Argentina programa 4.0

JDBC

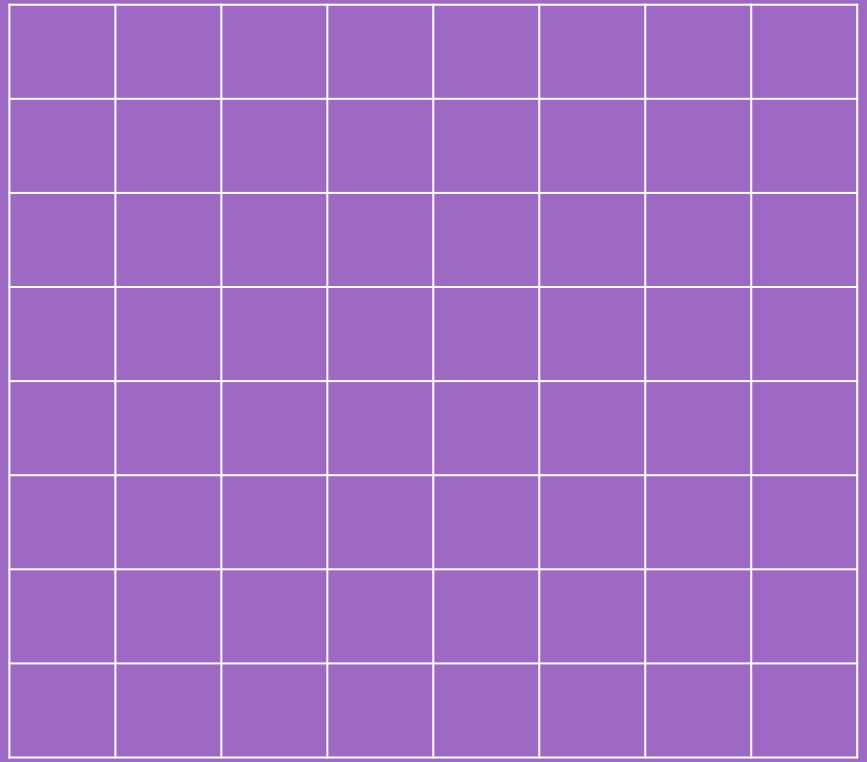
"Desarrollador Java Inicial"

Agenda

Argentina programa 4.0

- JDBC
  - API
  - Implementación
  - Conexión
  - Consultas

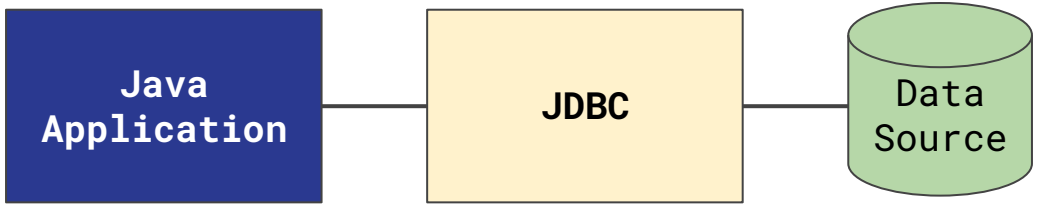
JDBC



JDBC

Argentina programa 4.0

- **JDBC** (**J**ava **D**atabase **C**onnectivity) conecta una aplicación Java a una fuente de datos como puede ser una DB.
- Es una API que es usada para:
  - Conectar a una fuente de datos
  - Enviar consultas y actualizaciones
  - Recupera y procesa el resultado
- JDBC utiliza drivers para conectarse a la DB.



JDBC

Argentina programa 4.0

- Java Database Connectivity (JDBC) es una interfase de acceso a bases de datos estándar SQL que proporciona un acceso uniforme a una gran variedad de bases de datos relacionales. JDBC también proporciona una base común para la construcción de herramientas y utilidades de alto nivel.

- Consiste en un conjunto de clases e interfases escritas en el lenguaje de programación Java. JDBC suministra un API estándar para los desarrolladores y hace posible escribir aplicaciones de base de datos usando un API puro Java.

