Προηγμένα Θέματα Βάσεων Δεδομένων

Διδάσκων: Άγγελος Μιχάλας

Περιεχόμενα

- Ανάκτηση-Εισαγωγή δεδομένων από/προς τη Βάση Δεδομένων χρησιμοποιώντας τη JAVA

Εγκατάσταση της MySQL

- **Εγκαταστήστε τη MySQL ακολουθώντας το σύνδεσμο:**

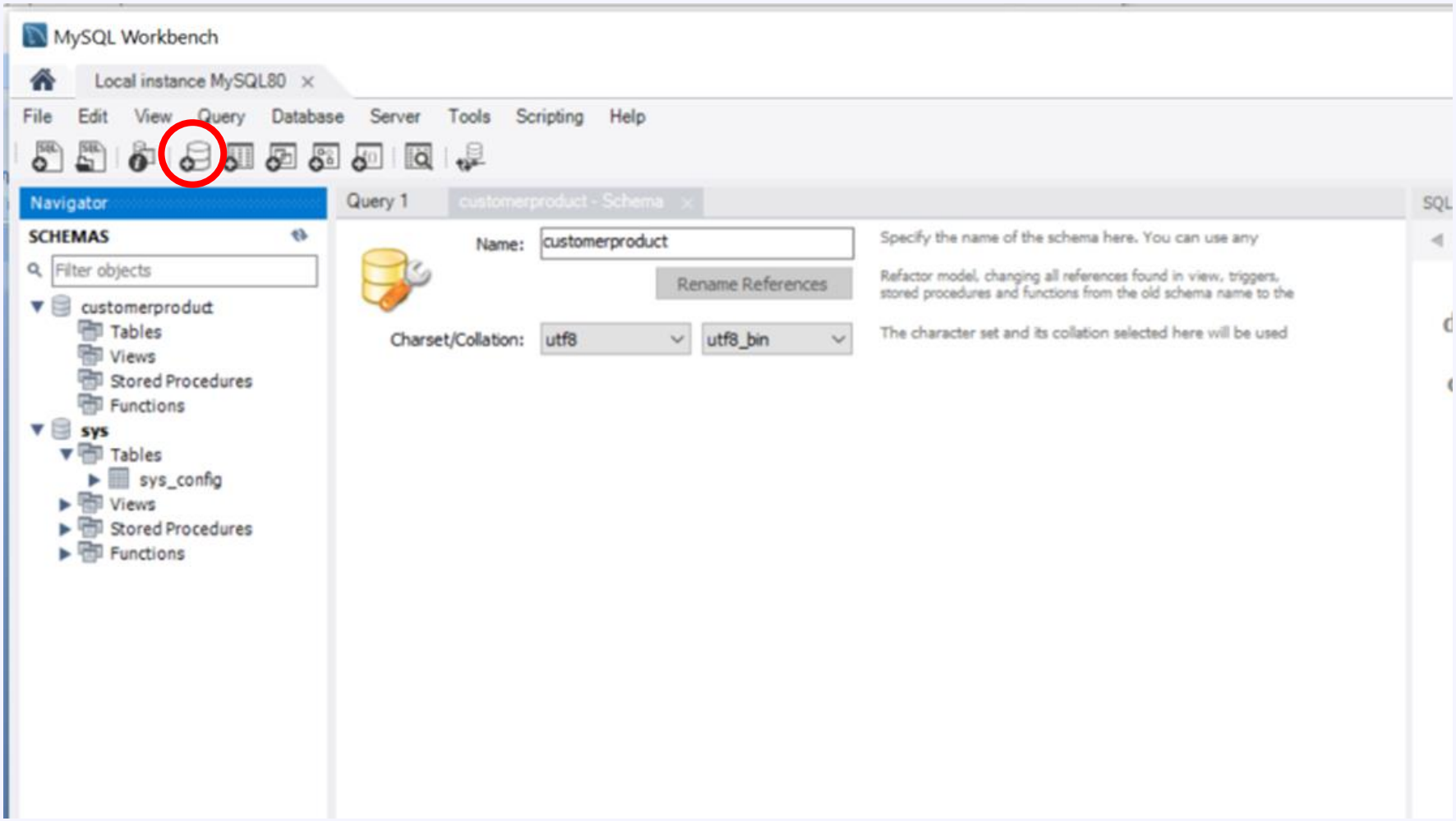
  https://dev.mysql.com/downloads/installer/

