TP 7 : Utilisation de JDBC et de JavaFX

Matière : ATELIER PROGRAMMATION OBJET AVANCEE | **Niveau :** DSI 2

Enseignants : Equipe pédagogique

gestion-bib-view.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<VBox alignment="CENTER" prefHeight="502.0" prefWidth="601.0"
spacing="20.0" xmlns="http://javafx.com/javafx/17.0.2-ea"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.bib.gestionbib.GestionBibController">
    <padding>
        <Insets bottom="20.0" left="20.0" right="20.0" top="20.0" />
    </padding>
    <children>
        <TabPane fx:id="tpMenu" prefHeight="503.0" prefWidth="561.0"
tabClosingPolicy="UNAVAILABLE">
            <tabs>
                <Tab text="Consultation, Suppression et Recherche">
                    <content>
                        <AnchorPane minHeight="0.0" minWidth="0.0"
prefHeight="180.0" prefWidth="200.0">
                            <children>
                                <HBox layoutX="14.0" layoutY="14.0"
prefHeight="44.0" prefWidth="533.0">
                                    <children>
                                        <Label text="Titre" />
                                        <TextField fx:id="titreR" />
                                        <Button fx:id="btnRechercher"
mnemonicParsing="false" text="Rechercher" />
                                    </children>
                                </HBox>
                                <TableView fx:id="tabLivres"
layoutX="10.0" layoutY="58.0" prefHeight="294.0" prefWidth="542.0">
                                    <columns>
                                        <TableColumn prefWidth="75.0"
text="C1" />
                                        <TableColumn prefWidth="75.0"
text="C2" />
                                    </columns>
                                </TableView>
                                <HBox alignment="CENTER" layoutX="11.0"
layoutY="355.0" prefHeight="34.0" prefWidth="541.0">
                                    <children>
                                        <Button fx:id="btnSupprimer"
mnemonicParsing="false" text="Supprimer" />
                                        <Button
```

```
fx:id="btnSupprimerTout" mnemonicParsing="false" text="Supprimer Tout"
/>
                                        <Button fx:id="btnQuitter"
mnemonicParsing="false" text="Quitter" />
                                    </children>
                                </HBox>
                            </children>
                        </AnchorPane>
                    </content>
                </Tab>
                <Tab text="Ajout">
                    <content>
                        <AnchorPane minHeight="0.0" minWidth="0.0"
prefHeight="180.0" prefWidth="200.0">
                            <children>
                                <VBox alignment="CENTER"
prefHeight="222.0" prefWidth="523.0" spacing="20.0">
                                    <padding>
                                        <Insets bottom="20.0"
left="20.0" right="20.0" top="20.0" />
                                    </padding>
                                    <children>
                                        <VBox prefHeight="200.0"
prefWidth="100.0">
                                            <children>
                                                <HBox
prefHeight="100.0" prefWidth="200.0">
                                                    <children>
                                                        <Label
prefWidth="100.0" text="Titre" />
                                                        <TextField
fx:id="tfTitre" prefHeight="25.0" prefWidth="396.0" />
                                                    </children>
                                                </HBox>
                                                <HBox
prefHeight="100.0" prefWidth="200.0">
                                                    <children>
                                                        <Label
prefHeight="17.0" prefWidth="95.0" text="Couleur" />
                                                        <ColorPicker
fx:id="cpCouleur" prefHeight="25.0" prefWidth="384.0" />
                                                    </children>
                                                </HBox>
                                                <HBox
prefHeight="100.0" prefWidth="200.0">
                                                    <children>
                                                        <Label
prefHeight="17.0" prefWidth="102.0" text="Nb Pages" />
                                                        <Label
fx:id="lbNbPages" prefHeight="17.0" prefWidth="55.0" text="Label" />
                                                        <Slider
fx:id="slNbPages" max="200.0" minorTickCount="5" prefHeight="14.0"
prefWidth="325.0" showTickLabels="true" showTickMarks="true" />
                                                    </children>
                                                </HBox>
                                                <HBox
alignment="CENTER" prefHeight="100.0" prefWidth="200.0">
                                                    <children>
```

