Trabajando con el archivos JAR y el archivo Manifest en Java

**Temario**

[El archivo MANIFEST.MF y el archivo POM.XML 1](#__RefHeading___Toc129_2494063844)

[Ubicación de los archivos MANIFEST.MF y POM.XML 1](#__RefHeading___Toc131_2494063844)

[Crear un proyecto NetBeans Maven 1](#__RefHeading___Toc133_2494063844)

[Paso 1: Crear una estructura Maven para un proyecto Java. 2](#__RefHeading___Toc135_2494063844)

[Paso 2: Crear la clase que contiene al método main. 4](#__RefHeading___Toc137_2494063844)

[Paso 3: Crear el archivo POM.XML 6](#__RefHeading___Toc139_2494063844)

[Ejecutar el programa desde el IDE 7](#__RefHeading___Toc157_2494063844)

[Paso 4: Crear el archivo JAR y el archivo MANIFEST.MF por defecto. 8](#__RefHeading___Toc141_2494063844)

[Paso 5: Ejecutar el archivo JAR. 11](#__RefHeading___Toc155_2494063844)

[El archivo manifiesto por defecto 11](#__RefHeading___Toc143_2494063844)

[Archivo de manifiesto personalizado 12](#__RefHeading___Toc145_2494063844)

[¿Qué tener en cuenta al manejar el archivo de Manifiesto personalizado? 12](#__RefHeading___Toc147_2494063844)

[Construir el archivo JAR con Maven o Ant 13](#__RefHeading___Toc149_2494063844)

[Configura la clase que tiene el método main en el IDE 15](#__RefHeading___Toc151_2494063844)

# El archivo MANIFEST.MF y el archivo POM.XML

El archivo MANIFEST.MF y el archivo POM.XML están relacionados

## Ubicación de los archivos MANIFEST.MF y POM.XML

* El archivo MANIFEST.MF en .../raizProyectoMaven/META-INF/MANIFEST.MF
* El archivo POM.XML en .../raizProyectoMaven/POM.XML

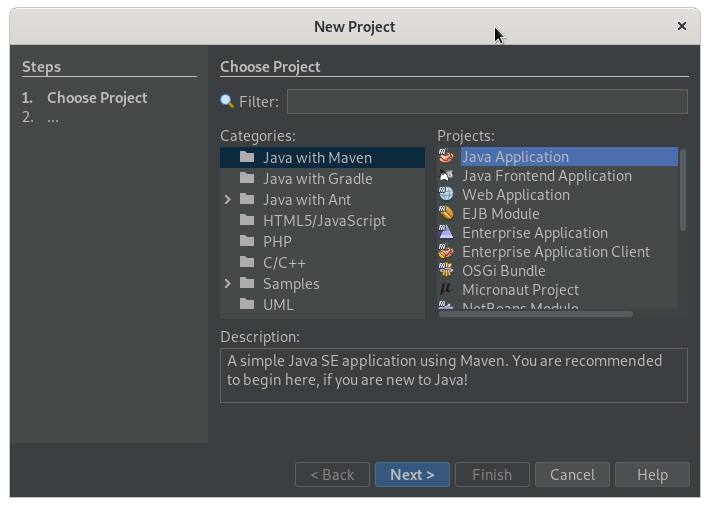
Cada vez que un desarrollador quiere distribuir una versión de su software, todo lo que quiere es distribuir un solo archivo y no una estructura de directorio llena de archivos de clases. Los archivos JAR fueron diseñados para este propósito. Un archivo JAR puede contener archivos de clases y otros tipos de archivos, como archivos de sonido e imagen, que pueden incluirse en el proyecto. Todos los archivos incluidos en un archivo JAR se comprimen con un formato similar a zip o rar.

