Programmazione avanzata

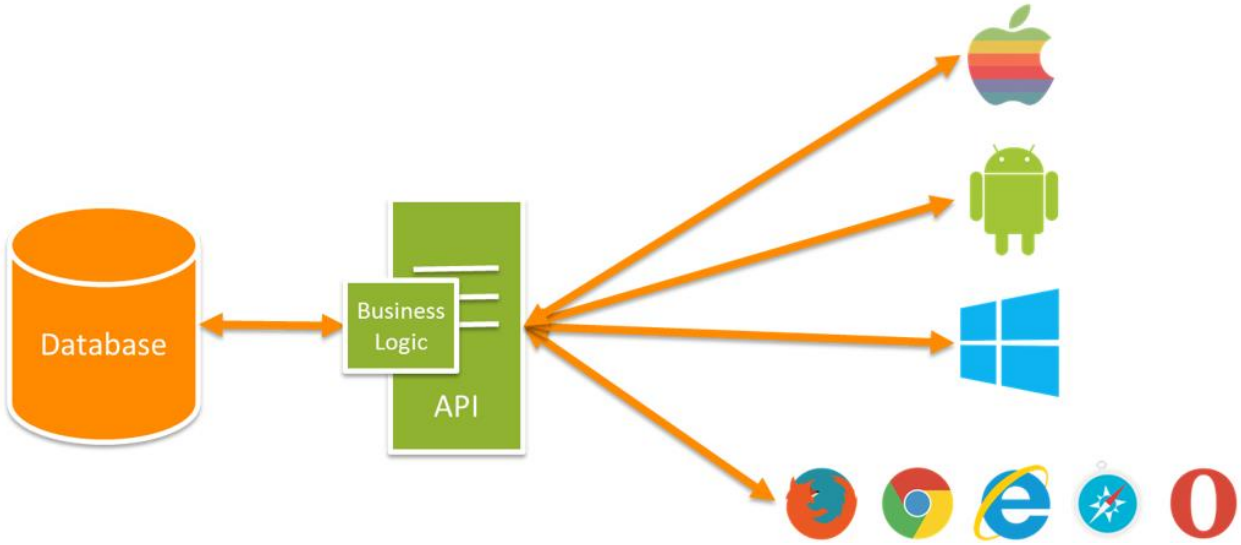
Lezione 8

Importazione dei dati da un servizio web

Visualizzazione dei dati a lista nell'interfaccia tramite TableView di JavaFX.

Servizio Web

Molto spesso le applicazioni recuperano delle informazioni da servizi web. Un servizio web è un particolare tipo di applicazione web che non espone un'interfaccia utente ma espone un'interfaccia per altre applicazioni per lo scambio di dati (maggiori dettagli verranno forniti alla magistrale o nel corso di Ingegneria del Software).



Questi servizi web forniscono dati alle applicazioni codificando i dati in formato JSON o XML.

Importazione dei dati

Esistono diversi servizi web disponibili, un elenco non esaustivo può essere recuperato attraverso il seguente sito web[1]: https://free-apis.github.io/

Un'applicazione può importare dati da un servizio web ed interpretarli utilizzando le librerie GSON o XSTREAM in maniera immediata. Nel caso di un flusso di dati di tipo JSON si può anche effettuare una lettura e interpretazione dei dati online senza definire una struttura dati che li rappresenta.

Il primo passo è quello di recuperare i dati dall'URL del servizio attraverso una richiesta http:

[1] Non tutti I servizi web sono aperti, alcuni richiedono un'autorizzazione e un'autenticazione per motivi di sicurezza. L'implementazione di tali funzionalità rimane fuori dallo scopo in questo momento.

```
1.  URL url = new URL("http://131.114.23.73:8080/api/f1/drivers?limit=10&offset=20");
2.  HttpURLConnection con = (HttpURLConnection) url.openConnection();
3.  con.setRequestMethod("GET");
4.
5.  BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
6.
7.  String inputLine;
8.  StringBuffer content = new StringBuffer();
9.  while ((inputLine = in.readLine()) != null) {
10.     content.append(inputLine);
11. }
12. in.close();
```

Alla riga 1 si specifica l'URL del servizio web da cui vogliamo scaricare i dati (fornito dal gestore del servizio) alla riga 2 si effettua una connessione attraverso la classe HttpURLConnection specificando il metodo di recupero dei dati (GET in questo caso, riga 3). In seguito, righe da 5 a 12 si recuperano i dati fino a quando il servizio non esaurisce l'invio inserendo tutto in una sola variabile[2].