```
                                                                <Button
fx:id="btnAjouter" mnemonicParsing="false" text="Ajouter" />
                                                                <Button
fx:id="btnAnnuler" mnemonicParsing="false" text="Annuler" />
                                                            </children>
                                                        </HBox>
                                                    </children>
                                                </VBox>
                                            </children>
                                        </VBox>
                                    </children>
                                </AnchorPane>
                            </content>
                        </Tab>
                        <Tab text="Modification">
                            <content>
                                <AnchorPane minHeight="0.0" minWidth="0.0"
prefHeight="180.0" prefWidth="200.0">
                                    <children>
                                        <VBox alignment="CENTER"
prefHeight="244.0" prefWidth="523.0" spacing="20.0">
                                            <padding>
                                                <Insets bottom="20.0"
left="20.0" right="20.0" top="20.0" />
                                            </padding>
                                            <children>
                                                <VBox prefHeight="200.0"
prefWidth="100.0">
                                                    <children>
                                                        <HBox
prefHeight="100.0" prefWidth="200.0">
                                                            <children>
                                                                <Label
prefHeight="17.0" prefWidth="94.0" text="Id" />
                                                                <ComboBox
fx:id="cbId" prefWidth="150.0" />
                                                            </children>
                                                        </HBox>
                                                        <HBox
prefHeight="100.0" prefWidth="200.0">
                                                            <children>
                                                                <Label
prefWidth="100.0" text="Titre" />
                                                                <TextField
fx:id="tfTitreM" prefHeight="25.0" prefWidth="396.0" />
                                                            </children>
                                                        </HBox>
                                                        <HBox
prefHeight="100.0" prefWidth="200.0">
                                                            <children>
                                                                <Label
prefWidth="100.0" text="Couleur" />
                                                                <ColorPicker
fx:id="cpCouleurM" prefHeight="25.0" prefWidth="384.0" />
                                                            </children>
                                                        </HBox>
                                                        <HBox
prefHeight="100.0" prefWidth="200.0">
```

```
                                                        <children>
                                                            <Label
prefHeight="17.0" prefWidth="102.0" text="Nb Pages" />
                                                            <Label
fx:id="lbNbPagesM" prefHeight="17.0" prefWidth="55.0" text="Label" />
                                                            <Slider
fx:id="slNbPagesM" max="200.0" minorTickCount="5" prefHeight="14.0"
prefWidth="325.0" showTickLabels="true" showTickMarks="true" />
                                                        </children>
                                                    </HBox>
                                                    <HBox
alignment="CENTER" prefHeight="100.0" prefWidth="200.0">
                                                        <children>
                                                            <Button
fx:id="btnModifier" mnemonicParsing="false" text="Modifier" />
                                                            <Button
fx:id="btnAnnulerM" mnemonicParsing="false" text="Annuler" />
                                                        </children>
                                                    </HBox>
                                                </children>
                                            </VBox>
                                        </children>
                                    </VBox>
                                </children></AnchorPane>
                            </content>
                        </Tab>
                    </tabs>
                </TabPane>
            </children>


</VBox>
```

Livre.java

```
package com.bib.gestionbib.data;

import javafx.scene.paint.Color;

public class Livre {
    private Integer id;
    private String titre;
    private Color couleur;
    private Integer nbPages;
    public Livre(Integer id, String titre, Color couleur, Integer
nbPages) {
        this.id = id;
        this.titre = titre;
        this.couleur = couleur;
        this.nbPages = nbPages;
    }

    public Integer getId() {
        return id;
    }

    public String getTitre() {
        return titre;
    }
```

```
    public Color getCouleur() {
        return couleur;
    }

    public Integer getNbPages() {
        return nbPages;
    }

    @Override
    public String toString() {
        return "Livre{" +
                "id=" + id +
                ", titre='" + titre + '\'' +
                ", couleur=" + couleur +
                ", nbPages=" + nbPages +
                '}';
    }
}
```

BibUtil.java

```
package com.bib.gestionbib.data;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.paint.Color;

import java.sql.*;

public class BibUtil {
    private static String dernierTitreErreur = "";
    private static String dernierMessageErreur = "";

    private static String JDBC_Driver = "com.mysql.cj.jdbc.Driver";

    private static String DB_Url =
"jdbc:mysql://db4free.net:3306/biblio?user=user2609&password=password26
09";

    public static String getDernierTitreErreur() {
        return dernierTitreErreur;
    }

    public static void setDernierTitreErreur(String dernierTitreErreur)
{
        BibUtil.dernierTitreErreur = dernierTitreErreur;
    }

    public static String getDernierMessageErreur() {
        return dernierMessageErreur;
    }

    public static void setDernierMessageErreur(String
dernierMessageErreur) {
        BibUtil.dernierMessageErreur = dernierMessageErreur;
    }

    public static ObservableList<Livre> getLivres() {
```

```
        ObservableList<Livre> liste =
FXCollections.observableArrayList();
        try {
            Class.forName(JDBC_Driver);
            System.out.println("Le pilote JDBC s'est chargé
correctement...");
            Connection conn = DriverManager.getConnection
                    (DB_Url);
            System.out.println("La connexion à la base s'est effactué
correctement...");
            Statement st = conn.createStatement();
            String sql = "select * from livre order by id;";
            ResultSet rs = st.executeQuery(sql);
            while (rs.next()) {
                Integer id = rs.getInt(1);
                String titre = rs.getString(2);
                Color couleur = Color.web(rs.getString(3));
                Integer nbPages = rs.getInt(4);
                Livre livre = new Livre(id, titre, couleur, nbPages);
                liste.add(livre);
            }
            st.close();
        } catch (ClassNotFoundException e) {
            dernierTitreErreur = "Problème de chargement du pilote";
            dernierMessageErreur = "Problème: " + e.getMessage();
            e.printStackTrace();
        } catch (SQLException e) {
            dernierTitreErreur = "Problème de connexion à la base";
            dernierMessageErreur = "Problème: " + e.getMessage();
            e.printStackTrace();
        }
        return liste;
    }

    public static Livre getLivreById(int id) {
        ObservableList<Livre> liste =
FXCollections.observableArrayList();
        try {
            Class.forName(JDBC_Driver);
            System.out.println("Le pilote JDBC s'est chargé
correctement...");
            Connection conn = DriverManager.getConnection
                    (DB_Url);
            System.out.println("La connexion à la base s'est effactué
correctement...");

            String sql = "select * from livre where id=?;";
            PreparedStatement pSt = conn.prepareStatement(sql);
            pSt.setInt(1, id);
            ResultSet rs = pSt.executeQuery();
            while (rs.next()) {
                Integer ident = rs.getInt(1);
                String titre = rs.getString(2);
                Color couleur = Color.web(rs.getString(3));
                Integer nbPages = rs.getInt(4);
                Livre livre = new Livre(ident, titre, couleur,
nbPages);
                liste.add(livre);
```

```
            }
            pSt.close();
        } catch (ClassNotFoundException e) {
            dernierTitreErreur = "Problème de chargement du pilote";
            dernierMessageErreur = "Problème: " + e.getMessage();
            e.printStackTrace();
        } catch (SQLException e) {
            dernierTitreErreur = "Problème de connexion à la base";
            dernierMessageErreur = "Problème: " + e.getMessage();
            e.printStackTrace();
        }
        if (liste.size() == 1)
            return liste.get(0);
        return null;
    }

    public static boolean ajouterLivre(String titre, String couleur,
int nbPage) {
        try {
            Class.forName(JDBC_Driver);
            System.out.println("Le pilote JDBC s'est chargé
correctement...");
            Connection conn = DriverManager.getConnection
                    (DB_Url);
            System.out.println("La connexion à la base s'est effactué
correctement...");
            if (!titre.isEmpty()) {
                String sql = "insert into livre(titre,couleur,nbPages)
values (?,?,?);";
                PreparedStatement pSt = conn.prepareStatement(sql);
                pSt.setString(1, titre);
                pSt.setString(2, couleur);
                pSt.setInt(3, nbPage);
                pSt.execute();
                pSt.close();
                conn.close();
                return true;
            } else {
                dernierTitreErreur = "Titre vide!";
                dernierMessageErreur = "Remplir le titre SVP!";
            }
        } catch (ClassNotFoundException e) {
            dernierTitreErreur = "Problème de chargement du pilote";
            dernierMessageErreur = "Problème: " + e.getMessage();
            e.printStackTrace();
        } catch (SQLException e) {
            dernierTitreErreur = "Problème de connexion à la base";
            dernierMessageErreur = "Problème: " + e.getMessage();
            e.printStackTrace();
        }
        return false;
    }

    public static boolean modifierLivre(int id, String titre, String
couleur, int nbPage) {
        try {
            Class.forName(JDBC_Driver);
            System.out.println("Le pilote JDBC s'est chargé
```