## Crear un proyecto NetBeans Maven

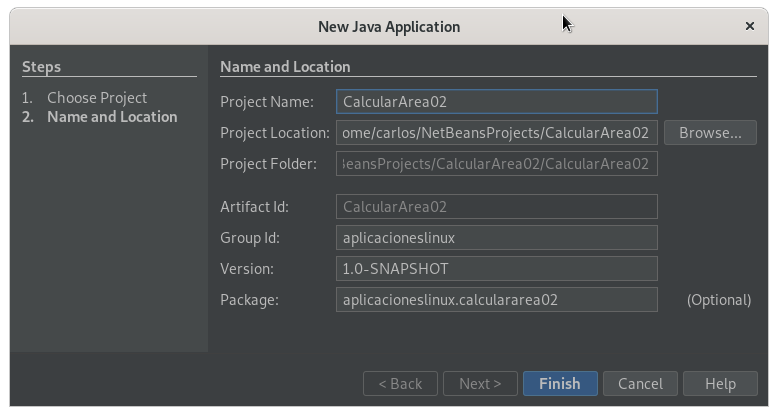
El siguiente tutor paso a paso en NetBeans tiene como objetivo generar un proyecto Java y los archivos correctos y básicos para todo proyecto Java.

* Paso 1: Crear una estructura Maven para un proyecto Java.
* Paso 2: Crear la clase que contiene al método main.
* Paso 3: Crear el archivo POM.XML con la configuración básica para construir el archivo JAR.
* Paso 4: Crear el archivo JAR y el archivo MANIFEST.MF por defecto.
* Paso 5: Ejecutar el archivo JAR.

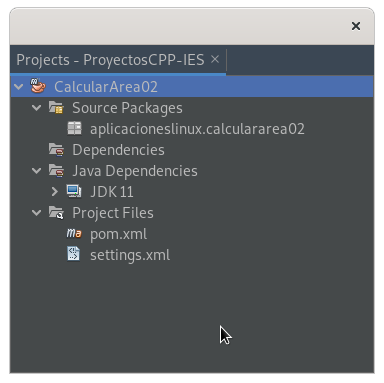
### Paso 1: Crear una estructura Maven para un proyecto Java.



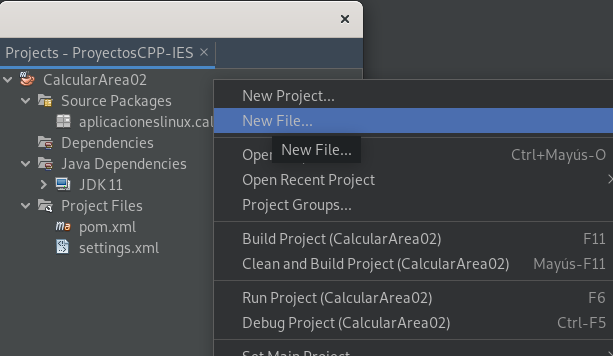
Configurar el proyecto Maven



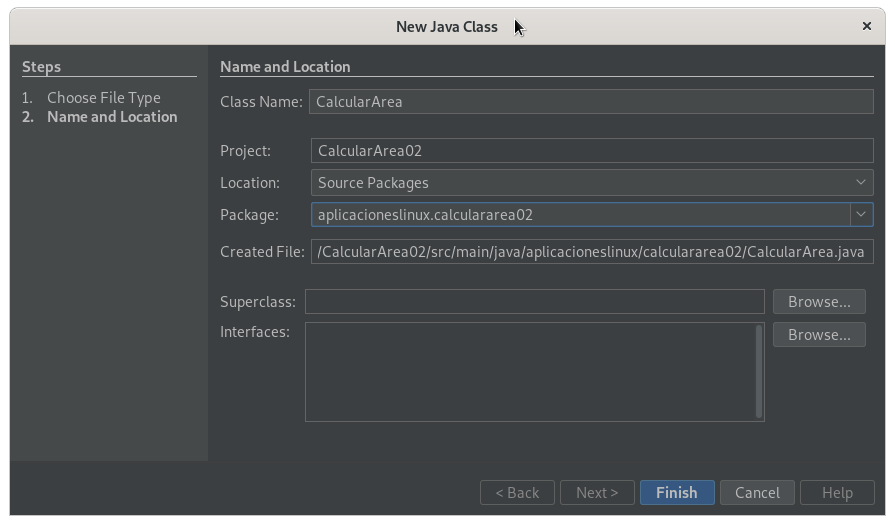
Tarea realizada



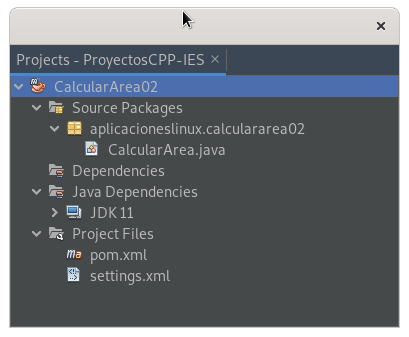
### Paso 2: Crear la clase que contiene al método main.



