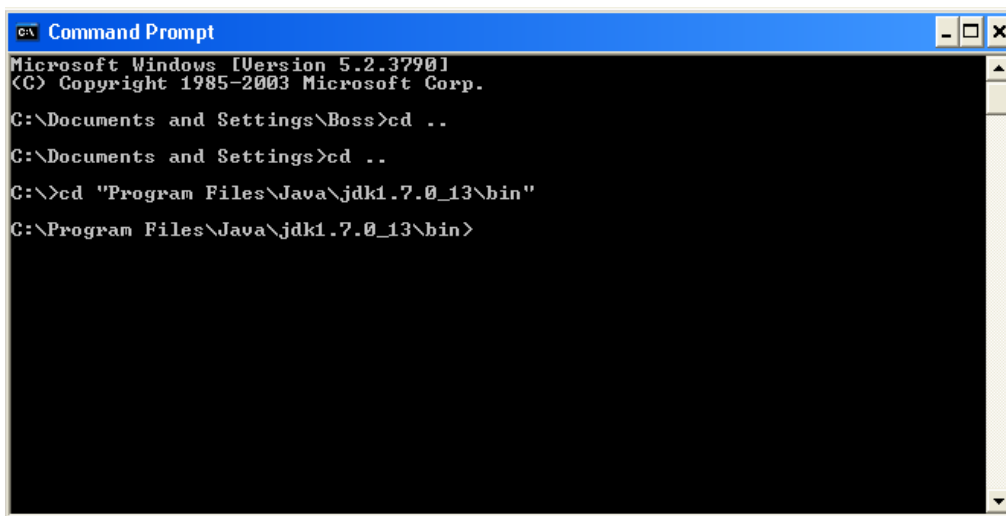


# A+ - D08

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 dirigimos a la unidad del disco duro y luego a la ruta `Program Files\Java\jdk1.7.0_13\bin`.



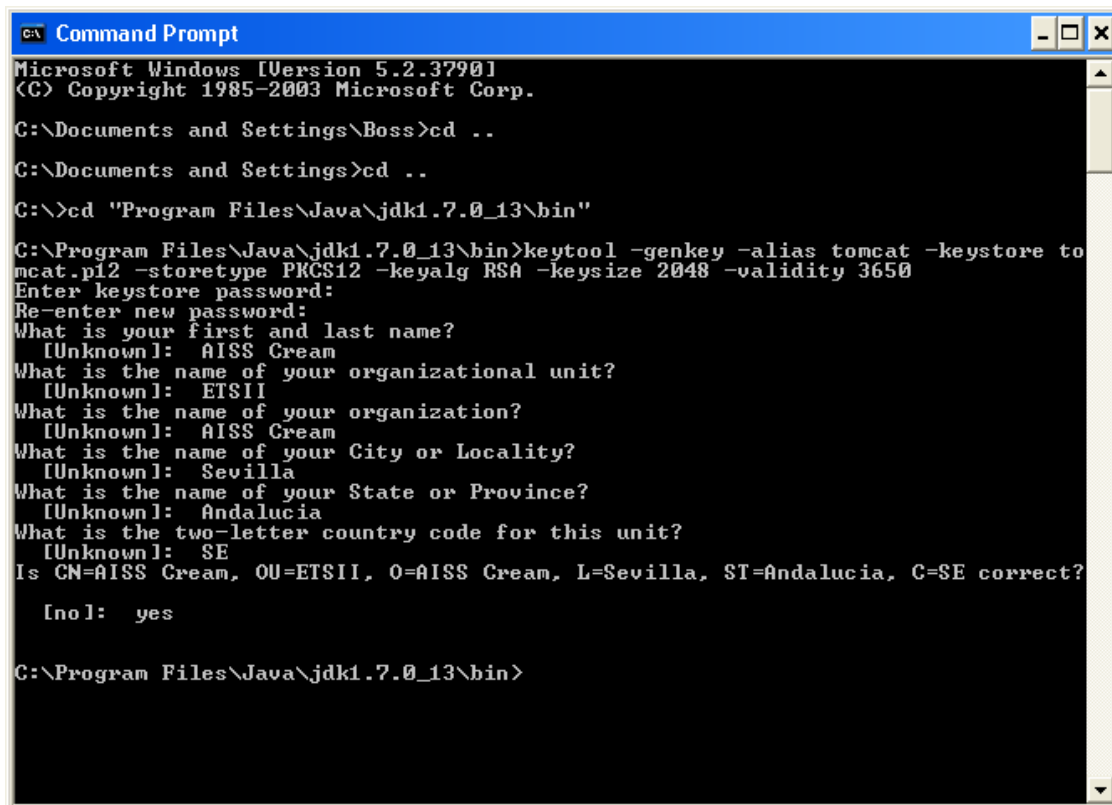
```
C:\ Command Prompt
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\Documents and Settings\Boss>cd ..
C:\Documents and Settings>cd ..
C:\>cd "Program Files\Java\jdk1.7.0_13\bin"
C:\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".