- **Επιλέγετε να εγκατασταθούν τα πακέτα: MySQL Server, MySQL Workbench, MySQL Shell, Connector/J.**

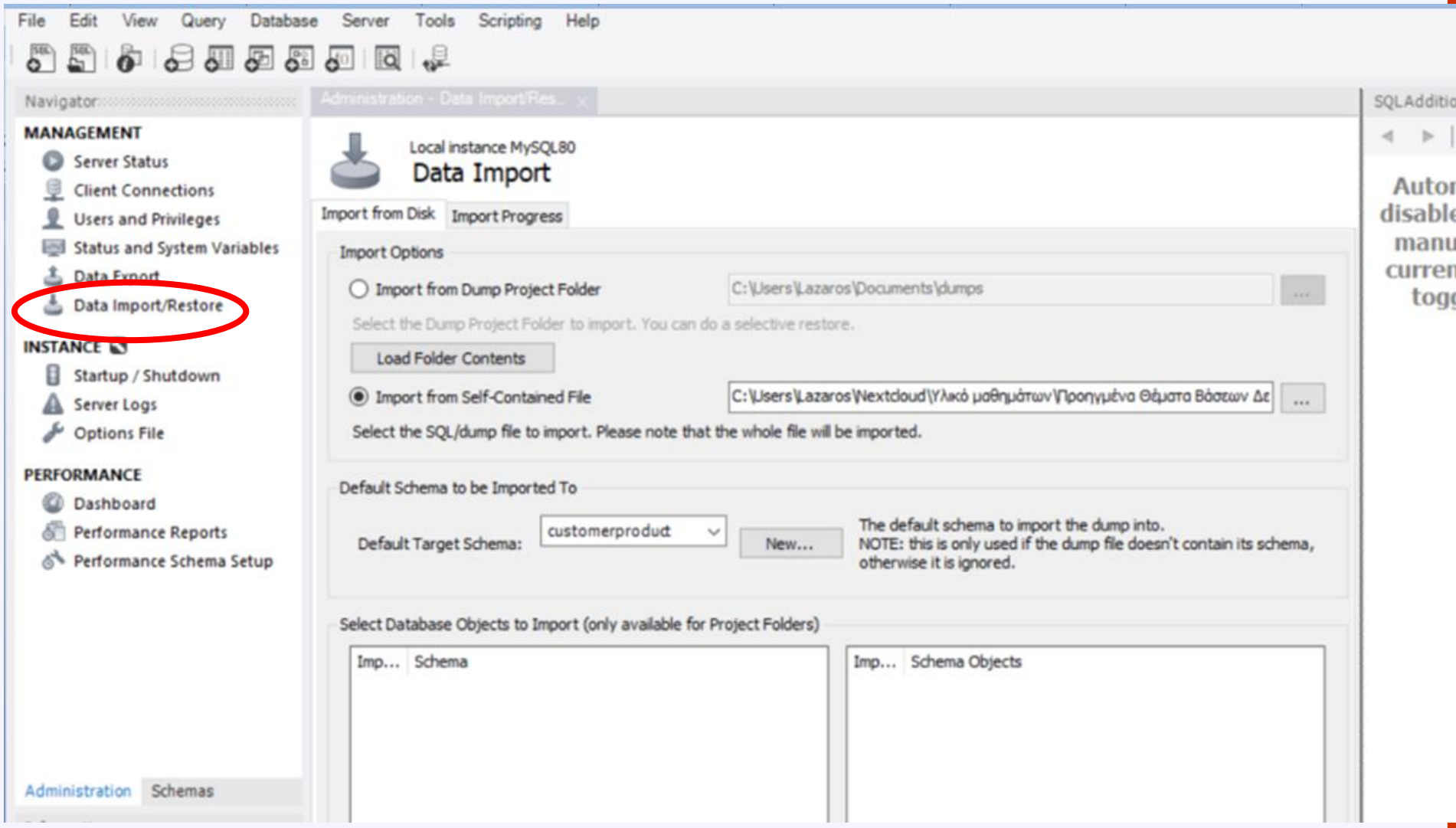
- **Οδηγίες για την εγκατάσταση υπάρχουν στο βίντεο:**