JDBC

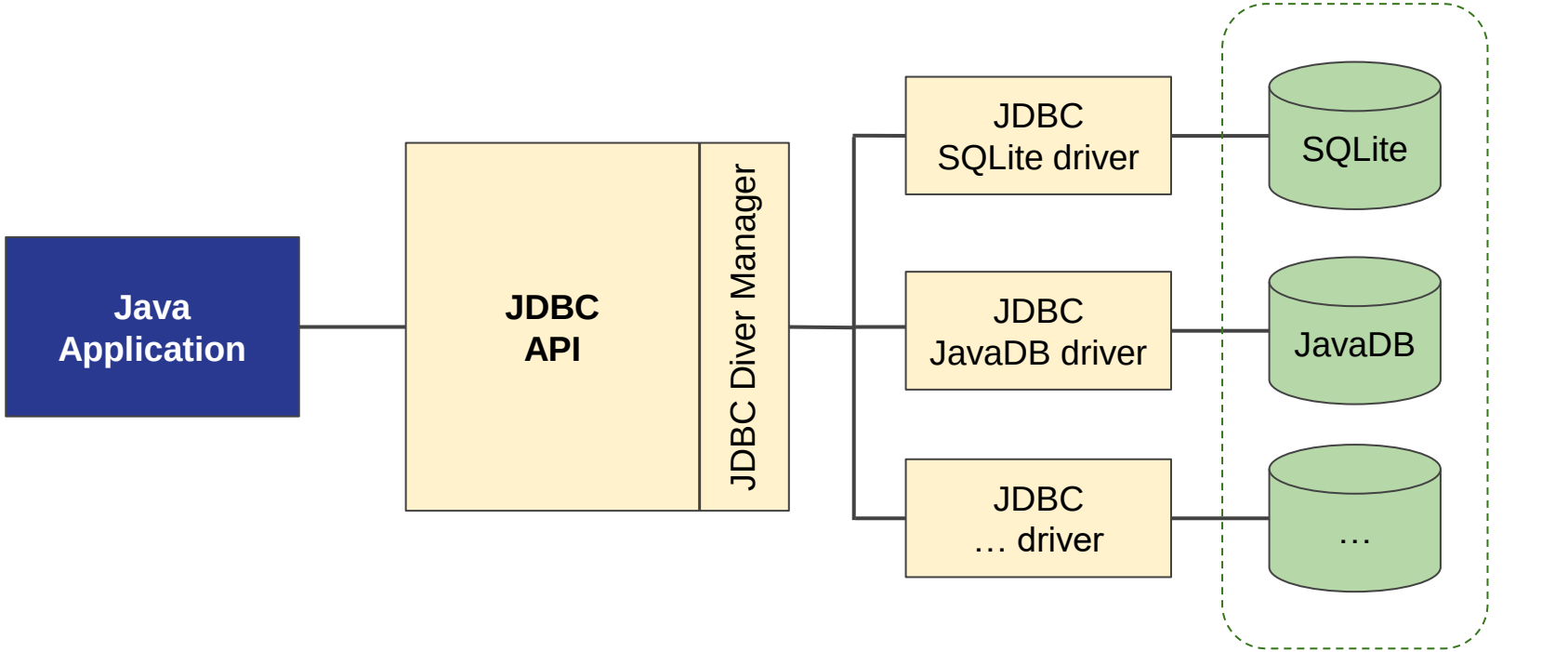
Argentina programa 4.0

- Usando JDBC es fácil enviar sentencias SQL virtualmente a cualquier sistema de base de datos. En otras palabras, con el API JDBC, no es necesario escribir un programa que acceda a una base de datos Sybase, otro para acceder a Oracle y otro para acceder a Informix.

- Un único programa escrito usando el API JDBC y el programa será capaz de enviar sentencias SQL a la base de datos apropiada. Y, con una aplicación escrita en el lenguaje de programación Java, tampoco es necesario escribir diferentes aplicaciones para ejecutar en diferentes plataformas.

- La combinación de Java y JDBC permite al programador escribir una sola vez y ejecutarlo en cualquier entorno.

JDBC - API


Java Application
JDBC API
JDBC Diver Manager
JDBC SQLite driver
JDBC JavaDB driver
JDBC … driver
SQLite
JavaDB
…
DBMS
Database Management System

¿Qué hace JDBC?

Argentina programa 4.0

- Simplemente JDBC hace posible estas tres cosas:

• Establece una conexión con la base de datos.

• Envía sentencias SQL

• Procesa los resultados.

Conectar una base de datos MySQL en Java

Argentina programa 4.0

- Para conectar la aplicación Java a la base de datos Mysql, debemos seguir algunos pasos que se enumeran a continuación:

❑ Descargue e instale la base de datos MySQL

❑ Crear una base de datos en MySQL

❑ Descargue el controlador JDBC e inclúyalo en Classpath

❑ Conéctese con MySQL

❑ Pruebe la conexión JDBC

Descargue e instale la base de datos MySQL

Argentina programa 4.0

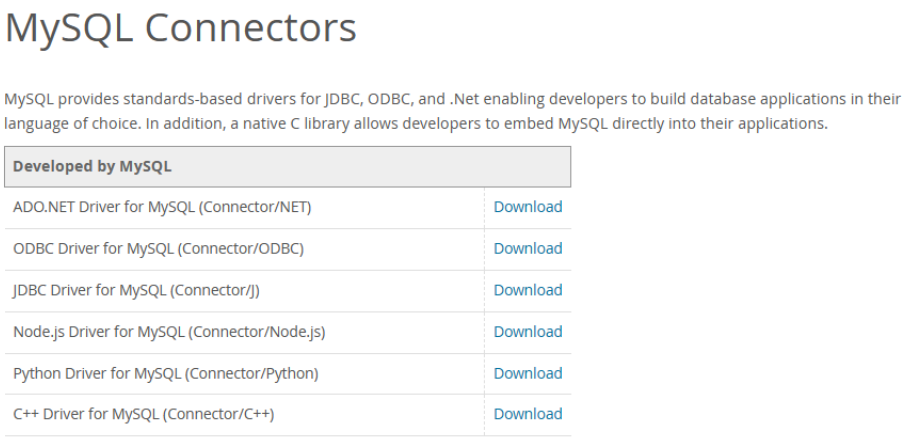
- Ver clase anterior.
- El Instalador de MySQL se puede descargar en este enlace.

