Maven – How to create a Java web application project

By mkyong | Last updated: October 24, 2018
Viewed: 1,143,403 (+262 pv/w)
Tags: jetty | junit 5 | maven | spring mvc | web project

In this tutorial, we will show you how to use Maven to manage a Java web project. At the end, we will create a Spring MVC web application, display a current date on a JSP page.

Technologies used :

1. Maven 3.5.3
2. JDK 8
3. Spring 5.1.0.RELEASE
4. JUnit 5
5. Logback 1.2.3
6. Jetty 9.4.x or Tomcat 8.5

1. Create a web project from Maven Template

Create a web project from Maven template maven-archetype-webapp

```
mvn archetype:generate
      -DgroupId={project-packaging}
      -DartifactId={project-name}
      -DarchetypeArtifactId={maven-template}
      -DinteractiveMode=false
```

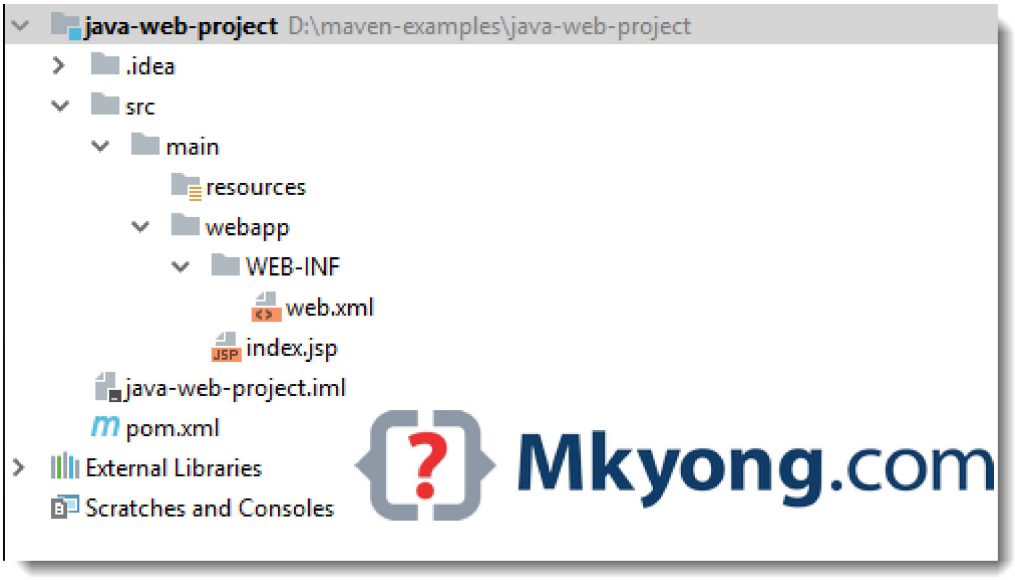
For example,

```
D:\>mvn archetype:generate -DgroupId=com.mkyong.web -DartifactId=java-web-project -DarchetypeArtifactId=maven-archetype-web

[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< org.apache.maven:standalone-pom >-------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:3.0.1:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.0.1:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO]
[INFO] --- maven-archetype-plugin:3.0.1:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Batch mode
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-webapp:1.0
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: basedir, Value: D:\
[INFO] Parameter: package, Value: com.mkyong.web
[INFO] Parameter: groupId, Value: com.mkyong.web
[INFO] Parameter: artifactId, Value: java-web-project
[INFO] Parameter: packageName, Value: com.mkyong.web
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: D:\java-web-project
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 6.509 s
[INFO] Finished at: 2018-10-04T15:25:16+08:00
[INFO] ------------------------------------------------------------------------
```

Note
Actually, this is optional to generate a web project from a Maven web template. You can always generate those folders with the classic mkdir command manually.

