Adv Java Day3

17 July 2020 09:10

1. JSP compilation

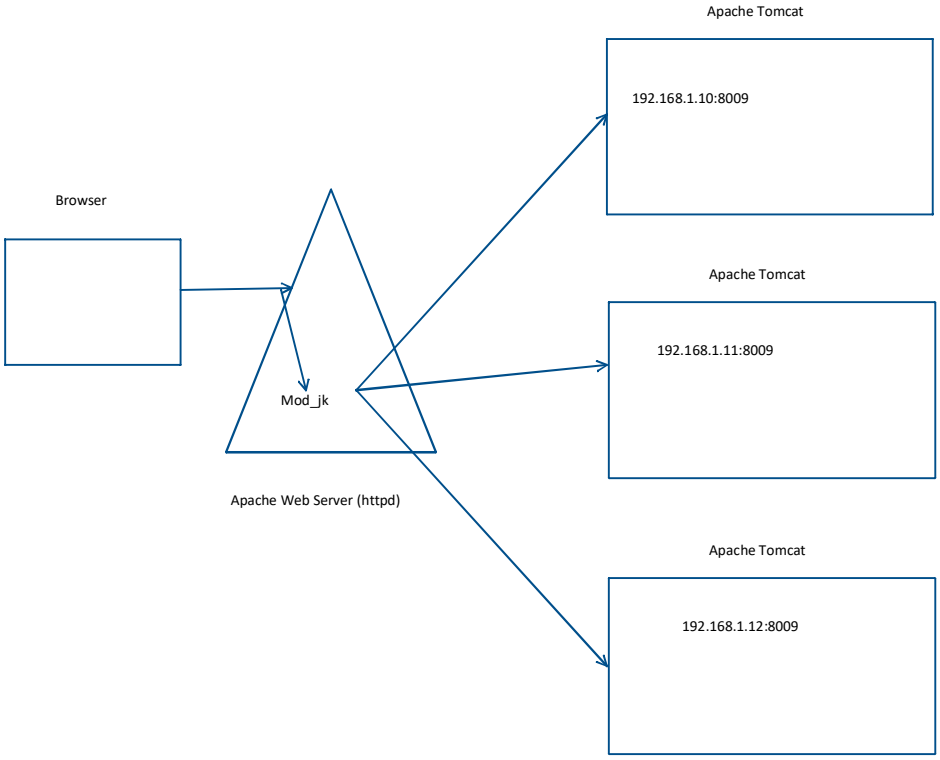
JSP (which Contains html) -->Web Container/JSP Engine converts the .jsp to Servlet--> Generates the Response to USER

GET/POST/PUT/DELETE ---> http Client

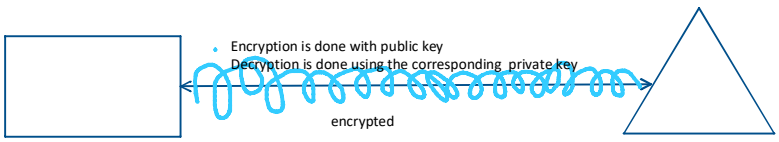
1. Browser (GET and POST using a Form)
2. Test clients like Postman to fire all http methods
3. Curl for the above tasks
4. You can write java code for http clients!!!
    a. Commons http client
    b. Java.net package
    c. Other 3rd party API like spring mvc, Jersey

Jakarta Tomcat(Catalina)-->given to Apache Foundation---> Apache Tomcat (Catalina)

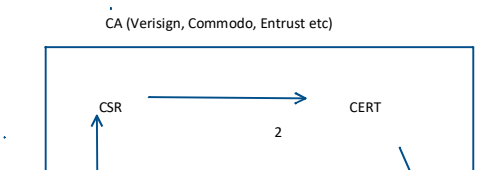
AJP--> Apache JServe Protocol



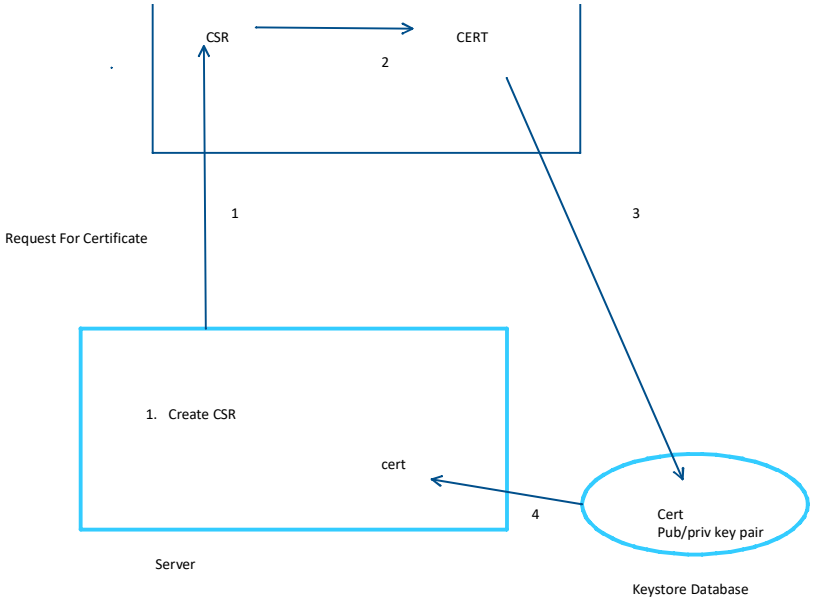
SSL (HTTPS)



