Mantenimiento de datos base de datos Java MySQL.

Para conectarnos a una base de datos de MySQL desde Java, lo primero que debemos de tomar en cuenta es la forma en que debemos realizar este procedimiento.

Java usa una característica denominada JDBC[1], es la tecnología que sirve de intermediaria entre los gestores de base de datos y Java, este implementa drivers que nos permiten poder comunicarnos entre los componentes que se acaban de mencionar, JDBC nos facilita el uso de cualquier servidor sin cambiar prácticamente nada en el código, ya que este es un estándar para los gestores para que ellos desarrollen las clases de los drivers sin tener que preocupar al desarrollador por estar adaptando sus códigos para el gestor que seleccione.

En nuestro caso al usar MySQL nos permite implementar el driver el cual por lo general se instala junto con el NetBeans el IDE que estamos utilizando, si en algún caso no está presente o no se instala lo podemos descargar de la siguiente dirección.

http://dev.mysql.com/downloads/connector/j/.

Conexión a la Base de datos desde el código.

Para conectarnos seguiremos los siguientes pasos.

1 - Crear una un paquete con nombre Base y clase dentro del paquete, la cual denominaremos Coneccion.

2 - importamos los paquetes que nos permitirán usar las características de JDBC

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
```

3 – Con los paquetes importados dentro de la clase declaramos 3 variables y un método el cual nos permitirá implementar un método que devolverá la conexión a la base de datos.

```
private final String user = "root";
private final String host = "jdbc:mysql://localhost";
private final String clave = "";

public Connection Conectar(){

}
```

[1] http://es.wikipedia.org/wiki/Java_Database_Connectivity

4 - Definimos el siguiente código.

```
Connection cn = null;
try {
    Class.forName("com.mysql.jdbc.Driver");
    cn = DriverManager.getConnection(host, user, clave);
} catch (ClassNotFoundException ex1) {
    System.out.println("Problemas con Clase contacta al administrador: ");
    System.out.println(ex1.getMessage());
    cn = null;
}catch ( SQLException ex2) {
    System.out.println("Problemas con Coneccion conactacta al administrador: ");
    System.out.println(ex2.getMessage());
    cn = null;
}
return cn;
```

Para que es el siguiente código.

a) Definimos la variable cn del tipo Connection y la igualamos a null.
b) Definimos un manejador de excepciones.
c) Dentro del bloque try definimos, un llamado a la clase que contiene el driver de mysql
d) Definimos la sentencia que permite realizar la conexión a la base de datos.
e) Creamos el bloque catch que manejara los problemas de la carga del driver (clase).
f) Creamos el boque catch que manejara los problemas referentes a la conexión.
g) Retornamos la conexión que esta almacena en la variable cn.

Probando la clase.

Ahora realizaremos una prueba la cual nos permitirá saber si nuestro código es funcional y nos podemos conectar a la base de datos.

1 – Creamos un paquete el cual llamaremos vista y agregamos a él una clase llamada Consola.

2 – Agregamos el siguiente código.

```
package vista;
import Base.Coneccion;
public class Consola {
    //Metodo para comprobar coneccion solo lo usaremos
    //en entornos de prueba.
    void probarConeccion(){
        Coneccion con = new Coneccion();
        if (con.Conectar() != null) {
            System.out.println("Conexion Exitosa");
        }
    }
    public static void main(String[] args) {
        Consola pb = new Consola();
        pb.probarConeccion();
    }
}
```

3 - Ejecutamos el archivo para verificar los resultados y depurar si fuese el caso.

Nota: si el código no se ejecuta o da un problema de librería debemos agregar la librería del driver dando clic derecho sobre la carpeta Libraries del proyecto ahí seleccionamos Add Library, en el cuadro de dialogo seleccionamos la librería MySQL JDBC Driver y damos clic en add.

Mantenimiento de Datos.

Para conectarnos a la base de datos crearemos un patrón basado en MVC, el cual nos permitirá poder hacer y separar la lógica de las funciones del código, así como el control de los datos de u na forma más práctica.

Creando clase para manejar modelos.

1 - Crearemos un paquete llamado Modelos, y dentro de este una clase Prueba.

2 - Creamos las variables para el control de los datos, los cuales serán similares o harán referencia a los campos que tenemos en la tabla de la base de datos.

