PANEVROPSKI UNIVERZITET „APEIRON“

FAKULTET INFORMACIONIH TEHNOLOGIJA

BANJA LUKA

JAVA KORISNIČKI INTERFEJS PREMA BAZI PODATAKA

Seminarski rad iz predmeta Objektno-orjentisano programiranje

Predavač :

prof. dr Saša Salapura

Student:

Jovana Dujaković

60-18/VITP-S

Banja Luka, jun 2021.

SADRŽAJ

UVOD

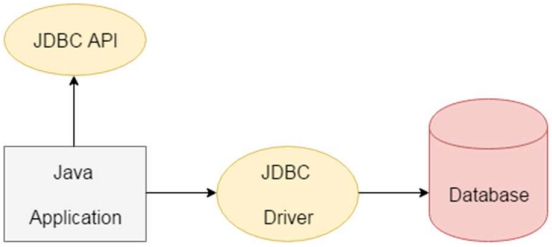
Tema ovog rada je korisnički interfejs prema bazi podataka. Ovim radom je pokazano koliko je širok spektar kodiranja i programiranja. Interesantna je, takođe činjenica da se uz pomoć kodiranja može pristupiti podacima u bazi podataka. Takođe, pomenuće se i istaknuti interfejsi, kao i poneki primjer.

JDBC

JDBC je skraćenica od Java Database Connectivity. JDBC je Java API za povezivanje i izvršavanje upita sa bazom podataka. Dio je od JavaSE-a (Java Standard Edition). JDBC API koristi JDBC upravljačke programe za povezivanje sa bazom podataka. Postoje četiri vrste JDBC pokretača:

1. Upravljački program mosta JDBC-ODBC,
2. Izvorni driver,
3. Upravljački program mrežnog protokola i
4. Tanki driver

JDBC API možemo koristiti za pristup tabličnim podacima pohranjenim u bilo kojoj relacionoj bazi podataka. Pomoću JDBC API-ja možemo sačuvati, ažurirati, brisati i pristupati podacima iz baze podataka. To je poput otvorene povezanosti baze podataka (ODBC) koju pruža Microsoft.



Ilustracija 1 - JDBC

Trenutna verzija JDBC-a je 4.3. Stabilno je izdanje od 21. septembra 2017. Zasnovano je na X / Open SQL interfejsu na nivou poziva. Paket java.sql sadrži klase i interfejs za JDBC API.

Popis popularnih *interfejsa* JDBC API-a dat je u nastavku:

- Driver interfejs
- Interfejs za povezivanje
- Interfejs izjave
- PreparedStatement intefejs
- CallableStatement interfejs
- ResultSet interfejs
- ResultSetMetaData interfejs
- DatabaseMetaData interfejs
- RowSet interfejs

Spisak popularnih *klasa* JDBC API-a dat je u nastavku:

→ Klasa DriverManager
→ Klasa blob
→ Klasa klapa
→ Tipovi klasa

Zašto bismo trebali koristiti JDBC

Prije JDBC-a, ODBC API bio je API baze podataka za povezivanje i izvršavanje upita sa bazom podataka. Ali, ODBC API koristi ODBC upravljački program koji je napisan na jeziku C (tj.zavisno od platforme). Zbog toga je Java definisala vlastiti API (JDBC API) koji koristi JDBC upravljačke programe (napisane na Java jeziku).

Možemo da koristimo JDBC API za rukovanje bazom podataka pomoću Java programa i možemo da obavljamo navedene aktivnosti:

- Povezivati se s bazom podataka
- Izvršiti upite i ažurirati izraze u bazi podataka
- Preuzeti rezultat primljen iz baze podataka

DRIVER MANAGER CLASS

Klasa DriverManager djeluje kao interfejs između korisnika i upravljačkih programa. On evidentira dostupne upravljačke programe i rukuje uspostavljanjem veze između baze podataka i odgovarajućeg upravljačkog programa. Klasa DriverManager održava popis klasa Driver koji su se sami registrovali pozivanjem metode DriverManager.registerDriver ().

METODA	OPIS
1) public static void registerDriver(Driver driver):	koristi se za registraciju datog upravljačkog programa u DriverManageru.
2) public static void deregisterDriver(Driver driver):	koristi se za odjavu datog upravljačkog programa (ispustite upravljački program sa liste) pomoću DriverManager-a.
3) public static Connection getConnection(String url):	koristi se za uspostavljanje veze sa navedenim URL-om.
4) public static Connection getConnection(String url,String userName,String password):	koristi se za uspostavljanje veze s navedenim URL-om, korisničkim imenom i lozinkom.