https://www.youtube.com/watch?v=OM4aZJW_Ojs&ab_channel=AmitThinks

Δημιουργία σχήματος customerproduct

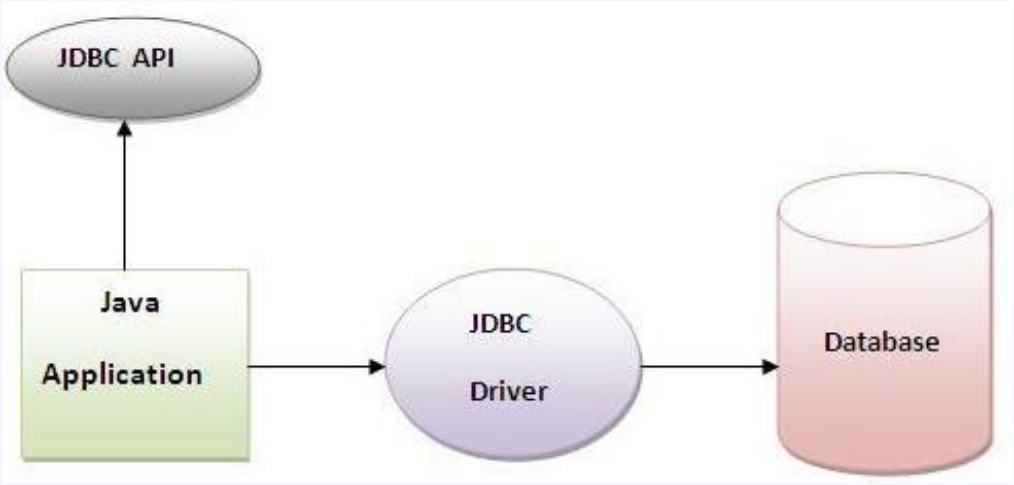


Εισαγωγή βάσης customerproduct στο σχήμα



Java Database Connectivity (JDBC)

- Java **JDBC** είναι ένα API για σύνδεση και εκτέλεση επερωτήσεων σε μία Βάση Δεδομένων (ΒΔ).
  - Δημιουργία σύνδεσης με ΒΔ.
  - Δημιουργία SQL statements.
  - Εκτέλεση SQL επερωτήσεων στη ΒΔ.
  - Προβολή και Επεξεργασία των αποτελεσμάτων (εγγραφών).



Java Database Connectivity (JDBC)

- Ο οδηγός (Driver) JDBC είναι λογισμικό που επιτρέπει σε μία εφαρμογή να αλληλεπιδρά με τη ΒΔ. Υπάρχουν 4 τύποι JDBC οδηγών:
  - JDBC-ODBC bridge driver
  - Native-API driver (partially java driver)
  - Network Protocol driver (fully java driver)
  - **Thin driver (fully java driver)**: Είναι υψηλής απόδοσης οδηγός διαθέσιμος για τη ΒΔ και παρέχεται συνήθως από τον κατασκευαστή της ΒΔ
    - Μετατρέπει κλήσεις JDBC απευθείας στο συγκεκριμένο πρωτόκολλο του κατασκευαστή της ΒΔ
    - MySQL's Connector/J driver είναι ένας thin driver.
    - Μπορεί να ληφθεί (κατέβει) δυναμικά

Εγκατάσταση Οδηγού ΒΔ

Μέθοδος 1η (Προσθήκη στο φάκελο lib μέσα στη JAVA)

