# JAVA Alkalmazások

# előadás beadandó feladat dokumentáció.

Készítette: Szász András Gábor

Neptun kód: ZYKV5N

GITHUB Repository: <https://github.com/SzaszZYKV5N/JAVAEloadasBeadando>

<https://youtu.be/X6KGsUP3L4I>

A feladatot egyedül készítettem el. A munkafolyamat során az elkészült munkarészeket a GITHUB -ra két profilon keresztül töltöttem fel, SzaszAndrasGabor és SzaszZYKV5N felhasználói fiókokat használva.

**A csatolt tömöritett mappában lévő feladat.db SQLite fájlt a feladat kiírásának megfelelően be kell másolni a c:\adatok nevű könyvtárba!**

A JAVA Alkalmazások elnevezésű tárgyhoz kapcsolódóan elkészítettem a JAVA FX beadandó feladatot. A feadat kiválasztásához a meghírdetett témák közül az 1-02-Vízvezeték szerelők adatbázist választottam, ennek megfelelően az alkalmazás is ehhez a témához kapcsolódóan lett kialakítva.

Az alkalmazás elkészítése során az IntelliJ segítségével létrehozott JavaFX sablont tekintettem kiindulási pontnak, melynek átalakítása (refaktorálása) során létrejöttek a MenuApplication.java, a MenuController.java és a menu.fxml állományok.

Az alkalmazás grafikus alapját képező menu.fxml állományban lett kialakítva a kezdőképernyő, amely megjelenés elemei egy vBox típusú külső virtuális doboz keretben lettek összefogva.

A vertikális elrendezésű Vbox-on belül AnchorPane elemen belül egy Menubar vezérlőelem, és egy másik, változó tartalmat megjelenítő AnchorBox került lehelyezésre.

Az elrendezés kialakításához a Scene Builder programot alkalmaztam, illetve az .fxml állományban szövegesen is szerkesztett állományt menu.fxml állományként mentettem el.

A Menubar vezérlőn belül a feladat kiírás szerinti menü és almenü pontok a Menu és Menuitem tagekkel lett létrehozva.

A változó tartalmat megjelenítő AnchorPane elem a rootPane fx azonosítót kapta, az menüpontokban kijelölt tartalmak ebben a konténerelemben jelennek majd meg.

A képen képernyőkép, szöveg, szoftver, Multimédiás szoftver látható

Automatikusan generált leírás

A MenuController fájlban az egyes menüpontokhoz az onAction tulajdonságoknál a következő függvény szerint lehet a rootPane konténerbe behívni az egyes menüpontokhoz tartozó grafikus tartalmakat:

@FXML  
private void Ujablak(ActionEvent event) throws IOException {  
 AnchorPane pane = FXMLLoader.*load*(getClass().getResource("second.fxml"));  
 rootPane.getChildren().setAll(pane);  
}

Azonban a beadandó feladat összesen 15 menüpontot tartalmaz, ezért nem célszerű minden menüponthoz új függvényt írni, ehelyett két egyszerű metódust alkalmaztam, melynek lényege, hogy mindegyik menüpont egy eseménykezelőt hív meg, amely megvizsgálja, hogy melyik menuitem hívta meg, és ennek megfelelően választja ki a fetöltendő .fxml fájlt.

@FXML  
private void handleMenuAction(ActionEvent event) {  
 MenuItem source = (MenuItem) event.getSource();  
 String fxmlFileName = "";  
  
 switch (source.getText()) {  
 case "Olvas":  
 fxmlFileName = "olvas.fxml";  
 break;  
 case "Olvas2":  
 fxmlFileName = "olvas2.fxml";  
 break;  
 // További esetek hozzáadása szükség szerint  
 default:  
 System.*out*.println("Ismeretlen menüpont: " + source.getText());  
 return;  
 }  
  
 loadFXMLToRootPane(fxmlFileName);  
}

A betöltést elvégző kódrészlet:

private void loadFXMLToRootPane(String fxmlFileName) {  
 try {  
 FXMLLoader loader = new FXMLLoader(getClass().getResource(fxmlFileName));  
 Parent newContent = loader.load();  
 rootPane.getChildren().setAll(newContent);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
}

## Adatbázis menü

Adatbázis létrehozása és tárolása

Az alkalmazásnak alkalmasnak könnyen hordozhatónak kell lennie, és nem szabad külső adatbázistól függenie. Ezt legegyszerűbben úgy lehet elérni, ha az adatokat vagy egyszerű szöveges állományban, vagy pl. SQLite állományban tároljuk, egy könnyez hozzáférhetőkönyvtárban. Az SQLite adatbázisok kezelése a JAVA rendszerekben támogatott, a tárolandó táblák és adatok pedig egyetlen fájlban megtalálhatók. A létrehozandó SQLite fájlt a c:/adatbazis könyvtárban helyeztem el, illetve hoztam létre. Az elkészült alkalmazás futtatásához a célgépen ezt a könyvtárat lére kell hozni és az adatállományt bele kell másolni.

Az adatbázis szerkezete a következő:

**Táblák:**

***szerelo*** (*az*, *nev*, *kezdev*)

*az* a szerelő azonosítója (szám), ez a kulcs

*nev* a szerelő neve (szöveg)

*kezdev* a szerelő ettől az évtől dolgozik a cégnél (szám)

***munkalap*** (*az*, *bedatum*, *javdatum*, *helyaz*, *szereloaz*, *munkaora*, *anyagar*)

*az* a munkalap azonosítója (szám vagy számláló), ez a kulcs

*bedatum* a munka felvételének dátuma (dátum)

*javdatum* a munka elvégzésének dátuma (dátum)

*helyaz* a munkavégzés helyének azonosítója (szám)

*szereloaz* a munkát elvégző szerelő azonosítója (szám)

*munkaora* a munka elvégzéséhez szükséges idő (szám)

*anyagar* a munka elvégzése során felhasznált anyag ára (szám)

***hely*** (*az*, *telepules*, *utca*)

*az* a javítás helyének azonosítója (szám), ez a kulcs

*telepules* a javítás helyének települése (szöveg)

*utca* a javítás helyének utca, házszáma (szöveg)

A képen szöveg, képernyőkép, Betűtípus, Téglalap látható

Automatikusan generált leírás

DB Browser for SQLite segítségével létrehoztam egy új adatbázist, ésbeimportáltam a txt fájlban tárolt adatokat.

A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Automatikusan generált leírás

A képen szöveg, képernyőkép, szám, szoftver látható

Automatikusan generált leírás

Pom.xml-ben hozzáadtam a függőségeket:

<dependency>  
 <groupId>org.xerial</groupId>  
 <artifactId>sqlite-jdbc</artifactId>  
 <version>3.43.0.0</version>  
</dependency>  
<dependency>  
 <groupId>com.github.gwenn</groupId>  
 <artifactId>sqlite-dialect</artifactId>  
 <version>0.1.1</version>  
</dependency>

Az adatbázis tábláknak megfelelően létrehoztam az entitás osztályokat, Hely, Munkalap, Szerelo néven konstruktorokkal, getterekkel és setterekkel.

Létrehoztam egy OlvasDAO. java osztályt a táblázatban megjelenítendő adatok részére. Az ebben az osztályban meghatározott változók az összetett lekérdezéstábláknál lettek felhasználva.

Az olvas.fxml-ben egy a menu.fxml-be beágyazandó felület lett kialakítva, AnchorPane konténeren belül. A táblázat kilakítását a főbb méretek miatt itt alakítottam ki.

A képen szöveg, képernyőkép, szoftver, Betűtípus látható

Automatikusan generált leírás

A menüponthoz önálló kontrollert hoztam létre OlvasController néven. Ebben a fájlban történik meg a táblázat oszlopváltozóinak és konfigurációjának kialakítása. Az initialize() lefutása során a táblázat előre beállított értékei törlődnek, majd megtörténik a táblázat oszlopainak és mezőtípusainak beállítása.

Ezt követően automatikusan meghívja a Tablabovit() metódus, ami feltölti a táblázatot.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

A táblázat sorait a következő formátumú osztály alapján állítjuk elő:

OlvasDAO newOlvasDAO = new OlvasDAO(1,"Budapest", "Podhorszky utca 68"," Kiss Lajos",3,15000);

Az SQL adatbázisból egy mindhárom tábla adatait feldolgozó lekérdezést készítettem, egyszerü jdbc módszerével.

OlvasDAO ujsor;  
Connection conn = DriverManager.*getConnection*("jdbc:sqlite:/c:/adatbazis/javabead.db");  
Statement stmt = conn.createStatement();  
ResultSet rs;  
rs = stmt.executeQuery(  
 "SELECT m.helyaz, h.telepules, h.utca, sz.nev, m.munkaora, m.anyagar " +  
 "FROM szerelo sz, munkalap m, hely h " +  
 "WHERE sz.az = m.szereloaz AND m.helyaz = h.az"+" ORDER BY m.helyaz"  
);  
while(rs.next()){  
 ujsor = new OlvasDAO(rs.getInt("helyaz"), rs.getString("telepules"), rs.getString("utca"),rs.getString("nev"), rs.getInt("munkaora"), rs.getInt("anyagar") );  
 tablazat.getItems().add(ujsor);  
  
}

Eredménye:

A képen szöveg, képernyőkép, szoftver, szám látható

Automatikusan generált leírás

Az Olvas2 menüpont

Itt egy ComboBox-ból lehet kiválasztani a szerelő nevét, szövegmezőben meg lehet adni a keresett utca nevének kezdetét vagy egészét, rádiógombokkal kiválasztható az értékhatár és egy jelölőnényzettel szűrhetőek a 2 munkaóránál többet igénylő munkák. Alapesetben az Olvas menüpont alatti listázás valósul meg a betöltéskor.

Az előzőekben kialakított Tablabovit() metódus lett átalakítva oly módon, hogy a lekérdezés törzsadatainak nevei és az azatbázis táblák nevei változatlanok maradtak, azonban a lekérdezés további részeit egy newFeltetel nevű StringBuilder változóban gyűjtöttem össze a vezérlőeszközök aktuális állapota függvényében, majd egy .toString() metódus meghívásával egy feltetel nevű stringváltozóba adtamát, végül ezt a változót fűztem hozzá az alap lekérdezéshez.

A szerelő kiválasztásához használatos cb1 nevű ComboBox elemet a Legordulo() metódussal töltöttem fel, az egyes sorokba Szerelo típusú osztályok kerültek, a legördülő listában megjelenő neveket a Szerelo osztályban létrehozott toString függvény állítja elő.

private void Legordulo() throws SQLException {  
 cb1.getItems().clear();  
 cb1.getItems().add(new Szerelo(0, "Válassz")); // Alapértelmezett elem  
 Connection conn = DriverManager.*getConnection*("jdbc:sqlite:/c:/adatbazis/javabead.db");  
 Statement stmt = conn.createStatement();  
 ResultSet rs = stmt.executeQuery("SELECT az, nev FROM szerelo");  
  
 while (rs.next()) {  
 int id = rs.getInt("az");  
 String nev = rs.getString("nev");  
 cb1.getItems().add(new Szerelo(id, nev));  
 }  
 cb1.getSelectionModel().selectFirst(); // Első elem kiválasztása  
}

A tábla szűrés nélküli lekérdezéséhez a következő Stringet hoztam létre:

private String feltetel=" AND sz.az = m.szereloaz AND m.helyaz = h.az ORDER BY m.helyaz";

A vezérlőeszközök lekérdezése a következő kódrészletben történik, amit a Szűrés nyomógomb megnyomásával aktiválunk :

public void szuresClick(ActionEvent event) throws SQLException {  
 int a = kivSzer();  
 StringBuilder newFeltetel = new StringBuilder(" WHERE sz.az = m.szereloaz AND m.helyaz = h.az ");  
 // Szerelő szűrés hozzáadása  
 if (a != 0) {  
 newFeltetel.append(" AND m.szereloaz = ").append(a);  
 }  
 // Utca név szűrés hozzáadása  
 if (!cimKereso.getText().isEmpty()) {  
 newFeltetel.append(" AND h.utca LIKE '").append(cimKereso.getText()).append("%'");  
 }  
 //értékhatár szűrés hozzáadása  
 if(rb1.isSelected())  
 {  
 newFeltetel.append(" AND m.anyagar < 5000 ");  
 }  
 if(rb2.isSelected()){  
 newFeltetel.append(" AND m.anyagar >= 5000 ");  
 }  
 if(ch1.isSelected()){  
 newFeltetel.append(" AND m.munkaora >2 ");  
 }  
 //rendezze sorba  
 newFeltetel.append(" ORDER BY m.helyaz");  
 feltetel = newFeltetel.toString();  
 Tablabovit();  
}

Eredmény: A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Automatikusan generált leírás

Ír almenü

Ennél a menüpontnál új szerelő felvételéra van lehetőség, az eddig rögzítettek kilistázása mellett. Beolvassa az új szerelő adatait a szövegmezőkből (ujSzerelo.getText(), belepesEve.getText()).

Ellenőrzi, hogy az adatok nem üresek és érvényesek.

Létrehoz egy adatbázis kapcsolatot és végrehajt egy INSERT SQL lekérdezést az új rekord hozzáadásához (pstmt.executeUpdate).

Frissíti a táblázatot a Tablabovit metódus meghívásával.

Kiüríti a szövegmezőket a következő bevitelhez (ujSzerelo.clear(), belepesEve.clear()).

public void felvesz(ActionEvent event) throws SQLException {

String szereloNev = ujSzerelo.getText();

int belepesEv = Integer.parseInt(belepesEve.getText());

if (!szereloNev.isEmpty() && belepesEv > 0) {

Connection conn = DriverManager.getConnection("jdbc:sqlite:/c:/adatbazis/javabead.db");

String query = "INSERT INTO szerelo (nev, kezdev) VALUES (?, ?)";

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setString(1, szereloNev);

pstmt.setInt(2, belepesEv);

pstmt.executeUpdate();

// Új rekord felvétele után frissítsd a táblázatot

Tablabovit();

// Mezők ürítése a következő bejegyzéshez

ujSzerelo.clear();

belepesEve.clear();

} else {

// Hibaüzenet, ha bármelyik mező üres vagy hibás

System.out.println("Kérjük, töltse ki a mezőket megfelelően!"); } }

A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Automatikusan generált leírás

Töröl almenü

Megjelenik a szerelo tábla tartalma, a legördülő menüből kiválasztható a törölni kívánt rekord. A legördülő menüben egyszerü lekérdezéssel lettek megadva az értékek:

private void Legordulo() throws SQLException {  
 cb1.getItems().clear();  
 cb1.getItems().add(0); // Alapértelmezett elem  
 Connection conn = DriverManager.*getConnection*("jdbc:sqlite:/c:/adatbazis/javabead.db");  
 Statement stmt = conn.createStatement();  
 ResultSet rs = stmt.executeQuery("SELECT az FROM szerelo");  
  
 while (rs.next()) {  
 int id = rs.getInt("az");  
 cb1.getItems().add(id);  
 }  
 cb1.getSelectionModel().selectFirst(); // Első elem kiválasztása  
}

Feldolgozása:

public void torol(ActionEvent event) throws SQLException {  
 int kivalasztottId = cb1.getSelectionModel().getSelectedItem();  
  
 if (kivalasztottId != 0) { // Ellenőrizd, hogy nem az alapértelmezett elem van kiválasztva  
 Connection conn = DriverManager.*getConnection*("jdbc:sqlite:/c:/adatbazis/javabead.db");  
 String query = "DELETE FROM szerelo WHERE az = ?";  
 PreparedStatement pstmt = conn.prepareStatement(query);  
 pstmt.setInt(1, kivalasztottId);  
 pstmt.executeUpdate();  
  
 // Rekord törlése után frissítia táblázatot és a ComboBox-ot  
 Tablabovit();  
 Legordulo();  
 } else {  
 // Hibaüzenet, ha az alapértelmezett elem van kiválasztva  
 System.*out*.println("Kérjük, válasszon egy szerelőt a törléshez!");  
 }  
}

Adatok módosítása

A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Automatikusan generált leírás

Itt a legördülőmenüből kiválasztott rekord adatai automatikusan betöltődnek a szövegmezőkbe, a gomb megnyomása után frissülnek az adatok a szerelo táblában.

private void adatBetoltes() throws SQLException {  
 int kivalasztottId = cb1.getSelectionModel().getSelectedItem();  
 if (kivalasztottId == 0) {  
 ujSzerelo.clear();  
 belepesEve.clear();  
 return;  
 }  
  
 String query = "SELECT nev, kezdev FROM szerelo WHERE az = ?";  
 try (Connection conn = DriverManager.*getConnection*(DB\_URL);  
 PreparedStatement pstmt = conn.prepareStatement(query)) {  
  
 pstmt.setInt(1, kivalasztottId);  
 try (ResultSet rs = pstmt.executeQuery()) {  
 if (rs.next()) {  
 ujSzerelo.setText(rs.getString("nev"));  
 belepesEve.setText(String.*valueOf*(rs.getInt("kezdev")));  
 }  
 }  
 }  
}

A tábla módosítása: public void modosit() throws SQLException {  
 int kivalasztottId = cb1.getSelectionModel().getSelectedItem();  
 String szereloNev = ujSzerelo.getText();  
 int belepesEv;  
  
 try {  
 belepesEv = Integer.*parseInt*(belepesEve.getText());  
 } catch (NumberFormatException e) {  
 System.*out*.println("Hibás év formátum!");  
 return;  
 }  
  
 if (kivalasztottId != 0 && !szereloNev.isEmpty() && belepesEv > 0) {  
 String query = "UPDATE szerelo SET nev = ?, kezdev = ? WHERE az = ?";  
 try (Connection conn = DriverManager.*getConnection*(DB\_URL);  
 PreparedStatement pstmt = conn.prepareStatement(query)) {  
  
 pstmt.setString(1, szereloNev);  
 pstmt.setInt(2, belepesEv);  
 pstmt.setInt(3, kivalasztottId);  
 pstmt.executeUpdate();  
  
 // Frissítsd a táblázatot  
 Tablabovit();  
 // Töröld a mezőket  
 ujSzerelo.clear();  
 belepesEve.clear();  
 cb1.getSelectionModel().selectFirst();  
 }  
 } else {  
 System.*out*.println("Kérjük, töltse ki a mezőket megfelelően!");  
 }  
}

## SoapKliens menü

A tanúsítványt nem lehet közvetlenül a futtatható állományba beépíteni, de a JavaFX alkalmazás futtatása során biztosíthatod, hogy a tanúsítvány elérhető legyen. Az alábbiakban bemutatom, hogyan lehet ezt megoldani: A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

1. Tanúsítvány Importálása a Java Keystore-ba

Nyisd meg a parancssort vagy terminált.

Navigálj a Java telepítési könyvtárába, ahol a keytool eszköz található (pl. C:\Program Files\Java\jdk-<version>\bin).

Futtasd az alábbi parancsot a tanúsítvány importálásához:

keytool -import -trustcacerts -file path\to\mnb\_cert.cer -alias mnb\_cert -keystore "%JAVA\_HOME%\lib\security\cacerts"

A keystore jelszava alapértelmezés szerint changeit.

1. SSLContext Konfigurálása a Kódban

A tanúsítványt használhatod a kódban az SSLContext konfigurálásával. Az alábbi példa bemutatja, hogyan lehet ezt megtenni:

Példa Kód

import javax.net.ssl.\*;

import java.security.KeyStore;

import java.io.FileInputStream;

import java.net.URL;

import javax.xml.namespace.QName;

import javax.xml.ws.Service;

public class Controller {

@FXML

private void handleDownload() {

DirectoryChooser directoryChooser = new DirectoryChooser();

directoryChooser.setTitle("Válassz egy mappát");

Stage stage = new Stage();

File selectedDirectory = directoryChooser.showDialog(stage);

if (selectedDirectory != null) {

try {

// Load the certificate

KeyStore keyStore = KeyStore.getInstance(KeyStore.getDefaultType());

try (FileInputStream fis = new FileInputStream("path/to/mnb\_cert.cer")) {

keyStore.load(fis, "changeit".toCharArray());

}

// Create a TrustManager that trusts the certificate

TrustManagerFactory tmf = TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());

tmf.init(keyStore);

SSLContext sslContext = SSLContext.getInstance("TLS");

sslContext.init(null, tmf.getTrustManagers(), null);

// Set the default SSL context

HttpsURLConnection.setDefaultSSLSocketFactory(sslContext.getSocketFactory());

URL wsdlURL = new URL("https://www.mnb.hu/arfolyamok.asmx?WSDL");

QName qname = new QName("http://www.mnb.hu/webservices/", "MnBArfolyamService");

Service service = Service.create(wsdlURL, qname);

MnBArfolyam mnBArfolyam = service.getPort(MnBArfolyam.class);

String data = mnBArfolyam.getAllData();

File file = new File(selectedDirectory, "data.txt");

try (FileWriter writer = new FileWriter(file)) {

writer.write(data);

}

showAlert("Siker", "Az adatok sikeresen letöltve!");

} catch (Exception e) {

showAlert("Hiba", "Hiba történt az adatok letöltése során.");

e.printStackTrace();

}

} else {

showAlert("Figyelmeztetés", "Nem választottál mappát.");

}

}

private void showAlert(String title, String message) {

Alert alert = new Alert(AlertType.INFORMATION);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(message);

alert.showAndWait();

}

}

## Párhuzamos menü

A párhuzamos programfutást bemutató menüpontban a feladatkiírásnak megfelelően az egyik Label típusú feliratban 1 másodpercenként, míg a másik Label feliratban két másodpercenként változik a szöveg.

A futást megvalósító kódrészlet:

public class ParhController {  
 @FXML  
 private Label label1;  
 @FXML  
 private Label label2;  
  
 private ScheduledExecutorService futtato;  
  
 @FXML  
 private void elindit() {  
 futtato = Executors.*newScheduledThreadPool*(2);  
  
 futtato.scheduleAtFixedRate(() -> {  
 Platform.*runLater*(() -> label1.setText("Label 1: " + System.*currentTimeMillis*()));  
 }, 0, 1, TimeUnit.*SECONDS*);  
  
 futtato.scheduleAtFixedRate(() -> {  
 Platform.*runLater*(() -> label2.setText("Label 2: " + System.*currentTimeMillis*()));  
 }, 0, 2, TimeUnit.*SECONDS*);  
 }  
  
 public void leallit() {  
 if (futtato != null) {  
 futtato.shutdown();  
 }  
 }

Megvalósulása:

A képen szöveg, képernyőkép, szoftver, számítógép látható

Automatikusan generált leírás

## Forex menü

Az Info menüpont alatt az aktuális számlainformációk jelennek meg a nyomógomb alatti szövegmezőben.

A képen szöveg, elektronika, képernyőkép, szoftver látható

Automatikusan generált leírás

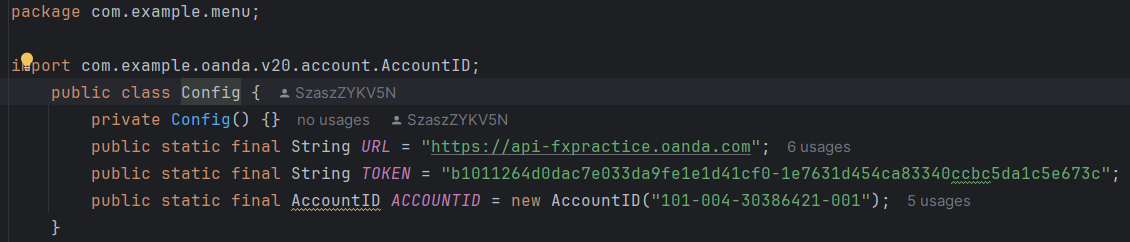
Az OANDA pénzügyi szolgáltató lehetőséget nyújt ingyenes demo számlanyitást, devizakereskedelmi modellezés céljából. A közzétett API felhasználásához a cég honlapján megtalálható a demo számlaszám és a működtetéshez generált token.

A működtetéshez a projekt fájljai közé le kell tölteni az OANDA API állományokat, és a module-info.java fájlban be kell importálni a következőket:

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

Egy Config.java fájlban meg kell adni a banki adatokat, majd erre hivatkozva tudjuk biztosítani a hozzáférést az összes lekérdezés során:



A számlainformációkat a SzamlainfoController által állítjuk elő, a lekérdezés során előállított devizapáronként lekérdezett instrumentumokat egy listában eltároljuk, majd azz egyes elemeit for ciklussal kiolvassuk, és megjelenítjük a szövegterületen.

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Automatikusan generált leírás

A historikus és aktuális áraknál is hasonlóképpen járunk el.

A képen szöveg, képernyőkép, szoftver, Weblap látható

Automatikusan generált leírásA képen szöveg, elektronika, képernyőkép, szoftver látható

Automatikusan generált leírás

A pozíció nyitás és zárás menüpontok banki órákban működnek.

A képen szöveg, képernyőkép, számítógép, szoftver látható

Automatikusan generált leírásA képen szöveg, képernyőkép, szoftver, számítógép látható

Automatikusan generált leírás

Végül a nyitott pozíciók lekérdezésekor az összes folyamatban lévő ügylet betöltődik.

A képen szöveg, elektronika, képernyőkép, szoftver látható

Automatikusan generált leírás