- MySQL :: Download MySQL Installer

Descargue el controlador JDBC e inclúyalo en Classpath

Argentina programa 4.0

- El controlador JDBC es un archivo JAR proporcionado por MySQL; es un conector que actúa como puente entre MySQL y las aplicaciones Java. Para descargar el archivo JAR visite el sitio oficial de MySQL y coloque los archivos en la carpeta lib de su proyecto java.



MySQL Connectors

MySQL provides standards-based drivers for JDBC, ODBC, and .Net enabling developers to build database applications in their language of choice. In addition, a native C library allows developers to embed MySQL directly into their applications.

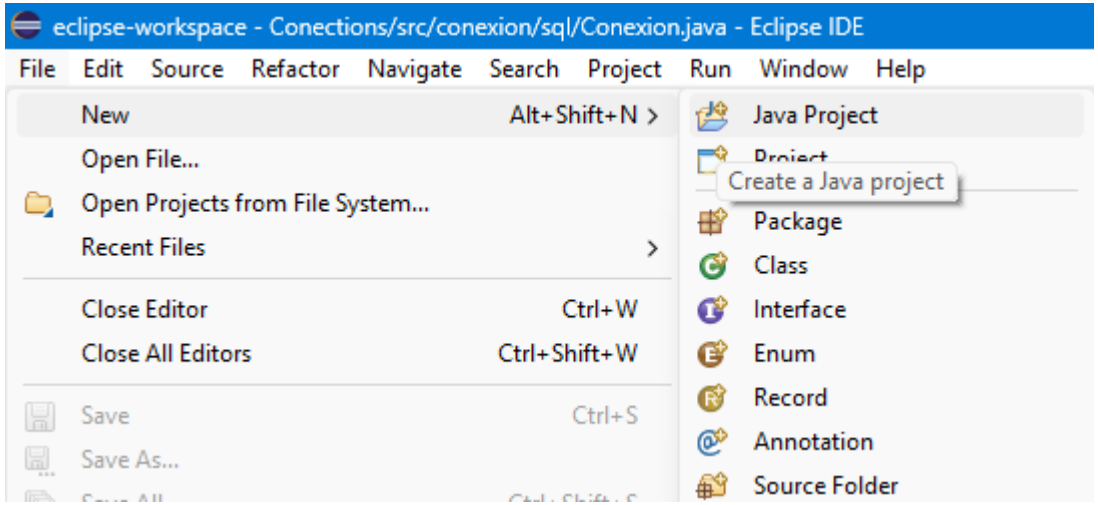
Developed by MySQL	
ADO.NET Driver for MySQL (Connector/NET)	Download
ODBC Driver for MySQL (Connector/ODBC)	Download
JDBC Driver for MySQL (Connector/J)	Download
Node.js Driver for MySQL (Connector/Node.js)	Download
Python Driver for MySQL (Connector/Python)	Download
C++ Driver for MySQL (Connector/C++)	Download

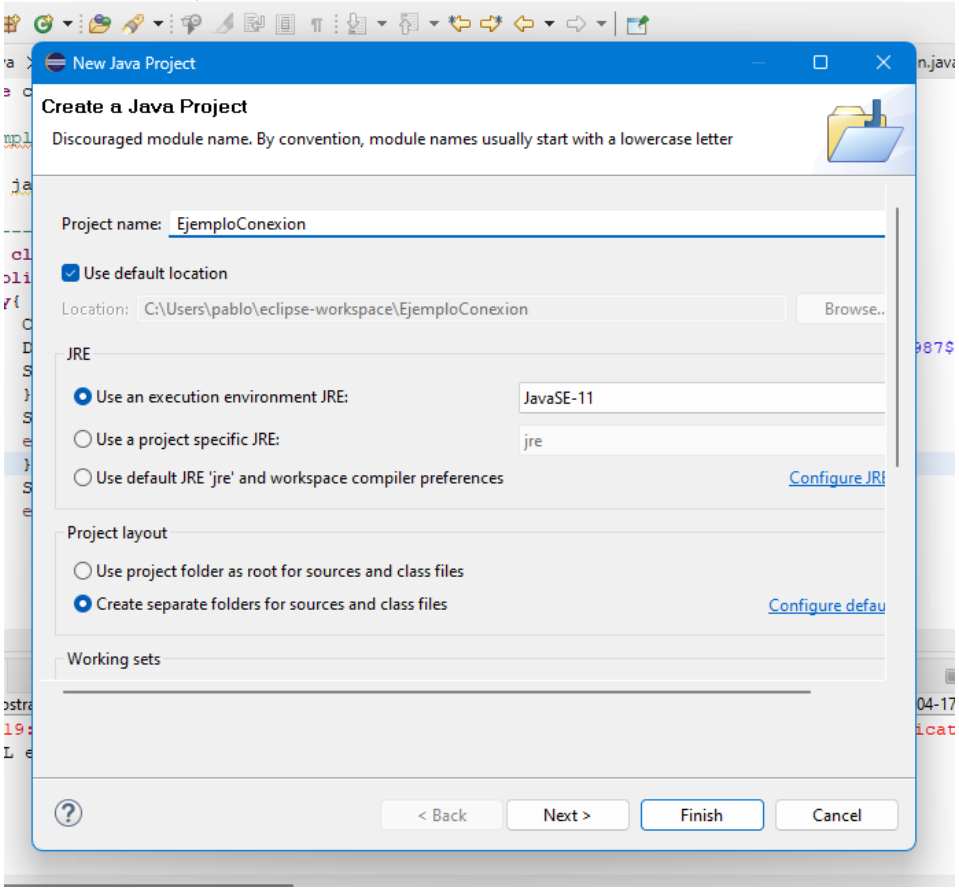
- Link en Google Drive
- https://drive.google.com/drive/u/1/folders/1Lvk9zq52M__jEljs_dcvOrZosv64KH3P

