

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <?import javafx.geometry.Insets?>
4 <?import javafx.scene.control.TableColumn?>
5 <?import javafx.scene.control.TableView?>
6 <?import javafx.scene.layout.VBox?>
7
8 <VBox alignment="CENTER" prefHeight="150.0" prefWidth="596.0" spacing="10.0" xmlns
  ="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1" fx:controller="
  com.example.vphw07.TableViewController">
9     <padding>
10         <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
11     </padding>
12     <children>
13         <TableView fx:id="mainTV" editable="true" prefHeight="200.0" prefWidth="575.
14             0">
15             <columns>
16                 <TableColumn fx:id="fullNameCol" prefWidth="120.0" text="Full Name" />
17                 <TableColumn fx:id="favoriteColorCol" prefWidth="101.0" text="Favorite
18                 Color" />
19                 <TableColumn fx:id="sportCol" minWidth="0.0" prefWidth="120.0" text="
20                 Sport" />
21                 <TableColumn fx:id="yearCol" prefWidth="100.0" text="# of Years" />
22                 <TableColumn fx:id="vegetarianCol" prefWidth="120.0" text="Vegetarian"
23             />
24             </columns>
25         </TableView>
26     </children>
27 </VBox>
```

```
1 package com.example.vphw07;
2
3 import javafx.application.Application;
4 import javafx.fxml.FXMLLoader;
5 import javafx.scene.Scene;
6 import javafx.stage.Stage;
7
8 import java.io.IOException;
9
10 public class MainApp extends Application {
11     @Override
12     public void start(Stage stage) throws IOException {
13         FXMLLoader fxmlLoader = new FXMLLoader(MainApp.class.getResource("table-
14         dialog-edit-demo.fxml"));
15         Scene scene = new Scene(fxmlLoader.load(), 596, 150);
16         stage.setTitle("TableDialogEditDemo");
17         stage.setScene(scene);
18         stage.show();
19     }
20     public static void main(String[] args) {
21         launch();
22     }
23 }
```

```
1 package com.example.vphw07.Model;
2
3 import javafx.scene.paint.Color;
4
5 public class Person {
6     private String fullName;
7     private Color favColor;
8     private String favSport;
9     private String years;
10    private boolean vegetarian;
11
12    public Person(String fullName, Color favColor, String favSport, String years,
13    boolean vegetarian) {
14        this.fullName = fullName;
15        this.favColor = favColor;
16        this.favSport = favSport;
17        this.years = years;
18        this.vegetarian = vegetarian;
19    }
20
21    public String getFullName() {
22        return fullName;
23    }
24
25    public void setFullName(String fullName) {
26        this.fullName = fullName;
27    }
28
29    public Color getFavColor() {
30        return favColor;
31    }
32
33    public void setFavColor(Color favColor) {
34        this.favColor = favColor;
35    }
36
37    public String getFavSport() {
38        return favSport;
39    }
40
41    public void setFavSport(String favSport) {
42        this.favSport = favSport;
43    }
44
45    public String getYears() {
46        return years;
47    }
48
49    public void setYears(String years) {
50        this.years = years;
51    }
52
53    public boolean isVegetarian() {
54        return vegetarian;
55    }
```

```
56     public void setVegetarian(boolean vegetarian) {  
57         this.vegetarian = vegetarian;  
58     }  
59 }  
60
```

