

DAM  
Desarrollo de Aplicaciones Multiplataforma  
2º Curso

AD  
Acceso a Datos

UD 5  
Programación de componentes  
de acceso a datos  
(Parte 1)

IES BALMIS  
Dpto Informática  
Curso 2022-2023  
Versión 1 (11/2022)

## UD5 – Programación de componentes de acceso a datos

### ÍNDICE

1. Introducción
2. Creación de componentes
3. Generar empaquetado de aplicaciones
  - 3.1 Empaquetado de una aplicación
  - 3.2 Ejecución de un empaquetado JAR
4. Generar componentes y reutilización
  - 4.1 Empaquetado de un componente
  - 4.2 Reutilización de un componente
5. Proyectos Maven y Dependencias
  - 5.1 Proyectos Maven con Java
  - 5.2 Proyectos Maven en NetBeans
6. Componentes visuales
  - 6.1 Swing
  - 6.2 Ejecución de aplicación de escritorio de tipo Window
  - 6.3 Alternativas en el desarrollo de software

# 1. Introducción

Un **componente** software es una clase creada para ser reutilizada y que proporciona una funcionalidad a una aplicación o servicio determinado, pudiendo en algunos casos ser accesible mediante interfaz pública.

Los componentes se definen por su estado que se almacena en un conjunto de propiedades, las cuales pueden ser modificadas para adaptar el componente al programa en el que se inserte. También puede tener un comportamiento que se define por los eventos ante los que responde y los métodos que ejecuta ante dichos eventos.

En Java disponemos de dos plataformas de computación:

- **Java SE:** Plataforma de computación formada por una máquina virtual de Java más una biblioteca de clases estándares en la que se pueden ejecutar aplicaciones java.
- **Java EE:** Plataforma de computación con el mismo planteamiento que Java SE, pero con una biblioteca de clases más amplia donde muchas de ellas están pensadas para aplicaciones empresariales.

Un **JavaBean** es un componente de software creado para ser ejecutado en Java SE que cumple con las siguientes especificaciones:

- Debe tener un **constructor vacío**
- Debe implementar la interfaz **Serializable**
- Las **propiedades/atributos** deben ser **privados**.
- Debe tener métodos **getters o setters** o ambos que permitan acceder a sus propiedades

Un JavaBean en NetBeans se crea desde:

- "Java => Java Application" o
- "Java => Java Class Library"

Un **EJB (Enterprise JavaBean)** es un componente de software creado par ser ejecutado en Java EE.

Un EJB en NetBeans se crea desde:

- "Java EE => Enterprise Application" o
- "Java EE => EJB Module"

Una **aplicación Java de escritorio (Desktop App)** está almacenada en un dispositivo y se ejecuta accediendo a ella a través de gestor de archivos de su SO.

Las App Desktop pueden ser de dos tipos:

- **De Consola:** serán ejecutadas con la utilidad **java.exe**
- **De Ventanas (Window):** serán ejecutadas con la utilidad **javaw.exe**

Una **aplicación java de tipo web (Web App)** está almacenada en un servidor y se ejecuta accediendo a ella a través de un navegador mediante pro protocolo HTTP o HTTPS.

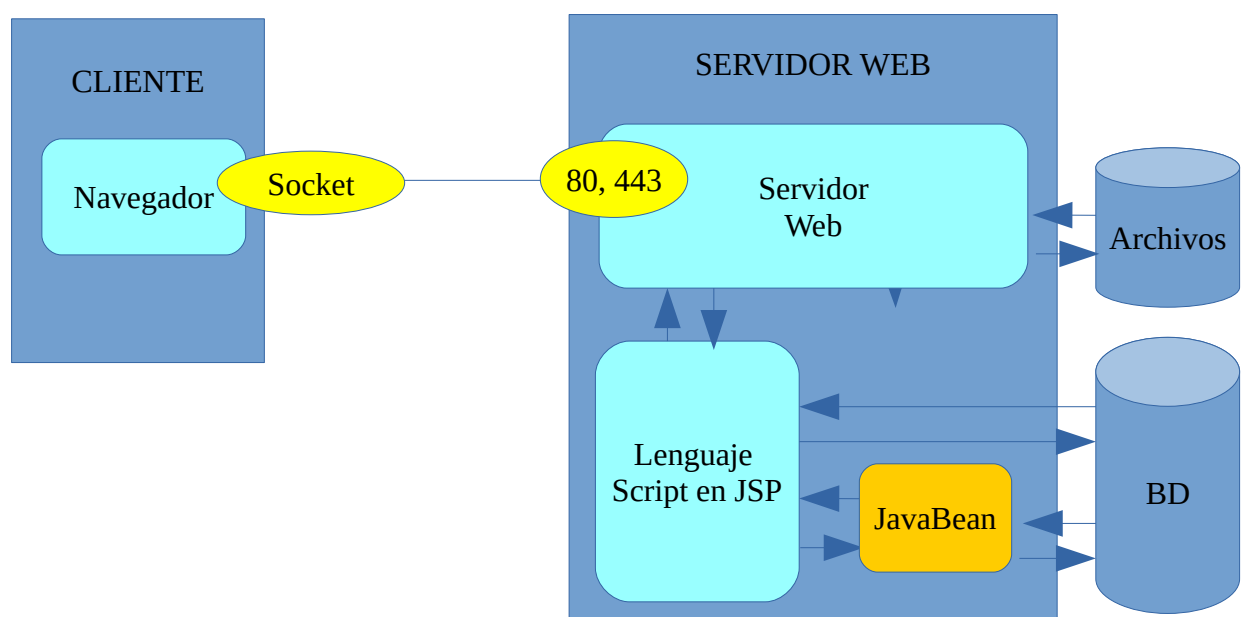
Las Web App de Java se desarrollan en utilizando el lenguaje script de servidor **Java Server Pages (JSP)**.

Los JSP pueden ejecutar código Java que genera páginas web dinámicas con tecnologías HTML+CSS+Javascript.

Un **Servlet** es un componente de Java utilizado para ampliar las capacidades de un servidor web, siendo utilizados por las páginas de tipo JSP que se ejecutan en el servidor.

Los **Servlets** son **JavaBean** pero para ser ejecutados en las páginas JSP y generalmente para implementar un **controlador**, es decir, un gestor de eventos y funcionalidad.

La mayoría de EJB y Servlets acceden a Bases de Datos, por lo que es necesario que tengan la funcionalidad de conexión al SGBD existente en el servidor.



## 2. Creación de componentes

Como en cualquier clase, un componente tendrá definido un estado a partir de un conjunto de atributos.

Los **atributos** son variables definidas por su nombre y su tipo de datos que toman valores concretos. Normalmente los atributos son privados y no se ven desde fuera de la clase que implementa el componente, se usan sólo a nivel de programación.

Las **propiedades** son un tipo específico de atributos que representan características de un componente que afectan a su apariencia o a su comportamiento. Son accesibles desde fuera de la clase y forman parte de su interfaz. Suelen estar asociadas a un atributo interno.

Una propiedad no es exactamente un atributo público, sino que tiene asociados ciertos métodos que son la única manera de poder modificarla o devolver su valor y que pueden ser de dos tipos:

### **GETTER**

Permiten leer el valor de la propiedad y tienen la estructura:

```
public <TipoPropiedad> get<NombrePropiedad>( )
```

Si la propiedad es booleana el método getter se implementa así:

```
public boolean is<NombrePropiedad>()
```

### **SETTER**

Permiten establecer el valor de la propiedad y tienen la estructura:

```
public void set<NombrePropiedad>(<TipoPropiedad> valor)
```

Si una propiedad no incluye el método set entonces es una propiedad de solo lectura.

Por ejemplo, si estamos generando un componente para almacenar datos de una **libro**, podemos tener, entre otras, una propiedad de sea **título**, que tendría asociados los siguientes métodos:

```
public void setTitulo(String titulo)
```

```
public String getTitulo()
```

Un **componente NO necesita** disponer de un método **main**, ya que se utilizará en otras aplicaciones.

Un **componente** bien definido debe incorporar al menos:

- La posibilidad de **Serializar** para conseguir la **persistencia** implementando la clase predefinida `java.io.Serializable` o `java.io.Externalizable`
- Las propiedades que almacenar
- Los constructores
  - Constructor sin parámetros
  - Constructor con todos los parámetros
  - En caso de existir una propiedad clave, el constructor de la dicha propiedad que no se repite
  - Los métodos getters y setters
  - El método `toString` sobrecargado para poder serializar a String
  - En caso de existir propiedades indexadas (`ArrayList`, `Set`, ...) es recomendable:
    - Añadir método para incorporar un elemento a la lista/conjunto
    - Añadir método para eliminar un elemento a la lista/conjunto

El **componente** debe incluirse en un **paquete** que posteriormente nos permitirá realizar el **import**.