Descargue e instale la base de datos MySQL

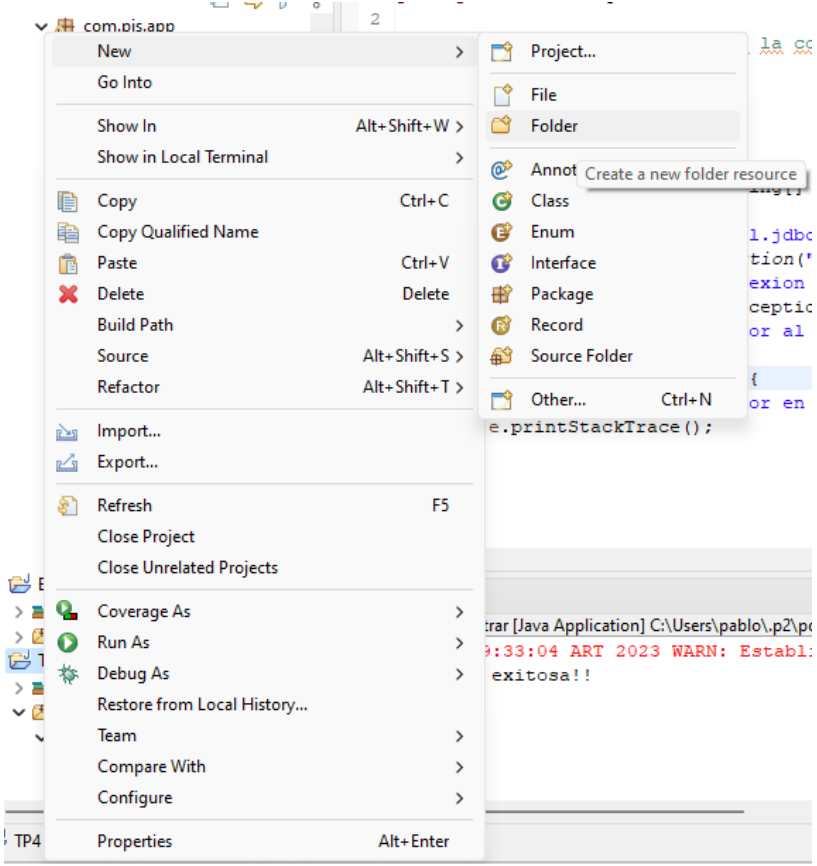
Argentina programa 4.0

- Una vez descargado, nos vamos a Eclipse y creamos un nuevo proyecto.