2. Maven Template

2.1 The following project directory structure will be created.



P.S Above figure is captured from IntelliJ IDEA, just ignore those IDE folders like .idea and java-web-project.iml

2.2 Review the generated pom.xml.

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
         http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mkyong.web</groupId>
  <artifactId>java-web-project</artifactId>
  <packaging>war</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>java-web-project Maven Webapp</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <finalName>java-web-project</finalName>
  </build>
</project>
```

P.S The generated files are not much value, we will update all of them later. First, delete the web.xml, we don't need this.

3. Update POM

3.1 Update the pom.xml file, add dependencies for Spring MVC for web framework, JUnit for unit test, Jetty server to test the web project, and also some Maven configuration.

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mkyong.web</groupId>
    <artifactId>java-web-project</artifactId>
    <packaging>war</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>java-web-project Maven Webapp</name>
    <url>http://maven.apache.org</url>
    <properties>
        <!-- https://maven.apache.org/general.html#encoding-warning -->
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <spring.version>5.1.0.RELEASE</spring.version>
    </properties>

    <dependencies>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>${spring.version}</version>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-test</artifactId>
            <version>${spring.version}</version>
        </dependency>

        <!-- logging , spring 5 no more bridge, thanks spring-jcl -->
        <dependency>
            <groupId>ch.qos.logback</groupId>
            <artifactId>logback-classic</artifactId>
            <version>1.2.3</version>
        </dependency>

        <!-- junit 5, unit test -->
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>5.3.1</version>
            <scope>test</scope>
        </dependency>

        <!-- unit test -->
```

```
    <dependency>
      <groupId>org.hamcrest</groupId>
      <artifactId>hamcrest-library</artifactId>
      <version>1.3</version>
      <scope>test</scope>
    </dependency>

    <!-- for web servlet -->
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>3.1.0</version>
      <scope>provided</scope>
    </dependency>

    <!-- Some containers like Tomcat don't have jstl library -->
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>jstl</artifactId>
      <version>1.2</version>
      <scope>provided</scope>
    </dependency>

  </dependencies>
  <build>
    <finalName>java-web-project</finalName>
    <plugins>
      <!-- http://www.eclipse.org/jetty/documentation/current/jetty-maven-plugin.html -->
      <plugin>
        <groupId>org.eclipse.jetty</groupId>
        <artifactId>jetty-maven-plugin</artifactId>
        <version>9.4.12.v20180830</version>
      </plugin>

      <!-- Default is too old, update to latest to run the latest Spring 5 + jUnit 5 -->
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.22.0</version>
      </plugin>

      <!-- Default 2.2 is too old, update to latest -->
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <version>3.2.2</version>
      </plugin>

    </plugins>
  </build>

</project>
```

3.2 Display the project dependencies.

```
D:\> mvn dependency:tree

...
[INFO] --- maven-dependency-plugin:2.8:tree (default-cli) @ java-web-project ---
[INFO] com.mkyong.web:java-web-project:war:1.0-SNAPSHOT
[INFO] +- org.springframework:spring-webmvc:jar:5.1.0.RELEASE:compile
[INFO] |  +- org.springframework:spring-aop:jar:5.1.0.RELEASE:compile
[INFO] |  +- org.springframework:spring-beans:jar:5.1.0.RELEASE:compile
[INFO] |  +- org.springframework:spring-context:jar:5.1.0.RELEASE:compile
[INFO] |  +- org.springframework:spring-core:jar:5.1.0.RELEASE:compile
[INFO] |  |  \- org.springframework:spring-jcl:jar:5.1.0.RELEASE:compile
[INFO] |  +- org.springframework:spring-expression:jar:5.1.0.RELEASE:compile
[INFO] |  \- org.springframework:spring-web:jar:5.1.0.RELEASE:compile
[INFO] +- org.springframework:spring-test:jar:5.1.0.RELEASE:compile
[INFO] +- ch.qos.logback:logback-classic:jar:1.2.3:compile
[INFO] |  +- ch.qos.logback:logback-core:jar:1.2.3:compile
[INFO] |  \- org.slf4j:slf4j-api:jar:1.7.25:compile
[INFO] +- org.junit.jupiter:junit-jupiter-engine:jar:5.3.1:test
[INFO] |  +- org.apiguardian:apiguardian-api:jar:1.0.0:test
[INFO] |  +- org.junit.platform:junit-platform-engine:jar:1.3.1:test
[INFO] |  |  +- org.junit.platform:junit-platform-commons:jar:1.3.1:test
[INFO] |  |  \- org.opentest4j:opentest4j:jar:1.1.1:test
[INFO] |  \- org.junit.jupiter:junit-jupiter-api:jar:5.3.1:test
[INFO] +- org.hamcrest:hamcrest-library:jar:1.3:test
[INFO] |  \- org.hamcrest:hamcrest-core:jar:1.3:test
[INFO] +- javax.servlet:javax.servlet-api:jar:3.1.0:provided
[INFO] \- javax.servlet:jstl:jar:1.2:provided
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 0.931 s
[INFO] Finished at: 2018-10-08T15:55:08+08:00
[INFO] ------------------------------------------------------------------------
```

4. Spring MVC + JSP + LogBack

4.1 Create a few files to bootstrap Spring MVC web project.

SpringConfig.java

```
package com.mkyong.web.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.view.InternalResourceViewResolver;
```

```
import org.springframework.web.servlet.view.JstlView;

@EnableWebMvc
@Configuration
@ComponentScan({"com.mkyong.web"})
public class SpringConfig implements WebMvcConfigurer {

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/resources/**")
                .addResourceLocations("/resources/");
    }

    @Bean
    public InternalResourceViewResolver viewResolver() {
        InternalResourceViewResolver viewResolver
                = new InternalResourceViewResolver();
        viewResolver.setViewClass(JstlView.class);
        viewResolver.setPrefix("/WEB-INF/views/");
        viewResolver.setSuffix(".jsp");
        return viewResolver;
    }
}
```

WebInitializer.java

```
package com.mkyong.web;

import com.mkyong.web.config.SpringConfig;
import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;

public class WebInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {

    @Override
    protected Class<?>[] getRootConfigClasses() {
        return null;
    }

    @Override
    protected Class<?>[] getServletConfigClasses() {
        return new Class[]{SpringConfig.class};
    }

    @Override
    protected String[] getServletMappings() {
        return new String[]{"/"};
    }

}
```

WelcomeController.java

```
package com.mkyong.web.controller;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

import java.util.Date;

@Controller
public class WelcomeController {

    private final Logger logger = LoggerFactory.getLogger(WelcomeController.class);

    @GetMapping("/")
    public String index(Model model) {
        logger.debug("Welcome to mkyong.com...");
        model.addAttribute("msg", getMessage());
        model.addAttribute("today", new Date());
        return "index";

    }

    private String getMessage() {
        return "Hello World";
    }

}
```

4.2 Move the index.jsp file into the WEB-INF folder, and update it

index.jsp

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<html>
<body>
<h1>${msg}</h1>
<h2>Today is <fmt:formatDate value="${today}" pattern="yyy-MM-dd" /></h2>
</body>
</html>
```

4.3 Logs to console.

logbacl.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
```

```
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
        <layout class="ch.qos.logback.classic.PatternLayout">

            <Pattern>
                %d{yyyy-MM-dd HH:mm:ss} [%thread] %-5level %logger{36} - %msg%n
            </Pattern>

        </layout>
    </appender>

    <logger name="com.mkyong.web" level="debug"
            additivity="false">
        <appender-ref ref="STDOUT"/>
    </logger>

    <root level="error">
        <appender-ref ref="STDOUT"/>
    </root>

</configuration>
```

5. Unit Test

A simple Spring MVC 5 + JUnit 5 example.

