

Un protocolo SLL (Secure Socket Layer) permite seguridad entre las comunicaciones cliente y servidor, implementando una encriptación de los datos y certificado de autenticación. Normalmente el término SSL es referido a “TSL” (Transport Layer ouSecurity) que esta basado en las especificación original de SSL.

En nuestro proyecto usamos Apache Tomcat como servidor, el cual permite la implementación del protocolo SSL.

Para su configuración primero deberemos empezar creando una Keystore.

Las claves que Tomcat utilizará para las conexiones SSL, serán almacenadas en un archivo protegido por contraseña llamado “Keystore”. Existen dos formas de crear este archivo, una importando la clave existente al keystore o creando una clave desde cero.

Por simplicidad hemos creado el Keystore con una herramienta llamada keytool incluido en el JDK 1.4 o superior, para acceder a esta keytool, pulsamos start y run (O directamente Ctrl + R), en la ventana emergente escribimos cmd lo cual nos llevará a la interfaz de comandos de windows.

Escribimos en ella, cd “%JAVA_HOME%\bin” y nos llevará a la carpeta java donde se encuentra nuestro JDK instalado y dentro de la misma a la carpeta bin.

En ella podremos encontrar nuestro keytool para generar el keystore con la clave privada del servidor y el certificado auto-firmado. Escribimos:

```
C:\Program Files\Java\jdk1.7.0_13\bin>keytool -genkey -alias tomcat -keyalg RSA
```

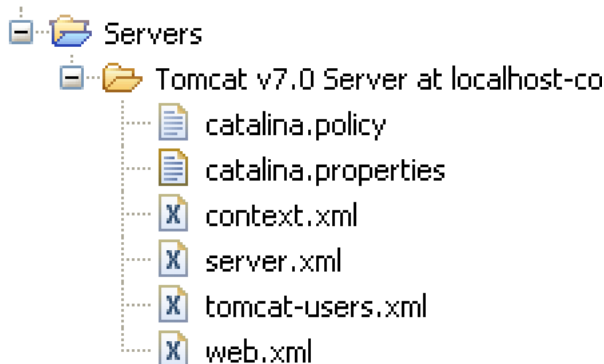
Nos pedirá que introduzcamos la siguiente información:

```
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]: Group 23 D&T
What is the name of your organizational unit?
[Unknown]: University of Seville
What is the name of your organization?
[Unknown]: Group 23 D&T
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]: ES
Is CN=Group 23 D&T, OU=University of Seville, O=Group 23 D&T, L=Sevilla, ST=Anda
lucia, C=ES correct?
[no]: yes
Enter key password for <tomcat>
(RETURN if same as keystore password):
```

El archivo creado se encontrará en "C:\Documents and Settings\Student" encontraremos en ella un archivo .keystore

El segundo paso será configurar Tomcat para que use HTTPS y el archivo ".keystore".

Para ello vamos a la carpeta servers asociada a nuestro proyecto y nos vamos al archivo server.xml (Estos mismos archivos se encuentran en la carpeta de configuración de Apache Tomcat v7)



Añadimos la siguiente línea de código

```
<Connector SSLEnabled="true" clientAuth="false"
keystoreFile="C:\Documents and Settings\Student\.keystore"
keystorePass="password" maxThreads="150" port="8443" scheme="https"
secure="true" sslProtocol="TLS"/>
```

Donde keystoreFile es la ruta en nuestro equipo del archivo .keystore (estará adjunto).