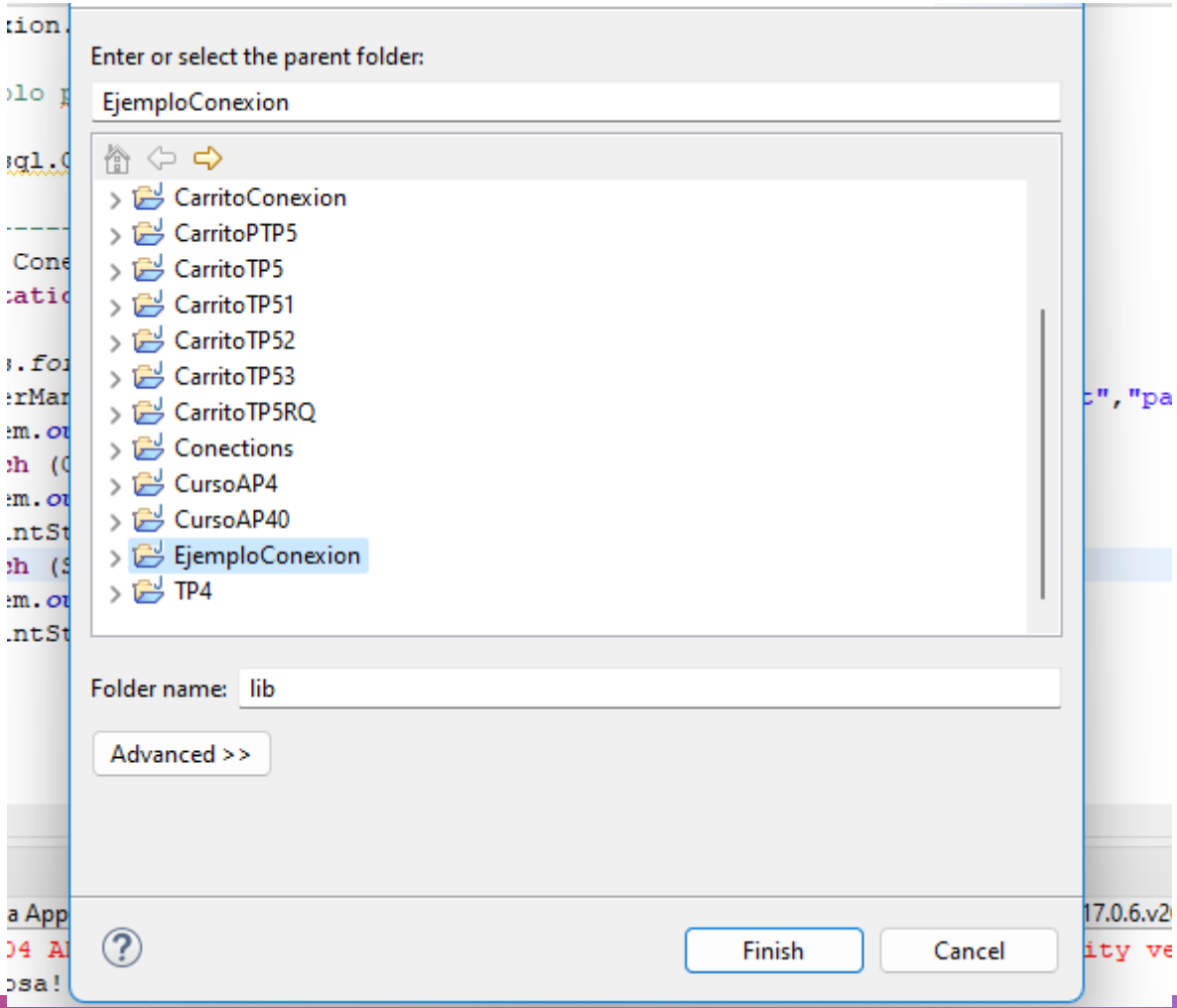




Argentina programa 4.0



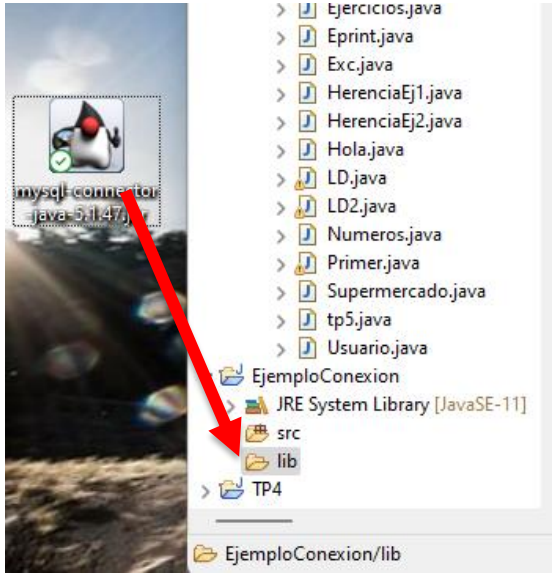
Dentro del nuevo proyecto, creamos una carpeta que se llamara lib