```
package Modelos;
public class Prueba {
    private int id, estado;
    private String nombre, apellido, email;


}
```

3 – Ahora nos creamos los métodos set y get necesarios para encapsular los campos que tenemos agregamos el siguiente código por cada campo (variable).

```
public int getId() {
    return id;
}

/**
 * @param id the id to set
 */
public void setId(int id) {
    this.id = id;
}
```

```
public int getEstado() {
    return estado;
}

/**
 * @param estado the estado to set
 */
public void setEstado(int estado) {
    this.estado = estado;
}
```

```
public String getNombre() {
    return nombre;
}

/**
 * @param nombre the nombre to set
 */
public void setNombre(String nombre) {
    this.nombre = nombre;
}
```

```
public String getApellido() {
    return apellido;
}

/**
 * @param apellido the apellido to set
 */
public void setApellido(String apellido) {
    this.apellido = apellido;
}
```

```
public String getEmail() {
    return email;
}

/**
 * @param email the email to set
 */
public void setEmail(String email) {
    this.email = email;
}
```

Estos son los métodos que nos permiten proteger nuestros campos podemos, acortar el proceso de crearlos si damos clic derecho buscamos la opción refactor, al expandirse buscar la opción Encapsulate Fields… y en el cuadro de dialogo damos clic en la opción selecta ll, y damos clic en finish.

Creando la clase para controlar los métodos para mantenimiento de la base de datos.

1 - Creamos el paquete Controlador y dentro de creamos la clase Mantenimiento.

2 – En la clase primero crearemos un método para insertar un nuevo registro nuestro código quedara así.

```
public class Mantenimiento{
    Coneccion con = new Coneccion();
    PreparedStatement ps = null;
    String msg = null;
    public void Insertar(String nombre,String apellido,String email,int estado){
        try {
            String sql = "INSERT INTO prueba ";
                   sql +="(idPrueba,nombre, apellido, email,estado) ";
                   sql += "VALUES(null,?,?,?,?)";
            ps = con.Conectar().prepareStatement(sql);
            ps.setString(1, nombre);
            ps.setString(2, apellido);
            ps.setString(3, email);
            ps.setInt(4, estado);
            if(ps.executeUpdate()== 1){
                msg = "Datos Almacenados";
                System.out.println(msg);
            }
        } catch (SQLException ex) {
               msg = "Error al Guardar los datos";
               msg += ex.getMessage();
               System.out.println(msg);
        }

    }
}
```

Detalles del código.

a) Creamos unas variables que nos servirán para definir elementos.
   a. El objeto de la conexión.
   b. Un objeto para manipular la consulta sql.
   c. Una cadena para controlar los mensajes.
b) Creamos el método para Insertar y le pasamos como parámetros los valores que vamos a almacenar en la tabla.
c) Definimos el manejador de excepciones.
d) En el bloque try creamos una cadena que almacena la sentencia de SQL, pero definimos unos comodines, estos son el símbolo de ? ya que el objeto PreparedStatement trata o procesa la sentencia, sustituyendo dichos comodines por los valores que se pasan en los métodos de su implementación.

e) Preparamos el statement, primero lo iniciamos al objeto de la coneccion y abstraemos el objeto pasándole la variable sql.
f) Después definimos los parámetros que se sustituirán internamente
g) Creamos una condicional para verificar si se ejecuta el método executeUpdate(), que es el encargado de disparar, la ejecución de la consulta.
h) Enviamos un mensaje para definir que se registraron los datos.
i) En el bloque catch se definen las instrucciones si no se pueden guardar los datos.

3 – Ahora probaremos si todo lo creado funciona nos dirigimos a la clase consola y la editamos para que el código quede de la siguiente forma.

