Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

JDBC

Készítette: Hegedűs Bettina

Alkalmazás fejlesztési lehetőségek

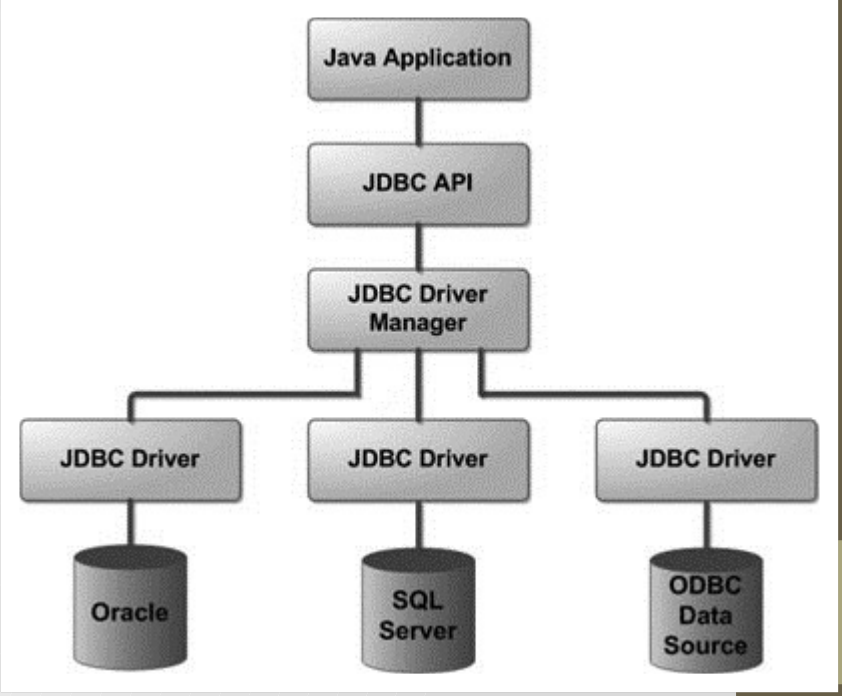
- SQL procedurális kiterjesztései (PL/SQL és társai)
- „Embendded SQL”: az SQL utasításokat beágyazzuk egy host nyelvbe (C,C++ stb.) ezeket fordítás előtt egy előfordító (precompiler) megfelelő függvényhívásokra cseréli.
- Speciális kapcsolódási eszközök, programkönyvtárak alkalmazásával, amelyek segítségével csatlakozni tudunk az adatforráshoz (pl.: ODBC, JDBC)

