How to create a Web Application Project with Maven

mkyong.com/maven/how-to-create-a-web-application-project-with-maven

By mkyong | December 21, 2012 | Updated : October 26, 2014 | Viewed : 982,644 | +2,276 pv/w

In this tutorial, we will show you how to create a Java web project (Spring MVC) with Maven.

maven-web-project-directory

Technologies used :

1. Maven 3.1.1
2. Eclipse 4.2
3. JDK 7
4. Spring 4.1.1.RELEASED
5. Tomcat 7
6. Logback 1.0.13

1. Create Web Project from Maven Template

You can create a quick start Java web application project by using the Maven `maven-archetype-webapp` template. In a terminal (*uix or Mac) or command prompt (Windows), navigate to the folder you want to create the project.

Type this command :

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

//for example
$ mvn archetype:generate -DgroupId=com.mkyong
 -DartifactId=CounterWebApp
 -DarchetypeArtifactId=maven-archetype-webapp
 -DinteractiveMode=false
```
Copy

For example :

```
$ pwd
/Users/mkyong/Documents/workspace

$ mvn archetype:generate -DgroupId=com.mkyong -DartifactId=CounterWebApp -
DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false

[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] ------------------------------------------------------------------------
[INFO]
[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: groupId, Value: com.mkyong
[INFO] Parameter: packageName, Value: com.mkyong
[INFO] Parameter: package, Value: com.mkyong
[INFO] Parameter: artifactId, Value: CounterWebApp
[INFO] Parameter: basedir, Value: /Users/mkyong/Documents/workspace
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir:
/Users/mkyong/Documents/workspace/CounterWebApp
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3.147s
[INFO] Finished at: Thu Dec 20 20:35:19 MYT 2012
[INFO] Final Memory: 12M/128M
[INFO] ------------------------------------------------------------------------
```
Copy

A new web project named “`CounterWebApp`“, and some of the standard web directory structure is created automatically.

2. Project Directory Layout

Review the generated project layout :

```
.
|____CounterWebApp
| |____pom.xml
| |____src
| | |____main
| | | |____resources
| | | |____webapp
| | | | |____index.jsp
| | | | |____WEB-INF
| | | | | |____web.xml
```
Copy

Maven generated some folders, a deployment descriptor `web.xml`, `pom.xml` and `index.jsp`.

Note
Please check this official Maven standard directory layout guide to understand more.
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</groupId>
  <artifactId>CounterWebApp</artifactId>
  <packaging>war</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>CounterWebApp 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>CounterWebApp</finalName>
  </build>
</project>
```
Copy

web.xml – Servlet 2.3 is too old, consider upgrade to 2.5

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
  <display-name>Archetype Created Web Application</display-name>
</web-app>
```
Copy

index.jsp – A simple hello world html file

```
<html>
<body>
<script async
src="//pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></script>
<ins class="adsbygoogle"
     style="display:block"
     data-ad-client="ca-pub-2836379775501347"
     data-ad-slot="8821506761"
     data-ad-format="auto"
     data-ad-region="mkyongregion"></ins>
<script>
(adsbygoogle = window.adsbygoogle || []).push({});
</script><h2>Hello World!</h2>
</body>
</html>
```
Copy

3. Eclipse IDE Support

To import this project into Eclipse, you need to generate some Eclipse project configuration files :

3.1 In terminal, navigate to “CounterWebApp” folder, type this command :

