



FACULTAD DE
INFORMÁTICA

Grado en Ingeniería Informática
Tecnologías de Desarrollo de Software

Gestión de proyectos con Maven

UNIVERSIDAD DE
MURCIA



Departamento DIS
Facultad de Informática
Universidad de Murcia

Introducción

- Proceso de construcción de un **proyecto** implica:
 - Crear una estructura de directorios
 - Realizar tareas como compilar, ejecutar tests, definir dependencias, crear ejecutables, generar documentación, empaquetar y desplegar.
- Herramientas *build* (*build automation tools*) facilitan la construcción automatizando algunas tareas.
- Ejemplos de herramientas: make, ant, **maven**, **gradle**.
- Maven es la más utilizada en la actualidad, pero aumenta el uso de Gradle (elegida por Google para Android).
 - Según Snyk, Maven 78% y Gradle 38%



Max Dimanov

Jun 15 · 4 min read ·  Listen



Detailed Comparison of Java Build Automation Tools: Maven vs Gradle



Gradle vs Maven Comparison

The following is a summary of the major differences between Gradle and Apache Maven: flexibility, performance, user experience, and dependency management. It is not meant to be exhaustive, but you can check the [Gradle feature list](#) and [Gradle vs Maven performance comparison](#) to learn more.

```
> gradle build --scan
```

```
> mvn clean install
```

[DZone](#) > [Java Zone](#) > [Gradle vs. Maven](#)

Gradle vs. Maven

When it comes to build automation tools, Gradle and Maven are the two heavy hitters. See each one's strengths and weaknesses and learn when to use each.



by [Angela Stringfellow](#)  · Jun. 30, 17 · [Java Zone](#) · Tutorial

 Like (95)

 Comment (55)

 Save

 Tweet

 346.66K Views

Ejemplo de Apache Ant

- Usada para proyectos Java en los primeros años de este siglo.
- Descripción de tareas en XML.

Proyecto

```
<project name="Mi-Proyecto" default="dist" basedir=". ">
```

```
<description>
```

```
    ejemplo sencillo de fichero build
```

```
</description>
```

```
<!-- propiedades globales para este build-->
```

```
<property name="src" location="src/main/java"/>
```

Propiedades

```
<property name="build" location="target/classes"/>
```

```
<property name="dist" location="target"/>
```

```
<target name="init">
```

Targets

```
<!-- Crear time stamp -->
```

```
<tstamp/>
```

Tareas

```
<!-- Crear estructura del directorio build usada al compilar-->
```

```
<mkdir dir="${build}"/>
```

```
</target>
```

Ejemplo de Apache Ant

- Usada para proyectos Java en los primeros años de este siglo.
- Descripción de tareas en XML.

...

```
<target name="compile" depends="init"
```

Targets

```
    description="compilar el fuente" >
```

Tareas

```
    <!-- Compilar el código Java de ${src} en ${build} -->
```

```
        <javac srcdir="${src}" destdir="${build}"/>
```

```
</target>
```

```
<target name="dist" depends="compile"
```

```
    description="generar la distribución" >
```

```
    <!-- Crear el directorio de distribución-->
```

```
        <mkdir dir="${dist}/lib"/>
```

```
    <!-- Colocar todo lo de ${build} en el fichero
```

```
        MiProyecto${DSTAMP}.jar -->
```

```
        <jar jarfile="${dist}/lib/MyProject-${DSTAMP}.jar"
```

```
        basedir= "${build}"/>
```

```
</target>
```

```
</project>
```

Extensible Markup Language (XML)

- Lenguaje de marcado (*markup language*).
- Datos **semi-estructurados**: esquema está implícito en los datos.
- Estructura **jerárquica**.
- Muy utilizado en la Web y para configuración de herramientas y sistemas.
- Útil para **serialización**: representar todo tipo de datos en un formato estándar.
- Representación de **metadatos**.

```
<?xml version="1.0" encoding="UTF-8"?>
<libro>
  <titulo> El Principito </titulo>
  <autor>
    <nombre>Antoine de Saint-Exupéry </nombre>
    <fecha_nacimiento> 29/06/1900 </fecha_nacimiento>
    <país>Francia</país>
  </autor>
  <fecha_publicacion> 1943 </fecha_publicacion>
  <editorial> Reynal & Hitchcock </editorial>
  <idioma> Francés </idioma>
  <paginas numero="93"/>
</libro>
```