Nell'esempio abbiamo scaricato i dati in formato JSON dal servizio web offerto all'URL:

http://131.114.23.73:8080/api/f1/drivers

(che è una nostra copia di http://ergast.com/api/f1/drivers.json)

che fornisce dati di tutti i piloti di piloti di F1. Al fine di suddividere i dati in gruppi gestibili, l'URL del servizio include alcuni parametri. Questi parametri che possono essere definiti all'interno di URL per il recupero dati dai servizi web sono chiamati query parameters e sono usati per specificare alcuni parametri della richiesta e variano da servizio a servizio. Nel caso del servizio che abbiamo preso in considerazione due parametri sono di nostro interesse:

- limit: per limitare il numero di piloti che vengono restituiti dal servizio
- offset: per richiedere un particolare gruppo di dati all'interno dell'insieme di tutti i dati forniti

Una volta recuperato il documento, ad esempio JSON, possiamo passare alla sua implementazione. Supponiamo che il servizio sia un servizio che fornisce dati sulla F1 e che in questo caso fornisca dati sui piloti e che il documento abbia questa struttura:

```
1.  {
2.    "MRData": {
3.      "DriverTable": {
4.        "Drivers": [
5.          {
6.            "driverId": "amon",
7.            "url": "http://en.wikipedia.org/wiki/Chris_Amon",
8.            "givenName": "Chris",
9.            "familyName": "Amon",
10.           "dateOfBirth": "1943-07-20",
11.           "nationality": "New Zealander"
12.         },
```

[2] Lo scaricamento di una grande quantità di dati richiede la gestione del frazionamento e del processamento dei dati per blocchi. Per semplicità lasciamo fuori questo aspetto che può essere facilmente compreso e implementato.

```
13.         {
14.           "driverId": "anderson",
15.           "url": "http://en.wikipedia.org/wiki/Bob_Anderson_(racing_driver)",
16.           "givenName": "Bob",
17.           "familyName": "Anderson",
18.           "dateOfBirth": "1931-05-19",
19.           "nationality": "British"
20.         },
21.       ]
22.     }
23.   }
24. }
25.
```

Questi dati possono essere interpretati attraverso il seguente codice:

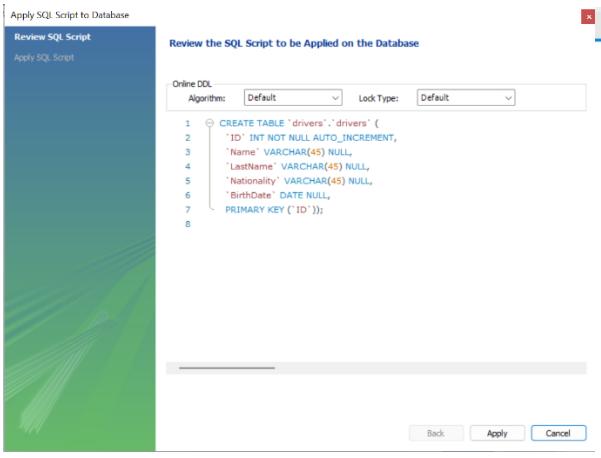
```
1.  Gson gson = new Gson();
2.
3.  JsonElement json = gson.fromJson(content.toString(), JsonElement.class);
4.  JsonObject rootObject = json.getAsJsonObject();
5.  JsonArray drivers =
    rootObject.get("MRData").getAsJsonObject().get("DriverTable").getAsJsonObject().get("Drivers
    ").getAsJsonArray();
6.
7.  for (int i = 0; i < drivers.size(); i++) {
8.      JsonObject d = drivers.get(i).getAsJsonObject();
9.      logger.info(d.get("givenName").toString());
10.
```

