

## ACCESO A DATOS JAVA CON MYSQL

### Objetivo general

Al término de la separata el alumno conocerá:

- Crear base de datos usando MySQL
- Elaboración de una Clase Conexión
- Uso de funciones con valores de retorno.
- Aplicación con manipulación de datos en Mysql

1

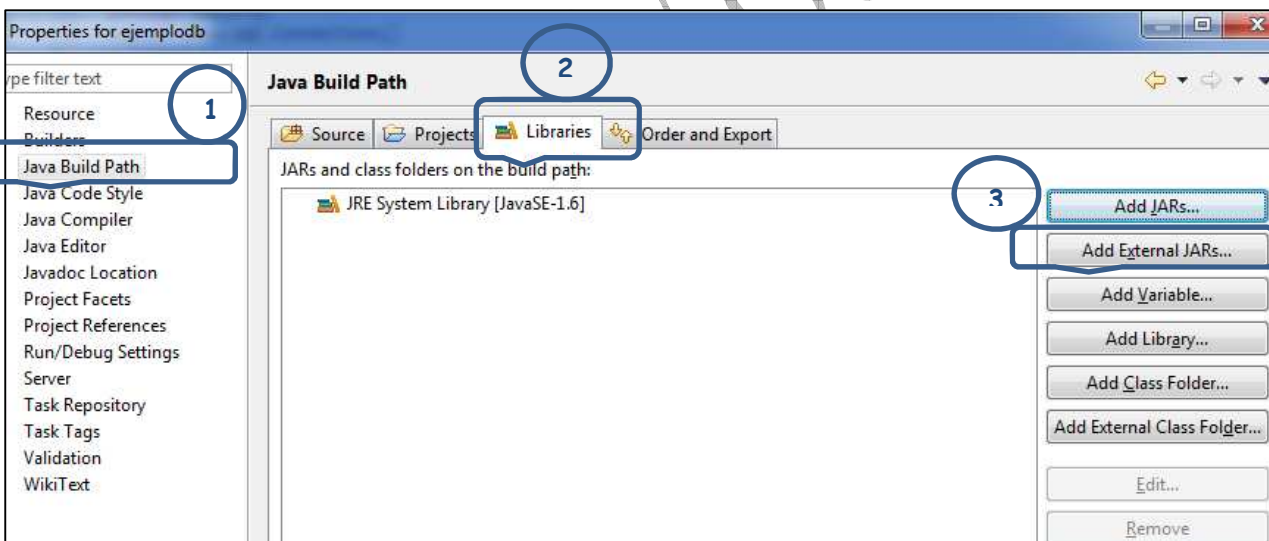
**Introducción:** En esta oportunidad vamos a conectar una aplicación en java con una Base de datos elaborado en el entorno Mysql, para eso necesitamos instalar el driver que contiene las clases que permite manipular datos en MySql.

### 1. Crear el Proyecto “JAVA\_MYSQL\_01”

#### 1.1. Instalar DRIVER o CONECTOR en el proyecto

##### Procedimientos o pasos a seguir:

- Menu *Projet* / Opción **Properties** / Mostrara la siguiente ventana:



- A) Un Clic en la Ficha **JavaBuild Path**
- B) Clic en la Ficha **Librerias**
- C) Clic en el **Boton Add External JARs**
- D) Buscar el Conector, Agregar y Aceptar.

#### 1.2. Habilitar los números de línea del editor de eclipse.

- A. Menú **Windows** / Opción **Preferences**
- B. Opcion **General** / **Editor** / **Text Editors** / Habilitar show line numbers

07

JAVA

PROGRAMING

LANGUAGE



- 1.3. Crear un paquete “**APLICACIÓN\_01**” en la carpeta **SRC**
- 1.4. Crear una Clase “**MYCONEXION**” en el paquete “**APLICACIÓN\_01**”
- 1.5. Importar los espacios de nombres.
- 1.6. Elaborar una función con retorno “**obtieneconexion()**” dentro de la clase.

```
package APLICACION_01;

// Importar el Espacio de Nombres DriverManager - Para el usar la Clase
DriverManager
import java.sql.DriverManager;
// Importar el Espacio de Nombres Connection - Para el usar la Clase Connection
import java.sql.Connection;

import javax.swing.JOptionPane;

public class MYCONEXION {
    //Elaborando una funcion con retorno de la cadena conexion
    public static Connection obtenelaconexion(){
        // Creando un objeto cn deriva de la clase "Connection"
        // Por defecto se asigna un valor nulo;
        Connection cn = null;

        try {
            //Cargary y Registrar el Driver de conexion (Clase Driver)
            Class.forName("com.mysql.jdbc.Driver");

            //Establecemos la conexion con la base de datos
            cn =
            DriverManager.getConnection("jdbc:mysql://localhost/bdEmpresa","root","");

            JOptionPane.showMessageDialog(null, "Conexion Exitosa");
        } catch (Exception e) {
            //Muestra el detalle del posible error
            e.printStackTrace();
        }

        return cn;
    }
}
```

