Online Auctioning

Tomcat Deployment Manual

**Slippery Rock University**

**Spring 2024**

Contributions by:

Jacob Johnston – jrj1014@sru.edu

Tiffany Morrow – tbm1005@sru.edu

Nicholas Vioral – ncv1005@sru.edu

Wolfgang Zoppelt – wsz1001@sru.edu

Table of Contents

[Enabling SSL Configuration in Project 3](#_Toc164611168)

[Enabling SSL within Eclipse Embedded Tomcat Server 3](#_Toc164611169)

[Generating a New Keystore 3](#_Toc164611170)

[Configure application.properties file 5](#_Toc164611171)

[Setting up a Local Tomcat Server 6](#_Toc164611172)

[Prerequisites 6](#_Toc164611173)

[Configuration 6](#_Toc164611174)

[Assigning JAVA\_HOME environment variable 6](#_Toc164611175)

[Configuring Tomcat Server 8](#_Toc164611176)

[Verifying Tomcat Install 11](#_Toc164611177)

[Deploying Project onto Tomcat Server 13](#_Toc164611178)

[Building WAR file 13](#_Toc164611179)

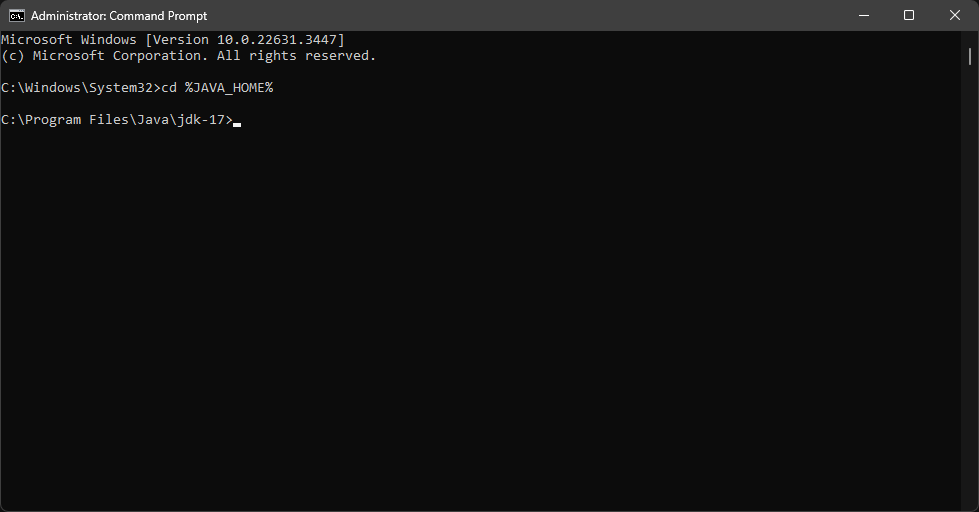
[Deploying Project onto Tomcat Server with WAR file 14](#_Toc164611180)

# Enabling SSL Configuration in Project

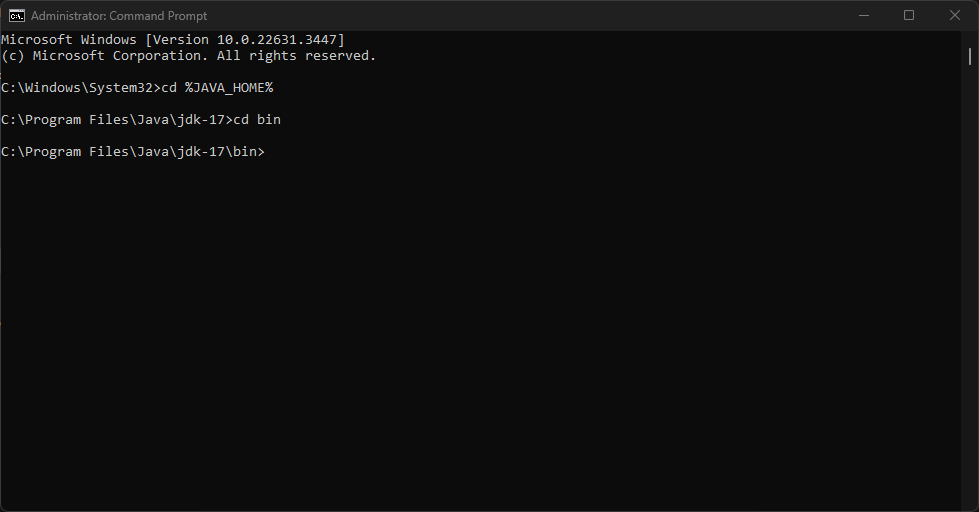
## Enabling SSL within Eclipse Embedded Tomcat Server