```

1 package com.example.vphw07;
2
3 import com.example.vphw07.Model.Person;
4 import javafx.beans.property.SimpleBooleanProperty;
5 import javafx.beans.property.SimpleStringProperty;
6 import javafx.beans.value.ChangeListener;
7 import javafx.beans.value.ObservableValue;
8 import javafx.collections.FXCollections;
9 import javafx.collections.ObservableList;
10 import javafx.fxml.FXML;
11 import javafx.geometry.Pos;
12 import javafx.scene.control.*;
13 import javafx.scene.control.Label;
14 import javafx.scene.control.cell.CheckBoxTableCell;
15 import javafx.scene.control.cell.ComboBoxTableCell;
16 import javafx.scene.control.cell.PropertyValueFactory;
17 import javafx.scene.control.cell.TextFieldTableCell;
18 import javafx.util.Callback;
19 import javafx.scene.paint.Color;
20
21 public class TableViewController {
22
23     @FXML
24     private TableColumn<Person, Color> favoriteColorCol;
25
26     @FXML
27     private TableColumn<Person, String> fullNameCol;
28
29     @FXML
30     private TableView<Person> mainTV;
31
32     @FXML
33     private TableColumn<Person, String> sportCol;
34
35     @FXML
36     private TableColumn<Person, Boolean> vegetarianCol;
37
38     @FXML
39     private TableColumn<Person, String> yearCol;
40
41     @FXML
42     void initialize() {
43         assert favoriteColorCol != null : "fx:id=\"favoriteColorCol\" was not
injected: check your FXML file 'table-dialog-edit-demo.fxml'.";
44         assert fullNameCol != null : "fx:id=\"fullNameCol\" was not injected:
check your FXML file 'table-dialog-edit-demo.fxml'.";
45         assert mainTV != null : "fx:id=\"mainTV\" was not injected: check your
FXML file 'table-dialog-edit-demo.fxml'.";
46         assert sportCol != null : "fx:id=\"sportCol\" was not injected: check your
FXML file 'table-dialog-edit-demo.fxml'.";
47         assert vegetarianCol != null : "fx:id=\"vegetarianCol\" was not injected:
check your FXML file 'table-dialog-edit-demo.fxml'.";
48         assert yearCol != null : "fx:id=\"yearCol\" was not injected: check your
FXML file 'table-dialog-edit-demo.fxml'.";
49
50         ObservableList<Person> personList = FXCollections.observableArrayList();

```

```

51     personList.add(new Person("Kathy Smith", Color.ORANGE, "Snowboarding", "5
    ", false));
52     personList.add(new Person("John Doe", Color.BLUE, "Rowing", "3", true));
53     personList.add(new Person("Sue Black", Color.GREEN, "Knitting", "2",
    false));
54     personList.add(new Person("Jane White", Color.RED, "Speed reading", "20"
    , true));
55     personList.add(new Person("Joe Brown", Color.PINK, "Pool", "10", false));
56
57     //Columns styles
58     yearCol.setStyle("-fx-alignment: CENTER-RIGHT;");
59     vegetarianCol.setStyle("-fx-alignment: CENTER;");
60     favoriteColorCol.setStyle("-fx-alignment: CENTER;");
61
62     //cellValueFactory and cellFactory
63     fullNameCol.setText("");
64     Label firstNameLabel = new Label("Full Name");
65     firstNameLabel.setTooltip(new Tooltip("This column shows the full name"
    ));
66     fullNameCol.setGraphic(firstNameLabel);
67     fullNameCol.setCellValueFactory(new PropertyValueFactory<>("fullName"));
68     fullNameCol.setCellFactory
69         (
70         column ->
71         {
72             return new TableCell<Person, String>()
73             {
74                 @Override
75                 protected void updateItem(String item, boolean
    empty)
76                 {
77                     super.updateItem(item, empty);
78                     setText( item );
79                     setTooltip(new Tooltip("value: " + item));
80                 }
81             };
82         });
83     fullNameCol.setOnEditCommit((TableColumn.CellEditEvent<Person, String>
    event) -> {
84         TablePosition<Person, String> pos = event.getTablePosition();
85
86         String newFullName = event.getNewValue();
87
88         int row = pos.getRow();
89         Person person = event.getTableView().getItems().get(row);
90
91         person.setFullName(newFullName);
92     });
93
94     favoriteColorCol.setText("");
95     Label favoriteColorLabel = new Label("Favorite Color");
96     favoriteColorLabel.setTooltip(new Tooltip("This column shows the favorite
    color"));
97     favoriteColorCol.setGraphic(favoriteColorLabel);
98     Callback factory = new Callback<TableColumn<Person, Color>, TableCell<
    Person, Color>>() {

```

