond the scope of this tutorial but all you should notice is that the **/*** indicates that now, any resource in your application can be accessed only with **https** be it Servlets or JSP's. The term **CONFIDENTIAL** is the term which tells the server to make the application work on SSL. If you want to turn the SSL mode for this application off then just turn don't delete the fragment. Just put the value as **NONE** instead of **CONFIDENTIAL**.

Resources

http://techtracer.com/2007/09/12/setting-up-ssl-on-tomcat-in-3-easy-steps/ - also provides steps