OSNOVNA USLUGA ZA UPRAVLJANJE SKUPOM JDBC UPRAVLJAČKIH PROGRAMA

DataSource intefejs (novo u JDBC 2.0 API), pruža još jedan način povezivanja s izvorom podataka. Korištenje DataSource objekta je poželjno sredstvo povezivanja s izvorom podataka.

Kao dio svoje inicijalizacije, klasa DriverManager pokušaće učitati klase upravljačkih programa na koje se odnosi sistemska osobina "jdbc.drivers". To omogućava korisniku da prilagodi JDBC upravljačke programe koje koriste njihove aplikacije. Na primjer, u datoteci ~ / .hotjava / properties možete navesti:

jdbc.drivers = foo.bah.Driver: wombat.sql.Driver: bad.taste.ourDriver

Metode od DriverManager getConnection i getDrivers poboljšane su da podržavaju mehanizam Java Service Provider Service Provider. JDBC 4.0 upravljački programi moraju sadržavati datoteku META-INF / services / java.sql.Driver. Ova datoteka sadrži ime implementacije JDBC upravljačkih programa java.sql.Driver. Na primjer, za učitavanje klase my.sql.Driver, datoteka META-INF / services / java.sql.Driver sadržavala bi unos:

my.sql.Driver

Aplikacije više ne trebaju eksplicitno učitavati JDBC upravljačke programe pomoću Class.forName (). Postojeći programi koji trenutno učitavaju JDBC upravljačke programe pomoću Class.forName () nastaviće raditi bez izmjena.

Kada se pozove metoda getConnection, DriverManager će pokušati pronaći odgovarajući upravljački program među onima koji se učitavaju prilikom inicijalizacije i onima koji se učitavaju eksplicitno koristeći isti učitavač klasa kao trenutni aplet ili aplikacija.

Počevši od Java 2 SDK, standardno izdanje, verzija 1.3, protok evidentiranja može se postaviti samo ako je dato odgovarajuće odobrenje. Obično se to radi s alatom PolicyTool, koji se može koristiti za davanje dozvole java.sql.SQLPermission "setLog".

JAWA SWING VODIČ

Vodič za Java Swing dio je Java Foundation Classes (JFC) koji se koristi za stvaranje aplikacija koje su + zasnovane na prozorima. Izgrađen je na vrhu API-ja AWT (Abstract Windowing Toolkit) i u potpunosti je napisan u Javi.

Za razliku od AWT-a, Java Swing nudi lagane komponente i one koje su nezavisne od platformi.

Paket javax.swing nudi klase za Java swing API kao što su JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser itd.

POVEZIVANJE JAVA BAZE BODATAKA SA MYSQL-OM

Da bismo povezali Java aplikaciju s MySQL bazom podataka, moramo slijediti 5 sljedećih koraka.

U ovom primjeru koristimo MySql kao bazu podataka. Dakle, moramo znati sljedeće informacije za mysql bazu podataka:

-Klasa upravljačkih programa: Klasa upravljačkih programa za mysql bazu podataka je com.mysql.jdbc.Driver.

-URL veze: URL veze za mysql bazu podataka je jdbc: mysql: // localhost: 3306 / sonoo gdje je jdbc API, **mysql** je baza podataka, **localhost** je ime servera na kojem je mysql pokrenut, možemo koristiti i IP adresu , gdje je npr. **3306** broj porta, a **sonoo** je ime baze podataka. Možemo koristiti bilo koju bazu podataka, u tom slučaju moramo zamijeniti sonoo imenom baze podataka.

-Korisničko ime: Podrazumijevano korisničko ime za mysql bazu podataka je **root**.

-Lozinka: To je lozinka koju je korisnik dao u vrijeme instalacije mysql baze podataka.

Kreira se tablica u mysql bazi podataka, ali prije stvaranja tablice prvo moramo stvoriti bazu podataka.

INTERFEJS ZA POVEZIVANJE

Veza označava sesiju između Java aplikacije i baze podataka. Interfejs za povezivanje je fabrika Statement-a, PreparedStatement-a i DatabaseMetaData, tj. Object Connection može se koristiti za dobijanje objekta Statement-a i DatabaseMetaData. Intefejs za povezivanje pruža mnoge metode za upravljanje transakcijama poput commit (), rollback () itd.