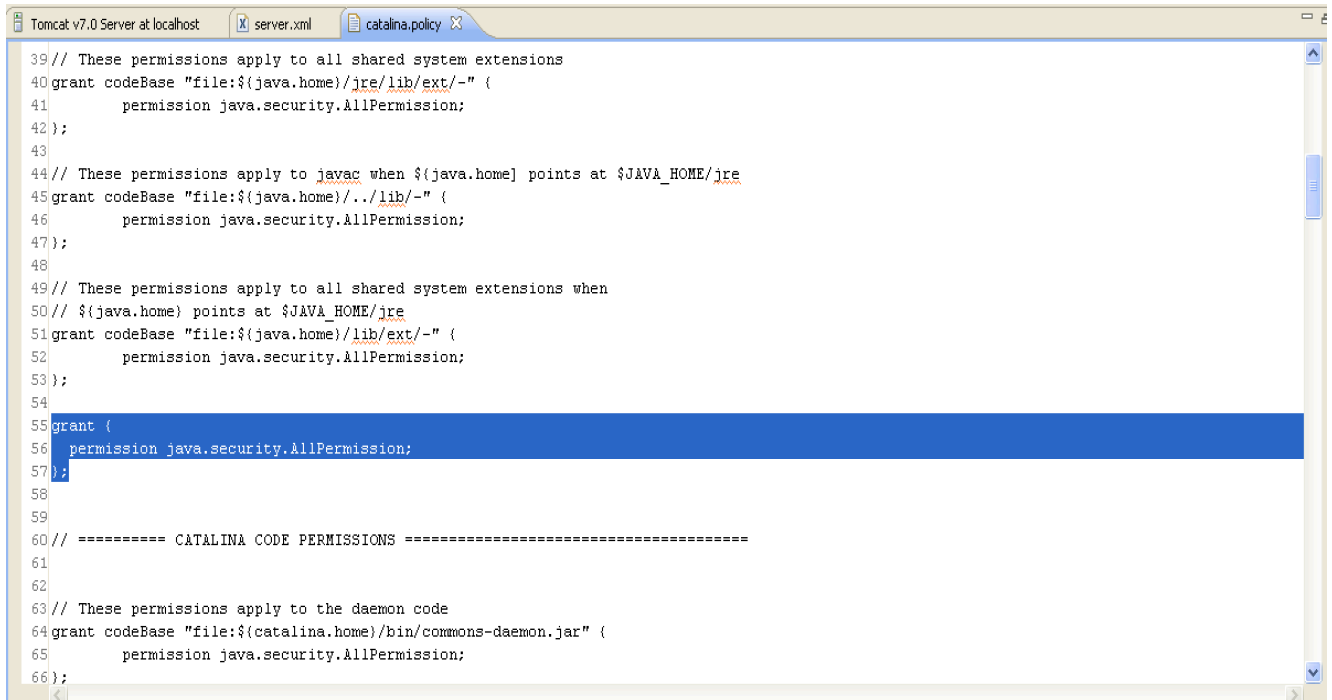
Si lanzamos Tomcat con la siguiente configuración nos devuelve un error al no poder iniciarse.

```
java.security.AccessControlException: access denied ("java.lang.RuntimePermission" "accessClassInPackage.org.apache.catalin
```

Como referencia al error encontramos la siguiente ayuda en stackoverflow (<https://stackoverflow.com/questions/2645298/how-to-sanely-configure-security-policy-in-tomcat-6>)

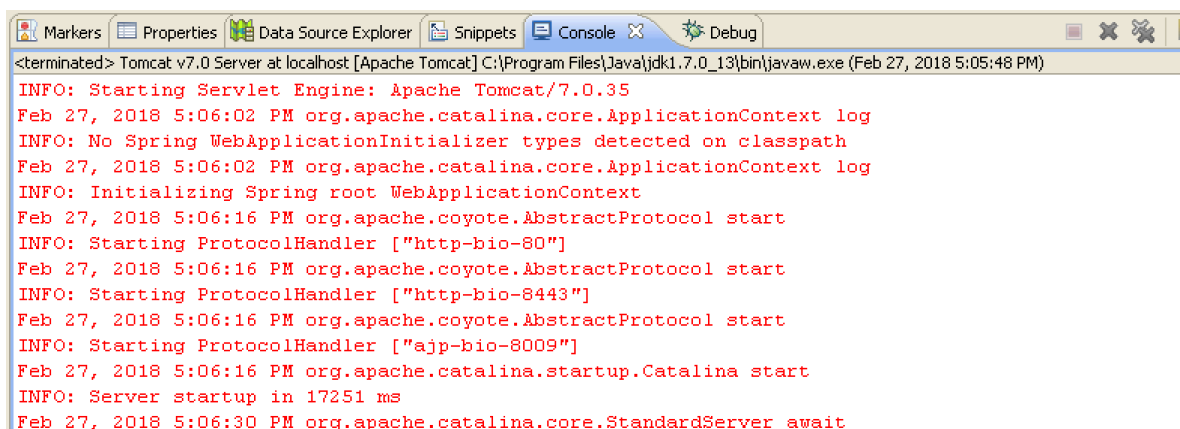
En el cual se menciona un *workaround* para dar permisos, y junto con otras respuestas pensamos en añadir esos permisos al catalina.policy y permitió lanzar la aplicación.

```
“grant {  
    permission java.security.AllPermission;  
};”
```