Para ilustrar cómo se implementa un componente con una herramienta como NetBeans, crearemos uno proyecto de tipo "**Java with Ant => Java Application**" que permita almacenar la información de una **persona**, guardaremos por ejemplo, su nombre, apellidos, teléfono y dirección:

- De tipo `int` tendremos:
  - **dni**
- De tipo `String` tendremos:
  - **nombre, apellidos, telefono**
- La **domicilio** será una propiedad compuesta, por la dirección, código postal, la población y la provincia.

Dado que la última propiedad no es simple, será necesario crear otro componente **domicilio** que la desarrolle, y del que dependerá nuestro componente principal **persona**.

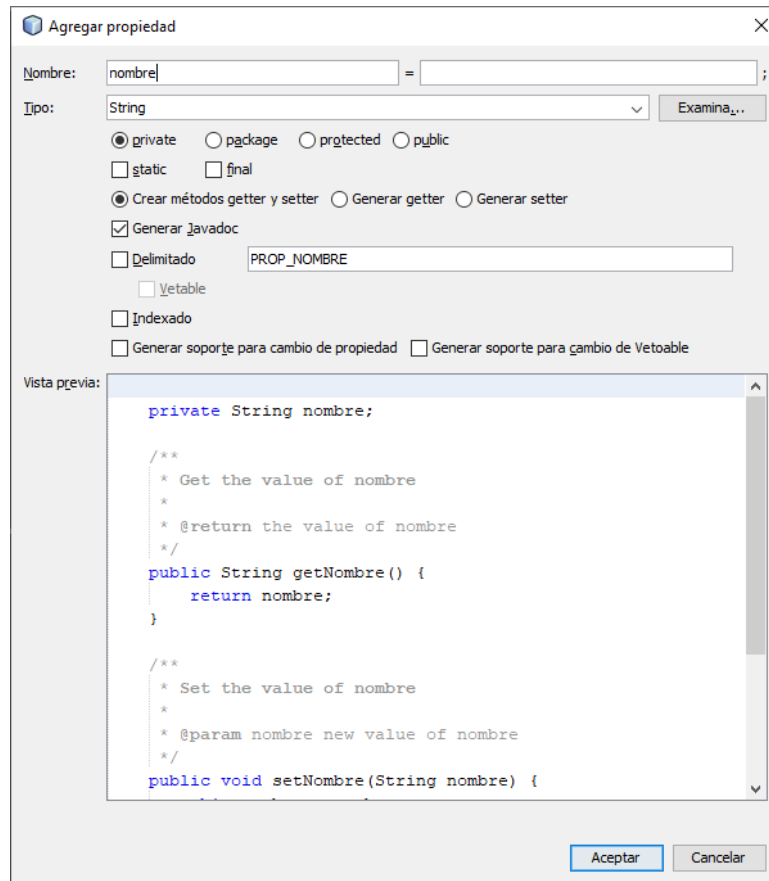
El proyecto contendrá la clase **public Persona**, y la clase **public Domicilio** .

Netbeans nos ofrece generación de código entre otros aspectos para:

- Constructores
- Getters y Setters
- `toString()`

Para ello, basta con pulsar en el editor con botón derecho y seleccionar "**Insert code**".

También podemos utilizar el asistente para añadir una propiedad con su **setter** y **getter** seleccionando "**Agregar propiedad**". Nos aparecerá en siguiente formulario:



**Agregar propiedad**

Nombre:  =  ;

Tipo:

☒ private    ☐ package    ☐ protected    ☐ public  
☐ static    ☐ final  
☒ Crear métodos getter y setter    ☐ Generar getter    ☐ Generar setter  
☒ Generar Javadoc  
☐ Delimitado   
☐ Vetable  
☐ Indexado  
☐ Generar soporte para cambio de propiedad    ☐ Generar soporte para cambio de Vetable

Vista previa:

```

private String nombre;

/**
 * Get the value of nombre
 *
 * @return the value of nombre
 */
public String getNombre() {
    return nombre;
}

/**
 * Set the value of nombre
 *
 * @param nombre new value of nombre
 */
public void setNombre(String nombre) {
  
```

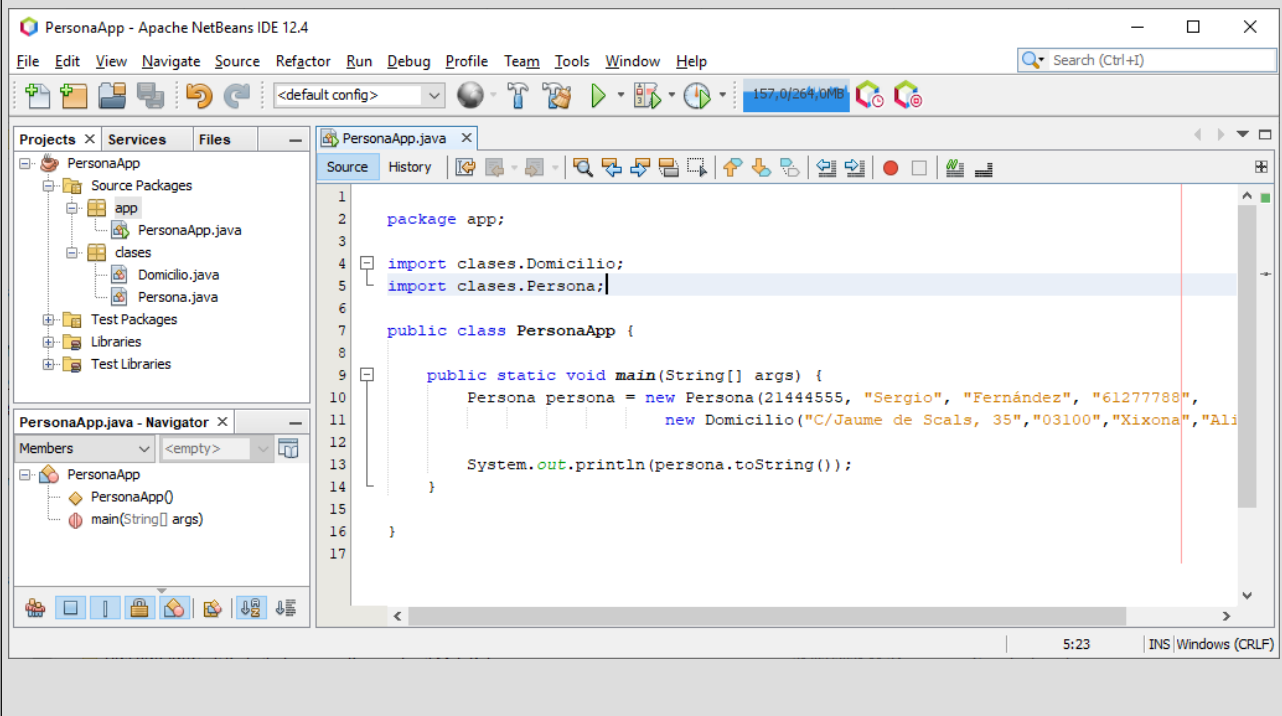
Los componentes visuales incorporan además varios recursos como:

- Un editor de propiedades que implementa la clase predefinida **PropertyEditor**
- La reacción ante **eventos** para capturar y procesar las acciones que se producen

Una de las principales características de un componente es que una vez instalado en un entorno de desarrollo, éste debe ser capaz de identificar sus propiedades simplemente detectando parejas de operaciones get/set, mediante la capacidad denominada **introspección**.

## PersonaApp

Realizaremos un proyecto que utilizará una clase persona que contiene un objeto de la clase Domicilio



### 3. Generar empaquetado de aplicaciones

El empaquetado de código compilado en Java implica la creación de un archivo binario.

El archivo de un código empaquetado de Java pueden tener diferentes extensiones como JAR, WAR o EAR.

Nuestros primeros proyectos serán de extensión **JAR**.

