Configuración de una aplicación con estructura no estándar de Maven para poder ejecutar las tareas de construcción de pruebas con Framework de Pruebas de Dextra (DTF)

Contents

[Pasos 2](#_Toc330415959)

[pom.xml generado 3](#_Toc330415960)

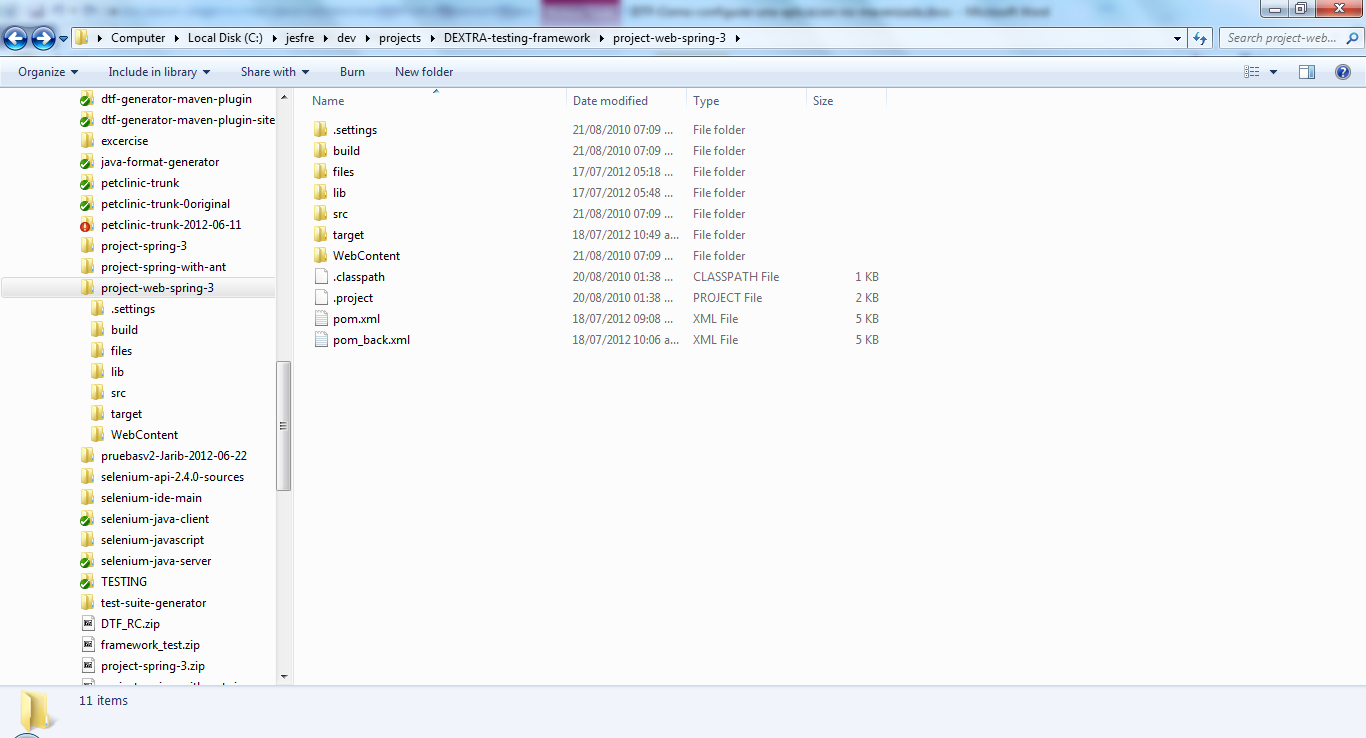
# Pasos

Mavenizar un proyecto Web Java requiere que el proyecto tenga una estructura de carpetas estándar de Maven y configurar el archivo pom.xml. Para mavenizar un proyecto con una estructura de carpetas distinta basta con correr el programa com.dextratech.dtf.utils.MavenizerUtil. El programa pedirá insertar la siguiente información:

* 1. La ubicación de la carpeta raíz del proyecto que se quiere mavenizar.
  2. La ubicación de la carpeta donde se encuentra el código fuente partiendo de la carpeta raíz.
  3. La ubicación de la carpeta donde se encuentran los archivos Web partiendo de la carpeta raíz.
  4. La ubicación de la carpeta WEB-INF del proyecto partiendo de la carpeta raíz.
  5. La ubicación de la carpeta que contiene las librerías del proyecto partiendo de la carpeta raíz.
  6. El nombre del proyecto Maven. Este es opcional, si no se especifica el programa tomará el nombre de la carpeta raíz del proyecto como el nombre del mismo.

Con esta información el programa generará un archivo pom.xml con la configuración necesaria que permitirá a Maven reconocer la estructura como propia.

El siguiente ejemplo muestra la ejecución del programa en un proyecto Web Java con la siguiente estructura:



El programa se ejecuta con el comando de Java:

**dtf-generator-maven-plugin/target>**java -classpath dtf-generator-maven-plugin-1.0-SNAPSHOT.jar com.dextratech.dtf.utils.MavenizerUtil

Root project directory: C:\jesfre\dev\projects\DEXTRA-testing-framework\project-web-spring-3

Source folder path from root directory: src

Webroot folder path from root directory: WebContent

WEB-INF folder path from root directory: WebContent/WEB-INF

Library folder path from root directory: WebContent/WEB-INF/li

Project name (optional):

El programa genera un nombre para el proyecto y obtiene las librerías del proyecto y genera la configuración de las dependencias.