1. Αντιγράψτε το αρχείο mysql-connector-java-version-bin.jar
   - Π.χ. mysql-connector-java-8.0.23.jar
   - Το .jar υπάρχει στο φάκελο της MySQL (C:\Program Files (x86)\MySQL\Connector J 8.0)
2. Επικολλήστε το αρχείο jar μέσα στο φάκελο jre/lib/ext όπου έχετε εγκαταστήσει το jdk

Εγκατάσταση Οδηγού ΒΔ

Μέθοδος 2η (cmd)

1. Αντιγράψτε το αρχείο mysql-connector-java-8.0.23.jar που υπάρχει στο φάκελο της MySQL (C:\Program Files (x86)\MySQL\Connector J 8.0)σε κάποιο δικό σας φάκελο (πχ στον φάκελο c:\lib)
2. Κάντε compile με

javac myprog.java

3. Εκτελέστε το με

java -cp .;c:\lib\mysql-connector-java-8.0.23.jar myprog

Εγκατάσταση Οδηγού ΒΔ

Μέθοδος 3η (Classpath και cmd)

1. Αντιγράψτε το αρχείο mysql-connector-java-8.0.23.jar σε κάποιο δικό σας φάκελο (πχ στον φάκελο C:\lib)
2. Πηγαίνετε στις μεταβλητές περιβάλλοντος και κάνετε click σε ένα νέο tab.
3. Στο όνομα μεταβλητής γράψτε CLASSPATH και στην τιμή της μεταβλητής επικολλήστε το μονοπάτι για το αρχείο mysql-connector.jar (πχ C:\lib\mysql-connector-java-8.0.23.jar)
4. Κάντε compile & εκτέλεση με javac & java κανονικά.

Εγκατάσταση Οδηγού ΒΔ

Μέθοδος 4η (Eclipse)

1. Δημιουργήστε ένα JAVA Project
2. Δημιουργήστε ένα φάκελο lib
3. Αντιγράψτε το αρχείο mysql-connector-java-8.0.30.jar και επικολλήστε το στο φάκελο lib
4. Δεξί click στο όνομα του Project και επιλέξτε τις Ιδιότητες (Properties)
5. Κάντε click αριστερά στην επιλογή “Java Build Path”
6. Επιλέξτε το tab “Libraries” κατόπιν επιλέξτε ‘Classpath’
7. Κάντε click στο κουμπί “Add JARs…”

Εγκατάσταση Οδηγού ΒΔ

Μέθοδος 4η (Eclipse)

8. Στο popup παράθυρο JAR Selection
   I. επιλέγετε τον φάκελο του project σας (πχ adbLabs)
   II. Επιλέξτε τον υποφάκελο lib και στην συνέχεια το αρχείο mysql-connector-java- 8.0.30.jar
   III. Επιλέξτε Apply and Close

Προσοχή στο package του project σας θα πρέπει να προσθέσετε στο αρχείο του module-info.java τον κώδικα:

```
module adbLabs {
        requires java.sql;
}
```

Βήματα Σύνδεσης με ΒΔ

1. Καταχωρήστε τη κλάση του driver
2. Δημιουργία σύνδεσης
3. Δημιουργία statement
4. Εκτέλεση επερωτήσεων
5. Τερματισμός της σύνδεσης

Καταχώρηση Driver Class (1/2)

- Η μέθοδος **forName()** της Class χρησιμοποιείται για να καταχωρήσουμε την driver class.
  - **Η κλήση της μεθόδου φορτώνει δυναμικά την driver class**
- Σύνταξη:

  public static void forName(String className) throws ClassNotFoundException
- Παράδειγμα:

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

Καταχώρηση Driver Class (2/2)

- Χρήση της static DriverManager.registerDriver() μεθόδου
- Παράδειγμα:

```
Driver myDriver = new com.mysql.jdbc.Driver();
DriverManager.registerDriver(myDriver);
```

DriverManager – Δημιουργία Σύνδεσης

- Η μέθοδος **getConnection()** της DriverManager class χρησιμοποιείται για δημιουργία σύνδεσης με τη ΒΔ.
- **Σύνταξη**:

  public static Connection getConnection(String url) throws SQLException

  public static Connection getConnection(String url, String name, String password) throws SQLException
- Παράδειγμα:

```
Connection con = DriverManager.getConnection(
dbUrl, dbUser, dbPwd);
```

DriverManager – Δημιουργία Σύνδεσης

- Δημοφιλή ονόματα JDBC driver και URL ΒΔ.