```
package vista;
import Controladores.Mantenimiento;
import Modelos.Prueba;
import java.util.Scanner;
public class Consola {
    //Metodo para comprobar coneccion solo lo usaremos
    //en entornos de prueba.
    void probarInsertar(){
        Scanner st = new Scanner(System.in);
        Prueba datos = new Prueba();
        System.out.println("Escriba el Nombre: ");
        datos.setNombre(st.next());
        System.out.println("Escriba el Apellido: ");
        datos.setApellido(st.next());
        System.out.println("Escriba el Correo: ");
        datos.setEmail(st.next());
        System.out.println("Escriba el Estado debe ser 1 o 0: ");
        datos.setEstado(st.nextByte());
        Mantenimiento mt = new Mantenimiento();
        mt.Insertar(datos.getNombre(),
                    datos.getApellido(),
                    datos.getEmail(),
                    datos.getEstado()
        );
    }
    public static void main(String[] args) {
        Consola pb = new Consola();
        pb.probarInsertar();
    }
}
```

Nota aquí hay que hacer una modificación en la variable host de la clase Coneccion en el paquete base, la línea quedara así.

```
private final String host = "jdbc:mysql://localhost/prueba";
```

Consultar los datos.

Para consultar los datos crearemos un método en la clase Mantenimiento con el siguiente código.

```
public void VerRegistros(){
    ResultSet rs = null;
    try {
        ps = con.Conectar().prepareStatement("SELECT * FROM prueba;");
        rs = ps.executeQuery();
        System.out.println("Id\t\tNombre\t\tApellidos\t\tMail\t\tEstado");
        while (rs.next()) {
            System.out.print(rs.getInt(1)+"\t\t");
            System.out.print(rs.getString(2)+"\t\t");
            System.out.print(rs.getString(3)+"\t\t");
            System.out.print(rs.getString(4)+"\t\t");
            System.out.print(rs.getInt(5)+"\n");
        }
    } catch (SQLException e) {
        System.out.println("Error " + e.getMessage());
    }
}
```

Lo nuevo que agregamos es un objeto ResultSet el cual será el encargado de controlar los datos que se consultan.

En el archivo consola probaremos el resultado del método agregando el siguiente código.