```
mvn eclipse:eclipse -Dwtpversion=2.0
```
Copy

Note
This option `-Dwtpversion=2.0` tells Maven to convert the project into an Eclipse web project (WAR), not the default Java project (JAR). For convenience, later we will show you how to configure this WTP option in `pom.xml`.

3.2 Imports it into Eclipse IDE – File -> Import… -> General -> Existing Projects into workspace.

Figure : In Eclipse, if you see a globe icon on top of the project, means this is a web project.

project structure

4. Update POM

In Maven, the web project settings are configure via this single `pom.xml` file.

1. Add project dependencies – Spring, logback and junit
2. Add plugins to configure this project

Read comments for self-explanatory.

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```xml
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</groupId>
<artifactId>CounterWebApp</artifactId>
<packaging>war</packaging>
<version>1.0-SNAPSHOT</version>
<name>CounterWebApp Maven Webapp</name>
<url>http://maven.apache.org</url>
<properties>
 <jdk.version>1.7</jdk.version>
 <spring.version>4.1.1.RELEASE</spring.version>
 <jstl.version>1.2</jstl.version>
 <junit.version>4.11</junit.version>
 <logback.version>1.0.13</logback.version>
 <jcl-over-slf4j.version>1.7.5</jcl-over-slf4j.version>
</properties>

<dependencies>


 <dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>${junit.version}</version>
 </dependency>


 <dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-core</artifactId>
  <version>${spring.version}</version>
  <exclusions>
   <exclusion>
    <groupId>commons-logging</groupId>
    <artifactId>commons-logging</artifactId>
   </exclusion>
  </exclusions>
 </dependency>

 <dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>jcl-over-slf4j</artifactId>
  <version>${jcl-over-slf4j.version}</version>
 </dependency>

 <dependency>
  <groupId>ch.qos.logback</groupId>
  <artifactId>logback-classic</artifactId>
  <version>${logback.version}</version>
 </dependency>

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


 <dependency>
  <groupId>jstl</groupId>
  <artifactId>jstl</artifactId>
  <version>${jstl.version}</version>
 </dependency>

</dependencies>

<build>
 <finalName>CounterWebApp</finalName>

 <plugins>

   <plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-eclipse-plugin</artifactId>
  <version>2.9</version>
  <configuration>

   <downloadSources>true</downloadSources>
   <downloadJavadocs>false</downloadJavadocs>

   <wtpversion>2.0</wtpversion>
  </configuration>
   </plugin>


   <plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>2.3.2</version>
  <configuration>
   <source>${jdk.version}</source>
   <target>${jdk.version}</target>
  </configuration>
   </plugin>


   <plugin>
  <groupId>org.apache.tomcat.maven</groupId>
  <artifactId>tomcat7-maven-plugin</artifactId>
  <version>2.2</version>
  <configuration>
   <path>/CounterWebApp</path>
  </configuration>
   </plugin>

 </plugins>
 </build>
</project>
```
Copy

Note
For convenience, declares `maven-eclipse-plugin` and configure `wtpversion` to avoid typing the parameter `-Dwtpversion=2.0`. Now, each time you use `mvn eclipse:eclipse`, Maven will convert this project into Eclipse web project.

```
mvn eclipse:eclipse --> Eclipse Java project (JAR)
mvn eclipse:eclipse -Dwtpversion=2.0 --> Eclipse Java web project (WAR)


mvn eclipse:eclipse --> Eclipse Java web project (WAR)
```
Copy

5. Update Source Code

In this step, we will create a few files and folders for Spring MVC and logback logging framework, the final project structure will look like this :

```
.
|____pom.xml
|____src
| |____main
| | |____java
| | | |____com
| | | | |____mkyong
| | | | | |____controller
| | | | | | |____BaseController.java
| | |____resources
| | | |____logback.xml
| | |____webapp
| | | |____WEB-INF
| | | | |____mvc-dispatcher-servlet.xml
| | | | |____pages
| | | | | |____index.jsp
| | | | |____web.xml
```
Copy

Note
Create the folder manually if it doesn't exist.

maven-web-project-directory

5.1 Create a Spring MVC controller class.

/src/main/java/com/mkyong/controller/BaseController.java

```
package com.mkyong.controller;

import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class BaseController {

 private static int counter = 0;
 private static final String VIEW_INDEX = "index";
 private final static org.slf4j.Logger logger =
LoggerFactory.getLogger(BaseController.class);

 @RequestMapping(value = "/", method = RequestMethod.GET)
 public String welcome(ModelMap model) {

  model.addAttribute("message", "Welcome");
  model.addAttribute("counter", ++counter);
  logger.debug("[welcome] counter : {}", counter);


  return VIEW_INDEX;

 }

 @RequestMapping(value = "/{name}", method = RequestMethod.GET)
 public String welcomeName(@PathVariable String name, ModelMap model) {

  model.addAttribute("message", "Welcome " + name);
  model.addAttribute("counter", ++counter);
  logger.debug("[welcomeName] counter : {}", counter);
  return VIEW_INDEX;

 }

}
```
Copy