Maven: Características

- Framework para **automatizar la gestión de proyectos**.
 - Extensible vía plugins
 - Basado en XML
- **Estructura predefinida** para los proyectos y flujos de tareas
 - «*Convención sobre configuración*»
- **Mecanismo de dependencias potente**
 - Dependencias son descargadas automáticamente de un repositorio local o del repositorio Central Maven.
- Basada en el concepto de “**ciclo de vida**” en el que se realizan tareas
- Genera muchos **tipos de proyectos**: jar, war, ..
- Genera **documentación** e informes sobre el proyecto, y el sitio web.
 - Javadoc, lista de dependencias, listas de mailing, informes de los tests, ..
- Otros:
 - Integración simple de **JUnit** para ejecutar pruebas unitarias
 - Se puede usar desde **consola** o **IDEs** como Eclipse

Maven for building Java applications - Tutorial

📖 Lars Vogel, Simon Scholz (c) 2013 - 2020 vogella GmbH – Version 2.3, 97.09.2020

This tutorial describes the usage of Maven for building Java applications.



Servidores VPS

Incluye procesadores Intel® Xeon® E5 y almacenamiento 100% SSD.

ionos.es

1. What is Apache Maven?



[Welcome](#)

[License](#)

ABOUT MAVEN

[What is Maven?](#)

[Features](#)

[Download](#)

[Use](#)



[Release Notes](#)

DOCUMENTATION

[Maven Plugins](#)

Documentation

Getting Started with Maven

- [Getting Started in 5 Minutes](#)
- [Getting Started in 30 Minutes](#)

Introductions

- [The Build Lifecycle](#)
- [The POM](#)
- [Profiles](#)
- [Repositories](#)
- [Standard Directory Layout](#)

POM: *Project Object Model*

- **Modelo conceptual de un proyecto**
 - **Fichero XML** (pom.xml en directorio raíz del proyecto) que contiene la información (metadatos) que configura al proyecto.
 - Nombre, directorios, dependencias, plugins, repositorios, informes, etc.
 - **Se genera automáticamente** y se puede modificar.
- Ejemplo de **POM** simple:

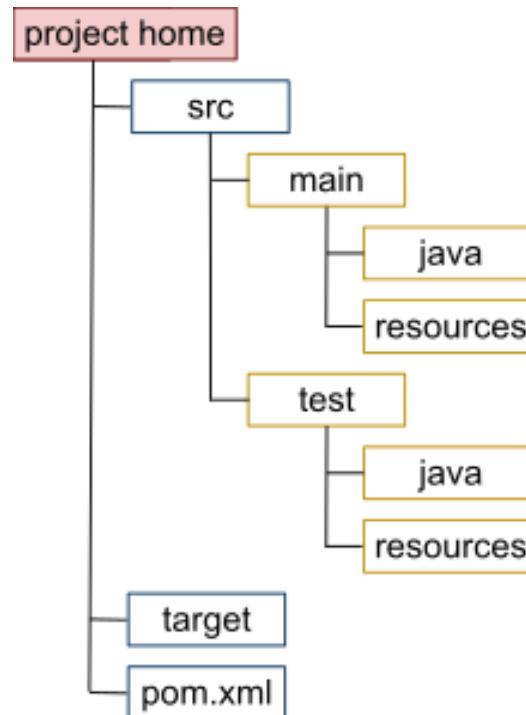
```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>umu.tds</groupId>
  <artifactId>MavenTestProject</artifactId>
  <name>Proyecto de prueba para Maven</name>
  <version>1.0</version>
  <packaging>jar</packaging>
</project>
```

POM para ejemplo de tutorial

```
<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>tds.maven.apps</groupId>
  <artifactId>ProyectoMaven</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name><ProyectoMaven</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>
</project>
```

