JAVA TECHNOLÓGIÁK 2009.
2. GYAKORLAT

Célok:

- a Java DB adatbázis-kezelő rendszer használatának ismertetése,
- adatbázisok használata Java alkalmazásokban - kétrétegű architektúra,
- egyszerű kliens-szerver architektúra használata hálózati alkalmazásokhoz.

I. A Java DB adatbázis-kezelő rendszer

`Java DB:`

- a SUN cég *Apache Derby* disztribúciója
- teljes egészében Java nyelven íródott

1. Adatbázis létrehozása

[Netbeans 6.0]

`Tools menüpont ----> Java DB Database ----> Create Database`

[Netbeans 6.1, 6.5, 6.7]

`Runtime window (felület bal felső sarkában levő ablak: Projects, Files, Services ... )`
`---> Services --->Databases ---> JavaDB jobb egérkattintás és Create Database`

`Database Name:` **studentdb**
`User Name:` **public**
`Password:` **public**
`Database location:` **hagyjuk meg az alapértelmezettet**

2. Adattábla létrehozása

a. `Runtime window ---> Services ---> Databases`

b. Adatbazis-nevre (`jdbc:derby://localhost:1527/studentdb`) jobb klikk `és Connect`

c. Adatbazis-nevre (`jdbc:derby://localhost:1527/studentdb`) jobb klikk `és Execute Command`

Ennek hatására megjelenik egy SQL parancsablak.

d.

```
CREATE TABLE student (
      id int NOT NULL GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
      name varchar(100) NOT NULL,
      address varchar(100) NOT NULL
)
```

3. Adattábla feltöltése

```
INSERT INTO student (name, address) values ('John Smith','Windows street 3');
```

JAVA TECHNOLÓGIÁK 2009.
2. GYAKORLAT

II. Java alkalmazás készítése – adatbázisok elérése

Készítsen egy konzolos Java alkalmazást, amely egy adatbázis egy adott adattáblájával a következő műveleteket képes végezni:

- a tábla tartalmának lekérdezése és megjelenítése
- új rekord hozzáadása az adattáblához
- a tábla adott kulcsú elemének a lekérdezése

Használja fel a következő adatbázis-kapcsolat osztályt:

```
import java.sql.*;
import javax.sql.*;
import java.util.*;


public class StudentDAO{

   public static final String DRIVER  = "org.apache.derby.jdbc.ClientDriver";
   public static final String URL     = "jdbc:derby://localhost:1527/studentdb";
   public static final String USERNAME ="public";
   public static final String PASSWORD ="public";


    public static final String GETALLDATA  ="select * from student";
    public static final String INSERTDATA  =
                            "insert into student (name, address) values (?,?)";


   private String driver;
   private String url;
   private String userName;
   private String password;

   private Connection con;

   public StudentDAO(){
     driver =DRIVER;
     url = URL;
     userName = USERNAME;
     password = PASSWORD;
   }

   public StudentDAO( String driver, String url,
                      String userName, String password){
     this.driver = driver;
     this.url = url;
     this.userName = userName;
     this.password = password;
   }
```

```
 public void connect(){
   try{
     //Driver betoltese
     Class.forName( driver );
     //Kapcsolat megteremtese
     con = DriverManager.getConnection(url,userName, password);
   }
   catch ( SQLException e1 ){
     System.out.println(
      "Hiba a kapcsolat megteremtesenel: "+url);
   }
   catch( ClassNotFoundException e2 ){
     System.out.println(
       "Hiba a driver betoltesenel: "+driver);
   }
 }


public void disConnect(){
   try{
     con.close();
   }
   catch( SQLException e){
     System.out.println("Kapcsolat lezarasi hiba");
   }
}

public List<String> getStudents(){
    Statement stmt = null;
    ResultSet rs = null;
    connect();
    if( con == null ) return null;
    List<String> list = new ArrayList<String>();
    try{
      stmt = con.createStatement();
      rs = stmt.executeQuery(GETALLDATA);
      while( rs.next()){
        int id = rs.getInt("id");
        String name = rs.getString("name");
        String address = rs.getString("address");

        StringBuffer str = new StringBuffer();
        str.append( ""+id ); str.append("; ");
        str.append( name ); str.append("; ");
        str.append( address ); str.append("; ");
        list.add(str.toString());
      }
    }
    catch (SQLException se) {
      se.printStackTrace();
    }
    finally {
      if(rs != null) {
          try { stmt.close(); }
          catch (Exception e) { e.printStackTrace(); }
      }
      if(stmt != null) {
          try { stmt.close(); }
          catch (Exception e) { e.printStackTrace(); }
```