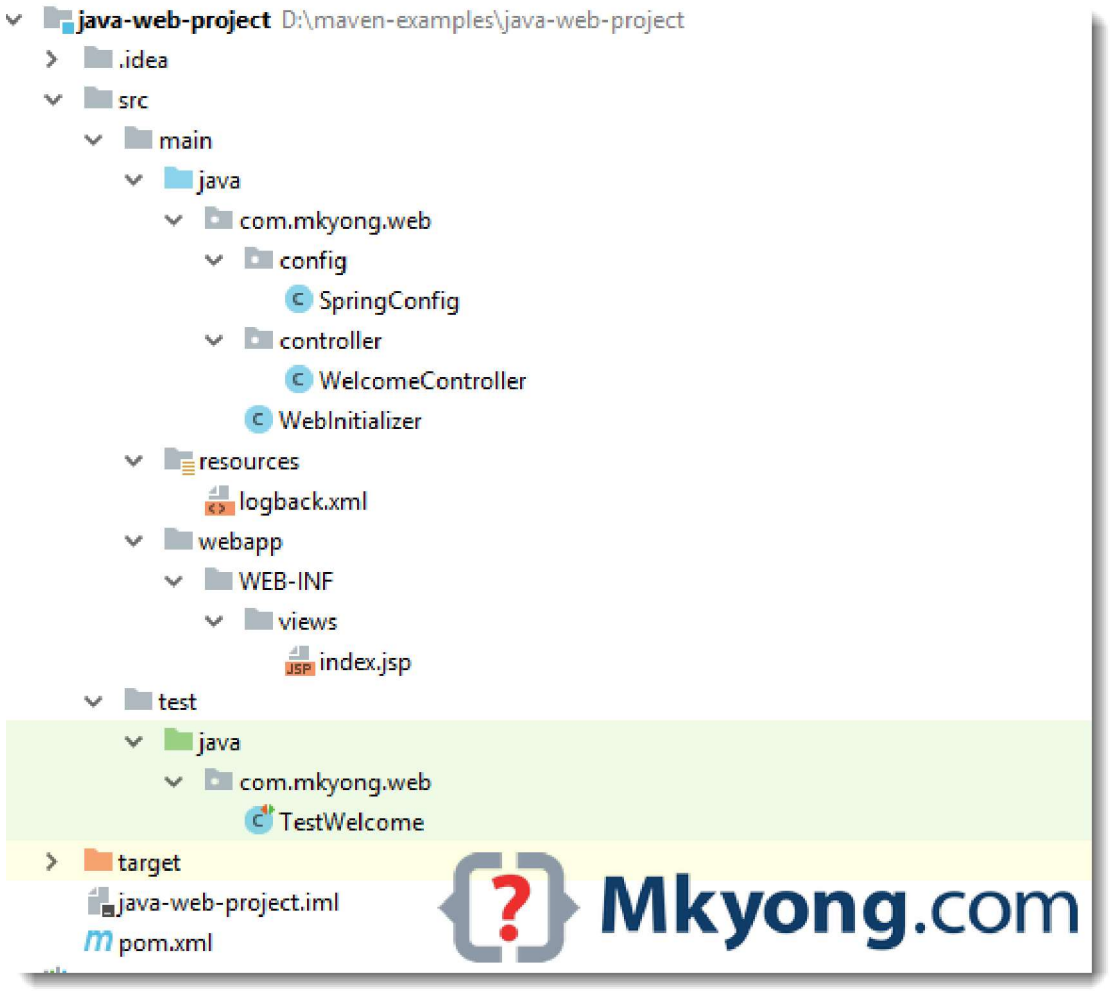
TestWelcome.java

```
package com.mkyong.web;

import com.mkyong.web.config.SpringConfig;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.junit.jupiter.web.SpringJUnitWebConfig;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.setup.MockMvcBuilders;
import org.springframework.web.context.WebApplicationContext;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
import static org.springframework.test.web.servlet.result.MockMvcResultHandlers.print;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.*;

@SpringJUnitWebConfig(SpringConfig.class)
public class TestWelcome {

    private MockMvc mockMvc;

    @Autowired
    private WebApplicationContext webAppContext;

    @BeforeEach
    public void setup() {
        mockMvc = MockMvcBuilders.webAppContextSetup(webAppContext).build();
```

```
    }

    @Test
    public void testWelcome() throws Exception {

        this.mockMvc.perform(
                get("/"))
                .andDo(print())
                .andExpect(status().isOk())
                .andExpect(view().name("index"))
                .andExpect(forwardedUrl("/WEB-INF/views/index.jsp"))
                .andExpect(model().attribute("msg", "Hello World"));
    }

}
```

6. Directory Structure

Review the final files and directory structure.