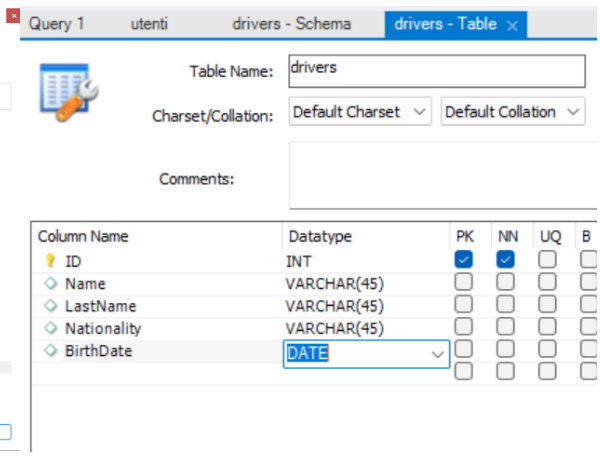
La funzione fromJson (riga 3) viene chiamata per recuperare un oggetto di tipo JsonElement che rappresenta l'intero documento. A partire da un JsonElement si puo' chiamare la funzione getAsJsonArray per recuperare una lista di elementi o la funzione getAsJsonObject per recuperare un oggetto singolo. Tramite queste due funzioni si possono recuperare i capi del documento e le liste. Una volta recuperato un oggetto si possono recuperare i campi attraverso la funzione get.

Esercizio

Importare i dati dei Drivers (anche solo una parte) forniti in un database dopo averli recuperati dal servizio web.

Suggerimento, il database deve essere creato nel seguente modo:





L’utilizzo della libreria mysql-client-java in un’applicazione JavaFX richiede anche l’aggiunta delle seguenti direttive nel file module-info.java:

```
1.      requires java.sql;
2.      requires java.base;
3.      requires java.desktop;
```

Visualizzazione dei dati in una tabella JavaFX

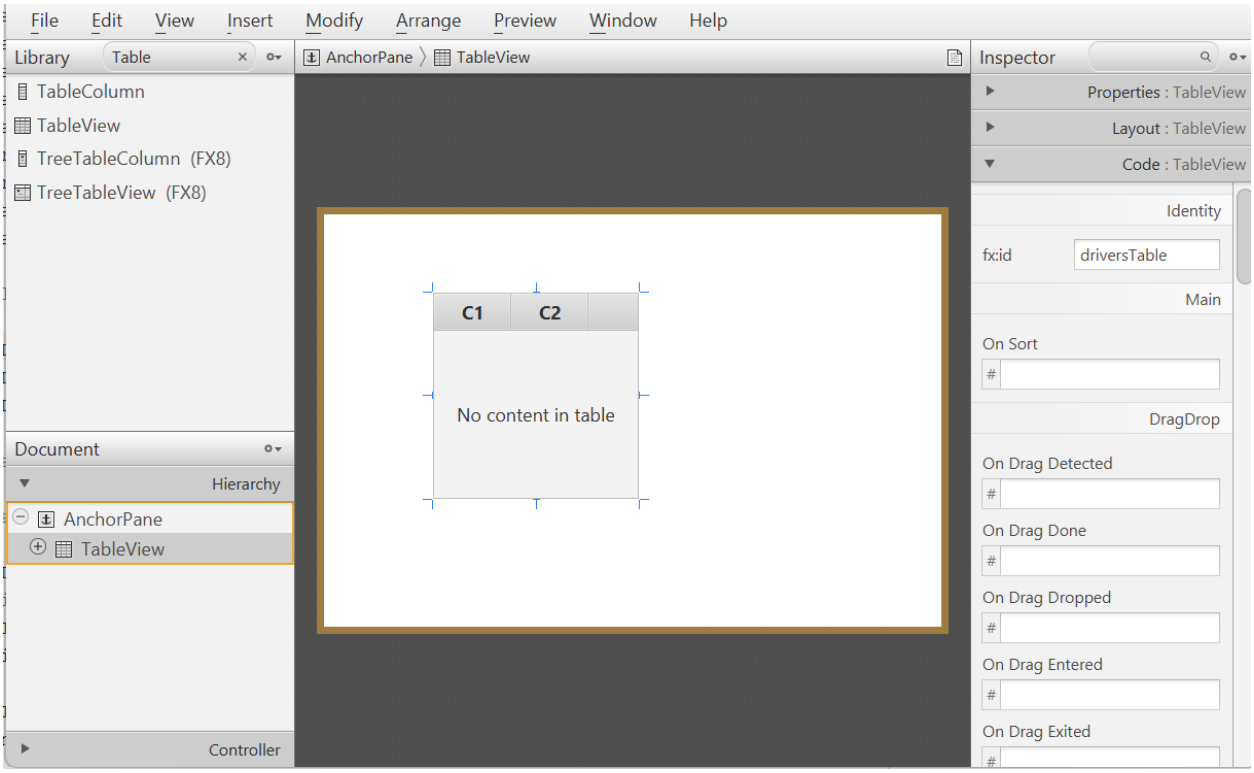
I dati inseriti all'interno del database potrebbero essere mostrati all'utente sotto forma di una tabella JavaFX come la seguente:

ID	Name	LastName	Nationality	BirthDate
1	"Chris"	"Amon"	"New Zeala...	1943-07-20
2	"Bob"	"Anderson"	"British"	1931-05-19
3	"Conny"	"Andersson"	"Swedish"	1939-12-28
8	"Frank"	"Armi"	"American"	1918-10-12
10	"RenÃ©"	"Arnoux"	"French"	1948-07-04

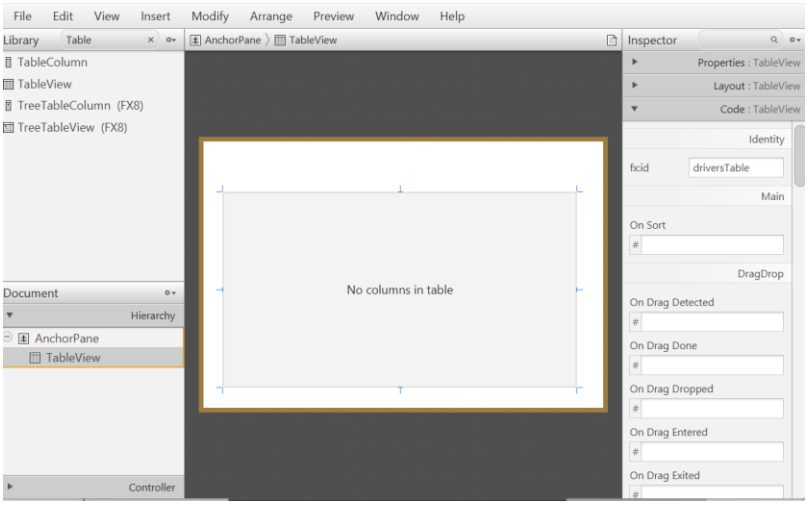
La visualizzazione dei dati sotto forma di tabella avviene collegando una lista particolare di elementi che rappresentano i dati mostrati alla tabella stessa. Il primo passo e' quindi di definire una classe Driver che rappresenta i dati dei piloti utilizzando una definizione basata su Java Beans:

```
1.  public class Driver {
2.      private final int id;
3.      private final String name;
4.      private final String lastname;
5.      private final String nationality;
6.      private final String birthdate;
7.
8.      public int getId() {
9.          return id;
10.     }
11.
12.     public String getName() {
13.         return name;
14.     }
15.
16.     public String getLastname() {
17.         return lastname;
18.     }
19.
20.     public String getNationality() {
21.         return nationality;
22.     }
23.
24.     public String getBirthdate() {
25.         return birthdate;
26.     }
27.
28.     public Driver(int id, String name, String lastname, String nationality, String
    birthdate) {
29.         this.id = id;
30.         this.name = name;
31.         this.lastname = lastname;
32.         this.nationality = nationality;
33.         this.birthdate = birthdate;
34.     }
35. }
```

Una volta creata la rappresentazione del dato, si va a creare nell'interfaccia un elemento di tipo TableView attraverso JavaFX Scene Builder:



E' importante cancellare le colonne create automaticamente (C1 e C2 in questo caso), perché andremo poi a creare le colonne nel codice e a collegarle ai campi della struttura dati che abbiamo creato. E' importante anche configurare il campo fx:id che verrà poi collegato ad un elemento creato all'interno del controllore che gestirà la tabella.