5.2 Create a Spring configuration file.

/src/main/webapp/WEB-INF/mvc-dispatcher-servlet.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:context="http://www.springframework.org/schema/context"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd">

 <context:component-scan base-package="com.mkyong.controller" />

 <bean
  class="org.springframework.web.servlet.view.InternalResourceViewResolver">
  <property name="prefix">
   <value>/WEB-INF/pages/</value>
  </property>
  <property name="suffix">
   <value>.jsp</value>
  </property>
 </bean>

</beans>
```
Copy

5.3 Update existing `web.xml` to support Servlet 2.5 (the default Servlet 2.3 is too old), and also integrates Spring framework via Spring's listener `ContextLoaderListener`.

/src/main/webapp/WEB-INF/web.xml

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
       http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
 version="2.5">

 <display-name>Counter Web Application</display-name>

 <servlet>
  <servlet-name>mvc-dispatcher</servlet-name>
  <servlet-class>
                      org.springframework.web.servlet.DispatcherServlet
              </servlet-class>
  <load-on-startup>1</load-on-startup>
 </servlet>

 <servlet-mapping>
  <servlet-name>mvc-dispatcher</servlet-name>
  <url-pattern>/</url-pattern>
 </servlet-mapping>

 <context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/mvc-dispatcher-servlet.xml</param-value>
 </context-param>

 <listener>
  <listener-class>
                 org.springframework.web.context.ContextLoaderListener
              </listener-class>
 </listener>
</web-app>
```
Copy

5.4 Moves `index.jsp` to folder `WEB-INF/pages`, in order to protect direct access. And updates the content :

/src/main/webapp/WEB-INF/pages/index.jsp

```
<html>
<body>
<h1>Maven + Spring MVC Web Project Example</h1>

<h2>Message : ${message}</h2>
<h2>Counter : ${counter}</h2>
</body>
</html>
```
Copy

5.5 Create a `logback.xml` file and puts in resources folder.

/src/main/resources/logback.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.controller" level="debug"
  additivity="false">
  <appender-ref ref="STDOUT" />
 </logger>

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

</configuration>
```
Copy

6. Eclipse + Tomcat

Here are few ways to deploy and test the web project.

6.1 To compile, test and package the project into a WAR file, type this :

```
mvn package
```
Copy

A new WAR file will be generated at `project/target/CounterWebApp.war` , just copy and deploy to your Tomcat.

6.2 If you want to debug this project via Eclipse server plugin (Tomcat or other containers), type this again :

```
mvn eclipse:eclipse
```
Copy

If everything is fine, the project dependencies will be attached to the project web deployment assembly.

eclipse-web-deployment-assembly

Figure : Right clicks on the project -> Properties -> Deployment Assembly

6.3 Since the Maven Tomcat plugin is declared :

pom.xml

```
<plugin>
 <groupId>org.apache.tomcat.maven</groupId>
 <artifactId>tomcat7-maven-plugin</artifactId>
 <version>2.2</version>
 <configuration>
  <path>/CounterWebApp</path>
 </configuration>
</plugin>
```
Copy

Type this command :

```
mvn tomcat:run
```
Copy

It will start Tomcat and deploy your project default to port 8080.

7. Demo

Deploy the WAR file on Tomcat :

7.1 *http://localhost:8080/CounterWebApp/*

maven-web-project-demo1

7.2 *http://localhost:8080/CounterWebApp/mkyong*

maven-web-project-demo2

7.3 *http://localhost:8080/CounterWebApp/android*

maven-web-project-demo3

Download Source Code

Download it – Maven-WebProject-CounterWebApp.zip (9 KB)

References

About the Author

mkyong

Founder of Mkyong.com, love Java and open source stuff. Follow him on Twitter, or befriend him on Facebook or Google Plus. If you like my tutorials, consider make a donation to these charities.

Comments