Estructura de un proyecto

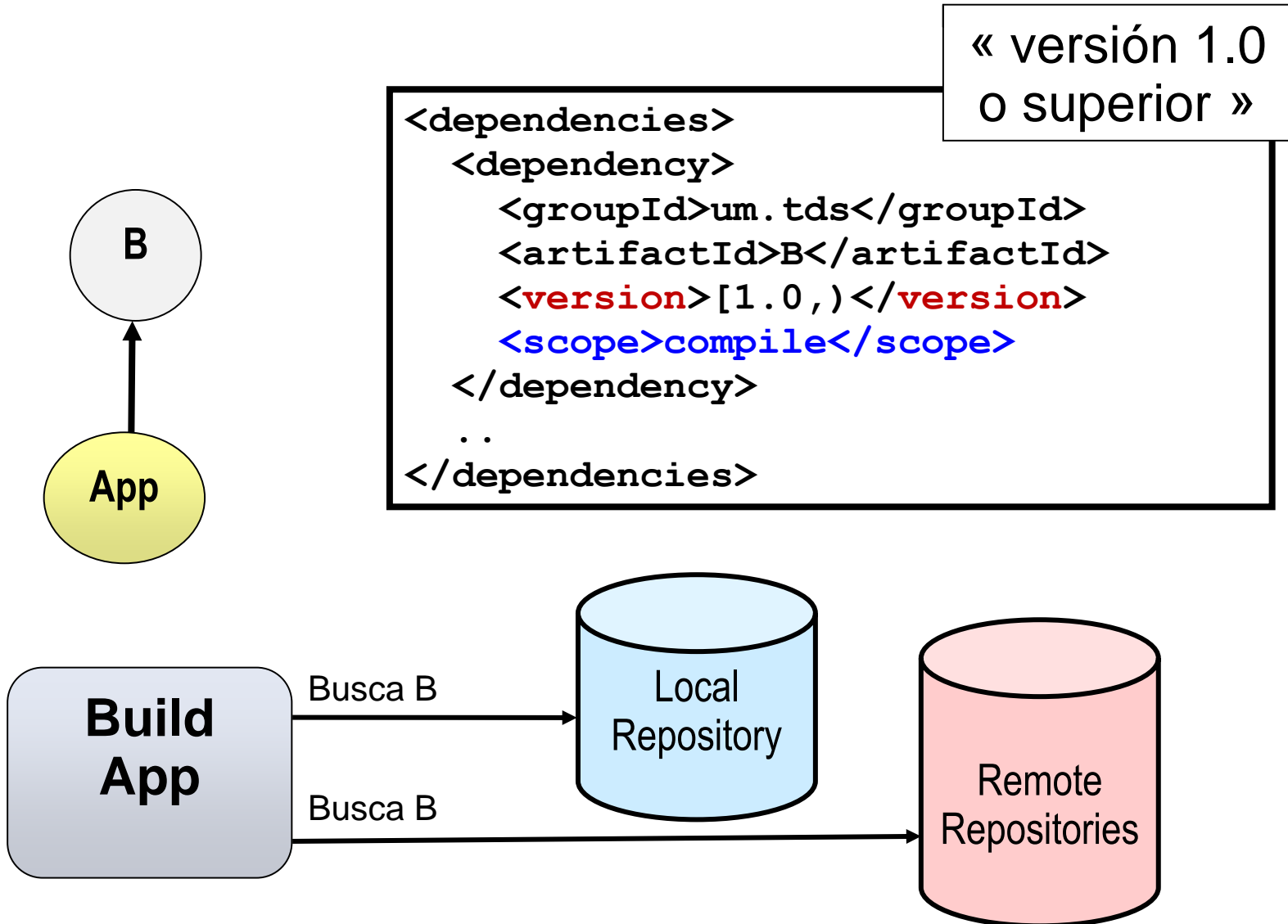
- Considera una estructura de directorios por defecto para todos los elementos software que componen un proyecto (puede ser modificada en el pom.xml).
- Directorio raíz con carpetas **src** (fuentes) y **target** (compilados, ejecutable, javadoc, informes, ..)



