

Manual Técnico

Juan Pablo de León Miranda

Introducción a la Programación y Computación 2

conexionDB

DatabaseConnection

- **URL:** La cadena de conexión `jdbc:mariadb://localhost:3306/proyecto1IPC2` especifica que estamos utilizando el controlador de MariaDB, que se encuentra en localhost (el mismo equipo donde se ejecuta la aplicación) y se conecta al puerto 3306 a la base de datos llamada proyecto1IPC2.
- **USER y PASSWORD:** Aquí se deben ingresar las credenciales correctas para acceder a la base de datos. Es importante asegurarse de que el usuario tenga los permisos necesarios.
- **Método getConnection():** Este método es crucial, ya que es el que realmente establece la conexión. Si la conexión es exitosa, se devuelve un objeto Connection. Si hay un problema, se lanza una excepción que puede ser manejada en el código que llama a este método.

```
1 public class DatabaseConnection {
2     private static final String URL = "jdbc:mariadb://localhost:3306/proyecto1IPC2";
3     private static final String USER = "juanpa"; // Reemplaza con tu usuario de MariaDB
4     private static final String PASSWORD = "hiraimomo"; // Reemplaza con tu contraseña de MariaDB
5
6     public static Connection getConnection() throws SQLException {
7         return DriverManager.getConnection(URL, USER, PASSWORD);
8     }
9 }
```

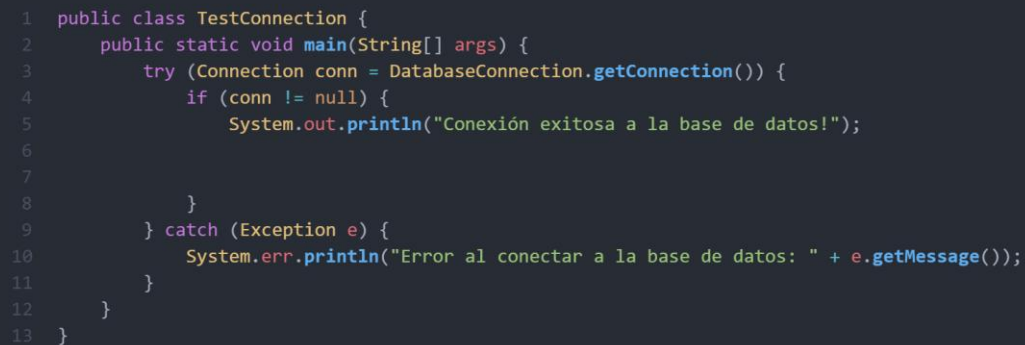
TestConnection

1. Bloque try-with-resources:

- `Connection conn = DatabaseConnection.getConnection();` Se intenta obtener una conexión a la base de datos utilizando un método estático `getConnection()` de la clase `DatabaseConnection`. Este método debe devolver un objeto de tipo `Connection`.
- Si la conexión es exitosa se imprime un mensaje de éxito en la consola.

2. Manejo de Excepciones:

- Si ocurre un error al intentar conectarse a la base de datos, se captura la excepción y se imprime un mensaje de error en la consola, mostrando el mensaje específico del error.

A screenshot of a code editor with a dark background and light-colored text. The code is in Java and defines a class named TestConnection. It has a main method that attempts to establish a database connection using DatabaseConnection.getConnection(). If the connection is successful (conn != null), it prints "Conexión exitosa a la base de datos!". If an exception occurs, it catches the Exception and prints "Error al conectar a la base de datos: " followed by the exception's message. The code is numbered from 1 to 13 on the left side of the editor.

```
1 public class TestConnection {  
2     public static void main(String[] args) {  
3         try (Connection conn = DatabaseConnection.getConnection()) {  
4             if (conn != null) {  
5                 System.out.println("Conexión exitosa a la base de datos!");  
6             }  
7         }  
8     } catch (Exception e) {  
9         System.err.println("Error al conectar a la base de datos: " + e.getMessage());  
10    }  
11 }  
12 }  
13 }
```