- 1.7. Verificar si la Cadena de Conexión está correcto.  
Elaborar una clase “**DEMOSTRACION\_01.java**”

```
package APLICACION_01;
import java.sql.Connection;

public class DEMOSTRACION_01 {
    public static void main(String[] args) {
        ///Invocando al método que Conecta con BD

        // El Formulario ya tiene la conexión con la base de datos
        Connection con = MYCONEXION.obtenelaconexion();
    }
}
```



## APLICATIVOS DIGITALES V

1.8. Elaborar una aplicación donde **CONSULTE REGISTROS** a una tabla

A) Insertar una Clase "**DEMOSTRACION\_02.java**"

B) Importar el espacio de Nombres.

```
import java.sql.*;
```

C) Escribir las siguientes instrucciones.

```
public class DEMOSTRACION_02 {

    //1. Crear el Metodo main
    public static void main(String[] args) {

        //2. Crear el Controlador de Errores
        try {
            // A) Llamar al objeto Conexion
            Connection con = MYCONEXION.obtienelaconexion();
            // B) Crear el objeto Statement envia instruccion(SQL)
            Statement st = con.createStatement();
            // C) Crear un Objeto ResultSet para guardar Datos
            // D) Enviar una Consulta(SQL) - Traer Información
            ResultSet rs = st.executeQuery("Select * from tbpersonal");
            // E) Recorrer Registros
            while (rs.next()) {
                System.out.println(rs.getInt("Codigo") + "\t" + rs.getString("Nombres"));
            }

            } catch (Exception e) {
                // 3. Devolver Mensaje de Error en caso se produce el error
                System.out.println(e);
            }
        }

    }
}
```



## APLICATIVOS DIGITALES V

1.9. Elaborar una aplicación donde **INSERTE** información a una tabla

A. Insertar una Clase “**DEMOSTRACION\_03.java**”

B. Importar el espacio de Nombres.

```
import java.sql.*;
```

C. Escribir las siguientes instrucciones.

```
public class DEMOSTRACION_03 {  
  
    //1. Creando el método main  
    public static void main(String[] args) {  
  
        try {  
            // 2. Estableciendo Conexion:  
            Connection con = MYCONEXION.obtienelaconexion();  
  
            //2. Creacion de Statement  
            // Objeto Statement permite realizar enviar instrucciones sql (Gestión de sentencias)  
  
            Statement st = con.createStatement();  
            // El Metodo executeUpdate podemos hacer los insert o update a una tabla  
            st.executeUpdate("insert into tbpersonal values(250,'yolfer R',2500.50,26)");  
  
        } catch (Exception e) {  
            // Muestra un mensaje si hay un error  
            System.out.println(e);  
        }  
    }  
}
```

07

JAVA

LANGUAGE

PROGRAMING



## 2. Elaboración de una aplicación grafica que realice mantenimiento a una tabla.

- A. Insertar una Clase “**MANTENIMIENTO\_PERSONAL.java**”
- B. Importar el espacio de Nombres.

```
import java.sql.*;
```

- C. Establecer la cadena de Conexión en el formulario.
- D. Elaborar el método que va a contener el diseño del formulario. Vea la imagen:

```
package APLICACION_01;
import java.sql.*;
import javax.swing.JFrame;

public class MANTENIMIENTO_PERSONAL extends JFrame {

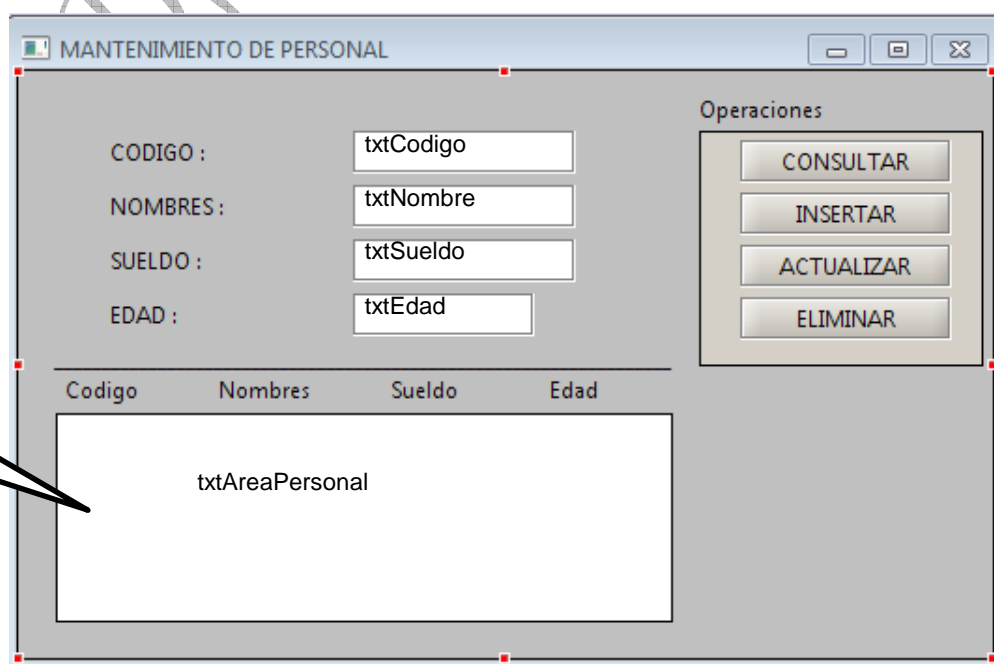
    //ESTABLECIENDO LA CONEXION AL FORMULARIO CON MYSQL
    Connection con = MYCONEXION.obtienelaconexion();

    //DISEÑO DEL FORMULARIO
    public MANTENIMIENTO_PERSONAL(){

    }

}
```