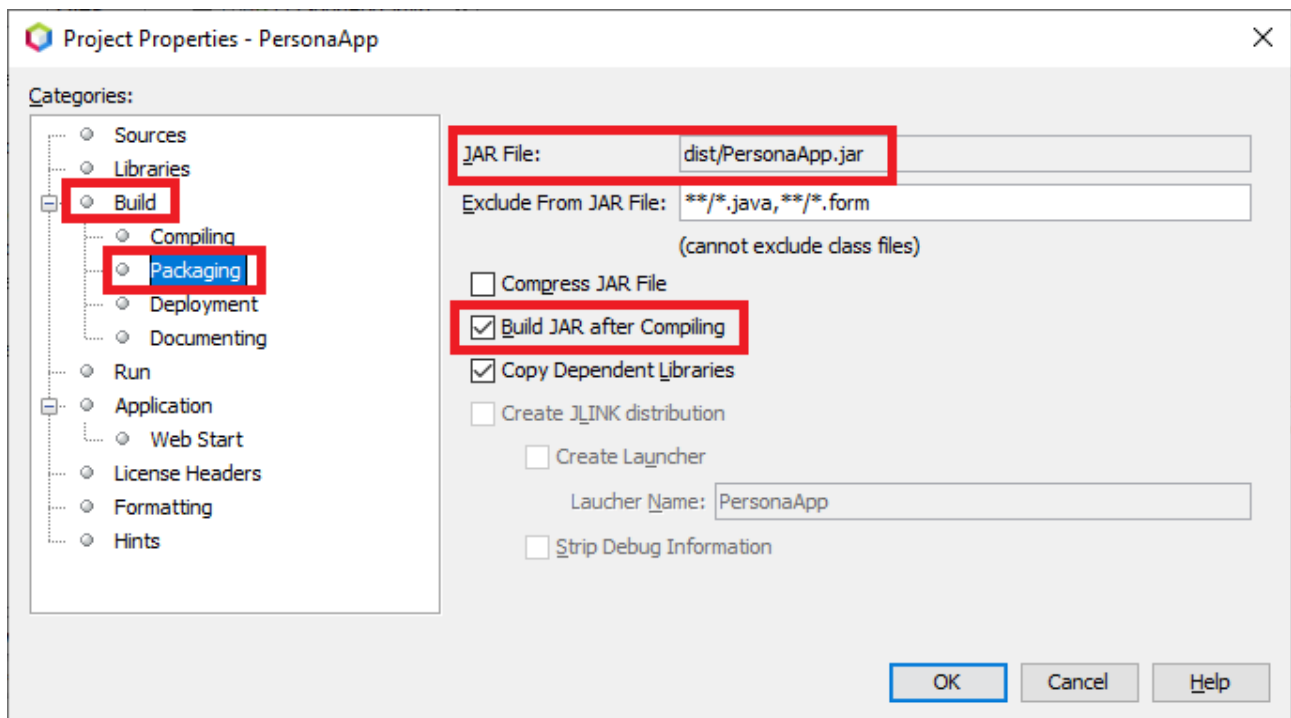
Veamos un ejemplo con el proyecto **PersonaApp**:

Crearemos el archivo JAR de la aplicación en Java que muestre el texto por consola.  
Utilizaremos para ello **System.out.println** con el método **toString()**.



### 3.1 Empaquetado de una aplicación

Una vez realizado el proyecto, lo primero es activar el empaquetado automático al compilar en propiedades del proyecto:



En este ejemplo, el proyecto **PersonaApp** creará el empaquetado **PersonaApp.jar** en la carpeta **dist**.

Para empaquetar un componente en NetBeans debemos generar el proyecto con el icono martillo o la tecla F11:



### 3.2 Ejecución de un empaquetado JAR

Una vez hayamos obtenido el archivo JAR lo podremos ejecutar. En nuestro caso, al ser una aplicación de escritorio de tipo consola, deberemos utilizar la utilidad **java.exe** con el siguiente comando:

```
C:\> cd "C:\Users\XXXX\Documents\NetBeansProjects\PersonaApp\dist"
C:\...\> java -jar PersonaApp.jar
Persona{dni=21444555, nombre=Sergio, apellidos=Fernández,
telefono=61277788, domicilio=Domicilio{direccion=C/Jaume de Scals, 35,
cpostal=03100, poblacio=Xixona, provincia=Alicante}}
```

Hay que tener en cuenta que si **java.exe** no se encuentra en el PATH de sistema, tendremos que indicar el camino completo. Por ejemplo:

```
C:\...\> "C:\Program Files\Java\jdk-X.X.X.X\bin\java" -jar PersonaApp.jar
```

## 4. Generar componentes y reutilización

### 4.1 Empaquetado de un componente

El empaquetado no está reservado exclusivamente a las aplicaciones, pues también se pueden crear componentes y empaquetarlos para poder reutilizarse en otros proyectos.

Veamos un ejemplo:

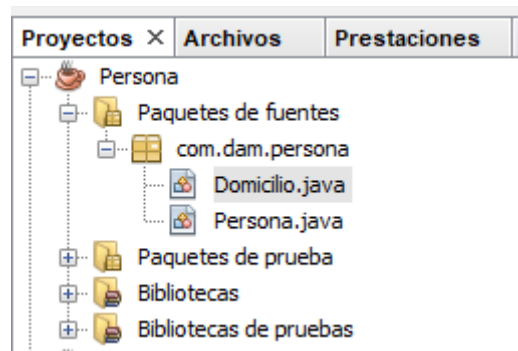
**Crear un componente en Java que contenga:**

- una clase `Domicilio` que almacene `direccion(String)`, `cpostal(String)`, `poblacion(String)` y `provincia(String)`
- una clase `Persona` que almacene un `dni(int)`, `nombre(String)`, `apellidos(String)`, `telefono(String)` y `domicilio(Domicilio)`

Las características del proyecto serán:

<b>TIPO</b>	Java Class Library (Es un Java Application <b>sin método main</b> )
<b>PROYECTO</b>	Persona
<b>JAVA PACKAGE</b>	<b>com.dam.persona</b>
<b>CLASES</b>	Serializables, Constructores, Getters y Setters, <code>toString()</code>

En NetBeans quedará:



Una vez activada la creación del JAR en propiedades al compilar, pulsamos en el martillo y generamos el JAR que se encontrará en la carpeta **dist**. El archivo tendrá el nombre de:

**Persona.jar**

## 4.2 Reutilización de un componente

Ahora crearemos una aplicación que use el componente creado.

Veamos un ejemplo:

### PersonaAppConJAR

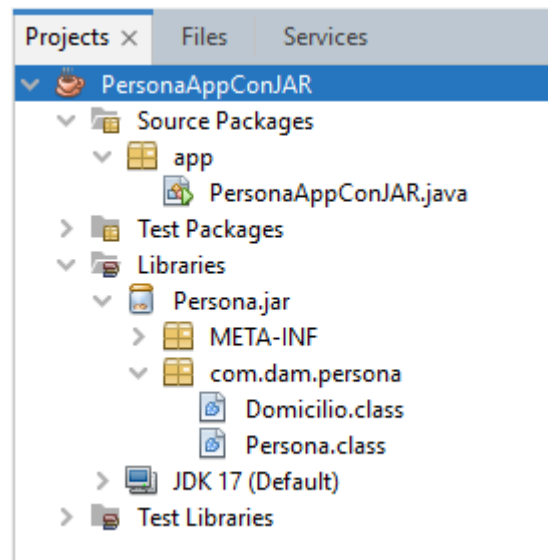
Crear una aplicación en Java que:

- Cree una instancia de **Persona** utilizando los constructores de **Domicilio** y **Persona**
- Mostrar su contenido utilizando **toString**

Crearemos un proyecto de tipo Java Application denominado **PersonaAppConJAR**.

Crearemos la carpeta **lib** y copiaremos el archivo **Persona.jar** del componente.