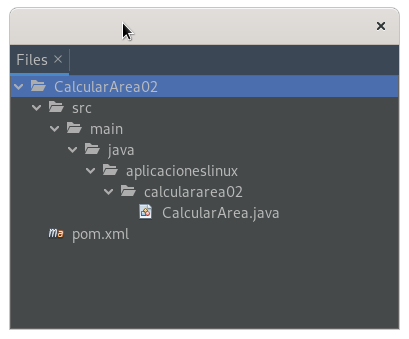
Colocar la nueva clase en el directorio, paquete, definido por Maven.



Tarea realizada en la pestaña Proyects



Tarea realizada en la pestaña Files



Escribir el siguiente código en la clase Calcular Area

package aplicacioneslinux.calculararea02;

public class CalcularArea {

public static void main(String[] args) {

int base = 6;

int altura = 3;

System.out.println("Área = " + (base \* altura));

}

}

### Paso 3: Crear el archivo POM.XML

Escribir el siguiente código en el archivo POM.XML

<build>

<plugins>

<plugin>

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

<artifactId>maven-jar-plugin</artifactId>

<version>3.2.0</version>

<configuration>

<archive>

<manifest>

<addClasspath>true</addClasspath>

<mainClass>aplicacioneslinux.calculararea02.CalcularArea</mainClass>

</manifest>

</archive>

</configuration>

</plugin>

</plugins>

</build>

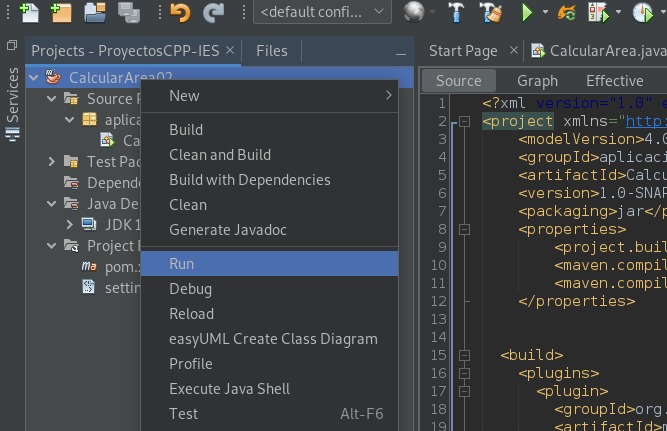
En el ejemplo, se utiliza el complemento maven-jar-plugin y se configura el elemento <archive> dentro del archivo pom.xml.

* <addClasspath>true</addClasspath>: Incluye las rutas de clase del proyecto en el atributo Class-Path del archivo de manifiesto. Esto permite que las dependencias del proyecto sean ubicadas automáticamente durante la ejecución.
* <mainClass>aplicacioneslinux.calculararea02.CalcularArea>: Especifica la clase principal que contiene el método main() de la aplicación. Asegúrate de reemplazar com.example.MainClass con la ruta completa de tu clase principal.

Puedes agregar más configuraciones dentro del elemento <manifest> para personalizar aún más el archivo de manifiesto, como atributos personalizados, permisos de seguridad, rutas de clase adicionales, dependencia, entre otros.

Recuerda que necesitarás ajustar la configuración según la estructura y necesidades de proyecto.