RDBMS	JDBC driver name	URL format
MySQL	com.mysql.jdbc.Driver	**jdbc:mysql://**hostname/ databaseName
ORACLE	oracle.jdbc.driver.OracleDriver	**jdbc:oracle:thin:@**hostname:port Number:databaseName
DB2	COM.ibm.db2.jdbc.net.DB2Driver	**jdbc:db2:**hostname:port Number/databaseName
Sybase	com.sybase.jdbc.SybDriver	**jdbc:sybase:Tds:**hostname: port Number/databaseName

Connection

- Μία σύνδεση (Connection) είναι ένα **session** ανάμεσα σε μία **java εφαρμογή** και τη **ΒΔ**.
- Οι πιο σημαντικές μέθοδοι της διασύνδεσης Connection είναι:
  - public Statement **createStatement()**: δημιουργία αντικειμένου τύπου Statement για την εκτέλεση SQL queries.
  - public Statement **createStatement (int resultSetType,int resultSetConcurrency)**
  - public void **setAutoCommit(boolean status)**: χρησιμοποιείται για την ανάθεση τιμής στην κατάσταση commit. Προεπιλογή είναι true.

Connection

- Μία σύνδεση (Connection) είναι ένα **session** ανάμεσα σε μία **java εφαρμογή** και τη **ΒΔ**.
- Οι πιο σημαντικές μέθοδοι της διασύνδεσης Connection είναι:
  - public void **commit()**: αποθηκεύει τις αλλαγές που έγιναν από το προηγούμενο μόνιμο commit/rollback.
  - public void **rollback()**: αφαιρεί όλες τις αλλαγές που έγιναν από το προηγούμενο μόνιμο commit/rollback και μετά.
  - public void **close()**: κλείνει τη σύνδεση και απελευθερώνει τους πόρους αμέσως

Driver & Connection – Παράδειγμα

```
try{
    Class.forName("com.mysql.cj.jdbc.Driver");
    Connection con=DriverManager.getConnection(
    "jdbc:mysql://localhost:3306/world","root","root");
    Statement stmt=con.createStatement();
    ResultSet rs=stmt.executeQuery("select * from city");
    while(rs.next())
          System.out.println(rs.getInt(1)+"  "+rs.getString(2));
    con.close();
} catch(Exception e){ System.out.println(e); }
```

Statement

- Η διαδύνδεση (interface) Statement παρέχει μεθόδους για την εκτέλεση επερωτήσεων στη ΒΔ.
  - Η διασύνδεση Statement χρησιμοποιείται για γενικού σκοπού πρόσβαση στη ΒΔ
  - Η διασύνδεση Statement δεν δέχεται παραμέτρους.
  - Χρήσιμο όταν χρησιμοποιείται στατικά SQL statements κατά τη διάρκεια εκτέλεσης.

Statement

- Οι πιο σημαντικές μέθοδοι της διασύνδεσης Statement είναι:
  - public ResultSet **executeQuery(String sql)**: χρησιμοποιείται για την εκτέλεση επερωτήσεων SELECT. Επιστρέφει ένα αντικείμενο τύπου ResultSet.
  - public int **executeUpdate(String sql)**: χρησιμοποιείται για την εκτέλεση συγκεκριμένων επερωτήσεων, όπως create, drop, insert, update, delete etc.

Statement – Παράδειγμα

- Παράδειγμα εκτέλεσης query

```
Statement stmt = dbConn.createStatement();
String query = "SELECT * FROM users";
ResultSet rs = stmt.executeQuery(query);
```

- Παράδειγμα εκτέλεσης διαγραφής (delete)

```
Statement stmt = dbConn.createStatement();
String query = "DELETE FROM users WHERE user_id=1";
int result = stmt.executeUpdate(query);
```

Statement - Παράδειγμα

```
class StatementExample{
    public static void main(String args[])throws Exception{
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con=DriverManager.getConnection(
    "jdbc:mysql://localhost:3306/world","root","roor");
        Statement stmt=con.createStatement();
        stmt.executeUpdate("insert into city values(4080, 'Kozani', 'GRC',
    'West Macedonia', 60000)");
        int result=stmt.executeUpdate("update city set Population=46956
    where ID=4080");
        //int result=stmt.executeUpdate("delete from city where ID=4080");
        System.out.println(result+" records affected");
        con.close();
    }
}
```

PreparedStatement

- Η διαδύνδεση PreparedStatement χρησιμοποιείται για την εκτέλεση **precompiled SQL statement**
- Η διαδύνδεση PreparedStatement **δέχεται παραμέτρους εισόδου κατά τη διάρκεια εκτέλεσης**.
- Όλες οι παράμετροι σε JDBC αναπαρίστανται από το σύμβολο **?**
  - Παροχή τιμών για κάθε παράμετρο πριν την εκτέλεση του SQL statement
- Χρήση της PreparedStatement όταν σχεδιάζετε να χρησιμοποιείσετε τα SQL statements πολλές φορές.

PreparedStatement

- Οι πιο σημαντικές μέθοδοι της διασύνδεσης PreparedStatement είναι:
  - public void **setInt(int paramIndex, int value)**: ανάθεση ακέραιας τιμής στην καθορισμένη παράμετρο.
  - public void **setString(int paramIndex, String value)**: ανάθεση αλφαριθμητικού στην καθορισμένη παράμετρο.
  - public void **setDouble(int paramIndex, double value)**: ανάθεση δεκαδικής τιμής στην καθορισμένη παράμετρο.
  - public ResultSet **executeQuery()**: χρησιμοποιείται για την εκτέλεσηSELECT query. Επιστρέφει ένα αντικείμενο ResultSet.
  - public int **executeUpdate()**: Χρησιμοποιείται για εκτέλεση εντολών όπως create, drop, insert, update, delete κ.α.

PreparedStatement – Παράδειγμα

- Παράδειγμα εισαγωγής τιμών

```
PreparedStatement stmt = con.prepareStatement(
"insert into users values(?, ?, ?)");
stmt.setInt(1,101);
stmt.setString(2,"Nikos");
stmt.setString(3,"Dimokas");
int result = stmt.executeUpdate();
```

PreparedStatement – Παράδειγμα

```
try{
    Class.forName("com.mysql.jdbc.Driver");
    Connection con=DriverManager.getConnection(
    "jdbc:mysql://localhost:3306/world","dimokas","dimok2016#");
    PreparedStatement stmt=con.prepareStatement("insert into city
    values(?,?,?,?,?)");
    stmt.executeUpdate("insert into city values(4080, 'Kozani', 'GRC', 'West
    Macedonia', 60000)");
    stmt.setInt(1,4081);   stmt.setString(2,"Volos");
    stmt.setString(3,"GRC");  stmt.setString(4,"Thessalia");
    stmt.setString(5,"95452");
    int i=stmt.executeUpdate();
    System.out.println(i+" records inserted");
    con.close();
}catch(Exception e){ System.out.println(e);}
```

PreparedStatement vs Statement

- Τα PreparedStatement είναι πιο γρήγορα επειδή χρησιμοποιούνται για την εκτέλεση προκατασκευασμένων SQL statement
- Τα PreparedStatement είναι **κατάλληλα για την εκτέλεση DML** εντολών όπως SELECT, INSERT, UPDATE και DELETE
- Τα PreparedStatement είναι κατάλληλα για την **ανάθεση java.sql.Array** με τη χρήση της **setArray μεθόδου** (ο οδηγός μετατρέπει java.sql.Array σε SQL Array).

PreparedStatement vs Statement

- Τα PreparedStatement αποτρέπει την έγχυση SQL (**SQL injection**), επειδή το κείμενο για όλες τις τιμές παραμέτρου escaped in java JDBC.
- PreparedStatement είναι κατάλληλο για
  - Αποθήκευση/ανάκτηση εικόνων και
  - Αποθήκευση/ανάκτηση στη ΒΔ (για παράδειγμα με τη χρήση BLOB, CLOB τύπων δεδομένων (LOB = Large Object for bitstreams, characters etc.)

Τύποι Δεδομένων

SQL	JDBC/Java	setXXX
VARCHAR	java.lang.String	setString
CHAR	java.lang.String	setString
LONGVARCHAR	java.lang.String	setString
BIT	boolean	setBoolean
NUMERIC	java.math.BigDecimal	setBigDecimal
TINYINT	byte	setByte
SMALLINT	short	setShort
INTEGER	int	setInt
BIGINT	long	setLong
REAL	float	setFloat
FLOAT	float	setFloat

Τύποι Δεδομένων

SQL	JDBC/Java	setXXX
DOUBLE	double	setDouble
VARBINARY	byte[ ]	setBytes
BINARY	byte[ ]	setBytes
DATE	java.sql.Date	setDate
TIME	java.sql.Time	setTime
TIMESTAMP	java.sql.Timestamp	setTimestamp
CLOB	java.sql.Clob	setClob
BLOB	java.sql.Blob	setBlob
ARRAY	java.sql.Array	setARRAY
REF	java.sql.Ref	SetRef
STRUCT	java.sql.Struct	SetStruct

PreparedStatement και SQL injection

- *SQL injection είναι μια τεχνική όπου οι κακόβουλοι* ***χρήστες*** *μπορούν να* ***δώσουν SQL εντολές σε ένα SQL statement*** *(για παράδειγμα μέσω μία ιστοσελίδας)*
- Παράδειγμα 1 (Web app form) userID: **1 or 1=1**

  Ο εξυπηρετητής θα εκτελέσει τον παρακάτω κώδικα:

  SELECT * FROM users WHERE userID = 1 or 1=1

  Αποτέλεσμα: Θα επιστραφούν όλες οι εγγραφές από τον πίνακα users

PreparedStatement και SQL injection

- *SQL injection είναι μια τεχνική όπου οι κακόβουλοι* ***χρήστες*** *μπορούν να* ***δώσουν SQL εντολές σε ένα SQL statement*** *(για παράδειγμα μέσω μία ιστοσελίδας)*
- Παράδειγμα 2 (Web app form) userID: **1; DROP TABLE users**

  Ο εξυπηρετητής θα εκτελέσει τον παρακάτω κώδικα:

  SELECT * FROM users WHERE userID = 1;

  DROP TABLE users;

  Αποτέλεσμα: Διαγραφή του πίνακα users

PreparedStatement και SQL injection

- ***Η λύση είναι το Prepared Statement***

```
String name = "test'); DROP TABLE info; -- ";

String query1 = "INSERT INTO info VALUES(?,?)";
PreparedStatement stmt1 = dbConn.prepareStatement(query1);
stmt1.setInt(1, 1);
stmt1.setString(2, name);
int i = stmt1.executeUpdate();

String query2 = "INSERT INTO info VALUES(1,'" + name + "')";
Statement stmt2 = dbConn.createStatement();
int j = stmt2.executeUpdate(query2);
```

- *1ο statement προσθέτει το κείμενο (όπως είναι) στη στήλη name*
- *2ο statement διαγράφει το πίνακα info*

ResultSet

- Αναπαριστά ένα σύνολο αποτελεσμάτων από την εκτέλεση μιας επερώτησης στη ΒΔ
- JDBC παρέχει τις ακόλουθες μεθόδους για τη δημιουργία statements με το επιθυμητό αντικείμενο ResultSet:
  - createStatement(int RSType, int RSConcurrency);
  - prepareStatement(String SQL, int RSType, int RSConcurrency);
  - prepareCall(String sql, int RSType, int RSConcurrency);