En NetBeans quedará:



El código de nuestro proyecto podría ser:

### PersonaAppConJAR

```
import com.dam.persona.Domicilio;
import com.dam.persona.Persona;

public class PersonaConJAR {

    public static void main(String[] args) {

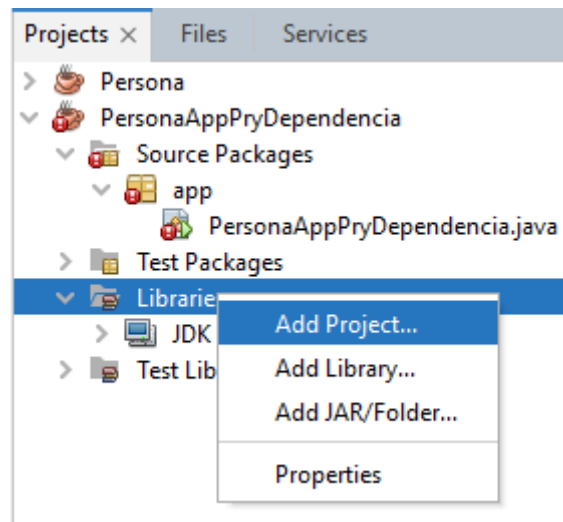
        Persona persona = new Persona(21444555, "Sergio", "Fernández", "61277788",
            new Domicilio("C/Jaume de Scals, 35", "03100", "Xixona", "Alicante"));

        System.out.println(persona.toString());
    }
}
```

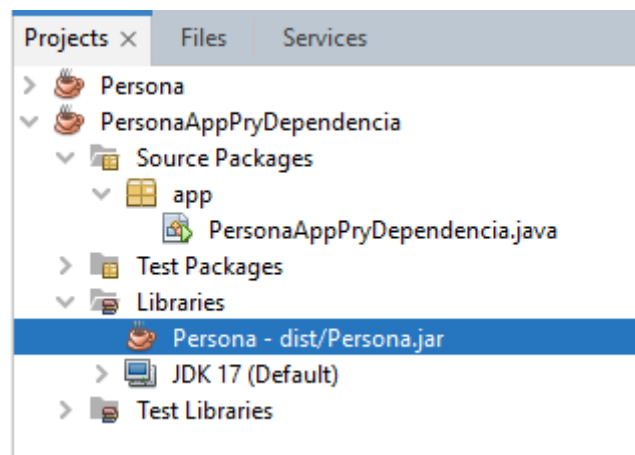
Podemos ver que el componente **Persona.jar** nos da la posibilidad de realizar **import** de sus clases **com.dam.persona.Domicilio** y **com.dam.persona.Persona**, como si las hubiéramos definido en el propio proyecto.

Muchos programadores, en entornos de desarrollo, lo que hacen es crear la dependencia entre proyectos para evitar tener que estar copiando el archivo JAR de la carpeta **dist** a la de **lib** del proyecto que la necesita.

Para ello crearemos un nuevo proyecto de tipo Java Application denominado **PersonaAppPryDependencia** con el mismo código que **PersonaAppConJAR** pero en Libraries añadiremos el proyecto Persona en vez de un JAR:



Una vez añadido, quedará la dependencia vinculada a la carpeta dist de Persona:



Esto nos permitiría realizar cambios en el proyecto **Persona** sin tener que copiar JAR después de cada compilación en el proyecto **PersonaAppPryDependencia**.

## 5. Proyectos Maven y dependencias

Existen multitud de componentes que podemos incorporar a nuestras aplicaciones y muchos de ellos dependen de otros, por lo que cuando un proyecto se hace grande y además hay actualizaciones es complejo gestionar el mantenimiento.

Por esta razón aparecen herramientas con repositorios de componentes. Para Java existen varios, pero los más usados son **Maven** y **Gradle**.

Gracias a estas herramientas, podemos añadir componentes y su documentación a nuestras aplicaciones de una manera sencilla, actualizada y segura.

### 5.1 Proyectos Maven con Java

**Apache Maven Project** es una herramienta de gestión y comprensión de proyectos de software basado en el concepto de un modelo de objeto de proyecto (POM).

Maven puede administrar la construcción, los informes y la documentación de un proyecto a partir de una información central.

#### *Apache Maven Project*

<https://maven.apache.org/>

**Maven ofrece la posibilidad de explorar online** los componentes pudiendo consultar diferente información organizada por **categorías** como: estadísticas de uso, revisiones y actualizaciones, tipos de licencia, fecha de publicación, ...

#### *Repository Maven*

<https://mvnrepository.com/>

<https://mvnrepository.com/repos/central>

<https://repo1.maven.org/maven2/> (para descargas directas de los JAR)

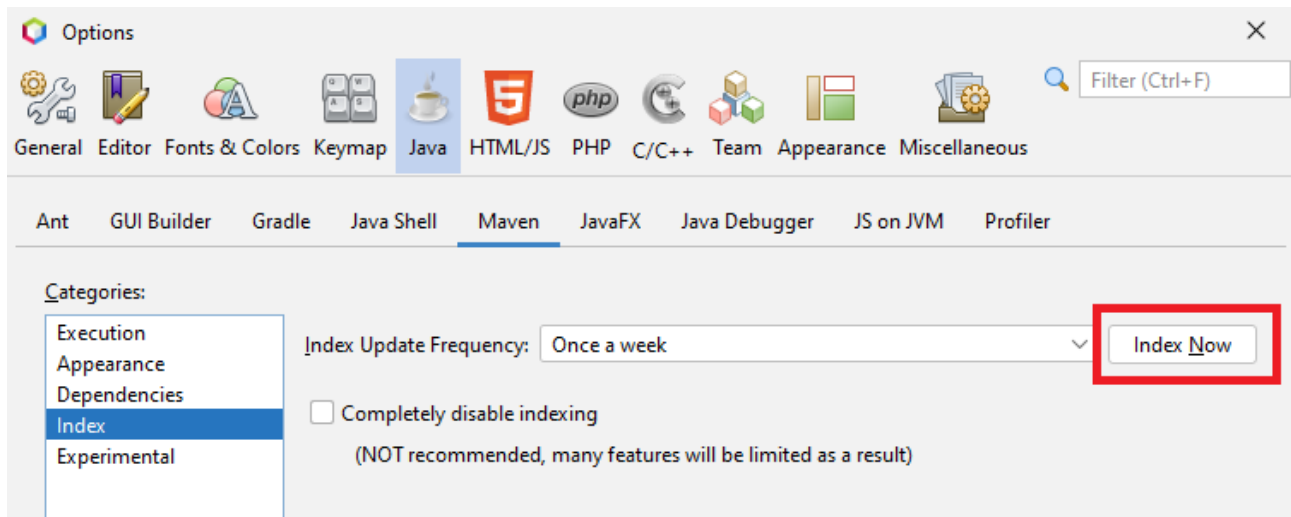
NetBeans necesita descargarse el índice de los millones de componentes disponibles.

Esta descarga ocupa más de 3GB, por lo que tardará un rato en terminar el proceso.

Para descargarlo o actualizarlo, podemos ir a:

**Tools → Options → Java → Maven → Index**

y pulsar en el botón **Index Now**:



La descarga se almacena en la carpeta:

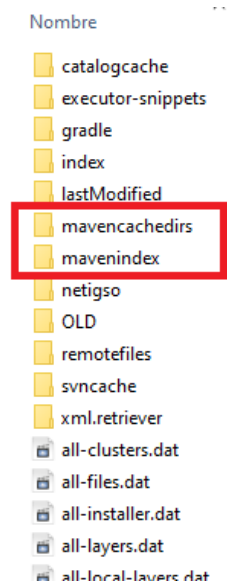
**C:\Users\XXXXXX\AppData\Local\NetBeans\Cache\XX**

Las dos carpetas que contienen el índice son:

- **mavencachedirs**
- **mavenindex**

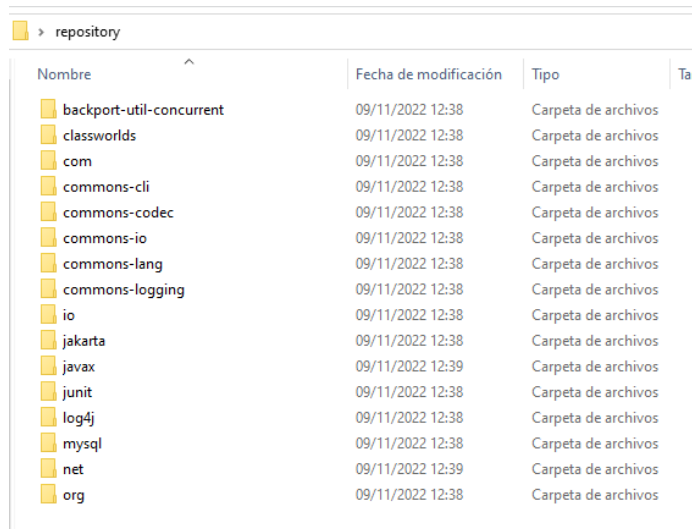
Podemos copiarlas de otro ordenador para disponer del índice más rápidamente.

Luego la actualización tardará menos tiempo.



La primera vez que ejecutemos un proyecto maven, **creará un repositorio local en tu disco duro**. En concreto, creará la carpeta **.m2** en la carpeta del usuario. En ella se guardarán todos los componentes (JAR) que utilice maven.