### Generating a New Keystore

To first ensure that your project is set up to run on SSL within your Eclipse environment, you will need to generate a new keystore. To do this, open your terminal environment and navigate to your JDK 17 folder.



Your next step is to navigate to the bin folder of your JDK folder.

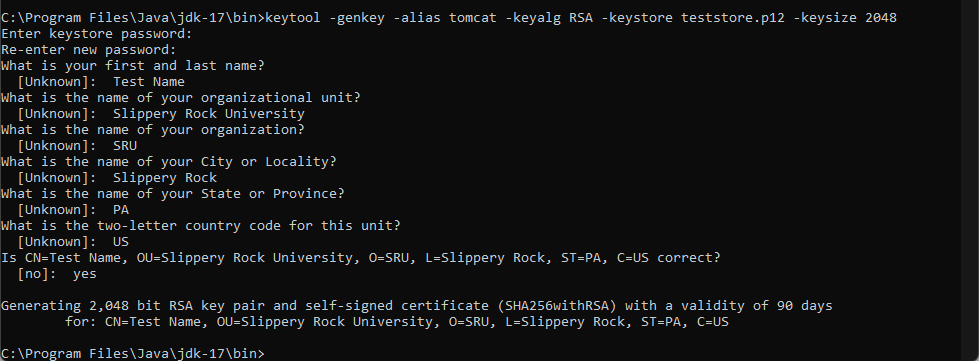


After navigating to this folder, you will want to run this command:

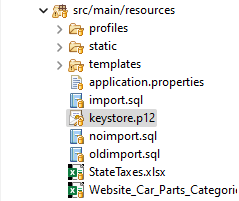
*keytool -genkey -alias tomcat -keyalg RSA -keystore keystore.p12 -keysize 2048*

The keytool will then ask you questions to generate a new keystore.

*\* IMPORTANT: Do take note of the keystore password in which you enter! You will need it in later steps. \**

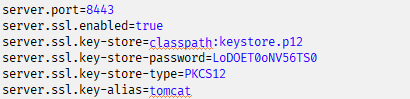


Now that you have generated this keystore, move it from the bin folder of your JDK to the resources folder within your project.



### Configure application.properties file

With the keystore.p12 in your project, you may now edit the application.properties file to reflect these changes. Within application.properties, ensure that these settings are configured.



\* *NOTE: The “server.ssl.key-store-password” property must be set to the password you used when you generated the keystore. Otherwise, this setup will not work. If you forgot the password, restart the process over again and be sure to note down the password. \**

Your embedded Eclipse Tomcat server should now be running through SSL. To ensure it is working, you will need to connect the server via <https://localhost:8443>.