A screenshot of an IDE window showing the 'catalina.policy' file. The file contains Java security policy rules. Lines 39-43 define permissions for the JRE. Lines 44-47 define permissions for javac. Lines 49-53 define permissions for all shared system extensions. Lines 55-57 show a 'grant' block with 'permission java.security.AllPermission;'. Lines 60-62 are comments for CATALINA CODE PERMISSIONS. Lines 63-66 define permissions for the daemon code.

```
39// These permissions apply to all shared system extensions  
40grant codeBase "file:${java.home}/jre/lib/ext/-" {  
41    permission java.security.AllPermission;  
42};  
43  
44// These permissions apply to javac when ${java.home} points at $JAVA_HOME/jre  
45grant codeBase "file:${java.home}/../lib/-" {  
46    permission java.security.AllPermission;  
47};  
48  
49// These permissions apply to all shared system extensions when  
50// ${java.home} points at $JAVA_HOME/jre  
51grant codeBase "file:${java.home}/lib/ext/-" {  
52    permission java.security.AllPermission;  
53};  
54  
55grant (  
56    permission java.security.AllPermission;  
57);  
58  
59  
60// ===== CATALINA CODE PERMISSIONS =====  
61  
62  
63// These permissions apply to the daemon code  
64grant codeBase "file:${catalina.home}/bin/commons-daemon.jar" {  
65    permission java.security.AllPermission;  
66};
```

Una vez realizado esto el proyecto es lanzado sin ningún problema, funcionando tanto en HTTP como en HTTPS.

A screenshot of the IDE console window showing the startup logs of Apache Tomcat. The logs indicate that the Servlet Engine is starting, the Spring root WebApplicationContext is initializing, and the server is starting successfully. The console also shows the time taken for the server to start up (17251 ms).

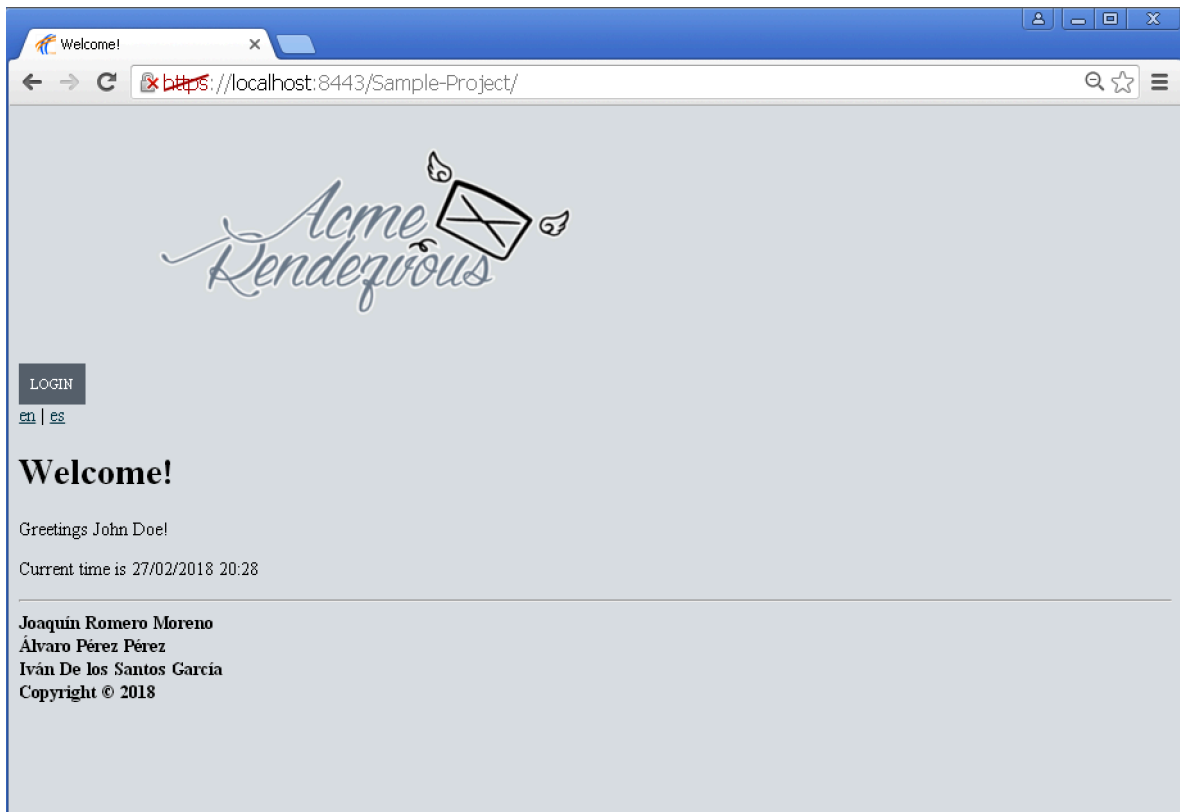
```
<terminated> Tomcat v7.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Feb 27, 2018 5:05:48 PM)  
INFO: Starting Servlet Engine: Apache Tomcat/7.0.35  
Feb 27, 2018 5:06:02 PM org.apache.catalina.core.ApplicationContext log  
INFO: No Spring WebApplicationContext types detected on classpath  
Feb 27, 2018 5:06:02 PM org.apache.catalina.core.ApplicationContext log  
INFO: Initializing Spring root WebApplicationContext  
Feb 27, 2018 5:06:16 PM org.apache.coyote.AbstractProtocol start  
INFO: Starting ProtocolHandler ["http-bio-80"]  
Feb 27, 2018 5:06:16 PM org.apache.coyote.AbstractProtocol start  
INFO: Starting ProtocolHandler ["http-bio-8443"]  
Feb 27, 2018 5:06:16 PM org.apache.coyote.AbstractProtocol start  
INFO: Starting ProtocolHandler ["ajp-bio-8009"]  
Feb 27, 2018 5:06:16 PM org.apache.catalina.startup.Catalina start  
INFO: Server startup in 17251 ms  
Feb 27, 2018 5:06:30 PM org.apache.catalina.core.StandardServer await
```