**C:\Users\XXXXX\.m2\repository**



Nombre	Fecha de modificación	Tipo	Tamaño
backport-util-concurrent	09/11/2022 12:38	Carpeta de archivos	
classworlds	09/11/2022 12:38	Carpeta de archivos	
com	09/11/2022 12:38	Carpeta de archivos	
commons-cli	09/11/2022 12:38	Carpeta de archivos	
commons-codec	09/11/2022 12:38	Carpeta de archivos	
commons-io	09/11/2022 12:38	Carpeta de archivos	
commons-lang	09/11/2022 12:38	Carpeta de archivos	
commons-logging	09/11/2022 12:38	Carpeta de archivos	
io	09/11/2022 12:38	Carpeta de archivos	
jakarta	09/11/2022 12:38	Carpeta de archivos	
javax	09/11/2022 12:39	Carpeta de archivos	
junit	09/11/2022 12:38	Carpeta de archivos	
log4j	09/11/2022 12:38	Carpeta de archivos	
mysql	09/11/2022 12:38	Carpeta de archivos	
net	09/11/2022 12:39	Carpeta de archivos	
org	09/11/2022 12:38	Carpeta de archivos	

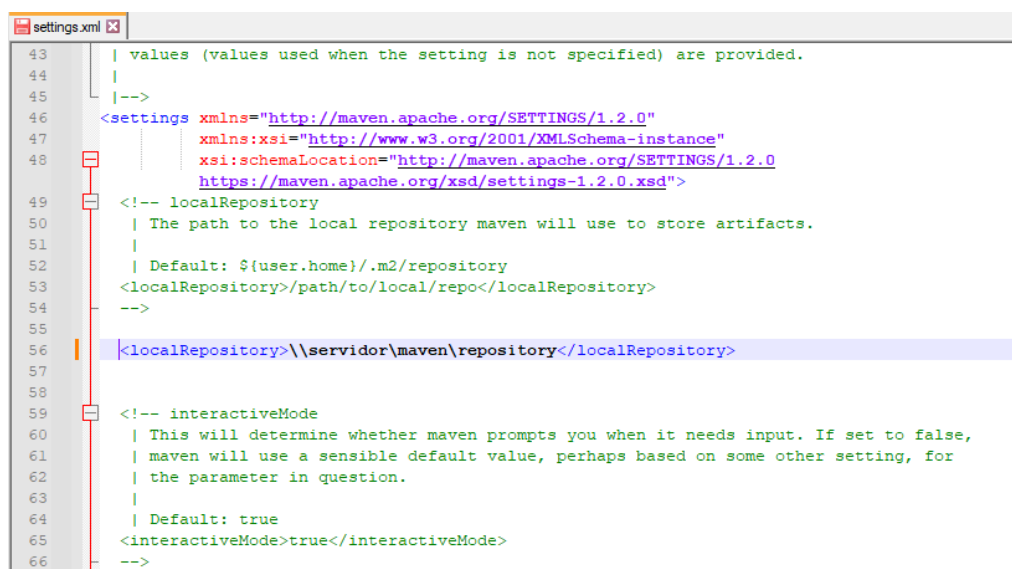
Esta carpeta será más o menos grande según los componentes que hayamos utilizado en nuestros proyectos.

### Compartir un repositorio ya descargado

También podríamos compartirlo entre varios ordenadores accediendo a una carpeta compartida que indicaríamos en el archivo:

**C:\Program Files\NetBeans-X\netbeans\java\maven\conf\settings.xml**

Por ejemplo, si la carpeta compartida estuviera en **\\servidor\maven\repository**, lo indicaríamos así:



```

43 | values (values used when the setting is not specified) are provided.
44 |
45 | -->
46 <settings xmlns="http://maven.apache.org/SETTINGS/1.2.0"
47 |         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
48 |         xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.2.0
49 |                             https://maven.apache.org/xsd/settings-1.2.0.xsd">
50 |   <!-- localRepository
51 |        | The path to the local repository maven will use to store artifacts.
52 |        | Default: ${user.home}\\.m2/repository
53 |   <localRepository>path/to/local/repo</localRepository>
54 |   -->
55 |
56 |   <localRepository>\\servidor\\maven\\repository</localRepository>
57 |
58 |   <!-- interactiveMode
59 |        | This will determine whether maven prompts you when it needs input. If set to false,
60 |        | maven will use a sensible default value, perhaps based on some other setting, for
61 |        | the parameter in question.
62 |        | Default: true
63 |   <interactiveMode>true</interactiveMode>
64 |   -->
65 |
66 |
```

## 5.2 Proyectos Maven en NetBeans

### MavenProject.java

Crea un proyecto con el contenido del proyecto **JDBC01Select** de la unidad didáctica de JDBC que ejecuta un SELECT de MySQL pero con Maven

Para crear un proyecto con Maven utilizaremos

**"Java with Maven => Java Application".**

El asistente nos permitirá añadir información al archivo **POM.xml** del proyecto mediante un formulario. Es muy importante el **ID del Grupo** porque será el Java Package que identificará nuestro proyecto, y deberá ser único para no crear colisiones con otros desarrollos externos.

Las empresas suelen usar su **dominio de Internet**, ya que es único. La única diferencia es que se escribe al contrario ya que se organizan como en el DNS por carpetas. En nuestro ejemplo, suponemos registrado el dominio **dam.com**, por lo que indicaremos **com.dam**

The screenshot shows the 'New Java Application' dialog box in NetBeans. The 'Steps' panel on the left indicates the current step is '2. Name and Location'. The 'Name and Location' panel contains the following fields:

- Project Name: MavenProject
- Project Location: C:\Users\vrigo\Documents\NetBeansProjects (with a 'Browse...' button)
- Project Folder: \Users\vrigo\Documents\NetBeansProjects\MavenProject
- Artifact Id: MavenProject
- Group Id: com.dam (highlighted with a red box)
- Version: 1.0-SNAPSHOT
- Package: com.dam.mavenproject (Optional)

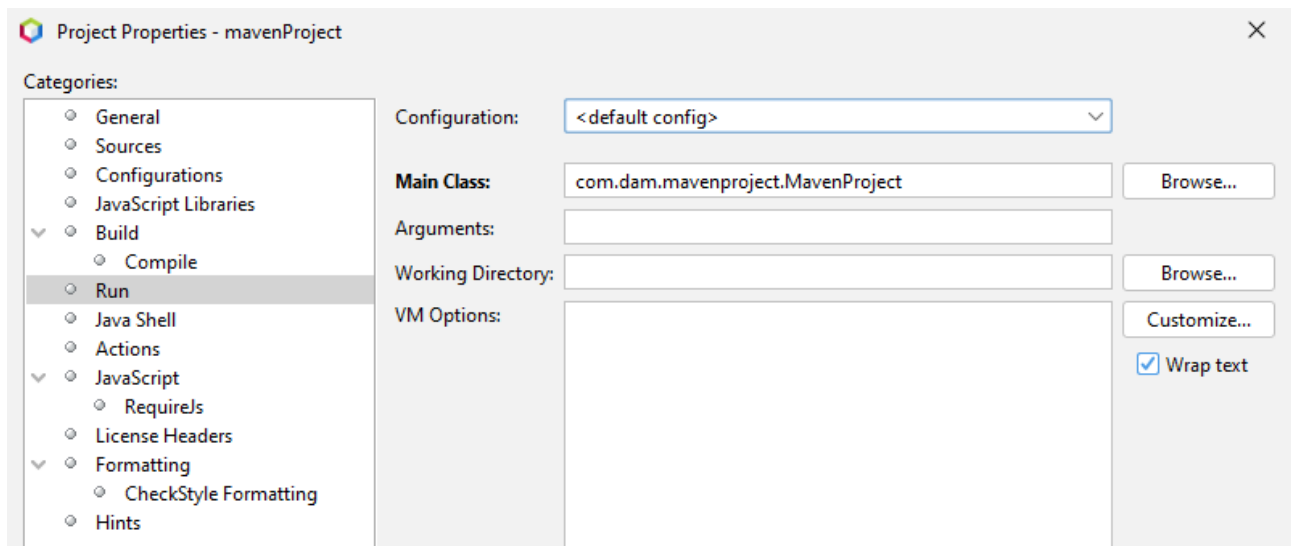
### Seleccionar la clase que contiene el método main

Pulsaremos con botón derecho sobre el proyecto => **Propiedades**

=> **Run => Main Class => Pulsar botón Browse**

- Seleccionar la clase que contiene el main





### Añadir componentes y sus dependencias

Realizaremos un proyecto que conecta con MySQL.

Las librerías o componentes están disponibles de forma online, por lo que es necesario disponer de conexión a internet.

### ANEXO Maven Proxy

Maven necesita hacer descargas desde sus repositorios. En el caso de tener **proxy**, habrá que configurar Netbeans para que lo utilice.