New project name: project-web-spring-3

Next file: commons-logging-1.1.1.jar - OK

Next file: jstl-1.2.jar - OK

Next file: spring-asm-3.0.3.RELEASE.jar - OK

Next file: spring-beans-3.0.3.RELEASE.jar - OK

Next file: spring-context-3.0.3.RELEASE.jar - OK

Next file: spring-core-3.0.3.RELEASE.jar - OK

Next file: spring-expression-3.0.3.RELEASE.jar - OK

Next file: spring-web-3.0.3.RELEASE.jar - OK

Next file: spring-webmvc-3.0.3.RELEASE.jar - OK

Next file: test.jar - OK

El programa guardará el archivo pom.xml generado dentro de la carpeta raíz del proyecto.

# pom.xml generado

El archivo pom.xml generado del ejemplo anterior tendrá el siguiente contenido con la información obtenida por el programa:

<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.dextratech.dtf</groupId>

<artifactId>project-web-spring-3</artifactId>

<packaging>war</packaging>

<version>1.0</version>

<properties>

<repo-dir>${project.basedir}/WebContent/WEB-INF/lib</repo-dir>

<src-dir>${project.basedir}/src</src-dir>

<webapp-dir>${project.basedir}/WebContent</webapp-dir>

<webinf-dir>${project.basedir}/WebContent/WEB-INF</webinf-dir>

</properties>

<dependencies>

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>1.1.1</version>

<scope>system</scope>

<systemPath>${repo-dir}/commons-logging-1.1.1.jar</systemPath>

</dependency>

<dependency>

<groupId>jstl</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

<scope>system</scope>

<systemPath>${repo-dir}/jstl-1.2.jar</systemPath>

</dependency>

<dependency>

<groupId>spring-asm</groupId>

<artifactId>spring-asm</artifactId>

<version>3.0.3.RELEASE</version>

<scope>system</scope>

<systemPath>${repo-dir}/spring-asm-3.0.3.RELEASE.jar</systemPath>

</dependency>

<dependency>

<groupId>spring-beans</groupId>

<artifactId>spring-beans</artifactId>

<version>3.0.3.RELEASE</version>

<scope>system</scope>

<systemPath>${repo-dir}/spring-beans-3.0.3.RELEASE.jar</systemPath>

</dependency>

<dependency>

<groupId>spring-context</groupId>

<artifactId>spring-context</artifactId>

<version>3.0.3.RELEASE</version>

<scope>system</scope>

<systemPath>${repo-dir}/spring-context-3.0.3.RELEASE.jar</systemPath>

</dependency>

<dependency>

<groupId>spring-core</groupId>

<artifactId>spring-core</artifactId>

<version>3.0.3.RELEASE</version>

<scope>system</scope>

<systemPath>${repo-dir}/spring-core-3.0.3.RELEASE.jar</systemPath>

</dependency>

<dependency>

<groupId>spring-expression</groupId>

<artifactId>spring-expression</artifactId>

<version>3.0.3.RELEASE</version>

<scope>system</scope>

<systemPath>${repo-dir}/spring-expression-3.0.3.RELEASE.jar</systemPath>

</dependency>

<dependency>

<groupId>spring-web</groupId>

<artifactId>spring-web</artifactId>

<version>3.0.3.RELEASE</version>

<scope>system</scope>

<systemPath>${repo-dir}/spring-web-3.0.3.RELEASE.jar</systemPath>

</dependency>

<dependency>

<groupId>spring-webmvc</groupId>

<artifactId>spring-webmvc</artifactId>

<version>3.0.3.RELEASE</version>

<scope>system</scope>

<systemPath>${repo-dir}/spring-webmvc-3.0.3.RELEASE.jar</systemPath>

</dependency>

<dependency>

<groupId>test</groupId>

<artifactId>test</artifactId>

<version>1</version>

<scope>system</scope>

<systemPath>${repo-dir}/test.jar</systemPath>

</dependency>

</dependencies>

<build>

<sourceDirectory>src</sourceDirectory>

<!--

<resources>

<resource>

<directory>files</directory>

</resource>

</resources>

-->

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-war-plugin</artifactId>

<version>2.2</version>

<configuration>

<warSourceDirectory>WebContent</warSourceDirectory>

</configuration>

</plugin>

<plugin>

<groupId>org.mortbay.jetty</groupId>

<artifactId>jetty-maven-plugin</artifactId>

<configuration>

<contextPath>/</contextPath>

<webAppSourceDirectory>${webapp-dir}</webAppSourceDirectory>

<scanIntervalSeconds>3</scanIntervalSeconds>

<!--

<webDefaultXml>src/main/resources/jetty-webdefault.xml</webDefaultXml>

-->

<scanTargetPatterns>

<scanTargetPattern>

<directory>${webinf-dir}</directory>

<excludes>

<exclude>\*\*/\*.jsp</exclude>

</excludes>

<includes>

<include>\*\*/\*.properties</include>

<include>\*\*/\*.xml</include>

</includes>

</scanTargetPattern>

</scanTargetPatterns>

</configuration>

</plugin>

<plugin>

<groupId>com.dextratech.dtf</groupId>

<artifactId>dtf-generator-maven-plugin</artifactId>

<version>1.0-SNAPSHOT</version>

<configuration>

<verbose>true</verbose>

<htmlFile>.html</htmlFile>

<finalName>.html</finalName>

<baseUrl>http://localhost/</baseUrl>

<timeout>6000</timeout>

<testSuitesBasePackage>com.dextratech.dtf</testSuitesBasePackage>

</configuration>

</plugin>

</plugins>

</build>

</project>