Uobičajene metode povezivanja interfejsa:

1) public Statement createStatement(): stvara objekat izraza koji se može koristiti za izvršavanje SQL upita.

2) public Statement createStatement(int resultSetType,int resultSetConcurrency): stvara objekat Statement koji će da generiše ResultSet objekte s danim tipom i paralelnošću.

3) public void setAutoCommit(boolean status): koristi se za postavljanje statusa urezivanja. Po defaultu je tačno.

4) public void commit (): trajno čuva promjene napravljene od prethodnog urezivanja / vraćanja.

5) public void rollback(): Ispušta sve promjene napravljene od prethodnog urezivanja / vraćanja.

6) prekida vezu i odmah objavljuje JDBC resurse.

RAD SA JAVA FX UI I JDBC APLIKACIJAMA AVATAR

Budući da JavaFX postaje sve jači kao Javin faktički GUI okvir, on će prije ili kasnije zamijeniti Swing. JavaFX UI i JDBC mogu biti efikasna kombinacija pri kreiranju aplikacije vođene bazom podataka, posebno u vanmrežnom ili ugrađenom sistemu. Ovaj članak u osnovi pokazuje kako se to može učiniti s primjerom scenarija.

PREGLED JDBC APLIKACIJE

Evolucija Java GUI okvira se sada zasniva na JavaFX biblioteci. Pruža snažnu, ali fleksibilnu alternativu razvoju grafičkog korisničkog interfejsa, za razliku od postojećeg Swing i AWT okvira. JavaFX pruža veliku lepezu ili kontrole i komponente koje pomažu u brzoj i efikasnoj izgradnji GUI interfejsa. Vrlo je lako razviti desktop aplikaciju koja komunicira sa pozadinskom bazom podataka. JDBC (Java Database Connectivity) aplikacija prvenstveno ima pozadinski sistem baza podataka kao što su MySQL, Derby, Oracle ili bilo koja druga baza podataka. Java kôd je napisan za dohvatanje zapisa iz jedne ili više tablica u bazi podataka. Upiti SQL (jezik strukturiranog upita) pokreću se iz Java koda i šalju mehanizmu baze podataka na obradu. JDBC pokretački program djeluje kao posrednik između Java programa i baze podataka i interpretira skup informacija tu i tamo, tako da se nenadmašna strana, poput baze podataka, i Java programa mogu pomiriti s izvedljivim rješenjem. Baza podataka apsolutno nema pojma o Java kodu, njegovim sintaksama ili bilo čemu o njemu. Jednostavno razumije SQL i može komunicirati samo s njim. S druge strane, Java je OOP (objektno orijentirano programiranje) jezik i nema pojma o SQL-u ili njegovim sintaksama. Da bi omogućio komunikaciju, dobavljač baze podataka isporučuje izvorne upravljačke programe zajedno s bazom podataka. To se naziva JDBC upravljački program. Imajte na umu da su na raspolaganju četiri vrste upravljačkih programa. U kolokvijalnom su nazivu pokretači tipa-1, tipa-2, tipa-3 i tipa-4. Izvorni pokretački programi su upravljački programi tipa 4 i najčešće se koriste. Oni su također efikasniji od ostalih vrsta. Java program može ove JDBC upravljačke programe uključiti kao vanjsku biblioteku u Java program, jer obično dolaze u JAR arhivskim datotekama.

JavaFX u sceni

Svaka aplikacija baze podataka zahtijeva interfejs tako da korisnik može komunicirati s informacijama iz baze podataka. Bolje je, ako se radi o GUI interfejsu gdje se ne moramo spuštati do zastrašujućeg naredbodavnog interfejsa niskog nivoa, već klikom na dugme dobiti ono što želimo. U ovom aspektu, JavaFX s JDBC može biti ubitačna kombinacija jer ima popriličan broj vizualno uzbudljivih GUI komponenata koje se mogu koristiti za predstavljanje zapisa baze podataka na smisleniji način. Na primjer, zapisi se mogu prikazati u tabličnom obliku pomoću kontrole TableView. Ili možemo stvoriti obrazac za dodavanje novih zapisa u tablicu baze podataka. Unos podataka od strane korisnika može se provjeriti

putem Java koda prije slanja u bazu podataka. Pozadinski mehanizam baze podataka dobija pauzu od provjere ispravnosti podataka i zaustavljene obrade zbog greške u unosu. Štaviše, krajnji korisnik može biti laik s malo ili nimalo informacija o ograničenjima ulaznih podataka. To se najbolje radi kada se obrazac za unos kreira pomoću TextField, Label, ComboBox i ListView kontrola u JavaFX-u. Događaji generisani pomoću button i drugih kontrola obrađuju se na takav način da je korisniku ugodno dok komunicira s GUI interfejsom.