# Setting up a Local Tomcat Server

## Prerequisites

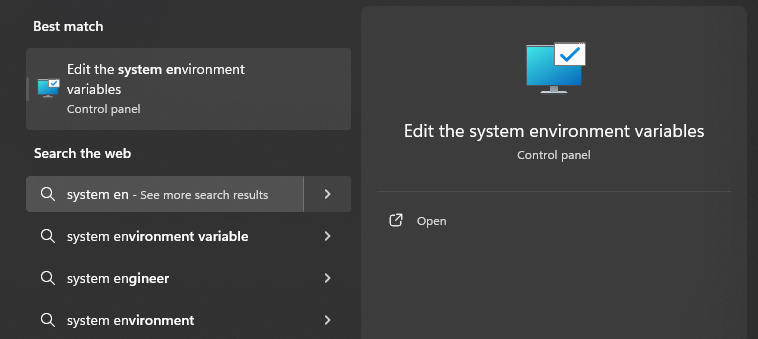
* **Java Development Kit 17:** Ensure that JDK 17 is installed. It can be installed via Oracles official website at this link: <https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html>
* **Apache Tomcat 10.0.x:** This is the Tomcat Server that you will be running. It is obtainable on the Apache Tomcat website: <https://archive.apache.org/dist/tomcat/tomcat-10/v10.0.27/bin/>

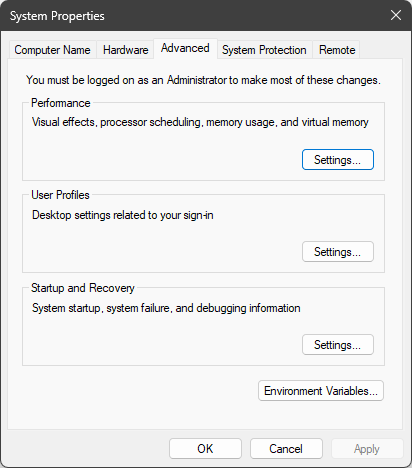
## Configuration

### Assigning JAVA\_HOME environment variable

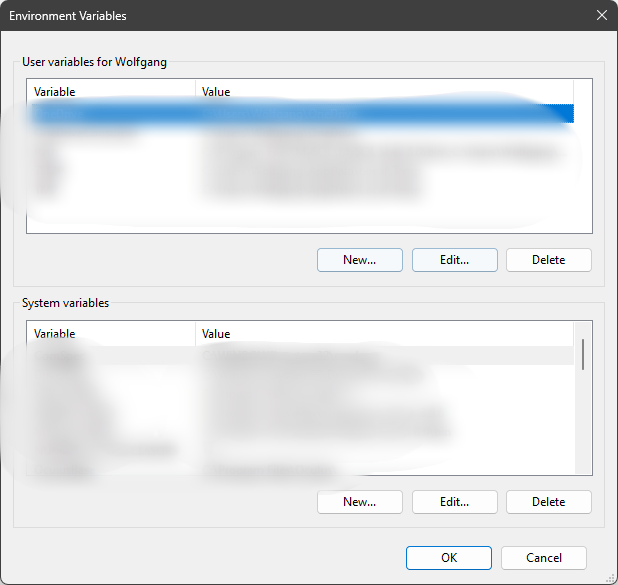
The first step is to ensure that your **JAVA\_HOME** system environment variable points to your JDK 17 installation. If this is already done, you may skip this section.

First navigate to your system environment variables menu.

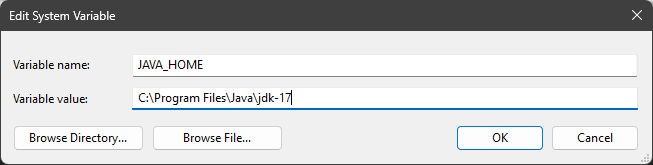




Once you navigate to this menu, you will want to click on **Environment Variables…**

****

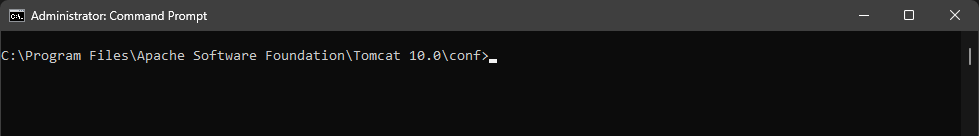
Click on the **New…** option. Enter **JAVA\_HOME** in the variable name field, and the path to your JDK installation in the variable value field.



Click **OK** and you are now done configuring your **JAVA\_HOME** variable.