ODBC és JDBC

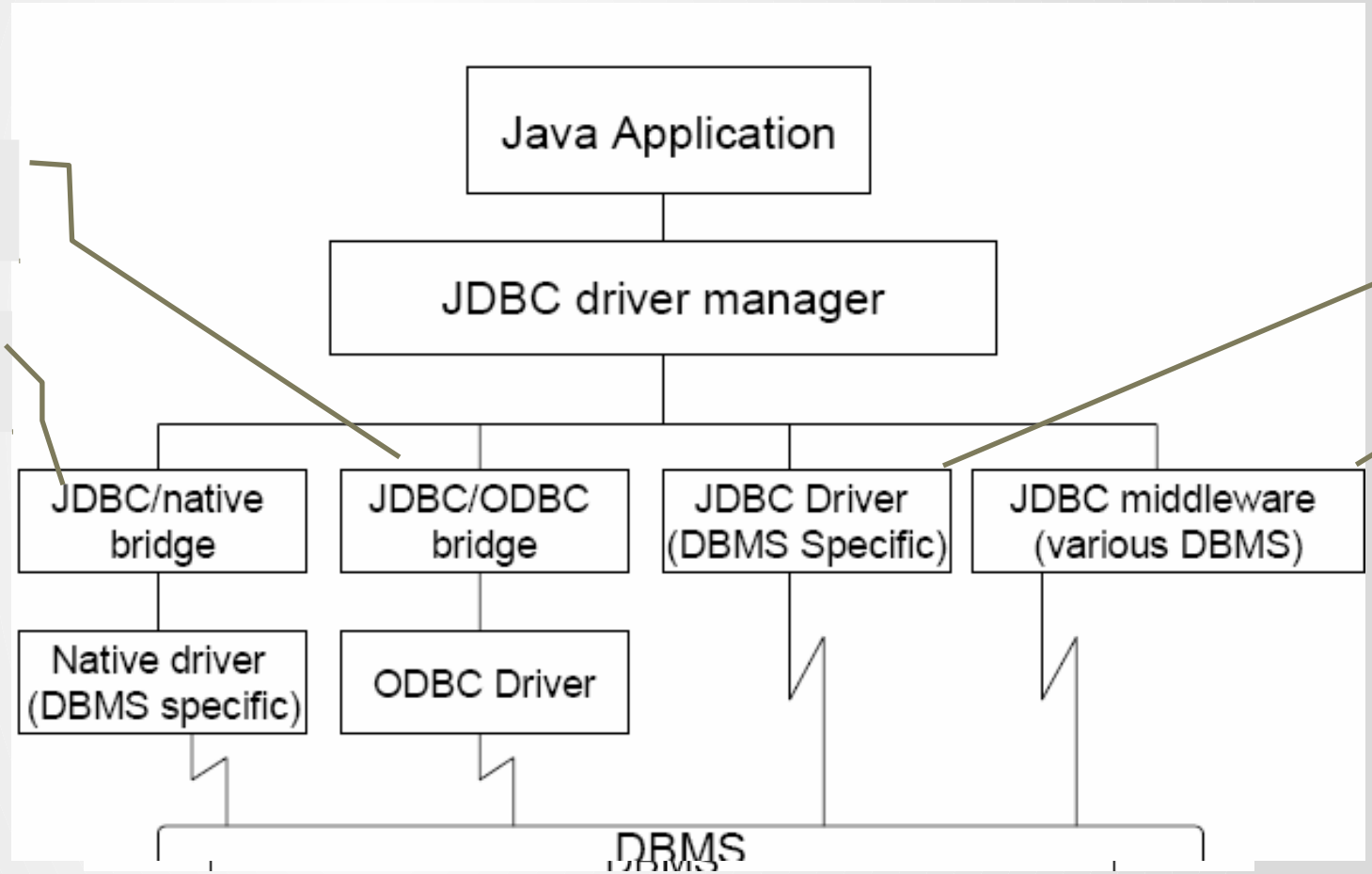
- API (Application Program Interface) mellyel az adatbázis szervert lehet elérni
- A JDBC API szolgáltatásai
  - Kapcsolat létrehozása egy adatforrással(data source)
  - Lekérdező valamint módosító parancsolat lehet küldeni az adatforrásnak
  - Lekérdezések eredményeinek feldolgozása
- Az ODBC (Open Database Connectivity; Microsoft) C, C++, C# és Visual Basic környezetben
- A JDBC (Java Database Connectivity) JAVA-hoz

JDBC felépítése

- Java alkalmazás (kapcsolat nyitás/zárás, SQL kód küldése)
- Driver Manager(ez tölti be az ún. JDBC Drivert)
- Driver(az adatbázis kezelő , adatforrás gyártója adja)
- Data Source (ami végrehajtja a lekérdezést)



JDBC felépítése



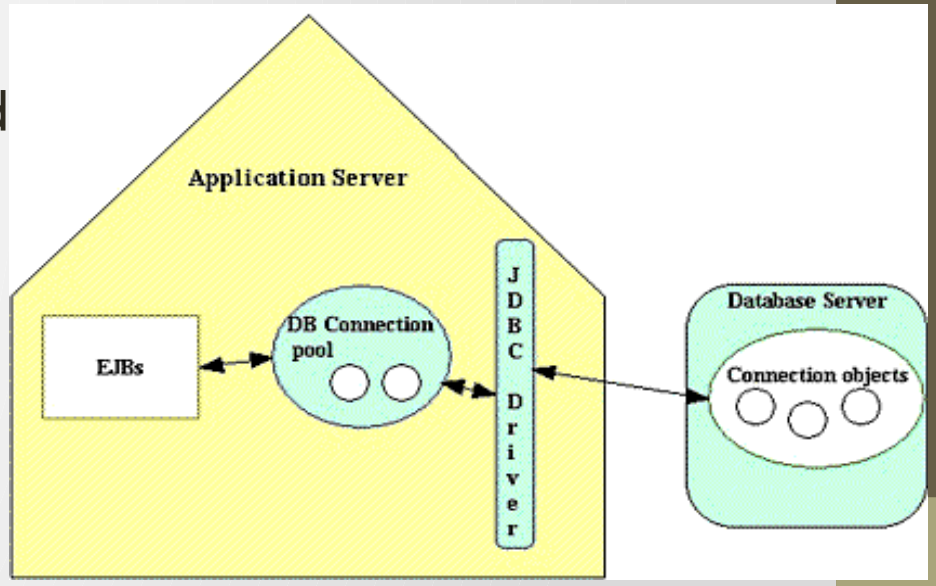
JDBC használati lépései

- Lépések:
  - Betöltjük a Java programunkba a Drivert
  - Csatlakozunk az adatforráshoz
  - Végrehajtjuk az SQL utasítást

  - Részleteiben
    - JVM betölti a Driver interfészt megvalósító osztályt és regisztrálja a DriverManager-nél a DriverManager-től kérünk egy Connection-t
    - A Connectiontól pedig egy Statement-et
    - A Statement-tel hajtjuk végre az SQL utasítást, aminek az eredménye egy ResultSet
    - Ez a ResultSet pedig iterátor segítségével bejárható
    - Végén lezárjuk a kapcsolatot.

JDBC driver betöltés módjai

- DataSource
  - DataSource interfész segítségével az adatforrás használata jobban optimalizálható,
  - adatforrás URL-jét nem szükséges fixen kódolnunk, JNDI-t elég ismerni -> hordozhatóság
- DataManager
  - Mindent kell ismerni a kód
    - host,
    - port,
    - username,
    - password,
    - driver class)



  - PL.:"jdbc:oracle:thin:@oracle.itk.ppke.hu:1521:gyak"

DataSource

```
import java.sql.*;
Import oracle.jdbc.pool.*;

public class ora {
public static void main(String[] args) throws ClassNotFoundException,
  SQLException
{
String url="jdbc:oracle:thin:@oracle.itk.ppke.hu:1521:pract";
String felh="user";
String pass="pass";
OracleDataSource ods=new OracleDataSource();
ods.setURL(url);
ods.setUser(felh);
ods.setPassword(pass);
Connection conn=ods.getConnection();
}
}
```

DriverManager

```
import java.sql.*;
import oracle.jdbc.pool.*;

public class ora{
public static void main(String[] args)
throws ClassNotFoundException, SQLException
{
String url="jdbc:oracle:thin:@oracle.itk.ppke.hu:1521:gyak";
String usr="user";
String pass=„pass";
String adatbazis="oracle.jdbc.driver.OracleDriver";
Class.forName(adatbazis);
Connection conn=
  DriverManager.getConnection(url,usr,pass);
}
}
```

JDBC URL

host:port:service

String
url="**jdbc:oracle:thin:@oracle.itk.ppke.hu:1521:gyak**

JDBC Driver név

Más adatbáziskezelő, más formátumú a JDBC URL

BC url: vigyázat, Oracle Pluggable Database
://technology.amis.nl/2016/01/10/create-jdbc-data-source-or-jdbc-url-database-connection-to-an-oracle-pluggable-database/

Statement használata

Statement:

- executeQuery: a paraméterében megadott SQL lekérdezést végrehajtja, majd az eredménytáblával (ResultSet) tér vissza.

  ```
  Statement stat1 = myCon.createStatement ();
  Stat1.executeQuery("SELECT * FROM test");
  ```

- executeUpdate: a paraméterében megadott SQL utasítást végrehajtja, majd a módosított sorok számával tér vissza.
- execute: a paraméterében megadott SQL utasítást hajtja végre. Az előző két metódus általánosításának tekinthető. Visszatérési értéke true, ha az utasítás által visszaadott eredmény ResultSet típusú, ekkor az a Statement.getResultSet metódussal kérdezhető le.

PreparedStatement:

Paraméterezhető SQL lekérdezés. Előnyei:

- Cache-elődik
- Lekérdezési terv csak egyszer készül el
- SQL Injection ellen véd

```
PreparedStatement stat2 =
myCon.prepareStatement (
"SELECT beer, price FROM Sells WHERE bar
= ?");
```

CallableStatement

Eljárás- és függvényhíváshoz

```
CallableStatement stat3 =
myCon.prepareCall ("{call JoeMenu(?, ?)}");
```

PreparedStatementek hívása

A JDBC megkülönbözteti a lekérdezéseket a módosításoktól:

- **executeQuery()**
- **executeUpdate()**

Query:

```
stat2.setString(1, "Joe’’s Bar");
ResultSet menu = stat2.executeQuery();
```

Update:

```
String sql="INSERT INTO Dept VALUES(?,?,?,?)";
PreparedStatment pstmt=con.prepareStatement(sql);
pstmt.clearParameters();
pstmt.setInt(1,depno);
pstmt.setString(2,depname);
pstmt.setInt(3, depmngr);
pstmt.setString(4,deploc);
int numRows = pstmt.executeUpdate();
```

ResultSet

- A ResultSet tartalmazza a lekérdezések eredményeit, soronként. Lényegében úgy viselkedik, mint egy kurzor/iterátor.
- **next()** metódus meghívásával tovább lép a következő sorra és igaz értékkel tér vissza
- Ha már nincs több elem a ResultSet-ben, akkor hamis értékkel tér vissza
- A ResultSet-ből a **getInt(i), getString(i)** stb. függvényekkel lehet kiszedni az

  adatokat, ahol az "i" az attribútum neve vagy sorszáma a relációban.
- **next()-**en kívül van még, **previous(), last(), first()** stb.
- Egy utasítás végrehajtása akkor zárul le, ha az összes visszaadott eredménytábla fel lett dolgozva (minden sora kiolvasásra került). Az utasítás végrehajtása manuálisan is lezárható a Statement.close metódussal.

ResultSet alkalmazása

```
ResultSet menu =
  stat1.executeQuery("SELECT beer, price
  FROM Sells WHERE bar = 'Joe''s Bar'
  ");

while (menu.next())
{
  theBeer = menu.getString(1);
  thePrice = menu.getFloat("price");
}
```

Tranzakció kezelés

Immediate az alapértelmezés

Autocommit van alapértelmezésben, amit át lehet

természetesen állítani és használni a commit(), rollback()

metódusokat igény szerint

CLOSE() fontos lezárni a kapcsolatot már csak a tranzakció kezelések miatt is

SQL Injection

1 Query is written in the backend script

```
$query = "SELECT user FROM tbl_user WHERE name = '$name' ";
```

2

Hacker inputs admin' OR 1=1 -- in the form field.
So $name=admin' OR 1=1 --

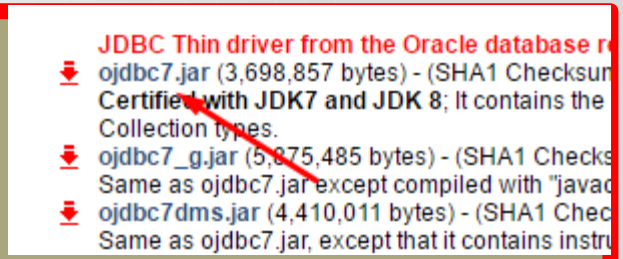
```
$query = "SELECT user FROM tbl_user WHERE name = 'admin' OR 1=1 --' ";
```

```
$query = "SELECT user FROM tbl_user WHERE name = 'admin' OR 1=1  --' ";
```

3 This becomes a logical expression which returns *True* for the query. Hacker can type any query even delete queries in the input string

TechRecite.com

- http://www.programmerinterview.com/index.php/database-sql/sql-injection-example/

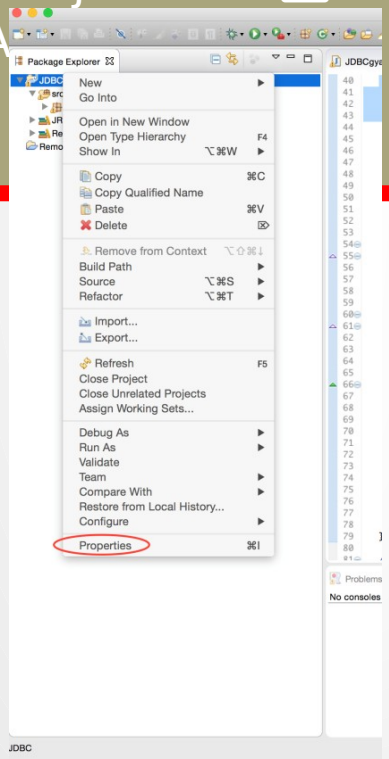


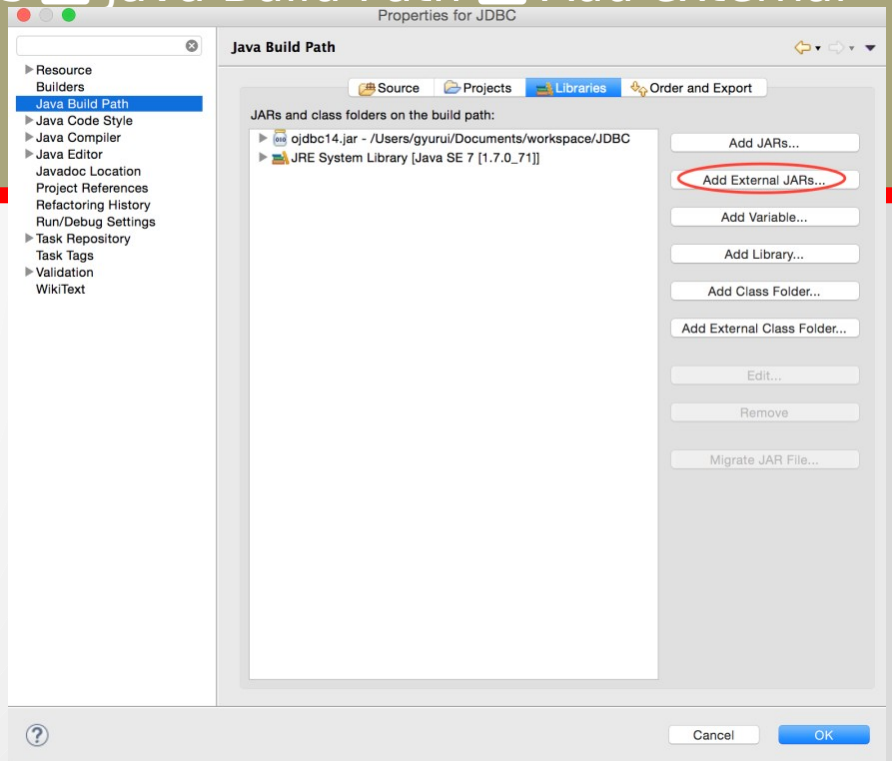
Feladat

Eclipse megnyitása, új Java projekt

Megfelelő JDBC driver letöltése (JDBC Thin for **12c Release 1**) http://www.oracle.com/technetwork/database/features/jdbc/default-2280470.html

Projekt név ✉ Properties ✉ Java Build Path ✉ Add external JA





Tesztelje a driver működését:
Létesítsen JDBC kapcsolatot az egyetemi Oracle adatbázissal.
Kapcsolati adatok mint az SQL developer-ben (az is egy Java alkalmazás, ami JDBC-n kapcsolódik az Oracle-hez)

JDBC url: vigyázat, Oracle Pluggable Database
https://technology.amis.nl/2016/01/10/create-jdbc-data-source-or-jdbc-url-database-connection-to-an-oracle-pluggable-database/

Hozza létre a következő táblát az adatbázisban. Az id értékeket ne kézzel kelljen megadni.
series tábla:

- series_id NUMBER
- series_name VARCHAR
- description VARCHAR
- next_episode DATE

- Készítsen egy új projektet javaban.
- Készítsen egy *SOROZAT* nyilvántartó alkalmazást a következő specifikáció alapján.
- Lehet gui-s vagy console-os az alkalmazás
- Hint a GUI építéshez:
  - Window Builder , Install new software
  - http://download.eclipse.org/windowbuilder/WB/release/R201406251200/4.4/

A program tudjon hozzáadni a *series* táblához nevet, leírást és következő rész dátumát.

Az alkalmazás tudjon törölni a *series* táblából. (id alapján)

A sorozat neveket egy listába legyen lehetőség lekérdezni.

Egy sorozat név kiválasztásakor jelenjen meg róla a leírás és a következő rész dátuma.

A sorozat dátuma és leírása legyen szerkeszthető.

A feladatokat *prepared statement* segítségével oldja meg.

Készítsen Logot a lekérdezésekhez (írja ki egy fájlba vagy jelenítse meg a console-on).