```
C:\> Command Prompt
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\Documents and Settings\Boss>cd ..
C:\Documents and Settings>cd ..
C:\>cd "Program Files\Java\jdk1.7.0_13\bin"
C:\Program Files\Java\jdk1.7.0_13\bin>keytool -genkey -alias tomcat -keystore to
mcat.p12 -storetype PKCS12 -keyalg RSA -keysize 2048 -validity 3650
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]: AISS Cream
What is the name of your organizational unit?
[Unknown]: ETSII
What is the name of your organization?
[Unknown]: AISS Cream
What is the name of your City or Locality?
[Unknown]: Sevilla
What is the name of your State or Province?
[Unknown]: Andalucia
What is the two-letter country code for this unit?
[Unknown]: SE
Is CN=AISS Cream, OU=ETSII, O=AISS Cream, L=Sevilla, ST=Andalucia, C=SE correct?
[no]: yes

C:\Program Files\Java\jdk1.7.0_13\bin>
```

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`.

Para redirigir el tráfico al puerto 8443, añadimos la línea:

```
<Connector connectionTimeout="20000" port="8080" 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.

Luego, añadimos al final del archivo web.xml (que se encuentra en la misma ruta), con el fin de que todo el tráfico hacia nuestra aplicación Acme-Chollos-Rifas vaya hacia el puerto 8443:

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

```
<!-- A "Connector" represents an endpoint by which requests
and responses are returned. Documentation at :
Java HTTP Connector: /docs/config/http.html (blocking
& non-blocking)
Java AJP Connector: /docs/config/ajp.html
APR (HTTP/AJP) Connector: /docs/apr.html
Define a non-SSL HTTP/1.1 Connector on port 80
-->

<Connector port="80" protocol="HTTP/1.1"
connectionTimeout="20000"
redirectPort="8443" />

<Connector port="8080" protocol="HTTP/1.1"
connectionTimeout="20000"
redirectPort="8443" />

<!-- A "Connector" using the shared thread pool-->

<!--<Connector executor="tomcatThreadPool"
port="80" protocol="HTTP/1.1"
connectionTimeout="20000"
redirectPort="8443" />-->

<!-- Define a SSL HTTP/1.1 Connector on port 8443
This connector uses the JSSE configuration, when using
APR, the
connector should be using the OpenSSL style
configuration
described in the APR documentation -->

<Connector port="8443" protocol="HTTP/1.1"
SSLEnabled="true"
keyStoreFile="C:\Program
Files\Java\jdk1.7.0_13\bin\tomcat.p12"
keyStorePass="changeit" keyStoreType="PKCS12"
maxThreads="150"
scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS" />

<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector port="8009" protocol="AJP/1.3"
redirectPort="8443" />
```

server.xml

```
<!-- customize) or returns a 404 status,
depending on the value of the -->
<!-- listings setting.
-->

<!--
-->

<!-- If you define welcome files in your
own application's web.xml
-->
<!-- deployment descriptor, that list
*replaces* the list configured -->
<!-- here, so be sure to include any of the
default values that you wish -->
<!-- to use within your application.
-->

<welcome-file-list>
<welcome-file>index.html</welcome-
file>
<welcome-file>index.htm</welcome-
file>
<welcome-file>index.jsp</welcome-
file>
</welcome-file-list>

<security-constraint>
<web-resource-collection>
<web-resource-name>Acme-
Rendezvous</web-resource-name>
<url-pattern>/*</url-pattern>
</web-resource-collection>

<user-data-constraint>
<transport-
guarantee>CONFIDENTIAL</transport-
guarantee>
</user-data-constraint>

</security-constraint>

</web-app>
```

web.xml

Tras los cambios, los archivos quedan como en la imagen. Estos cambios también se podrían hacer en el entorno de desarrollo, el certificado se generaría de igual manera y los cambios en la configuración del servidor serían en la ruta Acme-Chollos-Rifas\Servers\Tomcat v7.0 Server at localhost-config\server.xml.

Hemos añadido una versión sin cambios de Acme Chollos-Rifas, ya que los cambios realizados han sido sobre la configuración del servidor Tomcat (los mismos cambios se podrían añadir en el archivo Acme-Chollos-Rifas\src\main\resources\application.properties y en el Acme-Chollos-Rifas\src\main/

resources/spring/config/security.xml, sería una opción para en entorno de desarrollo esto supondría hacer los cambios para cada proyecto, por lo que hemos decidido que no es 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 los archivos server.xml y web.xml, abrimos el navegador y lanzamos nuestra aplicación en el puerto 8443 (<https://localhost:8443>), donde tendremos implementado las comunicaciones seguras a través del protocolo HTTPS. Podemos comprobar que el servidor está usando el certificado que hemos creado previamente (es normal que al entrar en la aplicación se nos aparezca un mensaje del tipo "Tu conexión no es privada", ya que estamos accediendo a través de un certificado autogenerado), al cual se le ha asignado las respuestas anteriores que hemos facilitado al crear el certificado.