El archivo a configurar es:

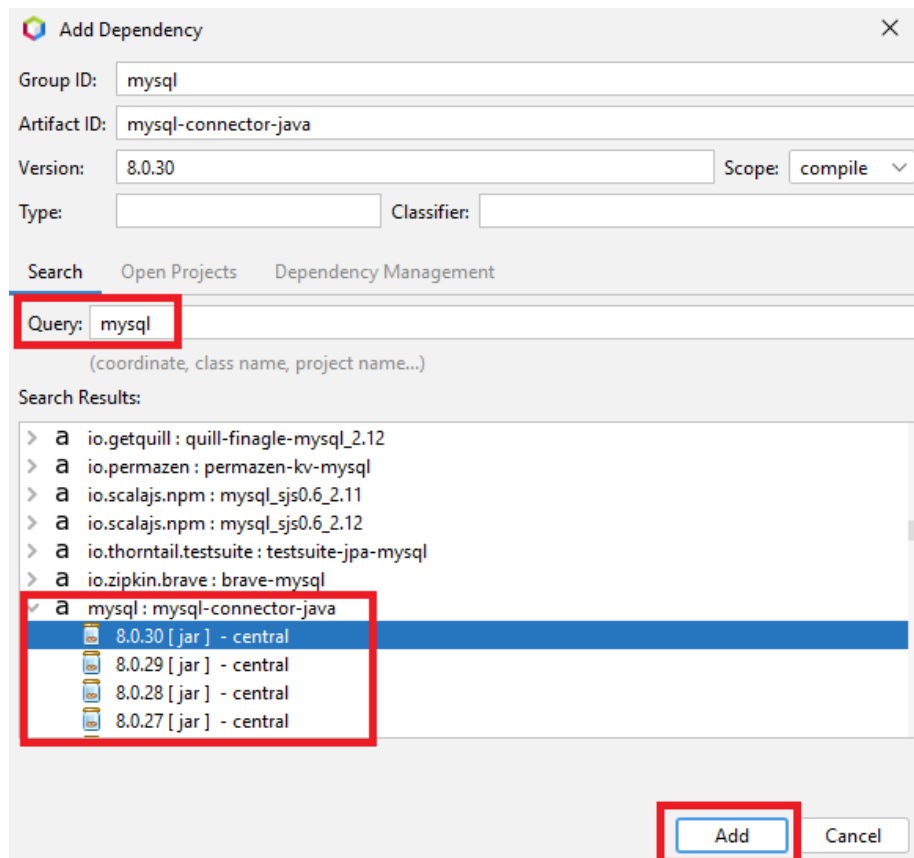
**C:\Program Files\NetBeans-X\netbeans\java\maven\conf\settings.xml**

donde sustituiremos el bloque **proxies** por el siguiente:

```
<proxies>
  <!-- proxy
  | Specification for one proxy, to be used in connecting
  | to the network.-->
  <proxy>
    <id>optional</id>
    <active>true</active>
    <protocol>http</protocol>
    <username>alumno</username>
    <password>alumno</password>
    <host>10.100.0.1</host>
    <port>8080</port>
    <!-- <nonProxyHosts>local.net|some.host.com</nonProxyHosts> -->
  </proxy>
  <!-- -->
</proxies>
```

Pulsando con botón derecho en "**Dependencies => Add Dependency**"

- En "**Query**" escribir **mysql**  
(la primera vez tardará un poco hasta que se descargue la información del repositorio de Maven)
- y buscar **mysql : mysql-connector-java**
- Pulsar en [+]
- Seleccionar la última (en este caso 8.X.X [jar])
- Agregar

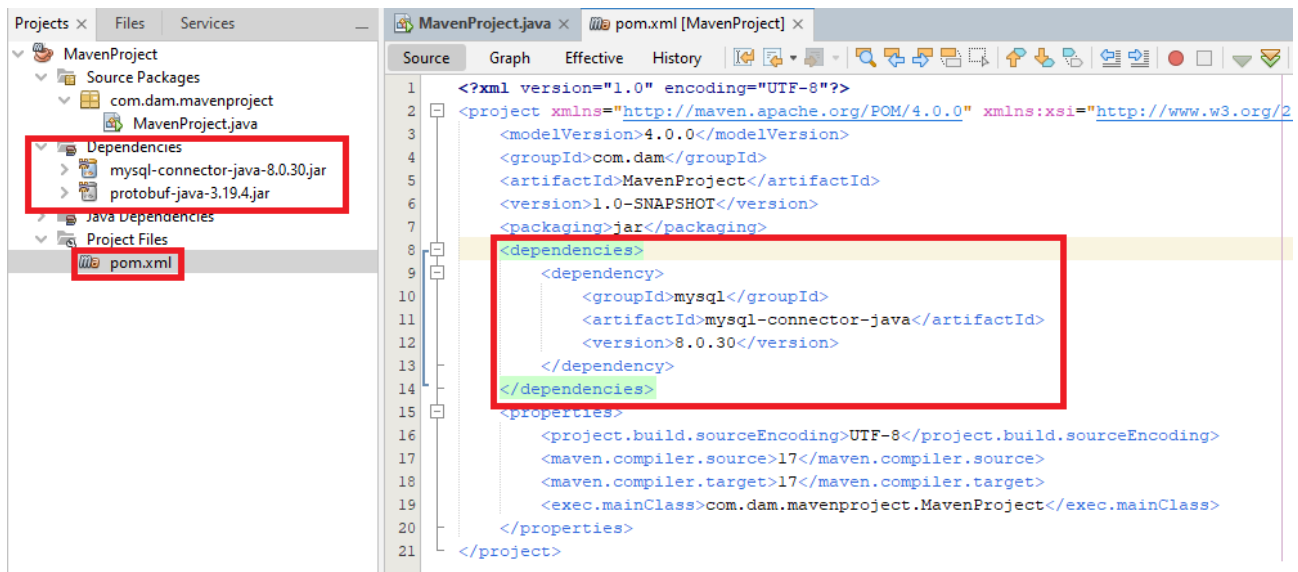


Al añadir un componente, Maven incorpora todas sus dependencias, por lo que no tenemos que preocuparnos por el resto de archivos.

Preferiblemente seleccionar **central**, aunque si ya se ha descargado antes también podremos seleccionar local.

## Añadir librerías por dependencia en pom.xml

Para añadir una librería, también podemos editar el fichero **pom.xml** y añadir la información de la librería en formato **Maven**. Por ejemplo:



Para conocer el texto de la dependencia, podemos consultar buscando en maven nuestra librería "**mysql-connector-java**". Accediendo al enlace tendremos el código:

<https://mvnrepository.com/artifact/mysql/mysql-connector-java>

## MVN REPOSITORY

**Indexed Artifacts (30.6M)**

**Popular Categories**

- Testing Frameworks
- Android Packages
- Logging Frameworks
- Java Specifications
- JSON Libraries
- Core Utilities
- JVM Languages
- Mocking
- Language Runtime
- Web Assets
- Annotation Libraries
- Logging Bridges
- HTTP Clients
- Dependency Injection
- XML Processing
- Web Frameworks
- I/O Utilities
- Configuration Libraries
- Defect Detection Metadata
- Code Generators
- Android Platform

[Home](#) » [mysql](#) » [mysql-connector-java](#) » 8.0.30

### MySQL Connector Java » 8.0.30

MySQL Connector/J is a JDBC Type 4 driver, which means that it is pure Java implementation of the MySQL protocol and does the Driver Manager, standardized validity checks, categorized SQLExceptions, support for large update counts, support for XML processing, support for per connection client information and support for the NCHAR, NVARCHAR ...

<b>License</b>	GPL 2.0
<b>Categories</b>	JDBC Drivers
<b>Tags</b>	database sql jdbc driver connector mysql
<b>Organization</b>	Oracle Corporation
<b>HomePage</b>	<a href="http://dev.mysql.com/doc/connector-j/en/">http://dev.mysql.com/doc/connector-j/en/</a>
<b>Date</b>	Jul 25, 2022
<b>Files</b>	pom (2 KB) jar (2.4 MB) View All
<b>Repositories</b>	Central
<b>Ranking</b>	#68 in MvnRepository (See Top Artifacts) #1 in JDBC Drivers
<b>Used By</b>	6,626 artifacts
<b>Vulnerabilities</b>	Vulnerabilities from dependencies: CVE-2022-3171

**Note:** There is a new version for this artifact

New Version 8.0.31

Maven
Gradle
Gradle (Short)
Gradle (Kotlin)
SBT
Ivy
Grape
Leiningen
Builder

```
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.30</version>
</dependency>
```

☒ Include comment with link to declaration

## Eliminar mensajes en la ejecución

Para eliminar los mensajes de Maven, podemos editar el archivo **simplelogger.properties** de la carpeta

**C:\Program Files\NetBeans-X\netbeans\java\maven\conf\logging\**

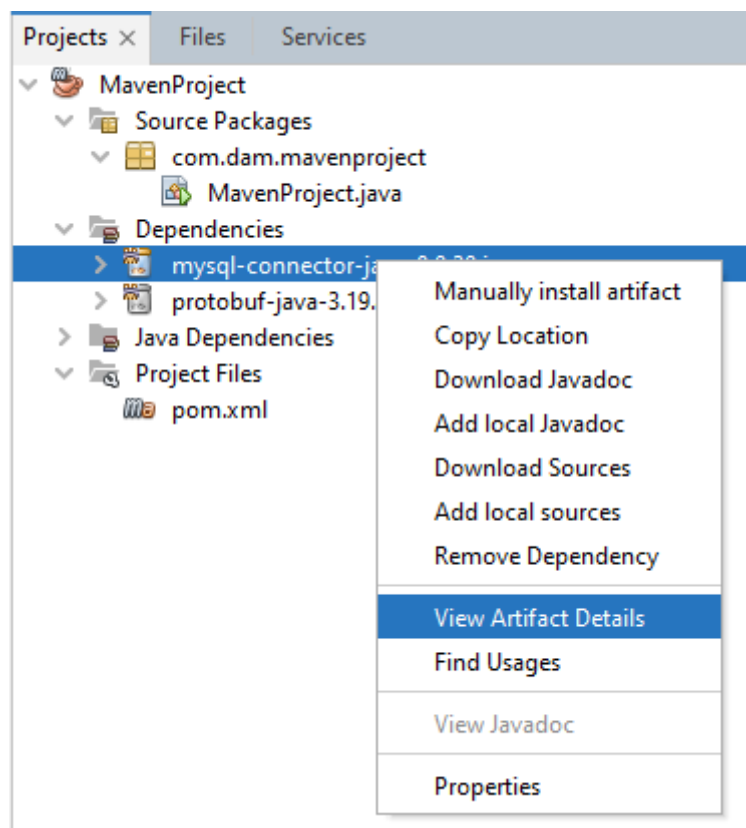
y cambiar la línea del **defaultLogLevel**:

**org.slf4j.simpleLogger.defaultLogLevel=error**

## Información sobre los componentes

Si queremos consultar la información disponible de cada componente añadido, pulsaremos con botón derecho sobre el **jar** => **View artifact details**, podremos tener acceso a datos sobre:

- Basic
- Project
- Classpath (Resolved Dependency List)



## Código de nuestra aplicación

Solo falta incluir en nuestra aplicación el código y probar a ejecutarlo. Añadiremos como prueba el código siguiente:

### Principal.java

```
package com.dam.mavenproject;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class MavenProject {
    public static void main(String[] args) {

        try {
            // Conexión a la BD
            String url;

            Class.forName("com.mysql.cj.jdbc.Driver");
            url = "jdbc:mysql://localhost:3306/biblioteca";
            url += "?zeroDateTimeBehavior=convertToNull";
            url += "&autoReconnect=true&useSSL=false&serverTimezone=UTC";

            String usuario = "root";
            String password = "1234";

            String sentenciaSQL = "SELECT titulo, autor FROM libros";

            try ( Connection con = DriverManager.getConnection(url, usuario, password);
                  Statement statement = con.createStatement();
                  ResultSet rs = statement.executeQuery(sentenciaSQL);
            ) {
                System.out.printf("%-60s %-60s\n", "Titulo", "Autor");
                System.out.print("-----");
                System.out.print(" ");
                System.out.print("-----");
                System.out.print("\n");
                while (rs.next()) {
                    System.out.printf("%-60s %-60s\n",
                                       rs.getString("titulo"), rs.getString("autor"));
                }
            } catch (SQLException ex) {
                System.out.println(ex.getErrorCode() + " " + ex.getMessage());
            } catch (ClassNotFoundException ex) {
                System.out.println("MySQL no accesible");
            }
        }
    }
}
```

Antes de ejecutarlo se deberá compilar, y se descargarán los **plugins de Maven** y las **librerías junto con sus dependencias al repositorio local**. Este proceso tardará más o menos según los que hayamos descargado previamente o no.

Después podremos ejecutarlo como un proyecto **Java with Ant**, pero teniendo los JAR en la carpeta del repositorio local.

## 6. Componentes visuales

Como ya hemos comentado anteriormente, en Java se pueden crear aplicaciones de escritorio de tipo Ventana. Esto permite su ejecución en entornos gráficos abriendo ventanas en el escritorio del dispositivo.

Para crear estas ventanas Java dispone de varios componentes visuales. Los 3 más usados son:

- AWT
- JavaFX con SceneBuilder
- Swing

### AWT

- Es la base de Swing, funciona bien pero carece de componentes avanzados.
- Si se tiene la intención de crear aplicaciones ricas, AWT probablemente no sea el camino a seguir.
- Podría usarse para aplicaciones GUI pequeñas que no requieran interfaces de usuario enriquecidas, ya que es un entorno muy probado.

### SWING

- Está basado en AWT como se indicó anteriormente.
- Al principio, se consideraba lento y con errores, e hizo que IBM creara SWT para Eclipse. Sin embargo, con Java 6 Swing se convirtió en el GUI de elección para crear nuevas aplicaciones.
- Swing tiene muchos componentes enriquecidos, pero todavía faltan en algunas áreas. Un ejemplo es que no hay un componente TreeTable con todas las funciones que pueda ordenar y filtrar / buscar.

### JavaFX

JavaFX es el GUI diseñado por Java/Oracle y se usa en el desarrollo de aplicaciones web o de escritorio enriquecidas.

- En JavaFX, los administradores de diseño son nodos
- JavaFX ha mejorado el manejo de eventos
- JavaFX admite propiedades
- JavaFX es parametriza el diseño con estilos de CSS
- JavaFX tiene controles más consistentes
- JavaFX tiene efectos especiales
- Las animaciones son más fáciles en JavaFX
- JavaFX es compatible con dispositivos táctiles modernos
- JavaFX no tiene equivalente a JOptionPane de Swing

**Fuente:** <https://www.dummies.com/programming/java/10-differences-between-javafx-and-swing/>

Todos estos componentes son similares, a la hora del diseño, pero cambian bastante en su uso por su diferente modelo de clases.

## 6.1 Swing

Para crear un proyecto con Swing seleccionaremos

**"Java with Ant => Java Application".**

Swing tiene dos formas de trabajo:

- Creación dinámica (en tiempo de ejecución) de elementos visuales
- Creación visual de ventanas

Veamos ejemplos.

### SwingDinamico

Aplicación que pida el nombre de un archivo, por ejemplo "prueba" y cree un archivo llamado "prueba.txt" en la carpeta "./datos/" con el texto:

"Creado desde la aplicación de escritorio de tipo Window"

A continuación si no hay error mostrará un mensaje "Archivo creado" y en caso contrario "Error al crear el archivo".

```
public class SwingDinamico {
    public static void main(String[] args) {

        // CREAR CARPETA DATOS SI NO EXISTE
        boolean crearDatos=true;
        File datos = new File("./datos/");
        if (datos.exists()) {
            if (datos.isDirectory()) {
                crearDatos=false;
            }
        }
        if (crearDatos) datos.mkdir();

        // CREAR ARCHIVO DE TEXTO
        String archivo = JOptionPane.showInputDialog("Introduce el nombre de un archivo: ");
        if (!archivo.isEmpty()) {
            try (PrintWriter fichero =
                new PrintWriter("./datos/" + archivo + ".txt", StandardCharsets.UTF_8)) {
                fichero.println("Creado desde la aplicación de escritorio de tipo Window");

                JOptionPane.showMessageDialog(null, "Archivo creado");
            } catch (FileNotFoundException e) {
                JOptionPane.showMessageDialog(null, "Error al crear el archivo");
            } catch (IOException ex) {
                JOptionPane.showMessageDialog(null, "Error al crear el archivo");
            }
        } else {
            JOptionPane.showMessageDialog(null, "Nombre de archivo no puede ser vacío");
        }
    }
}
```

### JOptionPane – Ventanas de Diálogo (Message + Input)

<https://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html>

La clase static **JOptionPane** permite mostrar ventanas de entrada de datos (**showInputDialog**) y ventanas de mensajes (**showMessageDialog**).