#### Ejecutar el programa desde el IDE



Salida después de ejecutar el programa

[INFO]

[INFO] --- exec-maven-plugin:3.0.0:exec (default-cli) @ CalcularArea02 ---

Área = 18

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

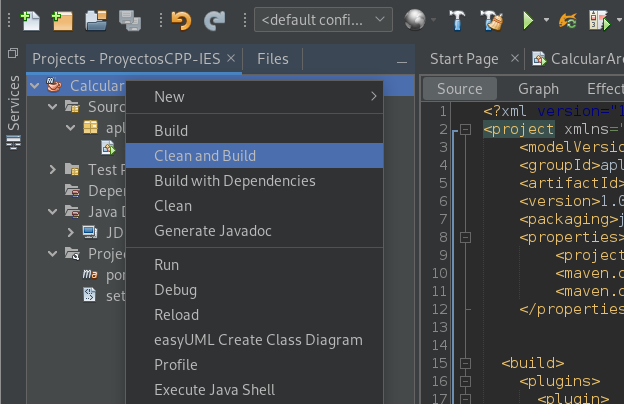
[INFO] ------------------------------------------------------------------------

[INFO] Total time: 1.167 s

[INFO] Finished at: 2023-05-28T21:02:35-03:00

[INFO] ------------------------------------------------------------------------

### Paso 4: Crear el archivo JAR y el archivo MANIFEST.MF por defecto.



Salida después de construir el archivo JAR

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

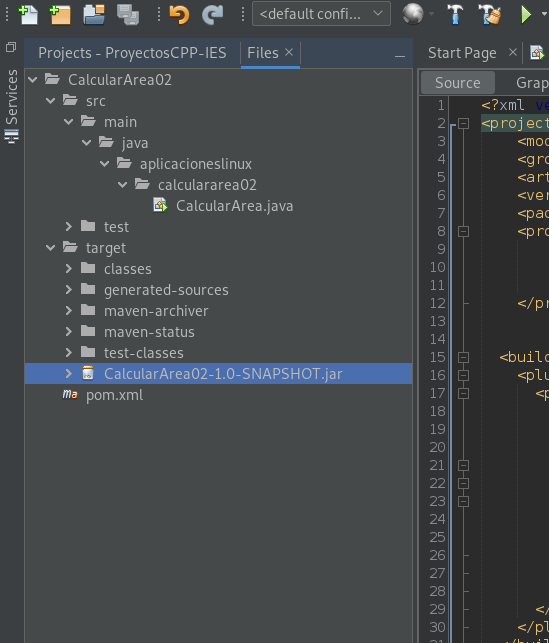
[INFO] ------------------------------------------------------------------------

[INFO] Total time: 1.286 s

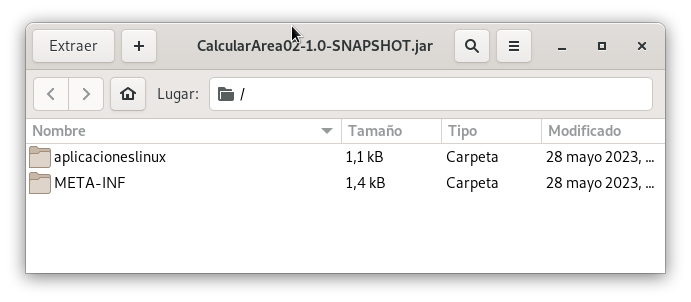
[INFO] Finished at: 2023-05-28T21:07:34-03:00