```
correctement...");
            Connection conn = DriverManager.getConnection
                    (DB_Url);
            System.out.println("La connexion à la base s'est effactué
correctement...");
            if (!titre.isEmpty()) {
                String sql = "Update livre set
titre=?,couleur=?,nbPages=? where id=?;";
                PreparedStatement pSt = conn.prepareStatement(sql);
                pSt.setString(1, titre);
                pSt.setString(2, couleur);
                pSt.setInt(3, nbPage);
                pSt.setInt(4, id);
                pSt.execute();
                pSt.close();
                conn.close();
                return true;
            } else {
                dernierTitreErreur = "Titre vide!";
                dernierMessageErreur = "Remplir le titre SVP!";
            }
        } catch (ClassNotFoundException e) {
            dernierTitreErreur = "Problème de chargement du pilote";
            dernierMessageErreur = "Problème: " + e.getMessage();
            e.printStackTrace();
        } catch (SQLException e) {
            dernierTitreErreur = "Problème de connexion à la base";
            dernierMessageErreur = "Problème: " + e.getMessage();
            e.printStackTrace();
        }
        return false;
    }

    public static ObservableList<Livre> rechercherLivres(String titreR)
{
        ObservableList<Livre> liste =
FXCollections.observableArrayList();
        try {
            Class.forName(JDBC_Driver);
            System.out.println("Le pilote JDBC s'est chargé
correctement...");
            Connection conn = DriverManager.getConnection
                    (DB_Url);
            System.out.println("La connexion à la base s'est effactué
correctement...");

            String sql = "select * from livre where titre like ? order
by id;";
            PreparedStatement pSt = conn.prepareStatement(sql);
            pSt.setString(1, "%" + titreR + "%");
            ResultSet rs = pSt.executeQuery();
            while (rs.next()) {
                Integer id = rs.getInt(1);
                String titre = rs.getString(2);
                Color couleur = Color.web(rs.getString(3));
                Integer nbPages = rs.getInt(4);
                Livre livre = new Livre(id, titre, couleur, nbPages);
                liste.add(livre);
```

```
            }
            pSt.close();

        } catch (ClassNotFoundException e) {
            dernierTitreErreur = "Problème de chargement du pilote";
            dernierMessageErreur = "Problème: " + e.getMessage();
            e.printStackTrace();
        } catch (SQLException e) {
            dernierTitreErreur = "Problème de connexion à la base";
            dernierMessageErreur = "Problème: " + e.getMessage();
            e.printStackTrace();
        }
        return liste;
    }

    public static boolean supprimerTout() {
        try {
            Class.forName(JDBC_Driver);
            System.out.println("Le pilote JDBC s'est chargé
correctement...");
            Connection conn = DriverManager.getConnection
                    (DB_Url);
            System.out.println("La connexion à la base s'est effactué
correctement...");

            String sql = "delete from livre;";
            PreparedStatement pSt = conn.prepareStatement(sql);
            pSt.execute();
            pSt.close();
            conn.close();
            return true;

        } catch (ClassNotFoundException e) {
            dernierTitreErreur = "Problème de chargement du pilote";
            dernierMessageErreur = "Problème: " + e.getMessage();
            e.printStackTrace();
        } catch (SQLException e) {
            dernierTitreErreur = "Problème de connexion à la base";
            dernierMessageErreur = "Problème: " + e.getMessage();
            e.printStackTrace();
        }
        return false;
    }

    public static boolean supprimerLivre(int id) {
        try {
            Class.forName(JDBC_Driver);
            System.out.println("Le pilote JDBC s'est chargé
correctement...");
            Connection conn = DriverManager.getConnection
                    (DB_Url);
            System.out.println("La connexion à la base s'est effactué
correctement...");

            String sql = "delete from livre where id= ?;";
            PreparedStatement pSt = conn.prepareStatement(sql);
            pSt.setInt(1, id);
            pSt.execute();
```

```
            pSt.close();
            conn.close();
            return true;

        } catch (ClassNotFoundException e) {
            dernierTitreErreur = "Problème de chargement du pilote";
            dernierMessageErreur = "Problème: " + e.getMessage();
            e.printStackTrace();
        } catch (SQLException e) {
            dernierTitreErreur = "Problème de connexion à la base";
            dernierMessageErreur = "Problème: " + e.getMessage();
            e.printStackTrace();
        }
        return false;
    }
}
```

GestionBibController.java

```
package com.bib.gestionbib;

import com.bib.gestionbib.data.BibUtil;
import com.bib.gestionbib.data.Livre;
import javafx.application.Platform;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import javafx.scene.paint.Color;

import java.net.URL;
import java.util.ResourceBundle;
import java.util.stream.Collectors;

public class GestionBibController implements Initializable {
    @FXML
    private TableView<Livre> tabLivres;
    @FXML
    private TextField titreR;
    @FXML
    private TextField tfTitre;
    @FXML
    private Button btnRechercher;
    @FXML
    private Button btnSupprimer;
    @FXML
    private Button btnSupprimerTout;
    @FXML
    private Button btnQuitter;
    @FXML
    private Button btnAjouter;
    @FXML
```