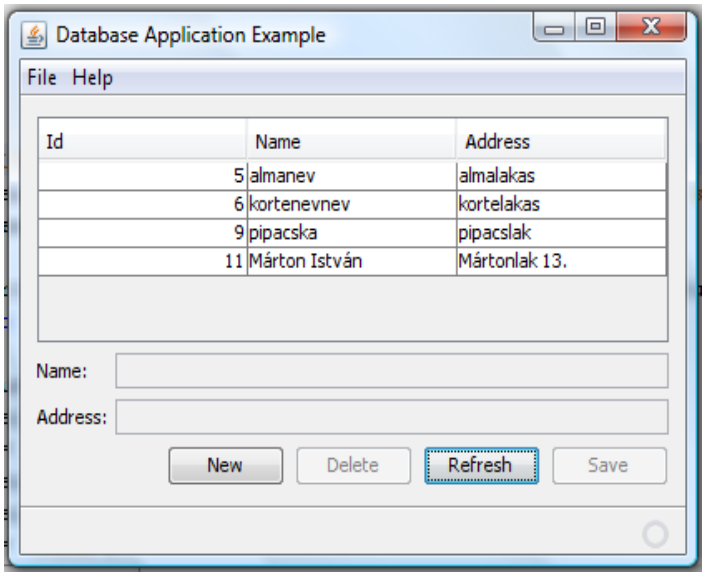
```
        }
        disConnect();
      }
      return list;
      }

public boolean insertStudent( String name, String address ) {
    PreparedStatement stmt = null;
    int res = 0;
    connect();
    if( con == null ) return false;
    try {
        stmt = con.prepareStatement(INSERTDATA);
        stmt.setString(1, name);
        stmt.setString(2, address);
        res = stmt.executeUpdate();

    }
    catch (SQLException e) {
        System.out.println("SQL insert error - " + e);
        return false;
    }
    finally {
        if(stmt != null) {
            try { stmt.close(); }
            catch (Exception e) { e.printStackTrace(); }
        }
        disConnect();
    }
    return res == 1;
}
}
```

III. Grafikus felület készítése

- Készítsen egy új *Java Desktop Application (Database Application)* típusú projektet. Rendelje hozzá a `studentdb` adatbázis `student` adattábláját. *Nem kell kódot írni, a Netbeans előállítja.*
- Tanulmányozza a kódot, és próbálja ki az alkalmazást.



JAVA TECHNOLÓGIÁK 2009.
2. GYAKORLAT

IV. Készítsen két Java alkalmazást: egy kliens-, illetve egy szerveralkalmazást. A szerver legyen párhuzamos architektúrájú és a kliensek számára biztosítsa a fenti három adatbázis-műveletet.

http://www.ms.sapientia.ro/~manyi/teaching/oop/oop_romanian/curs9/curs9.html

Szerver rész:

`Main.java` – `main` metódus

`Server.java`:

-felelősségek:

- szolgáltatás indítása egy adott porton: start()
- kapcsolat felépítése az ügyféllel: service()
- szolgáltatás leállítása: stop()

`StudentDAO.java`

`ClientHandle.java`

-felelősségek: ügyfél kiszolgálását végző szál osztály

-saját `TableAccess` példányt hoz létre minden klienshez (ezt később másképpen végezzük!!!)

//Szolgáltatás indítása

```
ServerSocket ss;
try{
	ss = new ServerSocket(port);
}
catch(Exception e){
	System.out.println("Server error: could'nt create Socket:\n"+e);
	System.exit(1);
}
System.out.println("Service running on "+ss.getInetAddress()+
		" at port "+ss.getLocalPort());
```

//Ügyfelek fogadása, ügyfél kiszolgáló szál indítása

```
Socket s= null;

while( isRunning ){
	try{
		s = ss.accept();
	}
	catch(Exception e){}
	System.out.println("Connection has been accepted from: "
		+s.getInetAddress().getHostAddress());
	ClientHandle ch = new ClientHandle( s );
	ch.start();
}
```