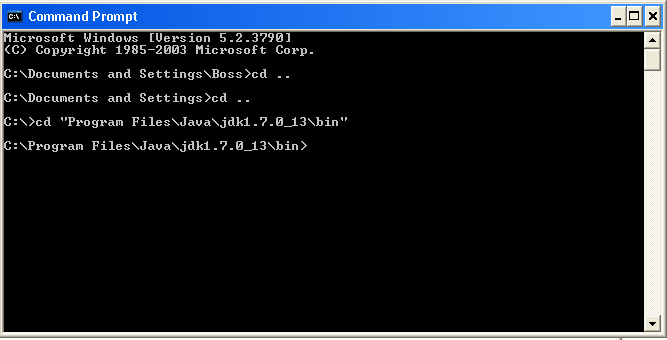
# A+ - D08

En el presente entregable se nos solicita establecer comunicaciones seguras usando el protocolo HTTPS para cumplir con la Ley de Protección de Datos. Para ello debemos de generar un certificado y cambiar la configuración del servidor Tomcat.

Comenzaremos explicando como generaremos el certificado. Una Keystore es un “almacén de claves” donde podemos almacenar claves privadas, certificados y claves simétricas. En nuestro caso, la Keystore es un archivo, pero el almacenamiento tambien puede manejarse de diferentes maneras, como por ejemplo usando un token criptográfico o utilizando el mecanismo propio del sistema operativo.

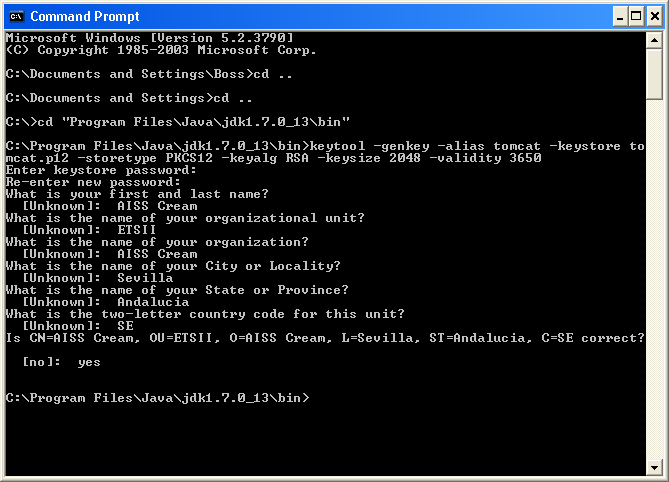
Abrimos como administrador (boss/$I=B0$$=U$3=P@$$) la consola de comandos o CMD del sistema de pre-producción provisto. Nos dirijimos a la unidad del disco duro y luego a la ruta Program Files\Java\jdk1.7.0\_13\bin.



A continuación, creamos el certificado en dicha ruta. Utilizaremos el comando keytool -genkey -alias tomcat -keystore tomcat.p12 -storetype PKCS12 -keyalg RSA -keysize 2048 -validity 3650. En cuanto a los argumentos del comando:

* keytool es una utilidad de gestión de claves y certificados.
* -genkey es el comando para generar nuestros propios certificados públicos/privados.
* -alias asigna un alias (pseudónimo) a la clave.
* -keystore asigna el nombre y formato al certificado generado.
* -storetype asigna el servicio de criptografía usado, puede ser JKS por defecto o PKCS12 (.p12) que usaremos ya que es más conveniente para el uso de certificados con claves privadas.
* -keyalg define el esquema de cifrado usado, en nuestro caso RSA,otros pueden ser DSA o EC.
* -keysize especifica el tamaño de cada clave que se generará, si usamos RSA será de 2048.
* -validity es el periodo de validez del certificado que vamos a generar.

Tras esto, nos pedirá que una contraseña y su confirmación, que declaramos como “changeit”, que es la contraseña por defecto que se utiliza. Nos aparecerán unas preguntas (nuestro nombre, nombre de la compañía, nombre de nuestra ciudad…), cuyas respuestas se añadirán al certificado. Por último confirmamos que los datos son correctos con “yes”.