U primjer scenarija

U sledećem ilustrovanom primjeru implementiraćemo operaciju pretraživanja ListView unosom teksta u TextField. Odabrana stavka u ListView preuzima se u skladu s tim - iz pozadinske baze podataka i prikazuje se u kontroli TableView. Dakle, to je prvenstveno aplikacija za preuzimanje i prikazivanje. Ostale operacije baze podataka - poput umetanja, brisanja i ažuriranja zapisa - ne provode se zbog ograničenja veličine.

Dakle, prije nego što započnemo, moramo stvoriti tablicu baze podataka i Java projekat. Mi ćemo koristiti MySQL kao pozadinsku bazu podataka, dok možete odabrati bilo koji drugi, ali obavezno uključite odgovarajuće upravljačke programe u svoju datoteku pom.xml(npr). Ovdje je dio SQL koda za kreiranje tablice, umetanje nekih lažnih podataka i neke druge operacije.

```
CREATE DATABASE addressbook;
USE DATABASE addressbook;

DROP TABLE IF EXISTS contact;

CREATE TABLE contact(
  id INT UNSIGNED NOT NULL AUTO_INCREMENT,
  name VARCHAR(100) NOT NULL,
  nick_name VARCHAR(20),
  address VARCHAR(128),
```

```
  home_phone VARCHAR(10),
  work_phone VARCHAR(10),
  cell_phone VARCHAR(10),
  email VARCHAR(100),
  birthday date,
  web_site VARCHAR(100),
  profession VARCHAR(100),
  PRIMARY KEY (id)
);

INSERT INTO contact (name, nick_name, address, home_phone,
  work_phone, cell_phone, email, birthday, web_site,profession)
  VALUES ('Bruce Wayne', 'batman', 'XYZ Batcave', '9876543210',
  '6278287326', '9182872363', 'batman@gmail.com',
  '1976/02/03', 'batblog.com', 'Super Hero');
...

INSERT INTO contact (...) VALUES (...);
```

Sada, kreiramo objekat domene koji ćemo koristiti i u ListView i u TableView jer su oba povezana, kako je navedeno u našem slučaju. TableView će sadržati vidljivi spisak ljudi (ContactPerson) na osnovu imena izabranog lica iz kontrole ListView. Takođe imamo TextField za brzu pretragu stavki (ime ContactPerson) sadržanih u ListView. Po odabiru određene stavke iz ListViewa, aktivira se SQL upit i dohvate se relevantni zapisi kako bi se popunila kontrola TableView u skladu s tim.

Objekt domene: ContactPerson

Klasa ContactPerson nije ništa drugo do POJO predstavljanje atributa tablice kontakata. Sadrži konstruktor i jednostavne metode dobijanja-postavljanja.

```
package org.mano.jdbc.examples;
```

```
import java.util.Date;
public class ContactPerson {
  private int id;
  private String name;
  private String nickName;
  private String address;
  private String homePhone;
  private String workPhone;
  private String cellPhone;
  private String email;
  private Date birthDate;
  private String webSite;
  private String profession;
  public ContactPerson() {
  }
  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public String getNickName() {
    return nickName;
  }
  public void setNickName(String nickName) {
```

```
  this.nickName = nickName;
}
public String getAddress() {
  return address;
}
public void setAddress(String address) {
  this.address = address;
}
public String getHomePhone() {
  return homePhone;
}<
public void setHomePhone(String homePhone) {
  this.homePhone = homePhone;
}
public String getWorkPhone() {
  return workPhone;
}
public void setWorkPhone(String workPhone) {
  this.workPhone = workPhone;
}
public String getCellPhone() {
  return cellPhone;
}
public void setCellPhone(String cellPhone) {
  this.cellPhone = cellPhone;
}
public String getEmail() {
  return email;
}
public void setEmail(String email) {
  this.email = email;
```

```
  }
  public Date getBirthDate() {
    return birthDate;
  }
  public void setBirthDate(Date birthDate) {
    this.birthDate = birthDate;
  }
  public String getWebSite() {
    return webSite;
  }
  public void setWebSite(String webSite) {
    this.webSite = webSite;
  }
  public String getProfession() {
    return profession;
  }
  public void setProfession(String profession) {
    this.profession = profession;
  }
}
```

Predmet pristupa podacima: ContactDAO

ContactDAO je klasa objekta pristupa podacima koja prvenstveno uključuje operaciju pristupa bazi podataka. Primjenjuje DAO interfejs. Ovaj interfejs možda nije važan u našem primjeru, ali se može dobro iskoristiti ako je aplikacija proširena s više klasa objekata pristupa podacima. Ovdje DAO interfejs uključuje niz veze, upravljački program i korisničko ime i lozinku za pristup MySQL bazi podataka.