[INFO] ------------------------------------------------------------------------

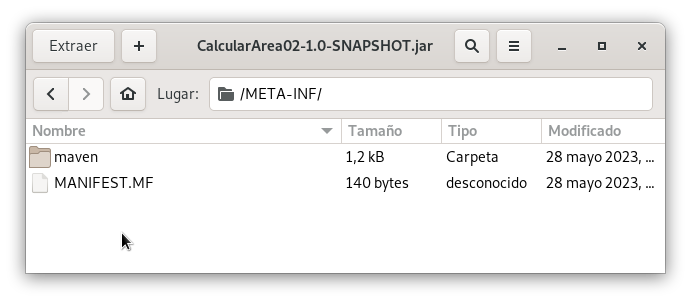
Verificar la tarea realizada



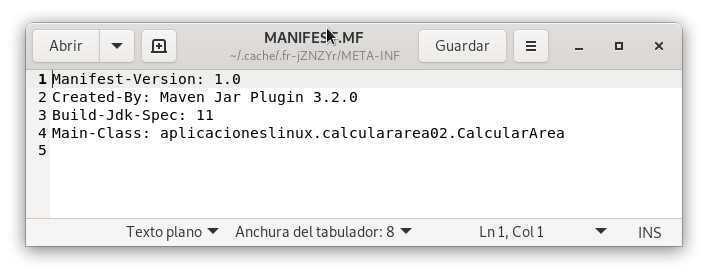
Verificar el contenido del archivo JAR



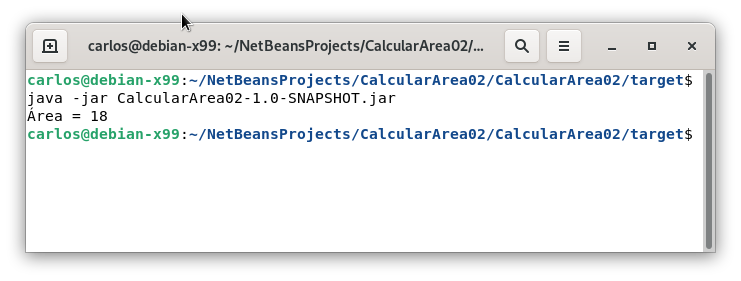
Verificar el archivo de manifiesto



Verificar el contenido del archivo de manifiesto



### Paso 5: Ejecutar el archivo JAR.



## El archivo manifiesto por defecto

Cada archivo JAR contiene un archivo de manifiesto que describe las características, dependencias y recursos que el archivo JAR necesita. El archivo de manifiesto por defecto contendrá es creado por la ejecución del archivo POM.XML. El archivo de manifiesto predeterminado se denomina MANIFEST.MF y está presente en el subdirectorio META-INF del archivo. Aunque el archivo de manifiesto predeterminado contiene solo dos entradas, los archivos de manifiesto complejos pueden tener muchas entradas referente a todos los recursos necesario para ejecutar el archivo JAR. Así es como se ve un archivo de manifiesto predeterminado:

Las entradas de los archivos de manifiesto se agrupan en secciones. Cada sección tiene dos entradas, el nombre de la sección y su valor.

## Archivo de manifiesto personalizado

Agregar un archivo de manifiesto personalizado (no recomendado, dejar que el archivo POM.XML administre las dependencias y recursos que el archivo JAR)

Para agregar un archivo de manifiesto hacer los siguientes pasos:

1. Crear la carpeta .../raizProyectoMaven/META-INF/
2. Crear archivo MANIFEST.MF en la carpeta META-INF

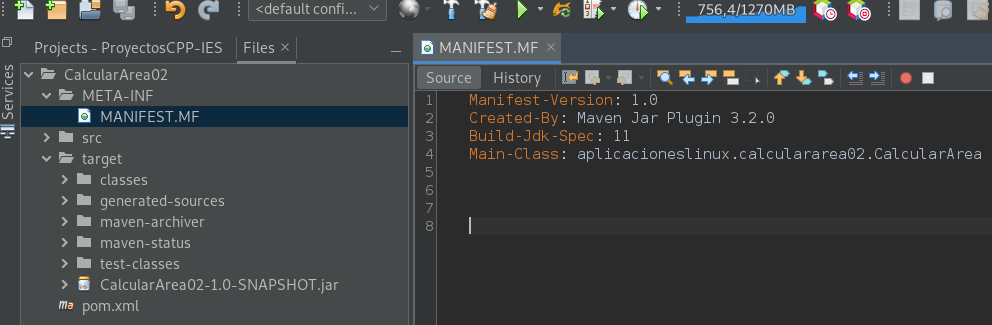
Agregar el siguiente contenido al archivo de manifiesto

Manifest-Version: 1.0

Created-By: Maven Jar Plugin 3.2.0

Build-Jdk-Spec: 11

Main-Class: aplicacioneslinux.calculararea02.CalcularArea



### ¿Qué tener en cuenta al manejar el archivo de Manifiesto personalizado?

Debe dejar espacio entre el nombre y el valor de cualquier sección en el archivo de manifiesto, como la Versión: 1.1 está en la sección válida en su lugar, escriba la Versión: 1.1, ese espacio entre dos puntos y 1.1 realmente importa mucho.