### Configuring Tomcat Server

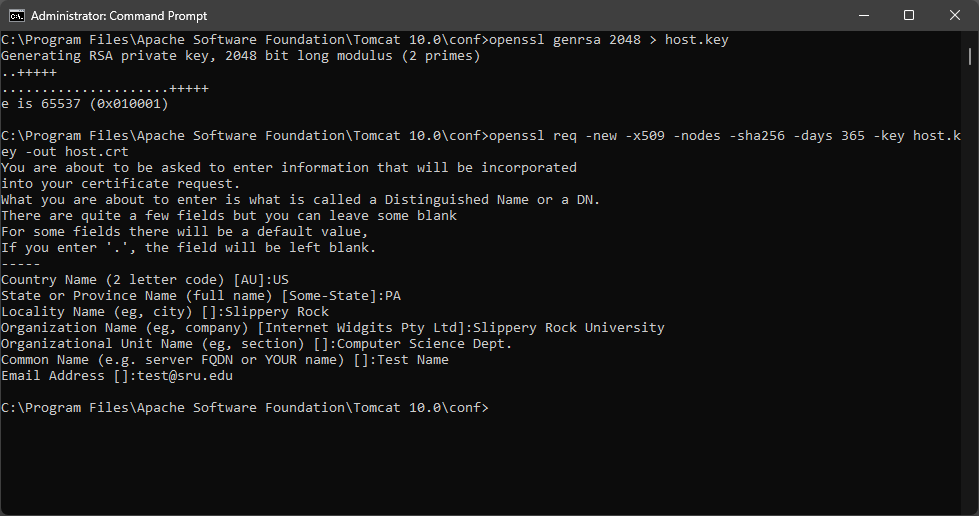
First, open a Command Prompt as Administrator and navigate to your Apache Tomcat Installation. Within your installation, also navigate to the **conf** folder.

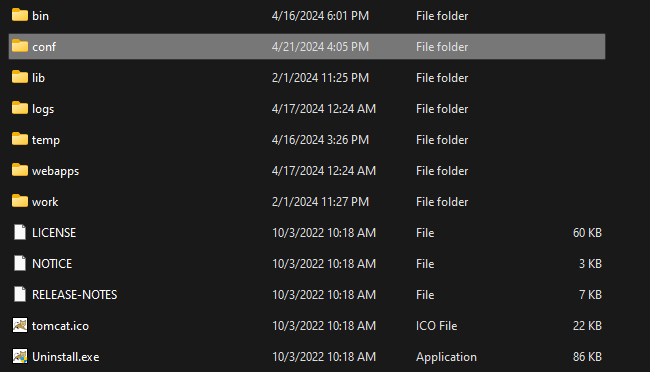


Now that you are here, run the following commands:

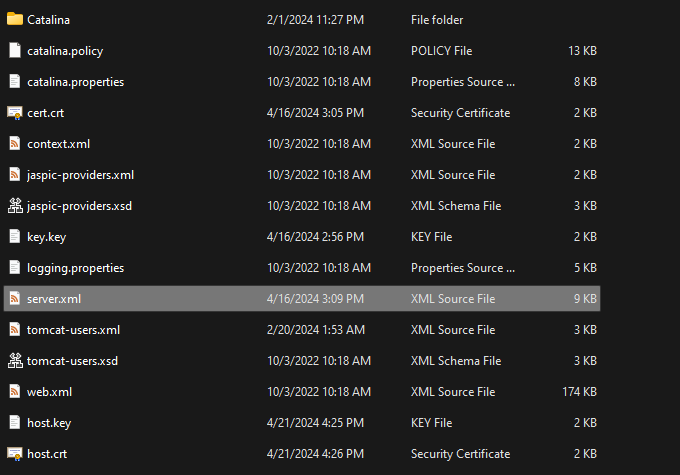
openssl genrsa 2048 > key.key

*openssl req -new -x509 -nodes -sha256 -days 365 -key key.key -out cert.crt*



Now navigate to your Apache Tomcat installation within your System Explorer. You will need to navigate to the **conf** folder. 