```
    private Button btnAnnulerM;
    @FXML
    private Button btnModifier;
    @FXML
    private Button btnAnnuler;
    @FXML
    private ColorPicker cpCouleur;
    @FXML
    private ColorPicker cpCouleurM;
    @FXML
    private Label lbNbPages;
    @FXML
    private Label lbNbPagesM;
    @FXML
    private Slider slNbPages;
    @FXML
    private Slider slNbPagesM;
    @FXML
    private TextField tfTitreM;
    @FXML
    private ComboBox cbId;
    @FXML
    private TabPane tpMenu;
    ObservableList<Integer> listId;
    boolean refrechCombo = true;

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        cpCouleur.setValue(Color.GREEN);
        slNbPages.setValue(100);
        actualiserNbPages();
        ajouterColonnes();
        remplir();
        ecouteurs();
        cbId.setItems(listId);
    }

    private void ajouterColonnes() {
        tabLivres.getColumns().clear();
        TableColumn<Livre, Integer> idCol = new TableColumn<>("Id");
        TableColumn<Livre, String> titreCol = new
TableColumn<>("Titre");
        TableColumn<Livre, Color> couleurCol = new
TableColumn<>("Couleur");
        TableColumn<Livre, Integer> nbPagesCol = new TableColumn<>("Nb
Pages");
        idCol.setCellValueFactory(new PropertyValueFactory<>("id"));
        titreCol.setCellValueFactory(new
PropertyValueFactory<>("titre"));
        couleurCol.setCellValueFactory(new
PropertyValueFactory<>("couleur"));
        nbPagesCol.setCellValueFactory(new
PropertyValueFactory<>("nbPages"));
        tabLivres.getColumns().addAll(idCol, titreCol, couleurCol,
nbPagesCol);
    }

    private void ecouteurs() {
```

```
        btnQuitter.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent actionEvent) {
                quitter();
            }
        });
        btnAjouter.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent actionEvent) {
                ajouter();
            }
        });
        btnAnnuler.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent actionEvent) {
                annuler();
            }
        });
        slNbPages.valueProperty().addListener(new
ChangeListener<Number>() {
            @Override
            public void changed(ObservableValue<? extends Number>
observableValue,
                                Number number, Number t1) {
                slNbPages.setValue(Math.round(t1.doubleValue()));
                actualiserNbPages();
            }
        });
        slNbPagesM.valueProperty().addListener(new
ChangeListener<Number>() {
            @Override
            public void changed(ObservableValue<? extends Number>
observableValue,
                                Number number, Number t1) {
                slNbPagesM.setValue(Math.round(t1.doubleValue()));
                actualiserNbPagesM();
            }
        });

        tabLivres.setOnMouseClicked(new EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent event) {
                if (event.getClickCount() == 2)
                    afficherModifier();
            }
        });
        btnAnnulerM.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent actionEvent) {
                annuler();
            }
        });
        btnModifier.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent actionEvent) {
                modifier();
            }
        });
```

```
        btnRechercher.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent actionEvent) {
                rechercher();
            }
        });
        btnSupprimer.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent actionEvent) {
                supprimer();
            }
        });
        btnSupprimerTout.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent actionEvent) {
                supprimerTout();
            }
        });
        cbId.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent actionEvent) {
                if (refrechCombo  && actionEvent.getEventType() ==
ActionEvent.ACTION)
                    actualiserLivre();
            }
        });
    }

    private void supprimerTout() {
        Alert dialog = new Alert(Alert.AlertType.CONFIRMATION);
        dialog.setTitle("Supprimer Tout?");
        dialog.setHeaderText("Etes vous sûr de supprimer tous les
livres?");
        dialog.showAndWait();
        if (dialog.getResult() == ButtonType.OK) {
            if (BibUtil.supprimerTout()) {
                remplir();
                tfTitreM.setText("");
                cpCouleurM.setValue(Color.WHITE);
                slNbPagesM.setValue(0);
            }
            else
                afficherErreur(BibUtil.getDernierTitreErreur(),
BibUtil.getDernierMessageErreur());
        }
    }

    private void supprimer() {
        TablePosition position =
tabLivres.getSelectionModel().getSelectedCells().get(0);
        if (position.getRow() >= 0) {
            Livre livreSelectionne =
tabLivres.getItems().get(position.getRow());
            int id = livreSelectionne.getId();
            if (BibUtil.supprimerLivre(id))
                remplir();
            else
                afficherErreur(BibUtil.getDernierTitreErreur(),
```