Para configurar la web con HTTPS, como referencia hemos usado la siguiente página (<https://dzone.com/articles/setting-ssl-tomcat-5-minutes>)

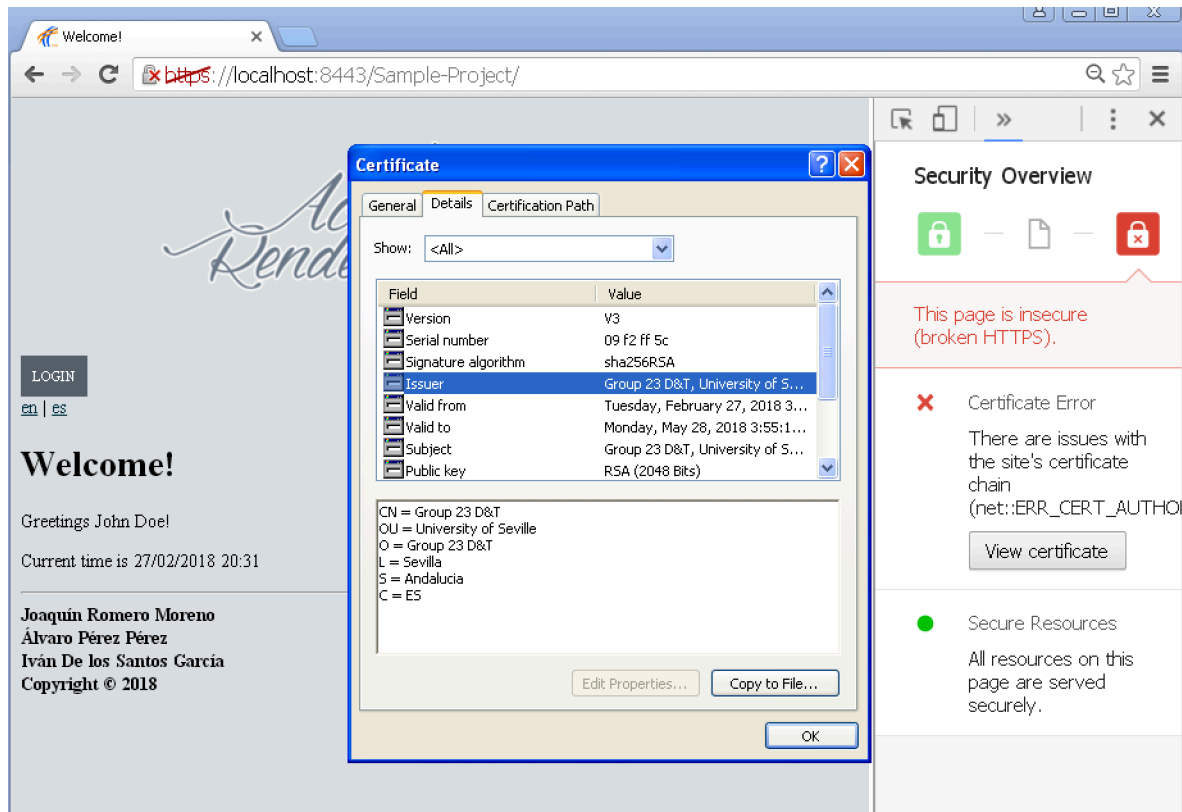
Para ello hemos configurado el web.xml añadiendo el siguiente código, para permitir lo anteriormente hablado.

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

Aquí se puede ver como todos los recursos de la web (/*) son servicios a través de un transporte confidencial.



Mostrando el certificado



Lo próximo que quedaría por explorar, fuera del alcance de este entregable, serían las distintas implementaciones del protocolo SSL en Tomcat (APR con OpenSSL o JSSE), y el uso de un certificado por parte de una web competente y autenticado, Certificate Authority (verisign.com, thawte.com, trustcenter.de) o usando un Online Certificate Status Protocol (OCSP), con el uso de OpenSSL.