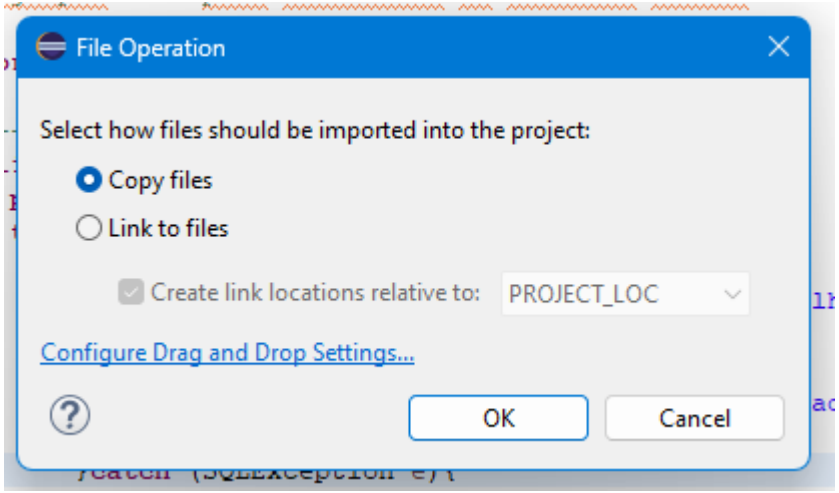


Argentina programa 4.0

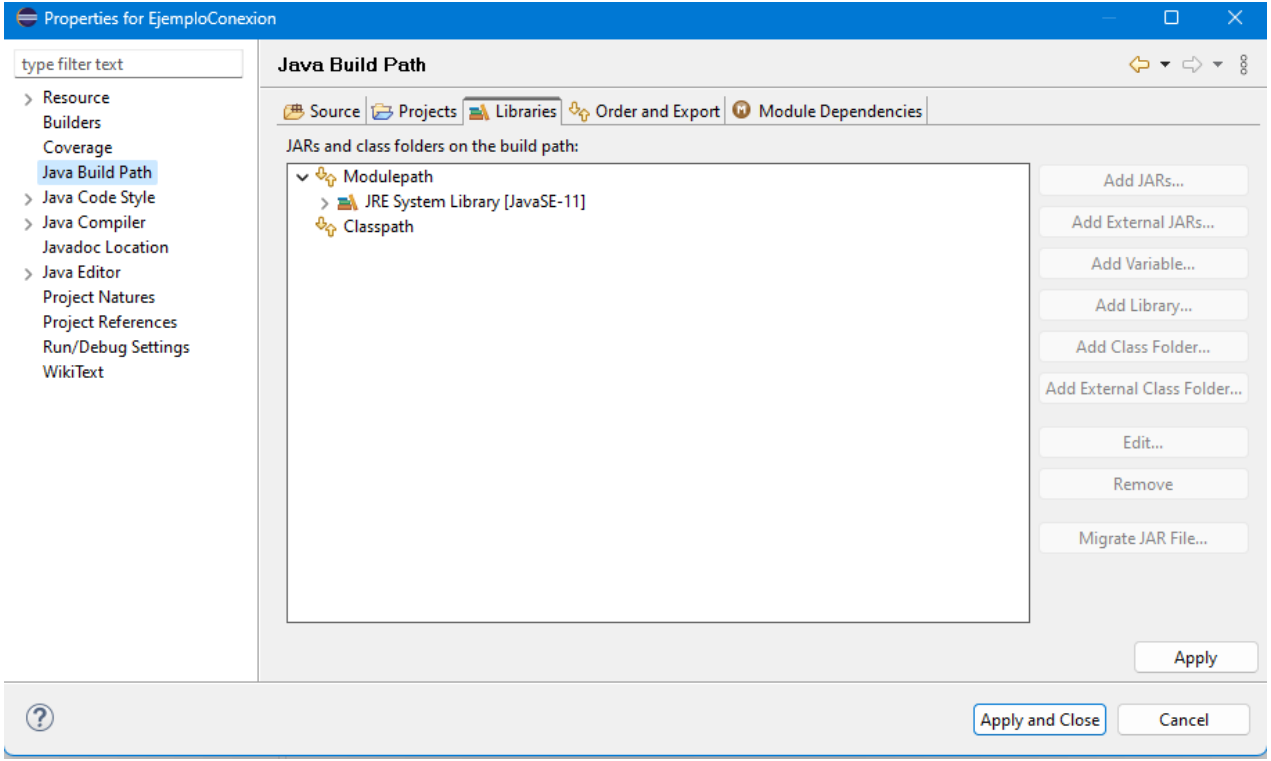
Copiamos el conector a la carpeta lib, de nuestro proyecto

- Luego de crear la carpeta, copiamos el conector descargado, en esa carpeta,





Ahora lo vamos a incluir en nuestras librerías...