```

99         @Override
100         public TableCell<Person, Color> call(TableCell<Person, Color> param
    ) {
101             return new TableCell<Person, Color>() {
102                 private ColorPicker colorPicker;
103
104                 public void ColorTableCell(TableCell<Person, Color> column
    , Color item) {
105                     this.colorPicker = new ColorPicker();
106                     this.colorPicker.editableProperty().bind(column.
    editableProperty());
107                     this.colorPicker.disableProperty().bind(column.
    editableProperty().not());
108                     this.colorPicker.setOnShowing(event -> {
109                         TableView<Person> tableView = getTableView();
110                         tableView.getSelectionModel().select(getTableRow().
    getIndex());
111                         tableView.edit(tableView.getSelectionModel().
    getSelectedIndex(), favoriteColorCol);
112                     });
113                     this.colorPicker.valueProperty().addListener((observable
    , oldValue, newValue) -> {
114                         if(isEditing()) {
115                             commitEdit(newValue);
116                         }
117                     });
118                     setContentDisplay(ContentDisplay.GRAPHIC_ONLY);
119                 }
120
121                 @Override
122                 protected void updateItem(Color item, boolean empty) {
123                     super.updateItem(item, empty);
124                     ColorTableCell(favoriteColorCol, item);
125
126
127                     //         this.colorPicker = new ColorPicker();
128                     //         setContentDisplay(ContentDisplay.GRAPHIC_ONLY);
129                     //
130                     //         if (empty || item == null) {
131                     //             setText(null);
132                     //         } else {
133                     //             setText("");
134                     //
135                     //             this.setStyle("-fx-background-color: " + item
    + ";");
136                     //         }
137
138                     setText(null);
139                     if(empty) {
140                         setGraphic(null);
141                     } else {
142                         this.colorPicker.setValue(item);
143                         this.setGraphic(this.colorPicker);
144                     }
145                 }
146

```

```

147         };
148     }
149 };
150     favoriteColorCol.setCellValueFactory(new PropertyValueFactory<>("favColor
151     "));
152     favoriteColorCol.setCellFactory(factory);
153     sportCol.setText("");
154     Label sportColLabel = new Label("Sport");
155     sportColLabel.setTooltip(new Tooltip("This column shows the sport"));
156     sportCol.setGraphic(sportColLabel);
157     ObservableList<String> sportsList = FXCollections.observableArrayList("
Snowboarding", "Rowing", "Knitting", "Speed reading", "Pool", "None of the above"
);
158     sportCol.setCellValueFactory(new Callback<TableColumn.CellDataFeatures<
Person, String>, ObservableValue<String>>() {
159         @Override
160         public ObservableValue<String> call(TableColumn.CellDataFeatures<
Person, String> param) {
161             Person person = param.getValue();
162             String favSport = person.getFavSport();
163             return new SimpleStringProperty(favSport);
164         }
165     });
166     sportCol.setCellFactory(ComboBoxTableCell.forTableColumn(sportsList));
167     sportCol.setOnEditCommit((TableColumn.CellEditEvent<Person, String> event
) -> {
168         TablePosition<Person, String> pos = event.getTablePosition();
169
170         String newFavSport = event.getNewValue();
171
172         int row = pos.getRow();
173         Person person = event.getTableView().getItems().get(row);
174
175         person.setFavSport(newFavSport);
176     });
177
178     yearCol.setText("");
179     Label yearColLabel = new Label("# of Years");
180     yearColLabel.setTooltip(new Tooltip("This column shows years"));
181     yearCol.setGraphic(yearColLabel);
182     yearCol.setCellValueFactory(new PropertyValueFactory<Person, String>("
years"));
183     yearCol.setCellFactory(TextFieldTableCell.<Person> forTableColumn());
184     yearCol.setOnEditCommit((TableColumn.CellEditEvent<Person, String> event
) -> {
185         TablePosition<Person, String> pos = event.getTablePosition();
186
187         String newYear = event.getNewValue();
188
189         int row = pos.getRow();
190         Person person = event.getTableView().getItems().get(row);
191
192         person.setYears(newYear);
193     });
194

```