Al especificar la clase principal, no debe agregar la extensión .class al final del nombre de la clase. Simplemente especifique la clase principal escribiendo:

Main-Class: Classname

Debe agregar una nueva línea al final del archivo. No necesita escribir \n para especificar una nueva línea, sino que deje en blanco la última línea de su archivo de manifiesto que servirá para ese propósito.

El archivo de texto para el manifiesto debe usar la codificación UTF-8; de lo contrario, puede tener problemas.

Si se ha creado manualmente el archivo manifest.mf dentro del directorio META-INF en el proyecto Maven sin respetar la estructura de directorios MAven, es probable que se sobrescriba o no sea reconocido correctamente durante el proceso de construcción del proyecto. Esto se debe a que Maven utiliza su propia estructura y convenciones de directorios para generar los archivos de salida, como son el archivo JAR y .class.

Cuando configuras el complemento maven-jar-plugin en el archivo pom.xml, Maven se encarga de generar automáticamente el archivo manifest.mf en la ubicación adecuada dentro del archivo JAR resultante. El contenido y las configuraciones del archivo de manifiesto se definen en el archivo pom.xml, y Maven se encarga de combinar esa información con otros metadatos del proyecto.

Si necesitas agregar configuraciones adicionales al archivo de manifiesto, es recomendable hacerlo a través de la configuración del complemento maven-jar-plugin en el archivo pom.xml. Esto asegurará que el archivo de manifiesto se genere correctamente y que todas las dependencias y metadatos del proyecto se reflejen correctamente en él.

En resumen, es mejor evitar crear manualmente el archivo manifest.mf en el proyecto Maven y utilizar la configuración adecuada en el archivo pom.xml para generar y personalizar el archivo de manifiesto de manera controlada.

## Construir el archivo JAR con Maven o Ant

En NetBeans se puede elegir entre los tipos de proyectosMaven y Ant, ¿Cuál es mejo?

Tanto Maven como Ant son herramientas de construcción ampliamente utilizadas en el desarrollo de software en Java, y cada una tiene sus ventajas y características distintas. La elección entre Maven y Ant dependerá de tus necesidades específicas y preferencias. Una descripción general de ambas herramientas:

Maven: es una herramienta de construcción y gestión de proyectos que se centra en la configuración por convención. Utiliza un enfoque declarativo basado en XML, lo que significa que configuras tu proyecto mediante un archivo pom.xml. Proporciona una estructura de directorios estandarizada y un conjunto de convenciones para organizar tu proyecto. Ofrece una gestión avanzada de dependencias y la capacidad de descargar automáticamente las bibliotecas necesarias desde repositorios remotos. Permite la ejecución de comandos para realizar tareas como compilar, empaquetar, probar y desplegar tu proyecto.

Ant: es una herramienta de automatización de construcción de software que se utiliza principalmente en proyectos de desarrollo de software. El nombre "Ant" es un acrónimo de "Another Neat Tool" (Otra Herramienta Genial, en español).

Ant se basa en XML (eXtensible Markup Language) para definir tareas y configuraciones de construcción. Proporciona un conjunto de tareas predefinidas que permiten compilar código fuente, crear archivos JAR, ejecutar pruebas, generar documentación y realizar otras actividades comunes en el proceso de desarrollo de software.

Una de las principales características de Ant es su capacidad para definir fácilmente flujos de construcción personalizados y automatizados. Los desarrolladores pueden crear archivos de construcción (llamados archivos build.xml) que contienen instrucciones y configuraciones para llevar a cabo diferentes tareas de construcción de manera secuencial o en paralelo. Esto permite automatizar el proceso de construcción y facilita la repetición de tareas de construcción complejas.

Ant es una herramienta independiente de plataforma y se puede utilizar en diferentes sistemas operativos. Es ampliamente utilizado en proyectos Java, pero también se puede utilizar para construir proyectos en otros lenguajes de programación.