El certificado se habrá creado en la ruta especificada, ahora prodecemos a cambiar la configuración de Tomcat. Nos dirigimos a la ruta C:\Program Files\Apache Software Foundation\Tomcat 7.0\conf y abrimos el archivo server.xml.

Comentamos la línea:

<Connector connectionTimeout="20000" port="80" protocol="HTTP/1.1" redirectPort="8443"/> Descomentamos la línea:

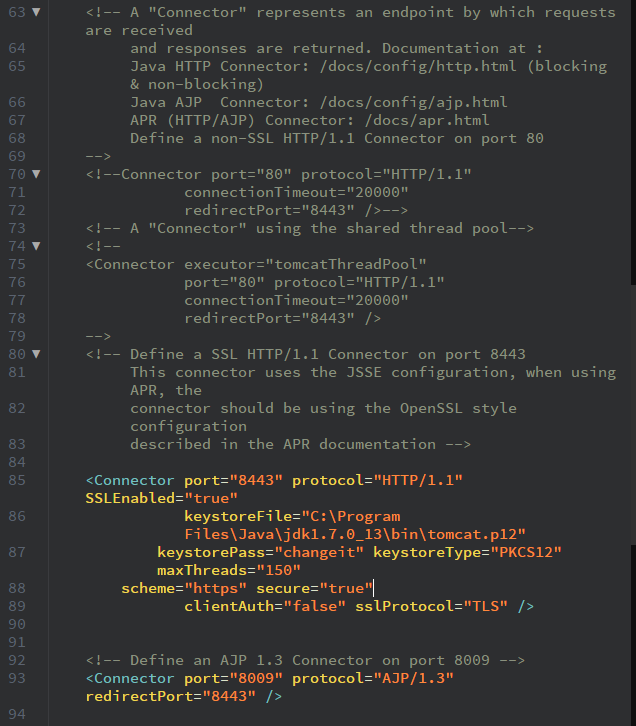
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true" maxThreads="150" scheme="https" secure="true" clientAuth="false" sslProtocol="TLS" />

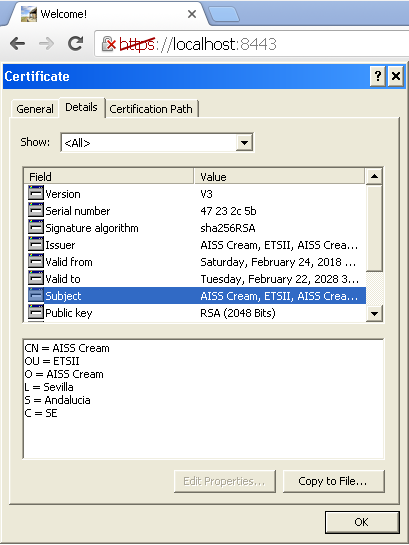
A la línea que acabamos de descomentar hemos de añadirle: keystoreFile="C:\Program Files\Java\jdk1.7.0\_13\bin\tomcat.p12" keystorePass="changeit" keystoreType="PKCS12"

Para especificar la ruta, la contraseña y el formato del certificado le añadimos:

* keystoreFile=”…”: para especificar la ruta en la que se halla el certificado.
* keystorePass=”…”: para detallar la contraseña del certificado.
* keystoreType=”…”: para expresar el servicio de criptografía usado.

Tras los cambios, el archivo queda:





Hemos añadido una versión sin cambios de Acme Rendezvous, ya que los cambios realizados han sido sobre la configuración del servidor Tomcat (se podría añadir en el archivo src/main/resources/application.properties, aunque hemos decidido que no era la mejor opción)y con estos, cualquier artefacto lanzado desde el servidor utilizaría el protocolo HTTPS en el puerto 8843.

Una vez realizados los cambios en el archivo server.xml, abrimos el navegador y lanzamos nuestra aplicación en el puerto 8443 (https://localhost:8843), donde tendremos implementado las comunicaciones seguras a traves del protocolo HTTPS. Podemos comprobar que el servidor esta usando el certificado que hemos creado previamente, al cual se le ha asignado las respuestas anteriores.