```
void probarResultados(){
    Mantenimiento mt = new Mantenimiento();
    mt.VerRegistros();
}
```

Y en el método main cambiaremos la línea de pb.probarInsertar() por pb.probarResultados().

Tarea.

Crear los métodos para modificar, eliminar Registros y además un método para buscar solamente un registró.

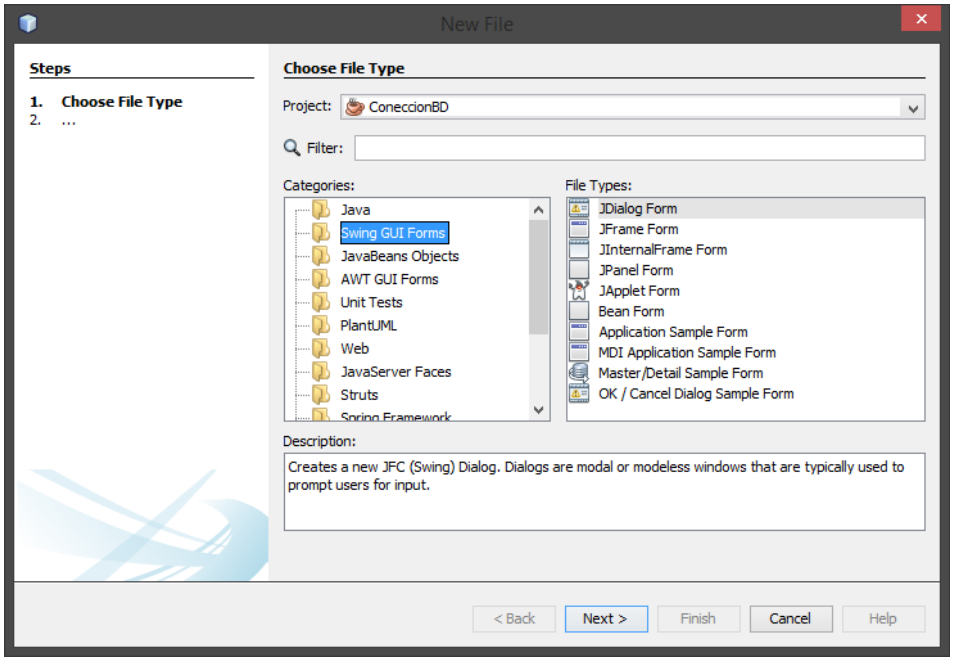
Usemos formularios.

En los anteriores ejemplos solo se a utilizado la consola de java para hacer las pruebas ahora crearemos una pequeña interfaz gráfica reutilizando las clases que hemos hecho en el anterior ejercicio pero las pondremos a disposición de una GUI usando Jform y las clases que implementa swing, el cual es el paquete que contiene todas las clases para la creación de formularios, botones y otros controles.

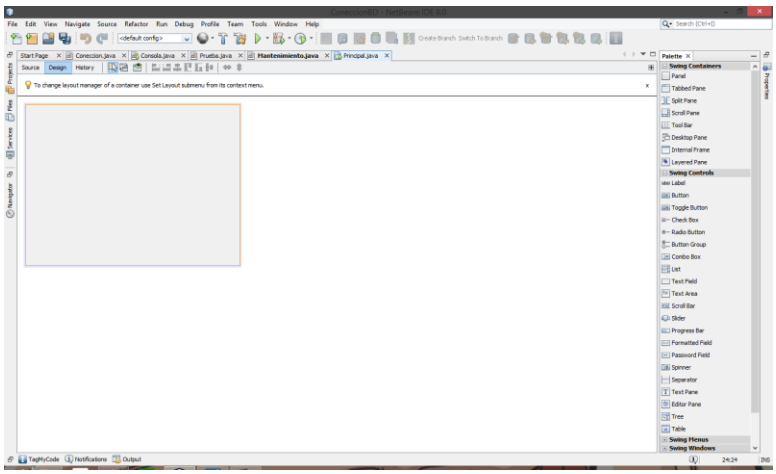
Creando un formulario.

Para crear un formulario realizaremos los siguientes pasos.

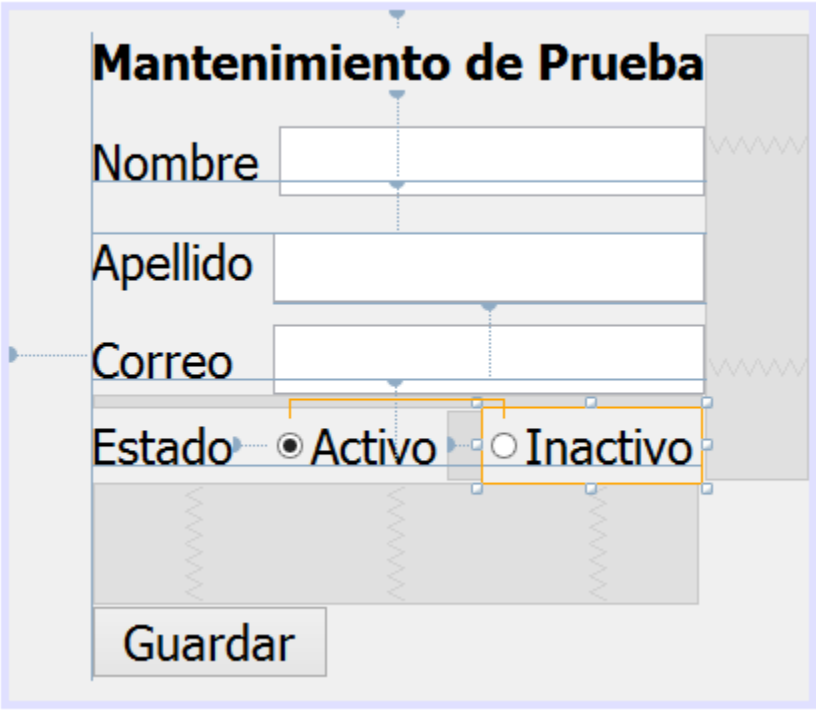
1 – vamos a archivo y seleccionamos un nuevo archivo, en las categorías vamos a seleccionar swing gui forms y en el tipo elegimos Jframe Form .



2 - Al elegir las opciones adecuadas damos clic en next y colocamos el nombre de Principal y lo asociamos al paquete vista damos clic en finish, al hacer esto Netbeans cargara una vista de diseño donde podremos crear de una manera gráfica nuestro, formulario, se carga además una paleta de controles, que al arrastrarlos o dando doble clic los veremos en nuestro formulario o ventana de diseño.



3 - Creamos el siguiente formulario.



Ahora a codificar lo que haremos es dar doble clic en el componente del botón con nombre Guardar y agregamos el siguiente código.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Mantenimiento mt = new Mantenimiento();
    Prueba datos = new Prueba();
    int estado=0;
    String msg = null;
    datos.setNombre(jTextField1.getText());
    datos.setApellido(jTextField2.getText());
    datos.setEmail(jTextField3.getText());

    if(jRadioButton1.isSelected())estado = 1;
    if(jRadioButton2.isSelected())estado = 0;
    datos.setEstado(estado);
   msg= mt.Insertar(
            datos.getNombre(),
            datos.getApellido(),
            datos.getEmail(),
            datos.getEstado()
    );
    javax.swing.JOptionPane.showMessageDialog(rootPane, msg);
    jTextField1.setText("");
    jTextField2.setText("");
    jTextField3.setText("");
    jRadioButton1.setSelected(true);
}
```