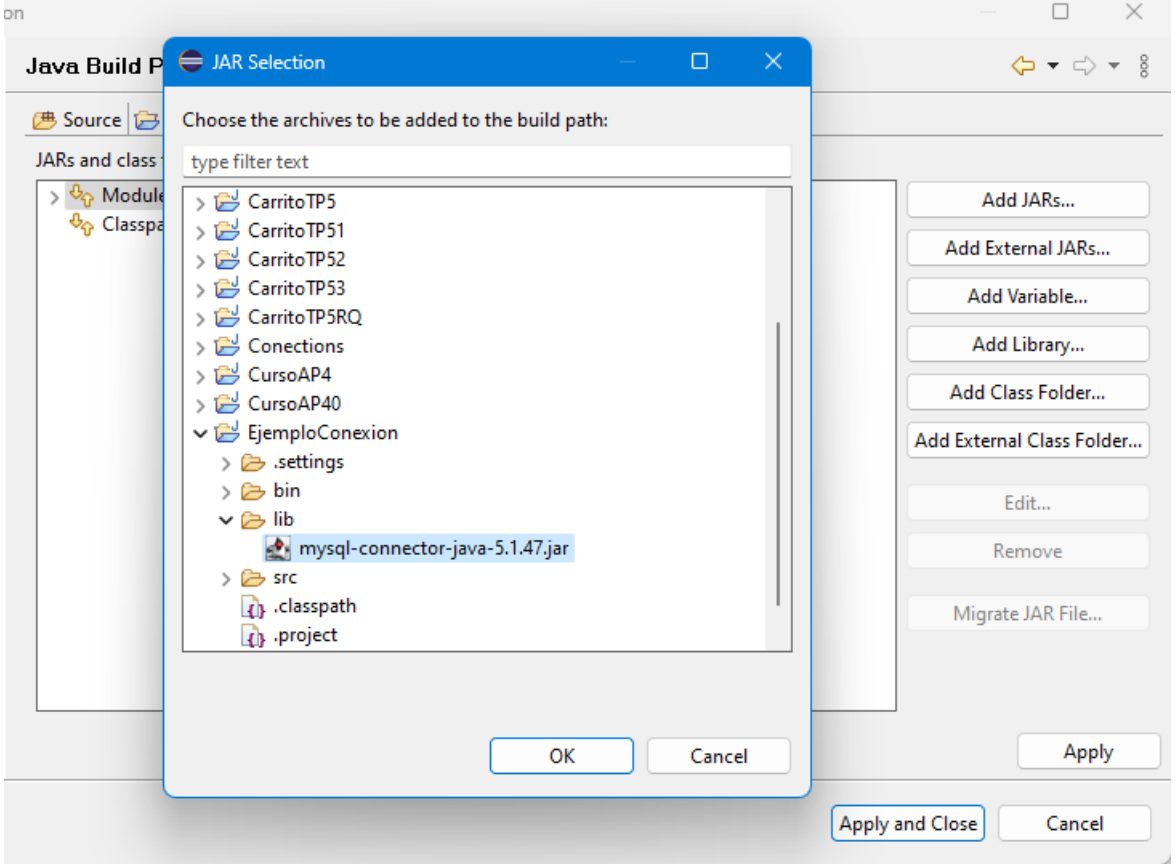


Click derecho sobre el proyecto, propiedades.

1- Se ejecuta la ventana y escogemos la opción Java Build Path..

2 - Luego la pestaña librerías

3 – Opción Add JARs



Argentina programa 4.0

Se abre una nueva ventana, buscamos la carpeta LIB, de nuestro proyecto, seleccionamos el conector. OK
Luego Aplicar y cerrar.

MySQL

Argentina programa 4.0

- **Vamos a MySQL y creamos una DB.**

```
create database carrito;

use carrito;
```

- **Creamos una tabla**

```
create table cliente(
ClienDni int(10)not null,
ClienNom varchar(30) not null,
ClienApe varchar(30) not null,
primary key(ClienDni)) engine=innodb;
```

Conectamos con MySQL

Argentina programa 4.0

Después de completar los procedimientos anteriores, escriba el código Java para la conectividad. Aquí, usamos:

- **el método class.forName() para cargar el JDBC Driver, que descargamos del sitio oficial de MySQL.**

- **El método getConnection() se utiliza para pasar la cadena de conexión:** MySQL:Port/Database/,username,dbpassword. Esta cadena se utiliza para autenticar al usuario y proporcionar acceso solo a usuarios autorizados.

- **Después de eso, usamos el método createStatement()** para crear una instancia que se usará para ejecutar consultas SQL usando el código.

nos a cargar el controlador, las clases que están dentro del JAR que agregamos recientemente.

Argentina programa 4.0

```
onexion.sql;
solo para establecer la conexion basica
.sql.Connection;
.sql.DriverManager;
.sql.SQLException;

s Conexion {
    public static void main(String[] args){
    try{
        Class.forName("com.mysql.jdbc.Driver"); // cargar el controlador/conector en memoria las clases necesarias para la conexion
        DriverManager.getConnection("jdbc:mysql://localhost:3306/qatar2022","root","pabLo987$"); // drivermanager en este método pasamos una cadena con los d

        System.out.println("Conexion exitosa!!");
        }catch (ClassNotFoundException e){ // sino se logra la conexión, con try Catch vamos a atrapar la excepción e interpretar el error al cargar el controlador
        System.out.println("Error al cargar el controlador!!!!");
        e.printStackTrace();
        }catch (SQLException e){
        System.out.println("Error en la conexion!!!!");
        e.printStackTrace();
    }
```

Probemos la conexion

Argentina programa 4.0

- Después de escribir el código, solo debes ejecutarlo. Si el código funciona bien, obtendrá el siguiente resultado en la consola:

Connected

JDBC

Argentina programa 4.0

- `Class.forName("com.mysql.jdbc.Driver");`
- `/*En esta línea es importante que indiquemos:`
- `el nombre de la base de datos --> qatar2022`
- `El usuario y contraseña que tengamos en nuestro gestor de base de datos MySQL*/`
- `conexion = DriverManager.getConnection("jdbc:mysql://localhost/carrito", "root", "pabLo987$");`
- `// aca ingresamos el camino y el nombre de la DB`