Dependencias

- Artefactos software (librerías, componentes, frameworks, etc.) requeridos en un proyecto, por ejemplo, el **Servicio de Persistencia**, **JUnit** y **JCalendar** en AppMusic.
- Los artefactos no se añaden al proyecto, sino que se **referencian** de un repositorio central o local y Maven obtiene automáticamente las dependencias.
 - Manejo automático de la **transitividad** entre dependencias.
- Elementos que describen las dependencias:
 - groupId: identifica a la organización
 - artifactId: identifica al componente
 - version: versión actual
 - scope: ámbito en el que se usa (compile, test,..)
 - ...

Dependencias



Plugins



- Existe un conjunto de **goals** asociados al ciclo de vida, que son implementados por **plugins** que forman el núcleo de Maven: “todo el trabajo se consigue mediante ejecución de plugins”. Se clasifican en categorías:
 - **Core**: clean, compiler, deploy, install, surefire, site,...
 - **Report**: javadoc, pmd, checkstyle,...
 - **Packaging types/tools**: ear, jar, war, rar, source,...
 - **Tools**: artifact, archetype, dependency, plugin,...
- Ejecutar un goal de un plugin en consola:
 - **mvn [nombre plugin]:[nombre goal]**
 - Ejemplos: **mvn compile**, **mvn install**
- Existen un gran número de plugins que se pueden instalar manualmente (plugin archetype) u obtener simplemente declarando su uso en el pom.xml.

Welcome

License

ABOUT MAVEN

What is Maven?

Features

Download

Use

Release Notes

DOCUMENTATION

Maven Plugins

Maven Extensions

Index (category)

User Centre

Plugin Developer Centre

Maven Repository Centre

Maven Developer Centre

Books and Resources

Security

COMMUNITY

Community Overview

Project Roles

How to Contribute

Getting Help

Issue Management

Getting Maven Source

The Maven Team

PROJECT DOCUMENTATION

Project Information

MAVEN PROJECTS

Archetype

Artifact Resolver

Doxia

Extensions

JXR

Maven

Parent POMs

Plugins

Plugin Testing

Plugin Tools

Available Plugins

Maven is - at its heart - a plugin execution framework; all work is done by plugins. Looking for a specific goal to execute? This page lists the core plugins and others. There are the build and the reporting plugins:

- Build plugins** will be executed during the build and they should be configured in the `<build/>` element from the POM.
- Reporting plugins** will be executed during the site generation and they should be configured in the `<reporting/>` element from the POM. Because the result of a Reporting plugin is part of the generated site, Reporting plugins should be both internationalized and localized. You can read more about the [localization of our plugins](#) and how you can help.

Supported By The Maven Project

To see the most up-to-date list browse the Maven repository, specifically the `org/apache/maven/plugins` subfolder. (Plugins are organized according to a directory structure that resembles the standard Java package naming convention)

Plugin	Type*	Version	Release Date	Description	Source Repository	Issue Tracking
Core plugins				Plugins corresponding to default core phases (ie. clean, compile). They may have multiple goals as well.		
clean	B	3.2.0	2022-04-01	Clean up after the build.	Git / GitHub	Jira MCLEAN
compiler	B	3.10.1	2022-03-11	Compiles Java sources.	Git / GitHub	Jira MCOMPILER
deploy	B	3.0.0	2022-07-16	Deploy the built artifact to the remote repository.	Git / GitHub	Jira MDEPLOY
failsafe	B	3.0.0-M7	2022-06-07	Run the JUnit integration tests in an isolated classloader.	Git / GitHub	Jira SUREFIRE
install	B	3.0.1	2022-07-17	Install the built artifact into the local repository.	Git / GitHub	Jira MINSTALL
resources	B	3.3.0	2022-07-23	Copy the resources to the output directory for including in the JAR.	Git / GitHub	Jira MRESOURCES
site	B	4.0.0-M3	2022-07-25	Generate a site for the current project.	Git / GitHub	Jira MSITE
surefire	B	3.0.0-M7	2022-06-07	Run the JUnit unit tests in an isolated classloader.	Git / GitHub	Jira SUREFIRE
verifier	B	1.1	2015-04-14	Useful for integration tests - verifies the existence of certain conditions.	Git / GitHub	Jira MVERIFIER
Packaging types/tools				These plugins relate to packaging respective artifact types.		
ear	B	3.3.0	2022-10-18	Generate an EAR from the current project.	Git / GitHub	Jira MEAR
ejb	B	3.2.1	2022-04-18	Build an EJB (and optional client) from the current project.	Git / GitHub	Jira MEJB
jar	B	3.3.0	2022-09-12	Build a JAR from the current project.	Git / GitHub	Jira MJAR
rar	B	3.0.0	2022-07-17	Build a RAR from the current project.	Git / GitHub	Jira MRAR
war	B	3.3.2	2021-09-10	Build a WAR from the current project.	Git / GitHub	Jira MWAR
app-client/acr	B	3.1.0	2018-06-19	Build a JavaEE application client from the current project.	Git / GitHub	Jira MACR
shade	B	3.4.1	2022-10-27	Build an Uber-JAR from the current project, including dependencies.	Git / GitHub	Jira MSHADE
source	B	3.2.1	2019-12-21	Build a source-JAR from the current project.	Git / GitHub	Jira MSOURCES

Archetypes (Arquetipos)

- Define el **prototipo o plantilla de proyecto** que se quiere crear.
 - Determina una estructura de proyecto y POM
 - Ficheros XML que son la entrada a plantillas de generación de código.
- Ejemplos:

maven-archetype-quickstart (básico)

maven-archetype-j2ee-simple

maven-archetype-webapp

- Orden en consola:

```
mvn archetype:generate -DgroupId=umu.tds
```

```
-DartifactId=proyectoJ2eeTDS
```

```
-DarchetypeArtifactId=maven-archetype-j2ee-simple
```

Ciclos de vida. Fases de Build

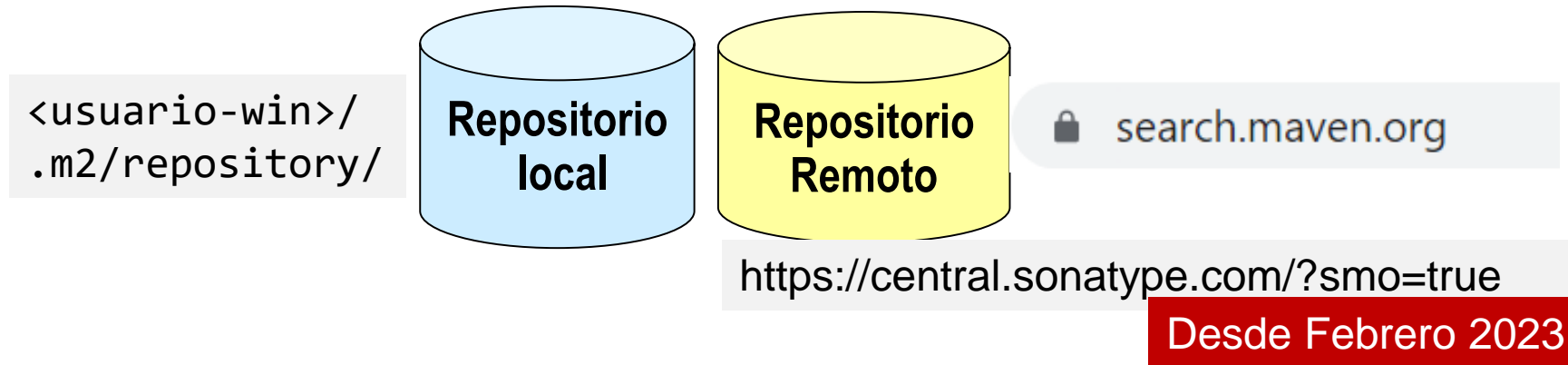
- Basado en el concepto de **ciclo de vida**: *build, clean y site*.
 - Cada ciclo de vida consta de varias fases.

Fases de Build

- **validate**: valida que el proyecto esté correcto y tiene toda la información necesaria para su construcción.
- **compile**: compila el código fuente del proyecto.
- **test**: lanza las pruebas unitarias
- **package**: toma las clases compiladas y recursos y crea un paquete con el proyecto (jar, war, ear)
- **integration-test**: ejecuta otros tests que requieren el paquete
- **verify**: realiza algún tipo de chequeo para comprobar si el paquete cumple unas normas de calidad.
- **install**: instala el paquete en el repositorio local para ser usado como dependencia por otros proyectos localmente.
- **deploy** : copia el paquete a un repositorio remoto para ser compartido con otros usuarios y proyectos.

Repositorios de artefactos


- Almacenan todo tipo de artefactos (incluido nuestro proyecto) en una estructura jerárquica (organizada por groupId o paquete): JAR, EAR, WAR, EJBs, ZIPs, plugins, etc.
 - Ejemplo: `<usuario-win>/ .m2/repository/com/toedter/jcalendar/1.4/`
- Dependencias se buscan primero en el repositorio local (por lo general `.m2/repository` en el directorio de usuario). Si no se encuentran allí, se buscan en el repositorio remoto (Maven central por defecto, no debe ser indicado en POM, salvo si se cambia URL)



https://central.sonatype.com/?smo=true

Find OSS Components

As stewards of Central for nearly 20 years and inventors of both software supply chain management and Nexus Repository, Sonatype knows that the integrity of your build is critical.

 jcalendar



[Advanced Options](#)

Curious about what else we offer?

Check out our free and paid tools that developers and security pros love!




<https://central.sonatype.com/?smo=true>

Find OSS Components

As stewards of Central for nearly 20 years and inventors of both software supply chain management and Nexus Repository, Sonatype knows that the integrity of your build is critical.




jcalendar 1.4 published 10 years ago in  **com.toedter**


JCalendar is a Java date chooser bean for graphically picking a date. JCalendar is composed of several other Java beans, a JDayChooser, a JMon...

jcalendarbutton 1.4.6 published 12 years ago in  **org.jbundle.thin.base.screen**


JCalendarButton is a simple Java Swing component that displays a popup calendar next to a date input field.

jcalendar-tz 1.3.3-4 published 13 years ago in  **com.luuuis**


A fork of the JCalendar project, with TimeZone support

jbundle-util-jcalendarbutton-repository 1.6.8 published 6 months ago in  **org.jbundle.config.repo**


jbundle-util-jcalendarbutton repository

org.jbundle.util.jcalendarbutton 1.6.8 published 6 months ago in  **org.jbundle.util.jcalendarbutton**

JCalendarButton is a simple Java Swing component that displays a popup calendar next to a date input field.

jbundle-util-jcalendarbutton-reactor 1.6.8 published 6 months ago in  **org.jbundle.util.jcalendarbutton**

Top level properties and features

jcalendarbutton 1.4.4 published 13 years ago in  **net.sourceforge.jcalendarbutton**

JCalendarButton is a simple Java Swing component that displays a popup calendar next to a date input field.

jcalendar

1.4

Used in 105 components

pkg:maven/com.toedter/jcalendar@1.4



Overview

Versions

Dependents

Dependencies

Overview

Description

JCalendar is a Java date chooser bean for graphically picking a date. JCalendar is composed of several other Java beans, a JDayChooser, a JMonthChooser and a JYearChooser. All these beans have a locale property, provide several icons (Color 16x16, Color 32x32, Mono 16x16 and Mono 32x32) and their own locale property editor. So they can easily be used in GUI builders. Also part of the package is a JDateChooser, a bean composed of an IDateEditor (for direct date editing) and a button for opening a JCalendar for selecting the date.

Snippets

```
<dependency>
  <groupId>com.toedter</groupId>
  <artifactId>jcalendar</artifactId>
  <version>1.4</version>
</dependency>
```

Apache Maven

Apache Maven

Gradle

Gradle (short)

Gradle (Kotlin)

sbt

ivy

grape

leiningen

buildr

Copy to clipboard

Using Maven within the Eclipse IDE - Tutorial

 Lars Vogel (c) 2016 - 2022 vogella GmbH – Version 1.6, 01.08.2021

This tutorial describes the usage of Maven within the Eclipse IDE for building Java applications.

1. Using Maven with the Eclipse IDE

The Eclipse IDE provides support for the Maven build. This support is developed in the *M2Eclipse* project.

It provides an editor for modifying the pom file and downloads dependencies if required. It also manages the classpath of the projects in the IDE. You can also use wizards to import existing Maven projects and to create new Maven projects.

2. Installation and configuration of Maven for Eclipse

2.1. Installation of the Maven tooling for the Eclipse IDE

Most Eclipse IDE downloads already include support for the Maven build system. To check, use **Help > About** and check if you can see the Maven logo (with the **M2E** sign).



If Maven support is not yet installed, the following description can be used to install it.

► [Installation of Maven support into the Eclipse IDE](#)

LoginRegistro

src

umu.tds

umu.tds.controlador

umu.tds.dao

umu.tds.dominio

RepositorioUsuario

Usuario.java

umu.tds.gui

JRE System Library [J2SE]

Referenced Libraries

Maven Dependencies

bin

lib

target

pom.xml

Dependencies

Dependencies

driverPersistencia : 2.0



Add...

Remove

Properties...

Manage...

To manage your transitive dependency exclusions, please use the [Dependency Hierarchy](#) page.

Overview Dependencies Dependency Hierarchy Effective POM pom.xml

```
1<project xmlns="http://maven.apache.org/POM/4.0.0"
2  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/
4
5  <modelVersion>4.0.0</modelVersion>
6  <groupId>umu.tds</groupId>
7  <artifactId>LoginRegistro</artifactId>
8  <version>0.0.1-SNAPSHOT</version>
9
10 <dependencies>
11   <dependency>
12     <groupId>umu.tds</groupId>
13     <artifactId>driverPersistencia</artifactId>
14     <version>2.0</version>
15   </dependency>
16 </dependencies>
17 </project>
```

Package Explorer ×

- instagram
- LoginRegistro
 - src
 - umu.tds
 - umu.tds.controlador
 - umu.tds.dao
 - umu.tds.dominio
 - RepositorioUsuarios.java
 - Usuario.java
 - umu.tds.gui

1 VentanaPrincipal

2 Login

3 Calculator

4 EjemploBotonesGBL

5 GridBagLayoutDemo

Run As

Run Configurations...

Organize Favorites...

Prin... PanelItemFot... PanelNuevaFo...

encies

encies

1 Maven build Alt+Shift+X, M

2 Maven build...

3 Maven clean

4 Maven generate-sources

5 Maven install

6 Maven test

Maven desde consola

1. Descargar maven (fichero Binary.zip) y descomprimir en C:\Program Files
2. Se puede crear un directorio para repositorio local o mantener el repositorio por defecto en .m2. En Windows:
usuarios/<usuario>/ .m2/repository
3. Crear y ejecutar un archivo .bat con el siguiente formato (se supone que es la version 3.6.3 de maven):

```
SET M2_HOME=C:\Program Files\apache-maven-3.6.3
SET M2=C:\Program Files\apache-maven-3.6.3\bin
SET PATH=%M2%;%JAVA_HOME%\bin;%PATH%
SET JAVA_HOME=C:\Program Files\Java\“Directorio con JDK”
```

4. Ejecutar `mvn -version`, debería aparecer un texto similar al siguiente

```
Apache Maven 3.6.3 (40f52333136460af0dc0d7232c0dc0bcf0d9e117; 2019-08-27T17:06:16+02:00)
Maven home: C:\Program Files\apache-maven-3.6.3\bin\..
Java version: 1.8.0_172, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk1.8.0_172\jre
Default locale: es_ES, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

Maven desde consola

5. Crear un proyecto maven

```
mvn archetype:generate -DgroupId=umu.tds.apps  
-DartifactId=AppGrupo32  
-DarchetypeArtifactId=maven-archetype-quickstart
```

Responder pulsando tecla "Enter" cada vez que se detenga la ejecución para pedir una entrada al usuario. Se instalarán plugins en el repositorio y se creará el directorio \MavenTestProject en el directorio actual con la carpeta src y el archivo pom.xml

6. Ejecutar **mvn compile** en el directorio \MavenTestProject que contiene el archivo pom.xml. Se crea la carpeta target con el código compilado.

7. Se pueden ir ejecutando comandos Maven: test, install, package,...