4 – haremos una pequeña modificación al código del método Insertar de la clase Mantenimiento en el paquete controlador. Quedará así.

```
public String Insertar(String nombre,String apellido,String email,int estado){
    try {
        String sql = "INSERT INTO prueba ";
               sql +="(idPrueba,nombre, apellido, email,estado) ";
               sql += "VALUES(null,?,?,?,?)";
        ps = con.Conectar().prepareStatement(sql);
        ps.setString(1, nombre);
        ps.setString(2, apellido);
        ps.setString(3, email);
        ps.setInt(4, estado);
        if(ps.executeUpdate()== 1){
            msg = "Datos Almacenados";
        }
    } catch (SQLException ex) {
           msg = "Error al Guardar los datos";
           msg += ex.getMessage();
    }
    return msg;
}
```

Nota si no queremos afectar la Clase consola entonces debemos de modificarla también para poder que se adapte a la devolución de la cadena de datos del método. Se puede codificar de esta manera.

```
String probarInsertar(){
    Scanner st = new Scanner(System.in);
    Prueba datos = new Prueba();
    System.out.println("Escriba el Nombre: ");
    datos.setNombre(st.next());
    System.out.println("Escriba el Apellido: ");
    datos.setApellido(st.next());
    System.out.println("Escriba el Correo: ");
    datos.setEmail(st.next());
    System.out.println("Escriba el Estado debe ser 1 o 0: ");
    datos.setEstado(st.nextByte());
    Mantenimiento mt = new Mantenimiento();
    return mt.Insertar(datos.getNombre(),
                       datos.getApellido(),
                       datos.getEmail(),
                       datos.getEstado()
    );
}
```

Ahora en el método main de la clase no debemos poner la línea pb.probarInsertar(); sino que deberá quedar así System.out.println(pb.probarInsertar());

Tareas.

- Investigar el Uso del JTable de Swing
- Investigar cómo usar un JCombo con datos extraídos de una tabla.
- Investigar como poder almacenar una fecha en la base de datos.
- Crear un mantenimiento en el cual se registren los datos de un entrenador Pokemon y su equipo, cree la base de datos con su normalización y cree cada modelo y controlador dependiendo de las tablas que tenga su modelo de base de datos.