DAO

LoginDAO - validate

1. **Carga del Controlador:** `Class.forName("org.mariadb.jdbc.Driver");` carga el controlador necesario para conectarse a la base de datos MariaDB.
2. **Conexión a la Base de Datos:** Se establece una conexión utilizando `DriverManager.getConnection()`, donde se especifica la URL de la base de datos, el usuario y la contraseña.
3. **Consulta SQL:** Se prepara una consulta SQL que busca un usuario con el nombre de usuario y la contraseña proporcionados. Los signos de interrogación (?) son marcadores de posición que se reemplazarán con los valores reales.
4. **Ejecución de la Consulta:** `preparedStatement.executeQuery()` ejecuta la consulta y devuelve un `ResultSet`. Si `rs.next()` devuelve true, significa que se encontró un usuario que coincide con las credenciales.
5. **Manejo de Excepciones:** Si ocurre un error durante la conexión o la consulta, se captura la excepción SQL y se maneja adecuadamente.

```
1  public static boolean validate(LoginBean loginBean) throws ClassNotFoundException, SQLException {
2      boolean status = false;
3
4      // Cargar el controlador de MariaDB
5      Class.forName("org.mariadb.jdbc.Driver");
6
7      // Establecer la conexión a la base de datos
8      try (Connection connection = DriverManager.getConnection(
9          "jdbc:mariadb://localhost:3306/proyecto1IPC2", "root", "hiraimomo")) {
10
11          // Consulta SQL para validar el usuario y la contraseña
12          String sql = "SELECT * FROM usuario WHERE nombreUsuario = ? AND password = ?";
13          try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
14              preparedStatement.setString(1, loginBean.getUsername());
15              preparedStatement.setString(2, loginBean.getPassword());
16
17              System.out.println(preparedStatement);
18              ResultSet rs = preparedStatement.executeQuery();
19              status = rs.next(); // Si hay un resultado, el usuario es válido
20          }
21      } catch (SQLException e) {
22          // Manejo de excepciones SQL
23          printSQLException(e);
24      }
25      return status;
26  }
```

LogInDAO – printSQLException

- **Iteración sobre Throwable:** El bucle for permite recorrer todos los errores relacionados con la excepción SQL. Esto es útil para capturar múltiples errores que pueden haber ocurrido.
- **Verificación de SQLException:** La condición if (e instanceof SQLException) asegura que solo se manejen objetos de tipo SQLException, evitando así errores de tipo.
- **Impresión de Detalles:** Utiliza printStackTrace(System.err) para imprimir la traza de la excepción en la salida de error estándar. Esto es crucial para el diagnóstico.
- **Información Adicional:** Se imprimen el estado SQL, el código de error y el mensaje de error, lo que proporciona contexto sobre el problema.
- **Causas de la Excepción:** El bloque while permite rastrear la causa de la excepción, lo que puede ser invaluable para resolver problemas complejos.

```
1 private static void printSQLException(SQLException ex) {
2     for (Throwable e : ex) {
3         if (e instanceof SQLException) {
4             e.printStackTrace(System.err);
5             System.err.println("SQLState: " + ((SQLException) e).getSQLState());
6             System.err.println("Error Code: " + ((SQLException) e).getErrorCode());
7             System.err.println("Message: " + e.getMessage());
8             Throwable t = ex.getCause();
9             while (t != null) {
10                 System.out.println("Cause: " + t);
11                 t = t.getCause();
12             }
13         }
14     }
15 }
```

Servlets

loginSV

- **processRequest(request, response);**: Este método se llama para procesar la solicitud antes de manejar la autenticación.
- **request.getParameter("username")**: Recupera el nombre de usuario enviado desde el formulario.
- **LoginDao.validate(loginBean)**: Este método verifica si las credenciales son correctas. Si lo son, se crea una sesión y se redirige al usuario a una página de éxito.
- **session.setAttribute("error", "Usuario o contraseña incorrectos")**: Si las credenciales son incorrectas, se almacena un mensaje de error en la sesión y se redirige al usuario de vuelta al formulario de inicio de sesión.

```
1  @Override
2  protected void doPost(HttpServletRequest request, HttpServletResponse response)
3      throws ServletException, IOException {
4      processRequest(request, response);
5      String username = request.getParameter("username");
6      String password = request.getParameter("password");
7      LoginBean loginBean = new LoginBean();
8      loginBean.setUsername(username);
9      loginBean.setPassword(password);
10
11     try {
12         if (LoginDao.validate(loginBean)) {
13             HttpSession session = request.getSession();
14             session.setAttribute("username", username);
15             response.sendRedirect("loginsuccess.jsp"); // Redirigir a la página de éxito
16         } else {
17             HttpSession session = request.getSession();
18             session.setAttribute("error", "Usuario o contraseña incorrectos");
19             response.sendRedirect("login.jsp"); // Redirigir de vuelta al login con un mensaje de error
20         }
21     } catch (ClassNotFoundException e) {
22         e.printStackTrace();
23     } catch (SQLException ex) {
24         Logger.getLogger(loginSV.class.getName()).log(Level.SEVERE, null, ex);
25     }
26
27 }
```

funcions

LoginBean

Es un modelo básico de usuario. Este modelo incluye atributos para el nombre de usuario y la contraseña, así como métodos para acceder y modificar estos atributos



```
1  private String username;
2      private String password;
3
4      public String getUsername() {
5          return username;
6      }
7
8      public void setUsername(String username) {
9          this.username = username;
10     }
11
12     public String getPassword() {
13         return password;
14     }
15
16     public void setPassword(String password) {
17         this.password = password;
18     }
```