DAO.java

```
package org.mano.jdbc.examples;
public interface DAO {
  public static final String DB_URL =
```

```
    "jdbc:mysql://localhost:3306/"+
  "addressbook?zeroDateTimeBehavior=convertToNull";
  public static final String DRIVER =
    "com.mysql.jdbc.Driver";
  public static final String USER = "root";
  public static final String PASS = "secret";
}
```

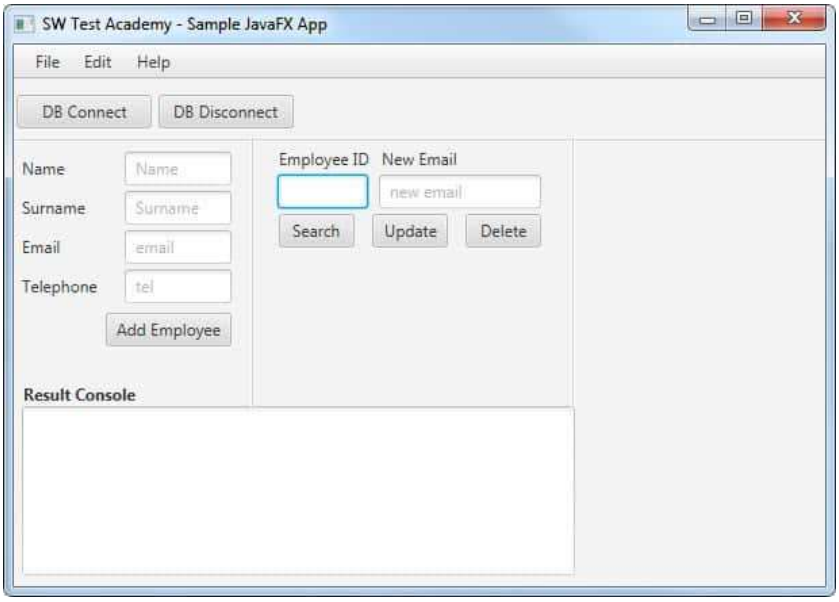
JAVA FX GUI INTERFEJS: CONTACTBROWSER

U JavaFX aplikaciji nazvanoj ContactBrowser, sve smo kontrole postavili programski. Ovo se također može postaviti pomoću FXML-a ili pomoćnih alata buildera, kao što je Scene Builder. Ali, prema različitim razmišljanjima, oni se mogu koristiti kada neko stekne dovoljno iskustva o onome što ide iza zavjesa u JavaFX-u. GUI je prvenstveno interakcija tri kontrole, kao što su TextField (searchField), ListView (listView) i TableView (contactTableView). Kôd je sam po sebi razumljiv, a komentari se daju na odgovarajućim mjestima. Lambda izraz koristi se gdje god je to primjenjivo za zadržavanje koda.

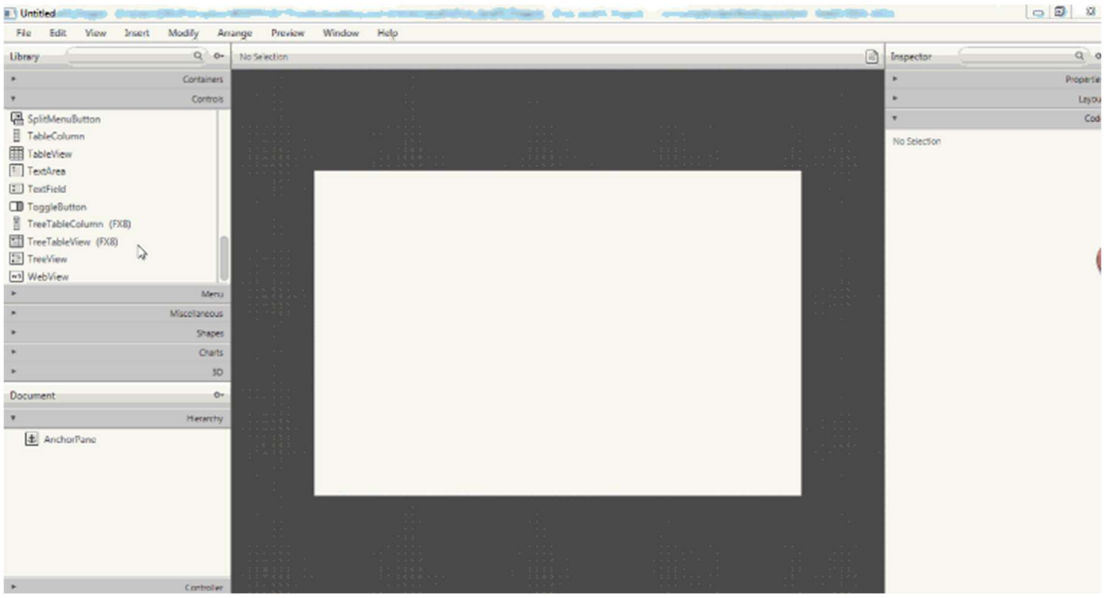
Zaključak

JDBC aplikacija s JavaFX-om u osnovi znači da je JavaFX GUI okvir korišten kao prednji razvojni mehanizam, a JDBC za pozadinsku interakciju s bazom podataka. Mogu biti vrsta tipova sa N brojem funkcionalnosti definisanom u njima. Osnovna je aplikacija CRUD. Implementirali smo dio operacije pretraživanja i prikaza. Evo što možete učiniti da ga proširite: implementirati operacije Stvaranje, Brisanje i Ažuriranje; također, možete dodati imena sa slikama u ListView.

Ovde ćemo se fokusirati na operacije baze podataka u JavaFX-u. U prvom dijelu kreirali smo primjer JavaFX projekta, dizajnirali nacrt verzije korisničkog interfejsa i postavili Oracle XE bazu podataka. U ovom dijelu ćemo stvoriti klase kontrolera, modela, DAO i Util za obavljanje DB operacija. Prije početka kodiranja, želim dati više informacija o našem primjeru projekta. Najnovija verzija korisničkog intefejsa prikazana je na donjoj slici.



UI možete dizajnirati pomoću SceneBuilder-a. Nakon povlačenja i ispuštanja bilo kojeg elementa u okno sidra, morate postaviti njihova svojstva fx: id. Ova svojstva moraju biti jedinstvena i u svom kodu možete koristiti elemente s njihovim jedinstvenim svojstvima fx: id. Također, možete kopirati i zalijepiti bilo koji element u okno sidra, a zatim promijeniti njegovo ime i vrijednost fx: id. Demonstrirano je kako dizajnirati GUI na donjoj slici.



U nastavku su navedeni detalji primjera:

Pretražite radnika pomoću ID-a radnika i prikažite rezultat u prikazu tabele i području teksta. (SELECT)

Pretražite sve radnike i pokažite ih na prikazu tabele. (SELECT *FROM)

Ažurirajte adresu e-mail-a radnika pomoću ID-a zaposlenika. (UPDATE)

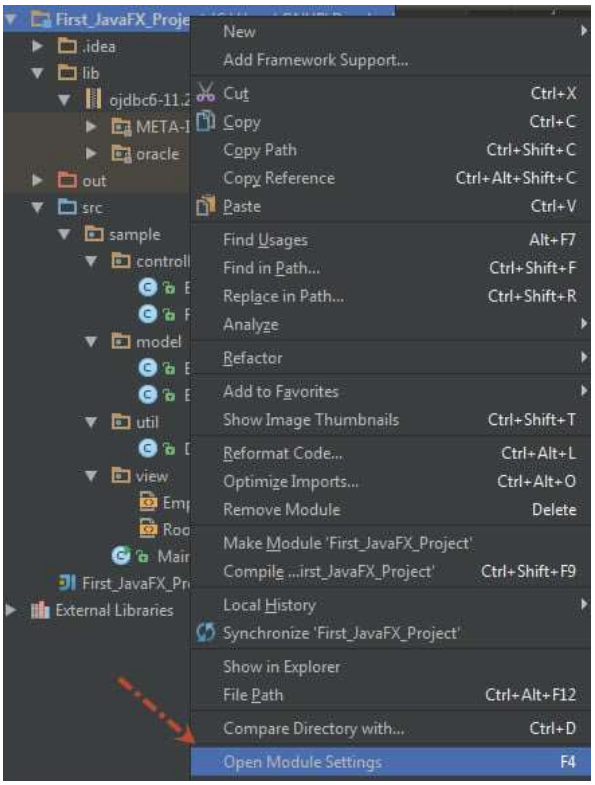
Izbrišite radnika pomoću ID-a zaposlenika. (DELETE)