- E) Ubicar el cursor en el cuerpo del método **MANTENIMIENTO\_PERSONAL()** para la correcta elaboración del formulario.
- F) Abrir el la Clase “**MANTENIMIENTO\_PERSONAL.java**” en el Editor de Formulario y Diseñe la siguiente interfaz.




## A) Programar en evento "ActionPerformed" del botón "CONSULTAR"

```
try {
    String sql = "Select * From tbPersonal";
    // 1. Preparamos la instruccion SQL para ser ejecutada
    PreparedStatement pst = con.prepareStatement(sql);

    // 2. Ejecutamos la instruccion SQL y los Registros
    // 3. Una Vez Seleccionados se almacenan en un objeto resultset
    // 4. El objeto ResultSet almacena a todo los registros Seleccionados

    // 4. Metodo executeQuery() Solo para las consultas
    ResultSet rs = pst.executeQuery();

    // el objeto ResultSet es tabla en memoria
    // Recorre todo el objeto ResultSet e imprime registro por registro

    while(rs.next()){
        // avanza eh imprime hasta que no encuentre registros next() = False
        txtAreaPersonal.append(rs.getInt(1)+ "\t" + rs.getString(2)+ "\t" +
        rs.getDouble(3)+ "\t" + rs.getInt(4)+ "\n");
    }

    } catch (Exception e) {
        // Controlar Errores
        e.printStackTrace();
    }
}
```

## B) Programar en evento "ActionPerformed" del botón "INSERTAR"

```
try {
    String sql="insert into tbpersonal values(?,?,?,?)";
    PreparedStatement pst= con.prepareStatement(sql);
    // 1. Asignando valores a los parametros
    pst.setInt(1,Integer.valueOf(txtcodigo.getText()));
    pst.setString(2,txtnombre.getText());
    pst.setDouble(3,Double.valueOf(txtsueldo.getText()));
    pst.setInt(4,Integer.valueOf(txtedad.getText()));

    // 2. Ejecutar La instrucción
    pst.executeUpdate();
    // Listar los Datos
    txtAreaPersonal.setText(""); // Ud. Elabore el método listarRegistro();
    listarRegistro();
    JOptionPane.showMessageDialog(null,"Registro Ok");

    } catch (Exception e) {
        // Controlando Errores
        e.printStackTrace();
    }
}
```



## C) Programar en evento "ActionPerformed" del botón "ACTUALIZAR"

```
try {
    String sql="update producto set nomproducto='" + txtprod.getText() +
        "','" + "precio=" + txtprecio.getText() +
        " where idproducto="+txtcod.getText();
    PreparedStatement pst= con.prepareStatement(sql);
    //2. Ejecutar La instruccion
    pst.executeUpdate();
    //3. Listar los Datos
    txtareaproducto.setText("");
    JOptionPane.showMessageDialog(null,"Registro Actualizado Ok");
} catch (Exception e) {
    System.out.println(e);
}
}
```

## D) Programar en evento "ActionPerformed" del botón "ELIMINAR"

```
try {

    String cod = JOptionPane.showInputDialog("Codigo: ");
    int codi = Integer.valueOf(cod);

    String sql="delete from tbpersonal where codigo=?";

    PreparedStatement pst= con.prepareStatement(sql);
    //1. Asignando valores a los parametros
    pst.setInt(1,codi);

    //2. Ejecutar La instruccion

    pst.executeUpdate();
    //3. Listar los Datos
    txtAreaPersonal.setText("");
    listarRegistro();
    JOptionPane.showMessageDialog(null,"Registro Eliminado Ok");
} catch (Exception e) {
    // TODO: handle exception
    e.printStackTrace();
    JOptionPane.showMessageDialog(null,"Existe un Error");
}
}
```

Tú futuro exitoso depende de tu presente, y tu presente lo diseñas tú con cada decisión que tomes

Conmigo hasta la próxima XD:

