שילוב בין jsp וטכנולוגיית JDBC

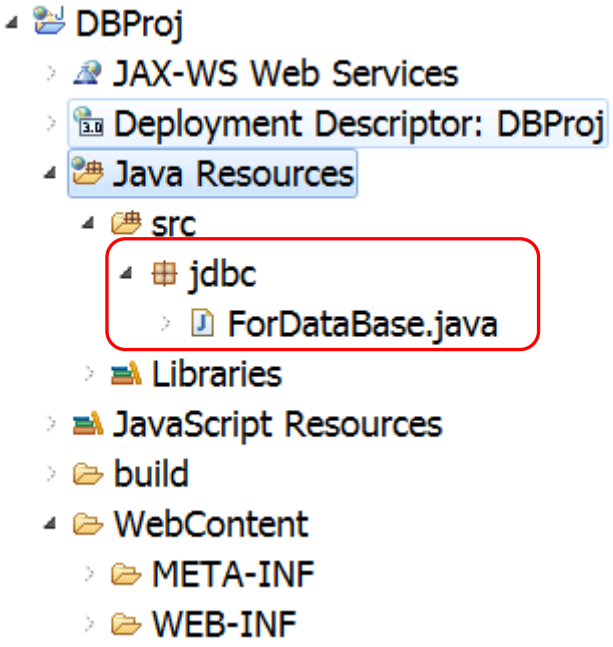
מבוסס על מחלקות שנכתבו ע"י: **אלה מריאנובסקי**

התקנת mysql-connector

1. הורד את הקובץ mysql-connector-java-5.1.14-bin.jar מהקישור הבא:
   http://www.kadman.net/MySQL.html (חומרי למידה >> תכנות בסביבת האינטרנט (תב"א) >> MySQL)
2. שמור את הקובץ בתיקייה lib שתחת Tomcat 7:
   C:\Program Files\Apache Software Foundation\Tomcat 7.0\lib

פעולה זו הינה חד פעמית.

שילוב המחלקה ForDataBase בפרויקט

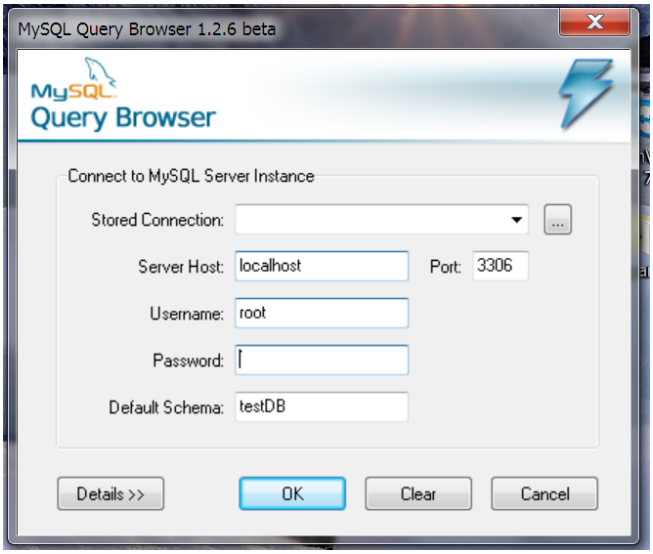


1. צור package בשם jdbc (שים לב! אות קטנה)
   תחת התיקייה: Java Resources\src
2. הורד את הקובץ ForDataBase
   מהקישור הבא:
   חומרי למידה >> תב"א >> טפסים וניהול מידע

   הקובץ הינו קובץ rar המכיל מחלקה בשם זה.
   פתח את קובץ ה- rar וחלץ מתוכו את המחלקה ForDataBase.java לתוך ה- package שיצרת.
   (ראה איור)

קבצי jsp יישמרו תחת התיקייה WebContent

יצירת מסד הנתונים במחשב הנוכחי



הפרויקט שלנו עובד עם מסד נתונים MySQL.
כשיצרנו את מסד הנתונים, הוא נשמר בכתובת localhost או בשם: 127.0.0.1 ב- port 3306. זהו מקום השרת המקומי שמפעיל Tomcat במחשב שלנו.

המסד שיצרנו יישמר במחשב האישי (כיבוי המחשב אינו מוחק אותו). הבעיה מתעוררת במחשב בית הספר שבדרך כלל אינו שומר את הנתונים, ולכן נריץ את הסקריפט שיצר את מסד הנתונים (קובץ טקסט שניתן לפתיחה גם באמצעות פנקס הרשימות):
www.kadman.net/taba/StudentsTbl.sql

שם הטבלה: studentsTbl
מיקום הטבלה: testdb

גישה למסד הנתונים בטכנולוגיית jdbc

לכל מסד נתונים יש שפה משלו ודרך משלו לניהול הנתונים. אובייקט jdbc מאפשר גישה למסד הנתונים, והדרייברים (Drivers) מתרגימים את הפנייה לשפה המובנת למסד הנתונים.

הדרייבר המתאים ל- MySql הוא MySql Connector אותו העתקנו ל- Tomcat

רכיבי JDBC - Java DataBase Connectivity:

- אובייקט **Connection** - מאפשר קישור למסד הנתונים. מכיל את כל התכונות של ההתחברות לנתונים.
- אובייקט **Statement** - אובייקט המכיל משפט sql באמצעותו מפעילים את השאילתות.
- אובייקט **Recordset** - מאפשר לשלוף מידע ממסד הנתונים. הנתונים מוחזרים בצורת טבלה.

אובייקטים אלו מוגדרים ב- package בשם java.sql

הגדרת חיבור למסד הנתונים – Connection:

טעינת דרייבר JDBC של שרת MySQL מבצעים בעזרת השורה הבאה:

```
Class.forName("com.mysql.jdbc.Driver").newInstance();
```

לאחר הטעינה של הדרייבר, ניתן ליצור חיבור למאגר עצמו. החיבור הוא אובייקט של Java , הנמצא ב - java.sql.Connection. את האובייקט באמצעות מנהל הדרייברים.(DriverManager) :

```
con = DriverManager.getConnection
          ("jdbc:mysql://127.0.0.1:3306/testdb","root","root");
```

פרמטר 1 : החיבור הוא לשרת מקומי (Tomcat) העובד בפורט : 127.0.0.1:3306 או localhost:3306
שם מסד הנתונים הוא testdb
פרמטר 2 : שם המשתמש : root
פרמטר 3 : סיסמה : root (אם לא קבענו סיסמה למסד הנתונים, נרשום מחרוזת ריקה : "")

con הינו תכונה מסוג אובייקט Connection במחלקה ForDataBase.
אוביקט זה ניגש למסד נתונים בשם testbd שנמצא בשרת המקומי, עם שם משתמש root וסיסמה root.

יצירת משפט sql - Statement

ניצור משפטי SQL באמצעות האובייקט Statement ודרכם נשלח את השאילתות (Query) :

```
Statement st = con.createStatement();
int num = st.executeUpdate(sql);
```

המשפט :

משמש להפעלת שאילתות שאינן מחזירות נתונים (CREATE, INSERT, UPDATE) ומחזירה את מספר השורה בטבלה שבה נעשה העדכון.

המשפט:
```
ResultSet rs = st.executeQuery(sql);
```
משמש להפעלת שאילתה מסוג SELECT המחזירה ResultSet המכיל את תוצאת השאילתה.
sql הינו מחרוזת, המכילה משפט בשפת sql, המתקבלת כפרמטר למשפט הביצוע

קבלת התוצאות - ResultSet

האובייקט rs מסוג ResultSet מכיל את התוצאה של השאילתה SELECT.
התוצאה היא מעין מערך דו-ממדי (מטריצה) וירטואלי שמספר שורותיו כמספר הרשומות העונות לתנאי של משפט ה- SELECT ומספר העמודות כמספר השדות שביקש משפט ה- SELECT להציג.

האובייקט מכיל סמן וירטואלי המפנה לשורה במטריצה, ועליו פועלות הפעולות הבאות:

beforeFirst()	פעולה המזיזה את הסמן לפני השורה הראשונה.
last()	פעולה המזיזה את הסמן אל השורה האחרונה בטבלה
next ()	פעולה העוברת לשורה הבאה בטבלה בתנאי שקיימת שורה כזו.
getRow()	פעולה המחזירה את מספרה של השורה עליה ניצב הסמן

הפעולות מופעלות על ידי האובייקט, למשל, פעולה הגורמת לתזוזת הסמן לשורה הבאה בטבלה, עבור האובייקט rs תהיה: rs.next();

שליפת המידע מהטבלה נעשית על ידי הפעולה getXXX(param) כאשר XXX מייצג את טיפוס הנתונים:
getInt() עבור נתון מספרי שלם, getDouble() עבור נתון מספרי ממשי, getString() עבור נתון מחרוזתי.
לכל טיפוס נתונים בסיסי יש פעולת get מתאימה.

הפונקציה getXXX() חייבת לקבל את שם או מספר העמודה ממנה היא מקבלת את הנתון.

למשל: קבלת שם מתוך עמודה בשם fNAme:
```
String name = rs.getString ("fName");
```
קבלת מספר מתוך עמודה מספר 3:
```
int k = rs.getInt (3);
```
(מספר העמודה יכול להיות קבוע מספרי או משתנה)

לפני שמציגים את הנתונים שבטבלה זו **<u>חייבים לסגור את האובייקט</u>**, ולכן נעתיק את תוכן הטבלה הוירטואלית למטריצה של מחרוזות בשם result

```
int m = res.getMetaData().getColumnCount();        // m מקבל את מספר העמודות בטבלה

res.last();                                        // הזזת הסמן לשורה האחרונה
int n = res.getRow();                              // n מקבל את מספר השורות בטבלה

result = new String[n][m];                         //       יצירת מערך המחרוזות
res.beforeFirst();                                 //       הזזת הסמן לתחילת הטבלה (לפני שורה ראשונה)

//---  לולאה להעתקת תוכן אובייקט ResultSet למטריצת המחרוזות ---
int i = 0;
while (res.next())
{
        for(int j=0 ; j < m ; j++)
           result[i][j] = res.getString(j+1);
        i++;
}
```

סגירת האובייקטים

```
rs.close();
st.close();
```

try & catch

קטע ההתקשרות עם מסד הנתונים ארוז בתוך מבנה מיוחד של try & catcj:

```
try
{
        קטע הקוד שמבקשים לבצע
}
catch (SQLException e)
{
        try -מה לעשות אם התגלתה שגיאה בחלק ה
}
```

אנו רגילים מתכניות java שאם מתרחשת שגיאה (exception) מסתיים ביצוע התכנית ומוצגת הודעת שגיאה.
ההוראה try & catch מונעת סיום התכנית, על הצעת קטע קוד חלופי.

דוגמאות לקוד חלופי:

```
e.printStackTrace();
```

סיום התכנית והצגת שרשרת הפעולות שזומנו וגרמו לשגיאה. (שרשרת הפעולות נשמרת בתוך מחסנית זמן ריצה)

```
System.out.println("Error in connecting");
```

הדפסת הודעת שגיאה, מבלי להפסיק את ריצת התכנית (במקום תוכן הטבלה תוצג הודעת השגיאה).

מבנה המחלקה ForDataBase.java:

```
package jdbc;
import java.sql.*;

public class ForDataBase
{
    private Connection con;

    /**
     * constructor - create object con ( Connection type)
     * invoking from the line: <jsp:useBean id="db" class = "jdbc.ForDataBase"/>
     */
    public ForDataBase()
    {
        :
    }

    /**
     *  This function performs one direction SQL commands and returns
     *  the number of updated rows
     */
    public int insertUpdateDelete(String sql)
    {
        :
        return num;
    }

      /**
       * This function performs Select SQL commands
       * and returns the result array of the data
       */
      public String [][] select(String sql)
      {
          :
          return result;
      }
}
```