Ubacite novog radnika u „tablicu zaposlenih“. (INSERT)

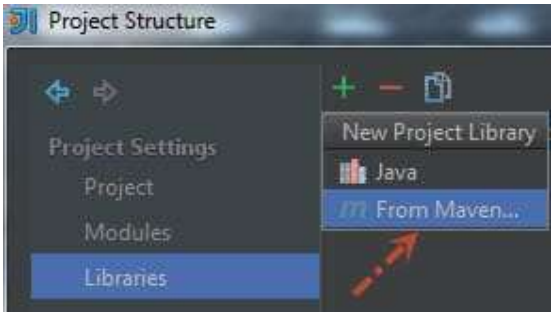
Prikaži rezultate svih operacija na području teksta (Print on Result Console)

Koristićemo Oracle XE bazu podataka i njenu zadanu HR shemu. Da bismo povezali Oracle DB, koristićemo JDBC upravljački program. JDBC upravljački program je softverska komponenta koja omogućava Java aplikaciji interakciju s bazom podataka. Da biste dodali JDBC pokretački program u naš projekt:

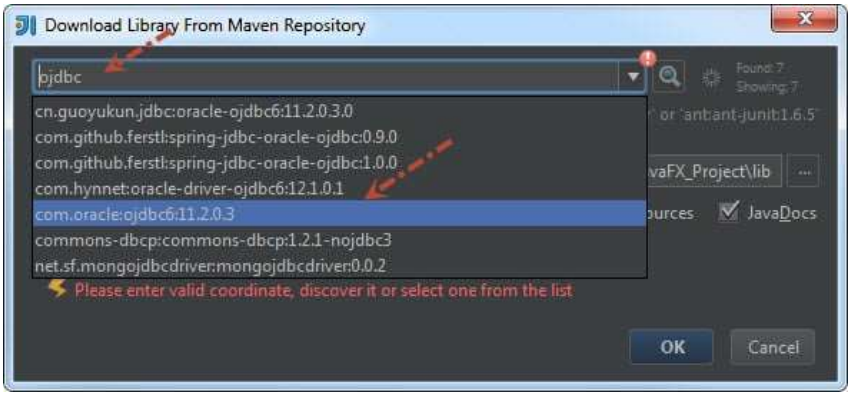
1) Idite na svoj projekt, kliknite desnom tipkom miša, a zatim kliknite **Open Module Settings.**

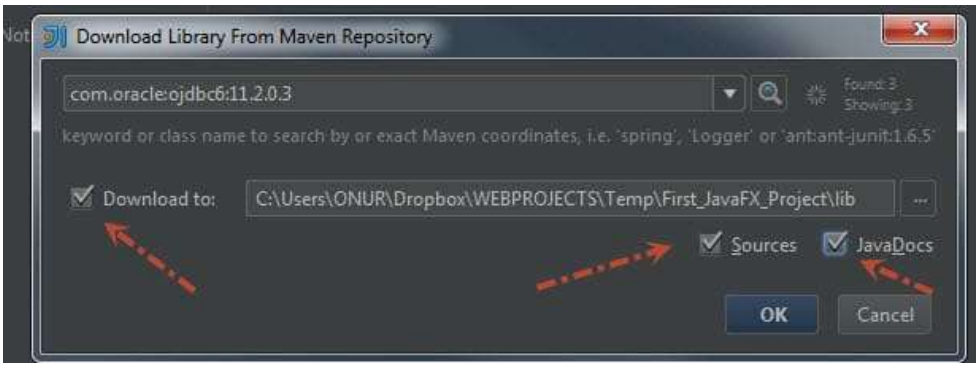


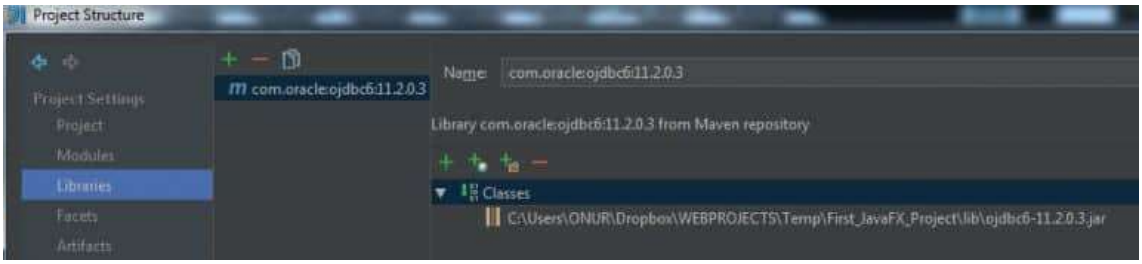
2) Kliknite **Libraries** , a zatim kliknite znak “+” , kliknite “**From Maven**”