```
BibUtil.getDernierMessageErreur());
        }
    }

    private void rechercher() {
        ObservableList<Livre> list =
BibUtil.rechercherLivres(titreR.getText());
        tabLivres.setItems(list);
    }

    private void quitter() {
        Platform.exit();
    }

    private void afficherModifier() {
        TablePosition position =
tabLivres.getSelectionModel().getSelectedCells().get(0);
        System.out.println("Double Click sur la ligne: " +
position.getRow());
        if (position.getRow() >= 0) {
            Livre livreSelectionne =
tabLivres.getItems().get(position.getRow());
            cbId.getSelectionModel().select(livreSelectionne.getId());
            tfTitreM.setText(livreSelectionne.getTitre());
            cpCouleurM.setValue(livreSelectionne.getCouleur());
            slNbPagesM.setValue(livreSelectionne.getNbPages());
            tpMenu.getSelectionModel().select(2);
        }
    }

    private void actualiserNbPages() {
        lbNbPages.setText(String.format("%.0f", slNbPages.getValue()));
    }

    private void actualiserNbPagesM() {
        lbNbPagesM.setText(String.format("%.0f",
slNbPagesM.getValue()));
    }

    private void actualiserLivre() {
        Integer id = (Integer) cbId.getValue();
        Livre livre = BibUtil.getLivreById(id);
        if (livre != null) {
            tfTitreM.setText(livre.getTitre());
            cpCouleurM.setValue(livre.getCouleur());
            slNbPagesM.setValue(livre.getNbPages());
            tpMenu.getSelectionModel().select(2);
        }
    }

    private void ajouter() {
        if (BibUtil.ajouterLivre(tfTitre.getText(),
                cpCouleur.getValue().toString(),
                (int) slNbPages.getValue())) {
            remplir();
            tfTitre.setText("");
            tpMenu.getSelectionModel().select(0);
        } else
```

```
            afficherErreur(BibUtil.getDernierTitreErreur(),
BibUtil.getDernierMessageErreur());
    }

    private void remplir() {
        ObservableList<Livre> listLivres = BibUtil.getLivres();
        tabLivres.setItems(listLivres);
        listId = listLivres.stream()
                .map(Livre::getId)

.collect(Collectors.toCollection(FXCollections::observableArrayList));

        refrechCombo = false;
        cbId.setItems(listId);
        refrechCombo = true;
    }

    public static void afficherErreur(String titre, String message) {
        Alert dialog = new Alert(Alert.AlertType.ERROR);
        dialog.setTitle(titre);
        dialog.setHeaderText(message);
        dialog.showAndWait();
    }

    private void annuler() {
        tpMenu.getSelectionModel().select(0);
    }

    private void modifier() {
        int id = Integer.parseInt(cbId.getValue().toString());
        if (BibUtil.modifierLivre(id, tfTitreM.getText(),
                cpCouleurM.getValue().toString(),
                (int) slNbPagesM.getValue())) {
            remplir();
            tpMenu.getSelectionModel().select(0);
        } else
            afficherErreur(BibUtil.getDernierTitreErreur(),
BibUtil.getDernierMessageErreur());
    }
}
```

GestionBibApplication.java

```
package com.bib.gestionbib;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.stage.Stage;

import java.io.IOException;

public class GestionBibApplication extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new
FXMLLoader(GestionBibApplication.class.getResource("gestion-bib-
view.fxml"));
```

```
        Scene scene = new Scene(fxmlLoader.load());
        stage.setTitle("Gestion Bibliothèque");
        stage.setScene(scene);
        stage.show();
    }
    public static void main(String[] args) {
        launch();
    }
}
```

Module-info.java

```
module com.bib.gestionbib {
    requires javafx.controls;
    requires javafx.fxml;
    requires java.sql;

    opens com.bib.gestionbib.data to javafx.base;
    opens com.bib.gestionbib to javafx.fxml;
    exports com.bib.gestionbib;
}
```

build.gradle

```
…
dependencies {
  implementation group: 'com.mysql', name: 'mysql-connector-j',
version: '8.0.33'
  testImplementation("org.junit.jupiter:junit-jupiter-
api:${junitVersion}")
  testRuntimeOnly("org.junit.jupiter:junit-jupiter-
engine:${junitVersion}")
}
…
```