Nella classe controllore andiamo a creare un elemento TableView e un elemento di tipo ObservableList che conterranno le istanze dei dati da mostrare:

```
1.  @FXML TableView<Driver> driversTable = new TableView<>();
2.
3.  private ObservableList<Driver> ol;
```

Nella funzione ‘initialize’ che viene chiamata automaticamente al momento dell’apertura della schermata inseriamo il seguente codice:

```
1.  @FXML
2.      public void initialize() {
3.          TableColumn idCol = new TableColumn("ID");
4.          idCol.setCellValueFactory(new PropertyValueFactory<>("id"));
5.
6.          TableColumn nomeCol = new TableColumn("Name");
7.          nomeCol.setCellValueFactory(new PropertyValueFactory<>("name"));
8.
9.          TableColumn lastCol = new TableColumn("LastName");
10.         lastCol.setCellValueFactory(new PropertyValueFactory<>("lastname"));
11.
12.         TableColumn nationalityCol = new TableColumn("Nationality");
13.         nationalityCol.setCellValueFactory(new PropertyValueFactory<>("nationality"));
14.
15.         TableColumn dateCol = new TableColumn("BirthDate");
16.         dateCol.setCellValueFactory(new PropertyValueFactory<>("birthdate"));
17.
18.         driversTable.getColumns().addAll(idCol, nomeCol, lastCol, nationalityCol, dateCol);
19.
20.         ol = FXCollections.observableArrayList();
21.
22.         driversTable.setItems(ol);
23.
```

Le righe dalla 3 alla 16 servono per dichiarare le colonne e collegarle ai dati delle istanze della classe Driver da mostrare, definendo prima degli elementi TableColumn e poi collegandoli ad una proprietà specifica dei dati. Alla riga 18 si collegano gli elementi TableColumn alla tabella. Nella riga 20 si crea un array osservabile e alla riga 22 si collega alla tabella in modo che i dati contenuti nell’array vengano mostrati automaticamente ogni volta che questi vengano aggiunti/rimossi.

L’aggiunta di un nuovo driver, leggendo i dati dal database (righe 1-5), può essere semplicemente fatta creando una nuova istanza della classe Driver (riga 6) e aggiungendo l’istanza all’array (riga 7) :

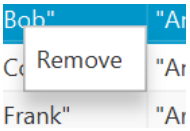
```
1.  try ( Connection co = DriverManager.getConnection(
2.                              "jdbc:mysql://localhost:3306/drivers", "root", "root");
3.        Statement st = co.createStatement();) {
4.           ResultSet rs = st.executeQuery("SELECT * FROM drivers");
5.           while (rs.next()) {
6.                Driver d = new Driver(rs.getInt("ID"), rs.getString("Name"),
7.                        rs.getString("LastName"), rs.getString("Nationality"),
8.                        rs.getString("BirthDate"));
9.                ol.add(d);
10.          }
11. } catch (SQLException e) {
12.      logger.error(e.getMessage());
13. }
14.
```

Esercizio

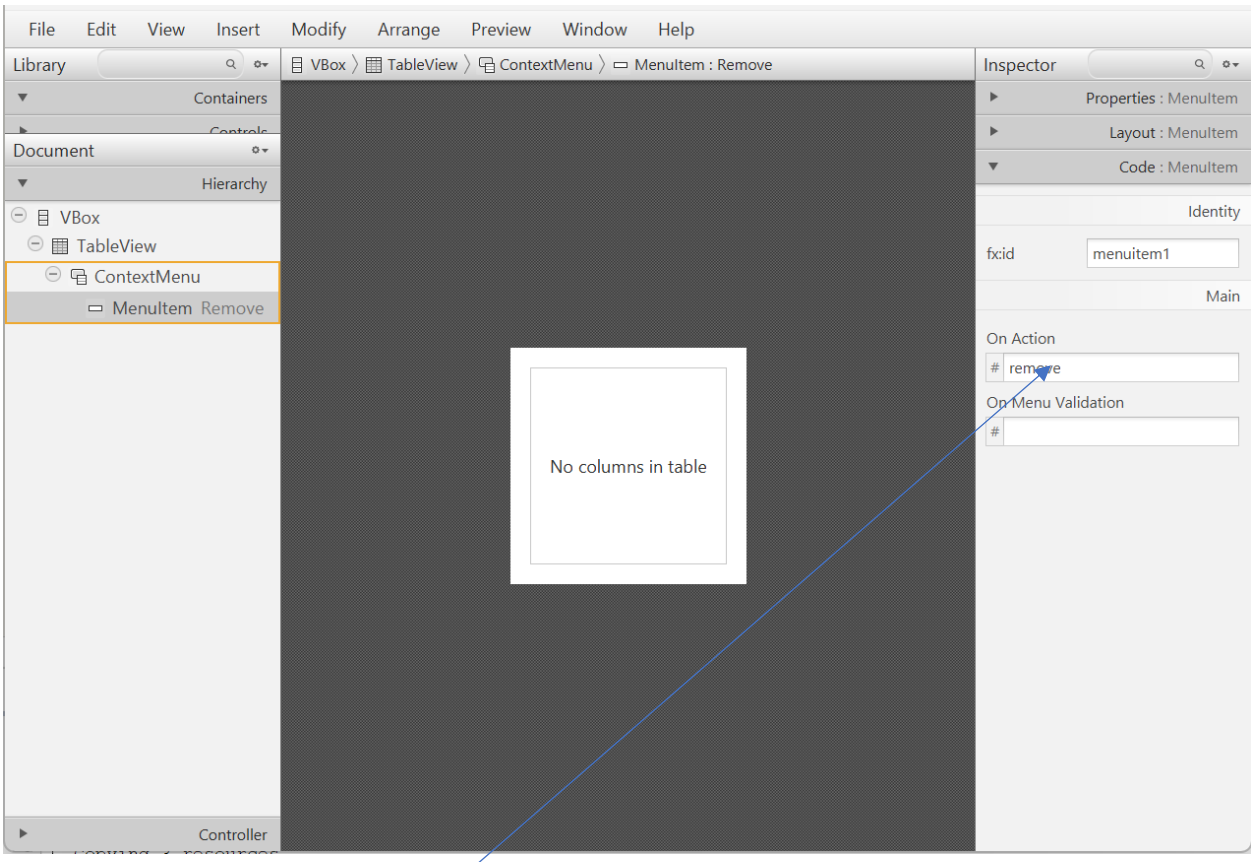
Aggiungere la tabella al programma e caricare, al momento dell’inizializzazione i dati dal database, i dati nella tabella.