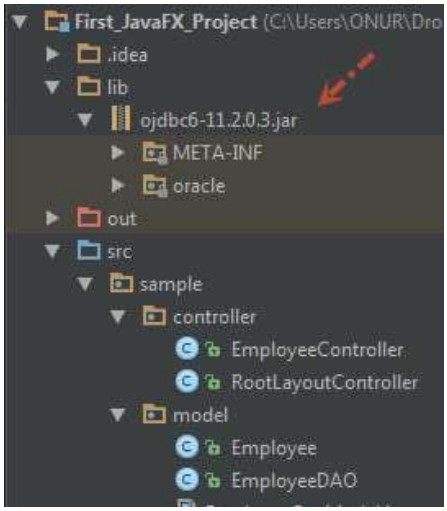
3) Napišite "**odjbc**" u prostoru za pretraživanje, a zatim kliknite enter. Zatim, izaberite"**oracle:ojdbc6:11.2.0.3**" i selektujte **sources** i **JavaDocs** i klik na OK.







4) Sada, možemo da vidimo **ojdbc6:11.2.0.3 library** u lib folderu na IntelliJ.



Konačno, spremni smo za početak kodiranja. Kao što je prikazano na donjoj slici, kreirana su 4 paketa: controller, model, view, i util.

Korišten je sam DAO Design Pattern za obavljanje poslova radnika. Ukratko ćemo objasniti DAO obrazac. U DAO uzorku, domena (poslovna) logika ne komunicira direktno s DB-om. Komunicira s DAO slojem i DAO sloj obrađuje DB operacije i šalje rezultate poslovnom sloju.

Filozofija DAO obrasca je da ako trebate promijeniti glavni mehanizam postojanja, to možete učiniti u DAO sloju, a ne na svim mjestima u poslovnom sloju. Također je vrlo važno da nijedan detalj osnovnog mehanizma povezanog s DB ne iscuri iz DAO sloja na poslovni sloj.

DBUTIL KLASA

Objasnićemo kôd klasi DBUtil. U našem primjeru, klasa DBUtil odgovorna je za DB vezu, DB prekid veze, upit baze podataka i operacije ažuriranja. DAO klasa (EmployeeDAO) koristi metode klase DBUtil za obavljanje DB operacija višeg nivoa.

U DBUtil klasi:

- metoda dbConnect () povezuje se s DB-om.

- metoda dbDisconnect () zatvara DB vezu.

- metoda dbExecuteQuery (String queryStmt) izvršava zadati SQL izraz i vraća skup cachedRowSet. Da bismo eliminisali grešku "java.sql.SQLRecoverableException: Zatvorena veza: sljedeća", vraćamo cachedRowSet umjesto ResultSet. Dakle, možemo koristiti cachedRowSet u drugim klasama i manipulisati tim podacima.

- metoda dbExecuteUpdate (String sqlStmt) izvršava zadate SQL operacije ažuriranja, umetanja i brisanja.

ZAKLJUČAK

Kako je pomenuto, u uvodu, pokušali smo sa približavnjem objašnjenja konecije između baze podataka i Java programskog jezika. Svakako je ovo oblast koja iziskuje dosta proučavanja i znanja, ali može da pomogne u svakodnevnoj primjeni za unos i manipulaciju sa podacima, te je iz tog razloga korisno poznavati bar dio ovoga. Ne podrazumjeva oblast sa kojom sam se prije susretala, ali svakako uz literaturu i pretraživanje se pronašao neki dio koji se mogao izučiti.

LITERATURA

1. https://www.javatpoint.com/java-jdbc, stranica posjećena 10.05.2021.
2. https://docs.oracle.com/javase/8/docs/api/java/sql/DriverManager.html, stranica posjećena 29.05.2021.
3. https://www.youtube.com/watch?v=6K9OpqDM_RM, stranica posjećena 14.06.2021.
4. https://www.developer.com/database/working-with-the-javafx-ui-and-jdbc-applications/, stranica posjećena 10.06.2021.