SSL Certificate:

SSL, or Secure Socket Layer, is a technology which allows web browsers and web servers to communicate over a secured connection. This means that the data being sent is encrypted by one side, transmitted, then decrypted by the other side before processing. This is a two-way process, meaning that both the server AND the browser encrypt all traffic before sending out data.

Another important aspect of the SSL protocol is Authentication. This means that during your initial attempt to communicate with a web server over a secure connection, that server will present your web browser with a set of credentials, in the form of a "Certificate", as proof the site is who and what it claims to be. In certain cases, the server may also request a Certificate from your web browser, asking for proof that *you* are who you claim to be.

This is known as "Client Authentication," although in practice this is used more for business-to-business (B2B) transactions than with individual users. Most SSL-enabled web servers do not request Client Authentication.

1. Generate Keystore

First, uses "keytool" command to create a self-signed certificate. During the keystore creation process, you need to assign a password and fill in the certificate's detail.

```
$Tomcat\bin>keytool -genkey -alias srinivas -keyalg RSA -keystore c:\cgkeystore
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  srinivas
What is the name of your organizational unit?
  //omitted to save space
  [no]:  yes

Enter key password for <srinivas>
        (RETURN if same as keystore password):
Re-enter new password:
```

```
$Tomcat\bin>
Copy
```

Here, you just created a certificate named "**cgkeystore**", which locate at "**c:**".

Certificate Details

You can use same "keytool" command to list the existing certificate's detail

```
$Tomcat\bin>keytool -list -keystore c:\cgkeystore
Enter keystore password:

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

mkyong, 14 Disember 2010, PrivateKeyEntry,
Certificate fingerprint (MD5): C8:DD:A1:AF:9F:55:A0:7F:6E:98:10:DE:8C:63:1B:A5

$Tomcat\bin>
Copy
```

2. **Connector in server.xml**

Next, locate your Tomcat's server configuration file at *$Tomcat\conf\server.xml*, modify it by adding a **connector** element to support for SSL or https connection.

File : $Tomcat\conf\server.xml

```
//...
<!-- 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"
           maxThreads="150" scheme="https" secure="true"
           clientAuth="false" sslProtocol="TLS"
```

```
                keystoreFile="c:\cgkeystore"
                keystorePass="password" />
 //...
```
Copy

Note
keystorePass="password" is the password you assigned to your keystore via "keytool" command.

3)**Open Chrome,hit the below URl.**

https://localhost:8443/

Enforce HTTPS

This is only applicable when you've SSL enabled. If not, it will break the application.

Once you've enabled SSL, it would be good to force redirect all HTTP requests to HTTPS for secure communication between user to Tomcat application server.

- Go to $tomcat/conf folder
- Modify `web.xml` by using `vi`
- Add following before `</web-app>` syntax

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

```
</user-data-constraint>
</security-constraint>
```

- Save the file and restart the Tomcat

Valves:

Acts like interceptor.

The RemoteAddrValve valve lets you selectively allow or block requests on the basis of their source IP address. It support two attributes – allow and block.

<Valve className="org.apache.catalina.valves.RemoteAddrValve" block="192\.168.*"/>

The RemoteHostValve valve operates like remote address filter but on client host names instead of client IP addresses.

<Valve className="org.apache.catalina.valves.RemoteHostValve" deny=".*badweb\.com"/>

The RequestDumperValve logs details of the incoming requests and therefore is useful for debugging purposes.

<Valve className="org.apache.catalina.valves.RequestDumperValve"/>

The single sign on valve, when included in a Host container, has the effect of requiring only one authentication for all the applications of that host. Without this valve, the user would have to enter his ID and password before using each separate application.

<Valve className="org.apache.catalina.valves.SingleSignOn"/>