Creazione di menu a scomparsa

La tabella può essere configurata per includere un menu a scomparsa in corrispondenza con gli elementi mostrati, ad esempio un menu per cancellare l'elemento attualmente selezionato.



A tal fine si deve aggiungere un elemento ContextMenu con almeno un elemento MenuItem nella tabella.



Una volta creato un MenuItem (una voce del menu) si deve specificare il nome della funzione da collegare all'invocazione della voce del menu che andrà poi dichiarata nella classe:

```
1.  @FXML
2.  public void remove() {
3.      try ( Connection co = DriverManager.getConnection("jdbc:mysql://localhost:3306/drivers",
4.                          "root", "root");
5.            PreparedStatement ps = co.prepareStatement("DELETE FROM drivers WHERE ID = ?");) {
6.
7.            ps.setInt(1, driversTable.getSelectionModel().getSelectedItem().getId());
8.            ps.executeUpdate();
9.
10.           ol.remove(driversTable.getSelectionModel().getSelectedItem());
11.     } catch (SQLException e) {
12.           logger.error(e.getMessage());
13.     }
14. }
```

Nel codice si effettua una connessione al database (riga 2-3) per rimuovere l'elemento selezionato (righe 7-8). Nella riga 10 infine si va a rimuovere l'elemento dalla tabella, rimuovendo l'istanza selezionata, il cui ID e' recuperato tramite la funzione "driversTable.getSelectionModel().getSelectedItem()", dall'array che contiene i dati mostrati in tabella.

Esercizio

Aggiungere il menu a scomparsa per la cancellazione degli utenti.

Gestione delle operazioni utilizzando i Task

Molte delle operazioni svolte sono operazioni che potrebbero richiedere un tempo considerevole per l'interazione con il database. Come abbiamo visto bisognerebbe svolgere tali operazioni all'interno di un Task per non lasciare l'interfaccia bloccata per troppo tempo:

```
1.  Task task = new Task<Void>() {
2.              @Override
3.              public Void call() {
4.                  try ( Connection co =
    DriverManager.getConnection("jdbc:mysql://localhost:3306/drivers", "root", "root");
    PreparedStatement ps = co.prepareStatement("DELETE FROM drivers WHERE ID = ?");) {
5.
6.                      ps.setInt(1,
    driversTable.getSelectionModel().getSelectedItem().getId());
7.
8.                      ps.executeUpdate();
9.
10.                     ol.remove(driversTable.getSelectionModel().getSelectedItem());
11.                 } catch (SQLException e) {
12.                     logger.error(e.getMessage());
13.                 }
14.
15.                 return null;
16.             }
17.         };
18.
19.
20.
21.         new Thread(task).start();
22. }
23.
```