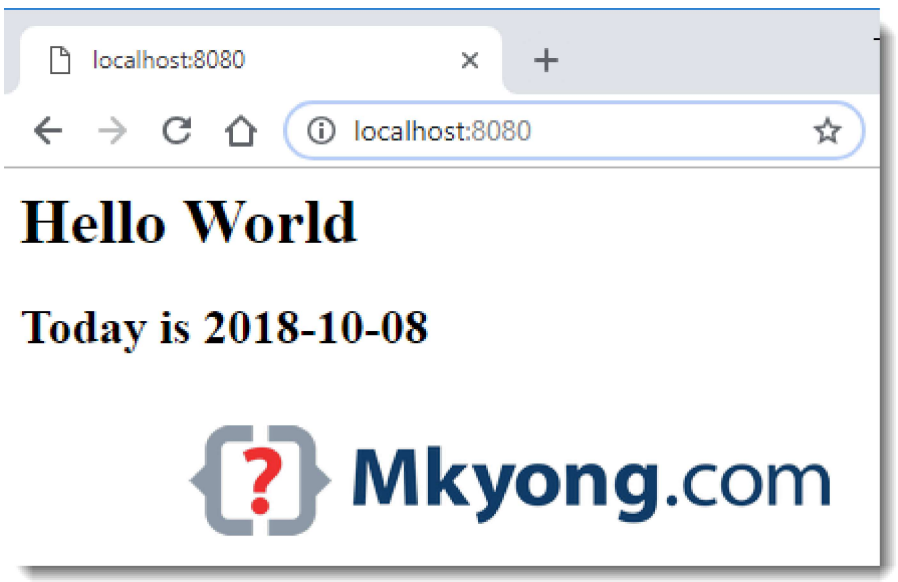
Read this Maven Standard Directory Layout.

7. Demo

7.1 Test the web project with Jetty web server – mvn jetty:run

```
D:\> mvn jetty:run

[INFO] webAppSourceDirectory not set. Trying src\main\webapp
[INFO] Reload Mechanic: automatic
[INFO] nonBlocking:false
[INFO] Classes = D:\java-web-project\target\classes
[INFO] Configuring Jetty for project: java-web-project Maven Webapp
[INFO] Logging initialized @4821ms to org.eclipse.jetty.util.log.Slf4jLog
[INFO] Context path = /
[INFO] Tmp directory = D:\java-web-project\target\tmp
[INFO] Web defaults = org/eclipse/jetty/webapp/webdefault.xml
[INFO] Web overrides =  none
[INFO] web.xml file = null
[INFO] Webapp directory = D:\java-web-project\src\main\webapp
[INFO] jetty-9.4.12.v20180830; built: 2018-08-30T13:59:14.071Z; git: 27208684755d94a92186989f695db2d7b21ebc51; j

...

[INFO] 1 Spring WebApplicationInitializers detected on classpath
2018-10-08 15:11:50 [main] DEBUG com.mkyong.web.WebInitializer - No ContextLoaderListener registered, as createR
[INFO] DefaultSessionIdManager workerName=node0
[INFO] No SessionScavenger set, using defaults
[INFO] node0 Scavenging every 660000ms
[INFO] Initializing Spring DispatcherServlet 'dispatcher'
[INFO] Started o.e.j.m.p.JettyWebAppContext@68a78f3c{/,file:///D:/java-web-project/src/main/webapp/,AVAILABLE}{file
[INFO] Started ServerConnector@3355168{HTTP/1.1,[http/1.1]}{0.0.0.0:8080}
[INFO] Started @6271ms
[INFO] Started Jetty Server
2018-10-08 15:12:01 [qtp1373051324-19] DEBUG c.m.web.controller.WelcomeController - Welcome to mkyong.com...
```

7.2 Access it via *http://localhost:8080/*



P.S CTRL + C to stop the Jetty web server.

8. Deployment

8.1 mvn package to generate a WAR file for deployment.

```
D:\> mvn package

...

[INFO] Packaging webapp
[INFO] Assembling webapp [java-web-project] in [D:\java-web-project\target\java-web-project]
[INFO] Processing war project
[INFO] Copying webapp resources [D:\java-web-project\src\main\webapp]
[INFO] Webapp assembled in [89 msecs]
[INFO] Building war: D:\java-web-project\target\java-web-project.war
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3.844 s
[INFO] Finished at: 2018-10-08T15:31:12+08:00
[INFO] ------------------------------------------------------------------------
```

The default directory for the generated WAR is target/finalName. Done.