SSL uses a combination of public /private key
Public key is distributed to clients and is used to encrypt the message.
The private key is not distributable, it is used to decrypt the message which has been encrypted by the corresponding public key
The certificate is sent to the client along with the public key



1. Your Server needs a digital certificate to prove its valid identity (CA Cert)
2. So, You apply to a CA (Certifying Authority) with a CSR (Certificate Signing Request)
3. CA ISSUES a certificate and you import the same to your key store
4. You configure this keystore to be used by your server
5. Now the server is ready to support communication using SSL (HTTPS)



Self Signed Certificate
(You are the CA and provide guarantee)

Usage of Keytool to create Self Signed Certificate and covert the keystore to pkcs12 format

1. Keytool -genkey -alias mykey -keystore mykeys.jks -keyalg RSA -validity 365
2. keytool -importkeystore -srckeystore mykeys.jks -destkeystore mykeys.jks -deststoretype pkcs12

After you have created the certificate and the keystore, open tomcat/conf/server.xml and locate connector for port 8443 and uncomment it. The configuration should match the one given below:

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
        maxThreads="150" SSLEnabled="true" secure="true" sslProtocol="TLS" keystoreFile="c:/ssl-temp/mykeys.jks"
keystorePass="welcome1">

  </Connector>
```

Save the file after you have updated the configuration and restart apache tomcat. Test the certificate using

https://localhost:8443/

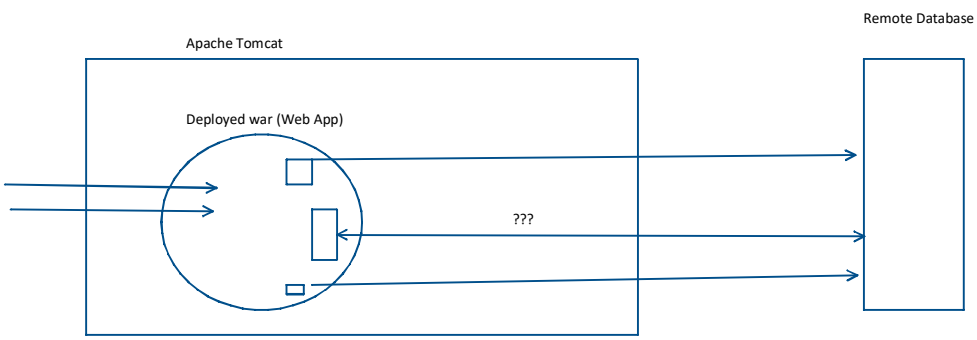
The browser will raise warning, indicates the certificate installation is successful.

Automatic Redirection to port 8443 even if you access http port

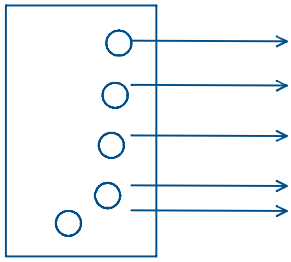
Add the following snippet to you web applications web.xml

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

Database connection with Apache Tomcat



1. We use JDBC
2. Use connection Pool (Data Source)
    a. Collection of pre initialized reusable Connection Objects



Connection Pool

JNDI (Java Naming and Directory Interface)

Data Source on Apache Tomcat

1. To Create a Datasource in Apache Tomcat, we need
    a. Configure the Database and its JDBC Driver
    b. Configuration for Apache Tomcat

Step1: initialize database with the given data
Run MYSQL database
Log into the MYSQL Client
C:\> mysql -u <user> -p

Mysql> create database demo;
>use demo;

- Paste the content from the given sample .sql file.

Step2: install the mysql jdbc driver in tomcat/lib folder (copy the driver .jar here)

Step 3: add the following snippet to tomcat/conf/context.xml to look as follows:

```
<Context >
<Resource name="jdbc/ds" auth="Container"
        type="javax.sql.DataSource" username="root" password="root"
        driverClassName="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost:3306/demo"
        maxTotal="8"
        />
<!--You have some more lines here-->
</Context>
```

Step4: Deploy the given Application to Apache Tomcat
Copy the given sample TestDataSource.war to tomcat/webapps/

Step5: restart Apache Tomcat and access the application using the following url:
http://host:port/TestDataSource/EmpInfo.html

What we did today

Tomcat Architecture
SSL on Tomcat
Database/DataSource on Tomcat with Testing
HTML tags