```

195     vegetarianCol.setText("");
196     Label vegetarianColLabel = new Label("Vegetarian");
197     vegetarianColLabel.setToolTipText(new Tooltip("This column shows whether is
vegetarian"));
198     vegetarianCol.setGraphic(vegetarianColLabel);
199     vegetarianCol.setCellValueFactory(new Callback<TableColumn.
CellDataFeatures<Person, Boolean>, ObservableValue<Boolean>>() {
200         @Override
201         public ObservableValue<Boolean> call(TableColumn.CellDataFeatures<
Person, Boolean> param) {
202             Person person = param.getValue();
203
204             SimpleBooleanProperty booleanProp = new SimpleBooleanProperty(
person.isVegetarian());
205
206             booleanProp.addListener(new ChangeListener<Boolean>() {
207
208                 @Override
209                 public void changed(ObservableValue<? extends Boolean>
observable, Boolean oldValue,
210                                     Boolean newValue) {
211                     person.setVegetarian(newValue);
212                 }
213             });
214             return booleanProp;
215         }
216     });
217
218
219     vegetarianCol.setCellFactory(new Callback<TableColumn<Person, Boolean>,
//
220         TableCell<Person, Boolean>>() {
221             @Override
222             public TableCell<Person, Boolean> call(TableColumn<Person, Boolean> p
) {
223                 CheckBoxTableCell<Person, Boolean> cell = new CheckBoxTableCell<
Person, Boolean>();
224                 cell.setAlignment(Pos.CENTER);
225                 return cell;
226             }
227         });
228
229     mainTV.setItems(personList);
230 }
231 }
232

```

TableDialogEditDemo

Full Name	Favorite Color	Sport	# of Years	Vegetarian
Kathy Smith	Orange	Snowboarding	5	<input type="checkbox"/>
John Doe	Blue	Rowing	3	<input checked="" type="checkbox"/>
Sue Black	Green	Knitting	2	<input type="checkbox"/>
Jane White	Red	Speed reading	20	<input checked="" type="checkbox"/>
	Pink			<input type="checkbox"/>

TableDialogEditDemo

Full Name	Favorite Color	Sport	# of Years	Vegetarian
Kathy Smith		Snowboarding	5	<input type="checkbox"/>
John Doe		Rowing	3	<input checked="" type="checkbox"/>
Sue Black		Knitting	2	<input type="checkbox"/>
Jane White		Speed reading	20	<input checked="" type="checkbox"/>
				<input type="checkbox"/>

TableDialogEditDemo

Full Name	Favorite Color	Sport	# of Years	Vegetarian
Kathy Smith		Snowboarding	5	<input type="checkbox"/>
John Doe		Rowing	3	<input checked="" type="checkbox"/>
Sue Black		Knitting	2	<input type="checkbox"/>
Jane White		Speed reading	20	<input checked="" type="checkbox"/>

TableDialogEditDemo

Full Name	Favorite Color	Sport	# of Years	Vegetarian
Kathy Smith	Orange	Snowboard...	5	<input type="checkbox"/>
John Doe	Blue	Snowboarding	3	<input checked="" type="checkbox"/>
Sue Black	Green	Rowing	2	<input type="checkbox"/>
Jane White	Red	Knitting	20	<input checked="" type="checkbox"/>

Snowboard...

Snowboarding


Rowing

Knitting

Speed reading

Pool

None of the above

Full Name	Favorite Color	Sport	# of Years	Vegetarian
Kathy Smith	 Oran... ▼	Snowboarding	5	<input type="checkbox"/>
John Doe			3	<input checked="" type="checkbox"/>
Sue Black			2	<input type="checkbox"/>
Tom White			1	<input type="checkbox"/>



Custom Color...

play.GRAPHIC_ONLY)

item, boolean empty

col item):