**SwingVisual**

Aplicación que pida seleccionar un archivo de texto y muestre su contenido en un control de tipo "area".

Cuando queramos añadir una ventana, utilizaremos archivo nuevo y el tipo deseado:

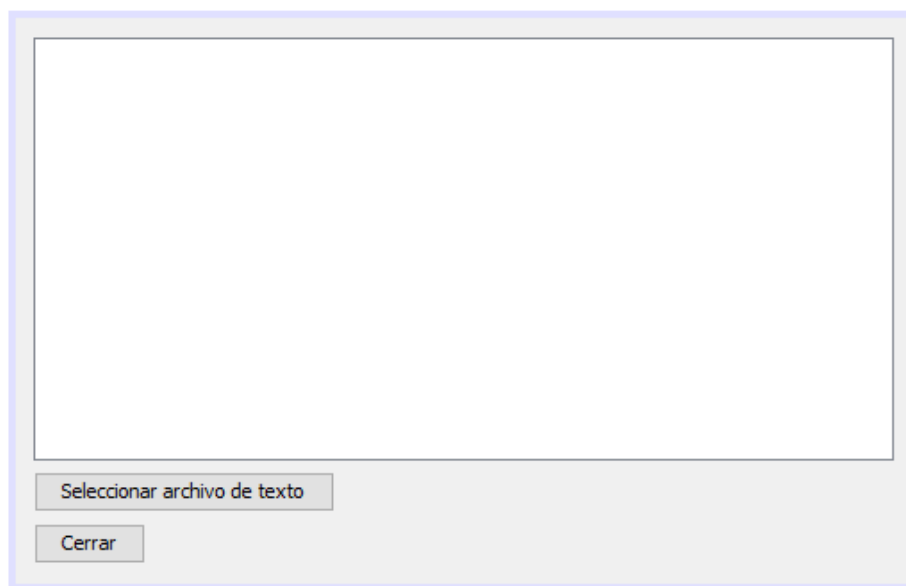
Formularios de interfaz gráfica de Swing =>

=> Formulario JDialog

=> Formulario JFrame

=> Formulario JPanel

En nuestro ejemplo crearemos un archivo nuevo de tipo **"Swing GUI Forms => JFrame Form"** denominado **WinJFrame** y le añadiremos un **TextArea** (área de texto) y dos **Button** (Botón).



Con botón derecho sobre los botones y **"Edit Text"** cambiaremos su contenido.

Ahora tenemos que introducir código en nuestra clase principal para llamar a la ventana:

```
public class SwingVisual {  
    public static void main(String[] args) {  
  
        java.awt.EventQueue.invokeLater(new Runnable() {  
            public void run() {  
                JFrame ventana = new WinJFrame();  
                ventana.setTitle("Cargar visor de archivo de texto:");  
                ventana.setVisible(true);  
            }  
        });  
    }  
}
```



Nuestro proyecto ya funciona pero no hace nada.

En el formulario añadiremos código al botón "**Cerrar**" haciendo doble click sobre él o mejor pulsando sobre él con botón derecho y luego

"Events → Actions → ActionPerformed" :

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}
```

Y al botón "Seleccionar archivo de texto":

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser jfc = new JFileChooser(new File("./datos/"));
    jfc.setDialogTitle("Elegir un fichero de texto: ");
    jfc.setFileSelectionMode(JFileChooser.FILES_ONLY);

    FileNameExtensionFilter filter =
        new FileNameExtensionFilter("TEXT FILES", "txt", "text");
    jfc.setFileFilter(filter);

    int returnValue = jfc.showOpenDialog(null);
    if (returnValue == JFileChooser.APPROVE_OPTION) {
        File selectedFile = jfc.getSelectedFile();

        try (BufferedReader ficheroEntrada =
            new BufferedReader(new InputStreamReader(
                new FileInputStream(selectedFile.getAbsolutePath()),
                StandardCharsets.UTF_8))) {
            jTextArea1.setText(null);
            String linea = null;
            while ((linea = ficheroEntrada.readLine()) != null) {
                jTextArea1.append(linea+"\n");
            }
        } catch (IOException ex) {
            JOptionPane.showMessageDialog(null, ex.getMessage());
        }
    }
}
```

Para probarlo, crear la carpeta **datos** en el proyecto y copiar un archivo de texto dentro.

## 6.2 Ejecución de aplicación de escritorio de tipo Window

Al compilar , obtendremos el **JAR** en la carpeta **dist** del proyecto.

Una vez hayamos obtenido el archivo JAR lo podremos ejecutar haciendo **dobles click** sobre él. En nuestro caso, al ser una aplicación de escritorio de tipo Windows, se usa por defecto la utilidad **javaw.exe**. También podemos ejecutarlo con el siguiente comando en un terminal CMD:

```
C:\> cd "C:\Users\XXXXXX\Documents\NetBeansProjects\SwingVisual\dist"  
C:\> javaw -jar SwingVisual.jar
```

Hay que tener en cuenta que si **javaw.exe** no se encuentra en el PATH de sistema, tendremos que indicar el camino completo. Por ejemplo:

```
C:\...\> "C:\Program Files\Java\jdk-X.X.X.X\bin\javaw" -jar SwingVisual.jar
```

## 6.3 Alternativas en el desarrollo de software

Una vez estudiada la creación de componentes podemos realizar las siguientes definiciones:

Un **Component** (componente de software) es una clase creada para ser reutilizada y que proporciona una funcionalidad a una aplicación o servicio determinado, pudiendo en algunos casos ser accesible mediante interfaz pública.

Un **Library** (librería de software) es conjunto de componentes relacionados entre sí que juntos se usan para un objetivo concreto.

Un **FrameWork** (marcos de trabajo) es conjunto de librerías de software que proporcionan una estructura base utilizada como punto de partida para elaborar un proyecto con objetivos específicos.

Los **Frameworks** ofrecen una funcionalidad definida y autocontenida, siendo contruidos usando patrones de diseño, y su característica principal es su alta cohesión y bajo acoplamiento.

Por lo tanto, en el **desarrollo de software** tenemos:

<b>IDE</b>	Software en el cual escribimos el código y que nos permite desarrollar de forma más cómoda y rápida.
<b>LENGUAJE DE PROGRAMACIÓN</b>	Conjunto de instrucciones que podemos usar en nuestro código y que podrá ser entendido por el compilador.
<b>FRAMEWORK</b>	Conjunto de librerías que se pueden reutilizar para simplificar y reducir el código a desarrollar para atender un requerimiento dado.

Nosotros estamos usando **NetBeans** como **IDE** y **Java** como **Lenguaje de Programación** y acabamos de ver **Swing** como **Framework** para crear interfaces de usuario gráficas.

Existen muchas alternativas para desarrollar la parte visual gráfica de un proyecto y las empresas eligen el Framework y Lenguaje de Programación con el que trabajar teniendo en cuenta multitud de características. Algunas de las más importantes son:

- Multiplataforma, desarrollo web
- Lenguaje de programación tradicional y conocido
- Tecnologías estandarizadas basadas en HTML, CSS, Javascript (Jquery, Bootstrap, ...)
- Formatos de archivo para los fuentes (TXT, JSON, XML, ...)

Algunos de los **Frameworks de desarrollo de interfaz gráfica** para el Desarrollo de Aplicaciones Multiplataforma más usados son:

<b>FRAMEWORK</b>	<b>LENGUAJE</b>	<b>DESARROLLADO POR</b>
<b>ANGULAR</b>	Javascript	Google
<b>REACT NATIVE</b>	Javascript	Facebook
<b>IONIC</b>	Javascript	Drifty
<b>VUE</b>	Javascript	Evan You
<b>SPRING</b>	Java	SpringSource
<b>DJANGO</b>	Python	Django Software Foundation
<b>KOTLIN</b>	Kotlin	JetBrains
<b>FLUTTER</b>	Dart	Google
<b>XAMARIN</b>	C#	Microsoft
<b>UNITY</b>	C#, C++	Unity Technologies
<b>.NET</b>	C#, C++	Microsoft

Independientemente del **Framework y Lenguaje de Programación** utilizado, podremos utilizar el **IDE** que mejor se adapte a nuestras necesidades. Algunos de los más usados son:

**Visual Studio, Eclipse, IntelliJ, Android Studio, ...**

Puedes consultar las siguientes estadísticas:

- <https://insights.stackoverflow.com/survey/2021#developer-profile-developer-roles>
- <https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-programming-scripting-and-markup-languages>
- <https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-databases>
- <https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-web-frameworks>
- <https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-cloud-platforms>