


## COMPONENTE JAVAFX TableView: carga datos mediante JDBC

El componente TableView permite **visualizar un conjunto amplio de datos en forma de tabla**, con las siguientes funcionalidades:

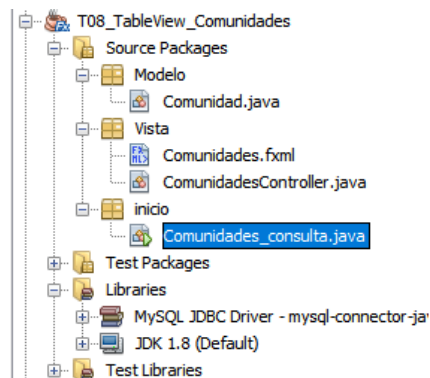
- permite cambiar el orden de cada columna pulsando sobre el encabezado de la columna
- ocultar/visualizar las columnas que se deseen mediante el icono 
- selección única ó múltiple de filas

Visualización de todas las comunidades usando Table...

Total 5 comunidades

Total	id	nombre comunidad	+
5	1	Andalucia	
10	2	Aragon	
20	4	Canarias	
15	3	Castilla La Mancha	
25	5	Comunidad Valenciana	

La estructura en capas de este proyecto será como la imagen:



## TEMA 8. CONEXIÓN A BDA Uso de componente TableView

**A.** En el método **initialize()** del Controller, tendremos el siguiente código:

- Obtenemos todas las comunidades usando: método **List<Comunidad> buscarComunidades()**
  - se prepara sentencia SQL de búsqueda (**select**) y se ejecuta (**executeQuery**),
  - se **recorre** los datos devueltos (ResultSet), y **crea y carga objetos** Comunidad en una colección (List<Comunidad>),
- se crea una **ObservableList** basada en el List<Comunidad> devuelto, y se **asigna al componente TableView** mediante el método **.setItems()**
- Se **asocia** cada columna/TableColumn ( **id.setCellValueFactory()** ) con un campo de la clase Comunidad (**new PropertyValueFactory<>("id")** )

```
29 public class ComunidadesController implements Initializable {
30     private ObservableList<Comunidad> listaComunidades;
31     @FXML
32     private Label total;
33     @FXML
34     private TableView<Comunidad> comunidadesTableView;
35     @FXML
36     private TableColumn<Comunidad, Integer> id;
37     @FXML
38     private TableColumn<Comunidad, String> nombre;
39     @FXML
40     private TableColumn<Comunidad, Integer> calculateColumn;
41
42     @Override
43     public void initialize(URL url, ResourceBundle rb) {
44         try {
45             //PASAMOS LOS DATOS A UNA ObservableList
46             listaComunidades = FXCollections.observableArrayList(buscarComunidades());
47
48             //ASIGNAMOS LOS DATOS A LA TABLEVIEW
49             comunidadesTableView.setItems(listaComunidades);
50
51             //PERMITIMOS SELECCION MULTIPLE
52             comunidadesTableView.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);
53
54             //ASOCIAMOS LAS COLUMNAS (TableColumn) CON LOS ATRIBUTOS/CAMPOS/PROPIEDADES DE LA CLASE Comunidad
55             id.setCellValueFactory(new PropertyValueFactory<>("id"));
56             nombre.setCellValueFactory(new PropertyValueFactory<>("nombre"));
57             //ESTA COLUMNA ES CALCULADA EN EL CONSTRUCTOR DE LA CLASE Comunidad
58             calculateColumn.setCellValueFactory(new PropertyValueFactory<>("total"));
59
60             total.setText("Total " + listaComunidades.size() + " comunidades");
61         } catch (Exception e) {
62             //
63         }
64     }
65 }
```

- B.** El método relativo a JDBC que establece conexión, prepara la sentencia y la lanza. Java cierra los recursos al terminar mediante el try-with-resources:

```
public List<Comunidad> buscarComunidades() throws SQLException {
    List<Comunidad> comunidades = new ArrayList<>();
    String consulta = "select * from comunidad ORDER BY nombre";
    String bd = "parques";
    String usuario = "root";
    String password = "";
    String ruta = "jdbc:mysql://localhost:3306/" + bd + "?serverTimezone=UTC";

    try (Connection conexion = DriverManager.getConnection(ruta, usuario, password);
        PreparedStatement ps = conexion.prepareStatement(consulta, ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_READ_ONLY);
        ResultSet rs = ps.executeQuery();) {

        while (rs.next()) {
            //CREAMOS EL OBJETO COMUNIDAD CON LOS DATOS DEL RESULTSET
            Comunidad comunidad = new Comunidad(rs.getInt("id"), rs.getString("nombre"));

            //ALMACENAMOS CADA OBJETO COMUNIDAD EN LA COLECCION
            comunidades.add(comunidad);
        }
    }

    return comunidades;
}
```

- C.** Definición de la clase Comunidad, con un campo *total* calculado en función del resto de campos:

```
public class Comunidad {
    private int id;
    private String nombre;
    private int total;

    public Comunidad(int id, String nombre) {
        this.id = id;
        this.nombre = nombre;
        this.total = id * 5; //CAMPO CALCULADO
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getTotal() {
        return total;
    }

    public void setTotal(int total) {
        this.total = total;
    }
}
```