Within this folder, edit the **server.xml** file.



In this file, you will want to navigate to the default **<Connector/>**. Comment out this connecter and replace it with this connector.

<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"

           maxThreads="150" SSLEnabled="true" >

    <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />

    <SSLHostConfig>

        <Certificate certificateKeyFile="conf/key.key"

                     certificateFile="conf/cert.crt"

                     type="RSA" />

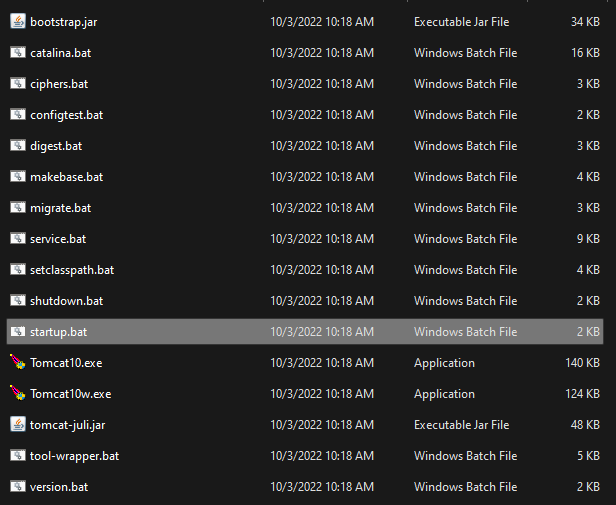
    </SSLHostConfig>

</Connector>

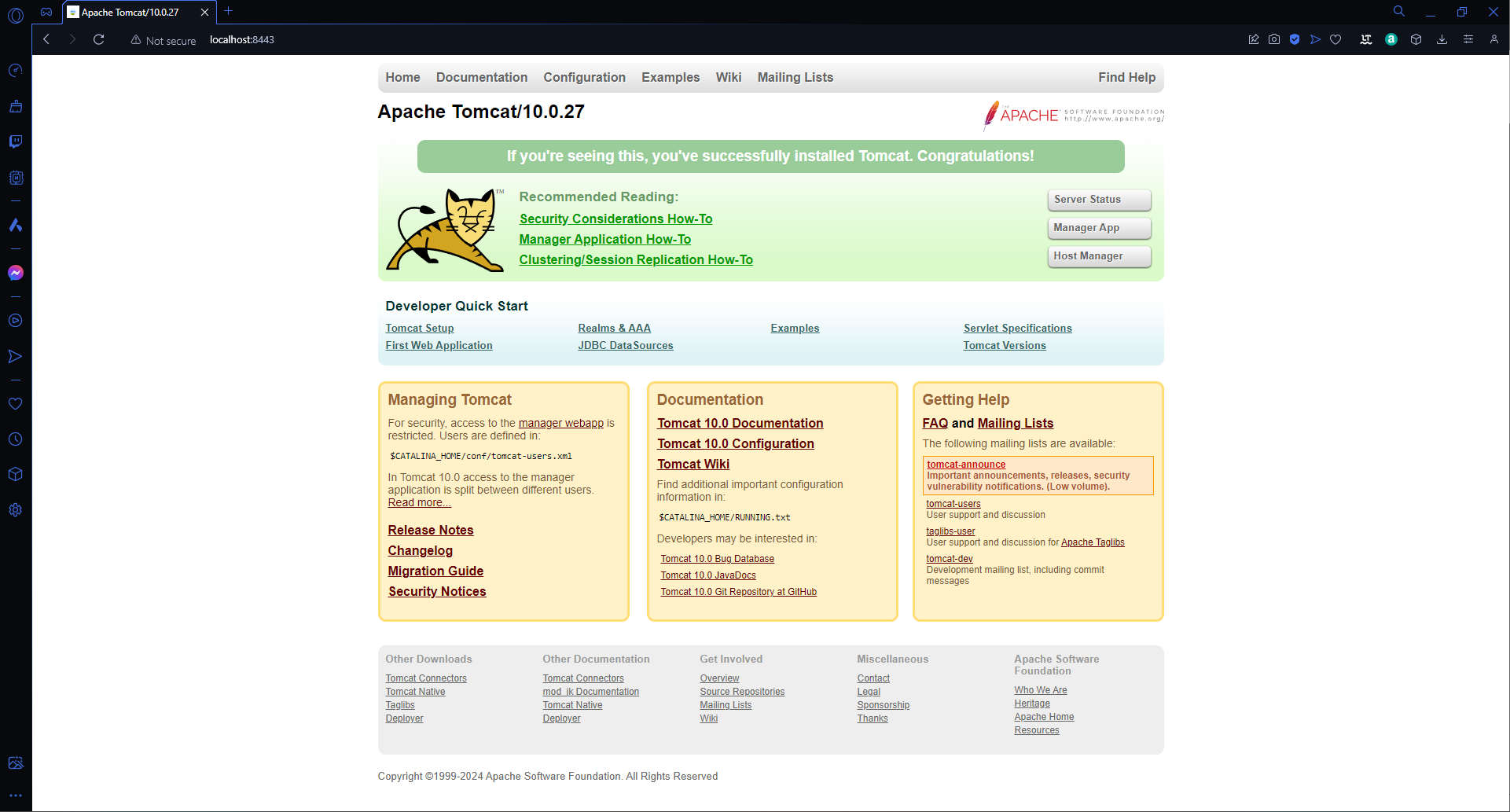
Ensure that the **certificateKeyFile** and **certificateFile** are properly named to the key and certificate you generated. Your Tomcat server is now configured.

### Verifying Tomcat Install

The next step is to make sure your Tomcat Server is working. To launch the server, navigate to the **bin** folder within your Tomcat installation. After navigating, start the server using the **startup.bat** file.



Once the server is running, go into an internet browser and navigate to [https:\\localhost:8443](https://localhost:8443). If your Tomcat Server is running, it will bring you to this page.



Congratulations! Your Tomcat server is up and running.

## Deploying Project onto Tomcat Server

### Building WAR file

The first step in deploying your project onto your newly made Tomcat Server is to build a Web Application Archive (WAR) file. To do so, open the terminal within your Eclipse Environment. Ensure that the current directory you are in is the one containing your **pom.xml** file. After doing so, execute either of the following commands.

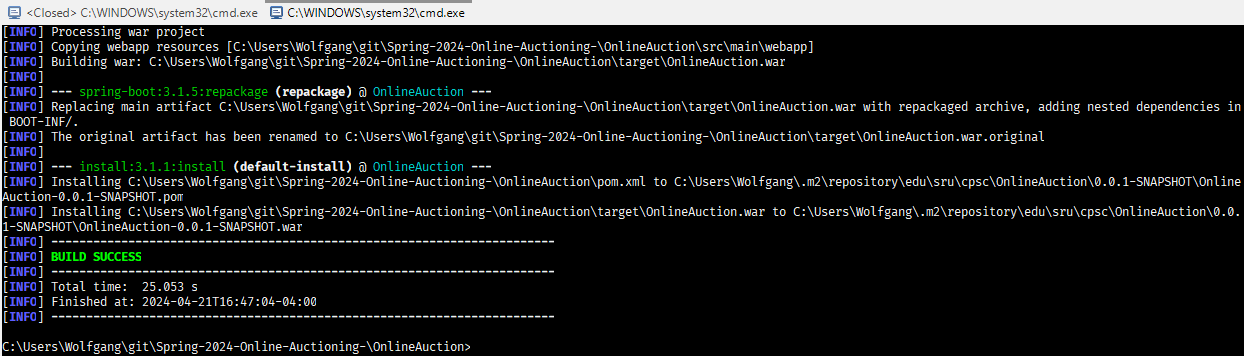
**To generate WAR file after running project tests:**

mvn clean install

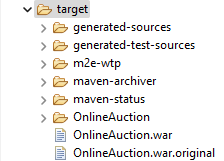
**To generate WAR file without running any project tests:**

mvn clean install -DskipTests

Attempt to run the command that includes project tests first. This will ensure that any JunitTests that are included within your project will be run before building the WAR file. If there are test failures, however, this command will fail and not generate a WAR file. If you wish to still create a WAR file with failed tests, use the second command given. You will know it is successful if you get this screen.

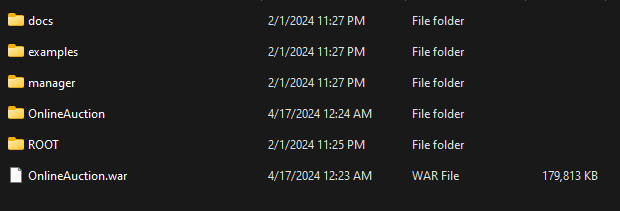


Your WAR file should now be generated within your **target** folder of your project.

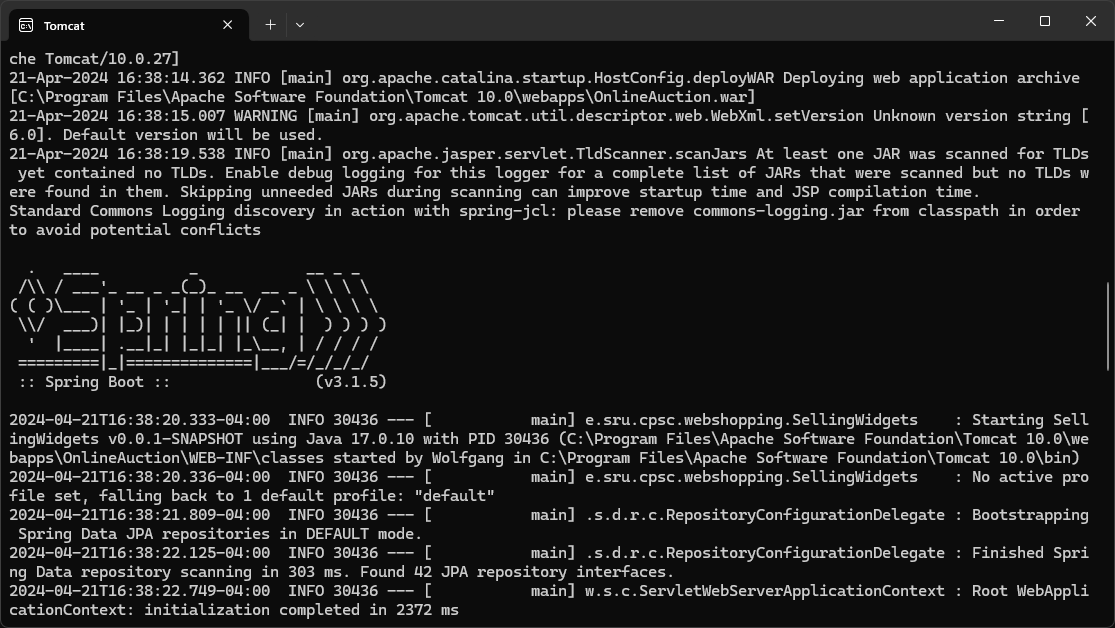


### Deploying Project onto Tomcat Server with WAR file

With this WAR file generated, you may now deploy the project onto your Tomcat Server. You will want to navigate to the **webapps** folder of your Tomcat Installation. Within this folder, simply copy and paste the generated WAR file into it.



Now, start up the Tomcat Server as you normally would. If the deployment was successful, you should see your Spring Boot application begin to launch within the Terminal that your Tomcat Server is running on.



You may now navigate to <https://localhost:8443/OnlineAuction>. If the deployment was successful you will see your application.