JDBC - Implementación

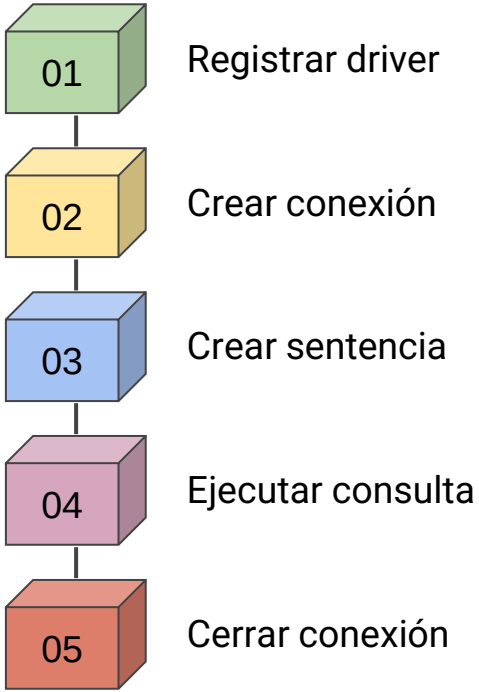
Argentina programa 4.0

- JDBC Driver es un componente de software que habilita a la aplicación java a interactuar con la DB.
- Existen 4 tipos de Drivers JDBC
  - JDBC-ODBC bridge driver
  - Native-API driver
  - Network Protocol driver
  - Thin driver

JDBC - Implementación

Argentina programa 4.0

Para interactuar con una DB debemos considerar 5 pasos:



JDBC - Implementación

Argentina programa 4.0

1. Registrar la 'clase' del driver - *Este método se utiliza para registrar la clase que se utilizará como driver.*

Sintaxis del método **forName()** method

```
public static void forName(String className)throws ClassNotFoundException
```

2. Crear el objeto de conexión - *Este método se utiliza para establecer conexión con la DB.*

Sintaxis del método **getConnection()** method

```
public static Connection getConnection(String url, String username, String password)throws SQLException
```

3. Crear la sentencia - *Este método es usado para crear la sentencia.*

Esta sentencia es la responsable de ejecutar las consultas a la DB.

Sintaxis del método **createStatement()** method

```
public Statement createStatement()throws SQLException
```

JDBC - Implementación

Argentina programa 4.0

4. Ejecutar consulta - *Este método devuelve el resultado de una consulta (filas).*

Sintaxis del método **executeQuery()** method

```
public ResultSet executeQuery(String sql)throws ClassNotFoundException
```

5. Cerrar la conexión - *Este método finaliza la conexión con la DB.*

Sintaxis del método **close()** method

```
public void close()throws ClassNotFoundException
```

JDBC - Conexión

Argentina programa 4.0

Para conectarnos a una DB (por ej. mysql) debemos conocer la siguiente información:

1. 'Clase' del driver: La clase para el driver de mysql es **`com.mysql.jdbc.Driver`**
2. URL de conexión: **`jdbc:mysql://localhost:3306/dbname`**
    - "**`jdbc`**" es la **API**
    - "**`mysql`**" es la **DB**
    - "**`localhost`**" es el nombre del **servidor** (se puede utilizar la dirección IP)
    - "**`3306`**" es el número de **puerto**
    - "**`dbname`**" es el nombre de la **DB**
3. Nombre de usuario: el usuario por defecto en mysql es **root**
4. Password: la contraseña asociada al usuario

JDBC - Conexión

Argentina programa 4.0

Ejemplo de conexión con *mysql*

```
import java.sql.*;
class MysqlCon{
      public static void main(String args[]){
            try{
                  Class.forName("com.mysql.jdbc.Driver");
                  Connection
            con=DriverManager.getConnection("jdbc:mysql://localhost:3306/dbname","root","password");
                  Statement stmt=con.createStatement();
                  //USO DE LA DB
                  con.close();
            } catch(Exception e){ System.out.println(e);}
      }
}
```

JDBC - Consultas

Argentina programa 4.0

Para realizar consultas a una DB utilizaremos los métodos de la interfaz **Statement**.

Los métodos más comunes son:

1. `public ResultSet executeQuery(String sql)`: utilizado para ejecutar sentencias SELECT. Devuelve un ResultSet.
2. `public int executeUpdate(String sql)`: utilizado para ejecutar sentencias CREATE, DROP, INSERT, UPDATE, DELETE, etc.

JDBC - Consultas

Argentina programa 4.0

Ejemplo de Statements con mysql

```
import java.sql.*;
class FetchRecord{
      public static void main(String args[])throws Exception{
            Class.forName("com.mysql.jdbc.Driver");
            Connection
            con=DriverManager.getConnection(“jdbc:mysql://localhost:3306/dbname","root","password”);
            Statement stmt=con.createStatement();

            ResultSet rs=stmt.executeQuery("select * from emp");
            while(rs.next())
            System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));

            int result=stmt.executeUpdate("delete from emp where id=33");
            System.out.println(result + " records affected");
            con.close();
      }
}
```

Referencias

Argentina programa 4.0

- https://www.xataka.com/basics/api-que-sirve
- https://www.javatpoint.com/java-jdbc
- https://dev.mysql.com/downloads/connector/j/
- https://codigoxules.org/conectar-mysql-utilizando-driver-jdbc-java-mysql-jdbc/

Argentina
programa
4.0

Gracias!