Ant es una herramienta de construcción basada en XML que se centra en la ejecución de tareas personalizadas y secuencias de comandos. Proporciona una flexibilidad y personalización extensas al permitirte definir tus propias tareas y secuencias de comandos en XML. No impone una estructura de directorios específica y no tiene una configuración por convención como Maven. Requiere una configuración manual y detallada, ya que debes especificar explícitamente las tareas y las dependencias de tu proyecto. Es adecuado para proyectos que requieren un alto grado de personalización y control en el proceso de construcción.

Resumen: Si se prefiere una configuración por convención, una gestión de dependencias más automatizada y una estructura de directorios estandarizada, Maven puede ser una buena opción. Si necesitas un mayor nivel de personalización y control en la construcción de tu proyecto, y estás dispuesto a configurar manualmente las tareas y las dependencias, Ant podría ser más adecuado.

Ambas herramientas son populares y ampliamente utilizadas en la comunidad de desarrollo de Java, por lo que puedes elegir la que mejor se ajuste a tus necesidades y con la que te sientas más cómodo trabajando.

Maven y Ant generan el archivo de manifiesto?

Sí, tanto Maven como Ant pueden generar y manipular el archivo de manifiesto en el contexto de proyectos Java.

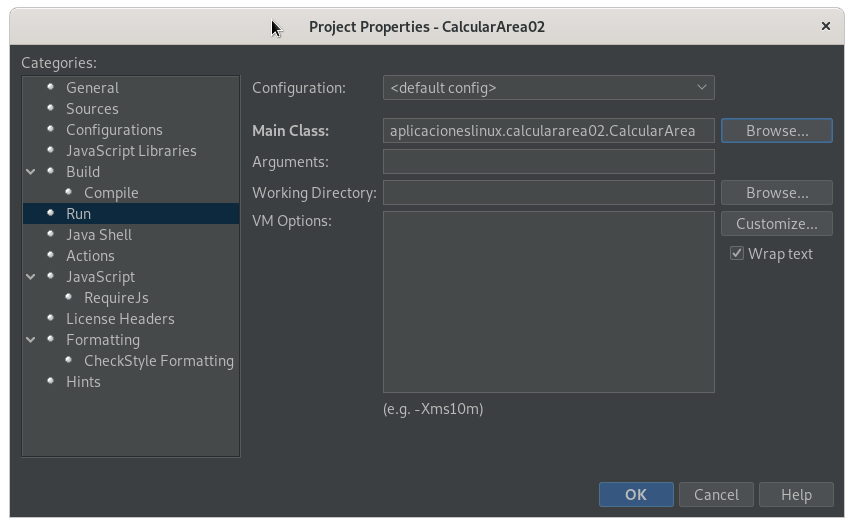
En el caso de Maven, el archivo de manifiesto se genera automáticamente durante el proceso de construcción según la configuración especificada en el archivo pom.xml. Maven utiliza el complemento maven-jar-plugin para crear el archivo JAR y generar el archivo de manifiesto correspondiente. Puedes configurar propiedades como la clase principal, dependencias, permisos de seguridad y otros atributos en el archivo pom.xml para que se reflejen en el archivo de manifiesto generado.

En cuanto a Ant, el archivo de manifiesto también se puede generar y manipular mediante tareas específicas. Ant proporciona la tarea manifest que permite configurar y personalizar el archivo de manifiesto. Puedes definir propiedades como la clase principal, rutas de clase adicionales, permisos y otros atributos en la tarea manifest para generar el archivo de manifiesto deseado.

En resumen, tanto Maven como Ant tienen la capacidad de generar el archivo de manifiesto en proyectos Java y ofrecen formas de configurarlo y personalizarlo según tus necesidades.

## Configura la clase que tiene el método main en el IDE

Ya no es necesario indicar en propiedades del proyecto el camino, paquete, a la clase que tiene el método main.



Es recomendado indicar el camino, paquete, donde se ubica la clase que tiene el método main. Si esta tarea no se realiza no importa, yo que el archivo POM.XML se hace cargo de esta tarea.

FIN