Τύποι ResultSet

Type	Description
ResultSet.TYPE_FORWARD_ONLY	Ο cursor μπορεί μόνο να κινηθεί μπροστά στο σετ αποτελεσμάτων. **Είναι προεπιλεγμένο.**
ResultSet.TYPE_SCROLL_INSENSITIVE	Ο cursor μπορεί να μετακινηθεί μπροστά και πίσω, και το set αποτελεσμάτων δεν είναι ευαίσθητο στις αλλαγές που έγιναν από άλλους χρήστες στη ΒΔ.
ResultSet.TYPE_SCROLL_SENSITIVE	Ο cursor μπορεί να μετακινηθεί μπροστά και πίσω, και το set αποτελεσμάτων είναι ευαίσθητο στις αλλαγές που έγιναν από άλλους χρήστες στη ΒΔ.

Συγχρονισμός του ResultSet

Type	Description
ResultSet.CONCUR_READ_ONLY	Δημιουργεί ένα σετ αποτελεσμάτων μόνο για ανάγνωση. **Είναι προεπιλεγμένο.**
ResultSet.CONCUR_UPDATABLE	Δημιουργεί ένα σετ αποτελεσμάτων που μπορεί να ενημερωθεί.

Πλοήγηση σε ένα ResultSet

Type	Description
public boolean next()	Χρησιμοποιείται για την μετακίνηση του cursor στην επόμενη εγγραφή από την τρέχουσα θέση.
public boolean previous()	Χρησιμοποιείται για την μετακίνηση του cursor στην προηγούμενη εγγραφή από την τρέχουσα θέση.
public boolean first()	Χρησιμοποιείται για την μετακίνηση του cursor στην πρώτη εγγραφή του σετ αποτελεσμάτων.
public boolean last()	Χρησιμοποιείται για την μετακίνηση του cursor στην τελευταία εγγραφή του σετ αποτελεσμάτων.
public boolean absolute(int row)	Χρησιμοποιείται για την μετακίνηση του cursor στην καθορισμένη γραμμή (εγγραφή) του σετ αποτελεσμάτων.

Get Μέθοδοι ενός ResultSet

Type	Description
public int getInt(int columnIndex)	Χρησιμοποιείται για να επιστρέψει τα δεδομένα με βάση το δείκτη ενός πεδίου της τρέχουσας γραμμής ως έναν ακέραιο.
public int getInt(String columnName)	Χρησιμοποιείται για να επιστρέψει τα δεδομένα με βάση το όνομα ενός πεδίου της τρέχουσας γραμμής ως έναν ακέραιο.
public String getString(int columnIndex)	Χρησιμοποιείται για να επιστρέψει τα δεδομένα με βάση το δείκτη ενός πεδίου της τρέχουσας γραμμής ως ένα αλφαριθμητικό (String).
public String getString(String columnName)	Χρησιμοποιείται για να επιστρέψει τα δεδομένα με βάση το όνομα ενός πεδίου της τρέχουσας γραμμής ως ένα αλφαριθμητικό (String).

ResultSet – Παράδειγμα

```
Statement stmt = dbConn.createStatement();
String query = "SELECT * FROM user";
ResultSet rs = stmt.executeQuery(query);
while (rs.next()){
   String firstname = rs.getString("firstname");
    String lastname = rs.getString("lastname");
   User s = new User();
   s.setFirstname(firstname);
   s.setLastname(lastname);
   users.add(s);
}
```

ResultSet.TYPE_FORWARD_ONLY
ResultSet.CONCUR_READ_ONLY

Συναλλαγή

- Οι συναλλαγές (Transactions) επιτρέπουν να **ελέγχετε εάν, και πότε, οι αλλαγές θα εφαρμοστούν** στη ΒΔ.
- Αντιμετωπίζει ένα μοναδικό SQL statement ή ένα σύνολο από SQL statements ως μία λογική μονάδα, και **εάν κάποιο statement αποτύχει, τότε ολόκληρή η συναλλαγή αποτυγχάνει**.

```
conn.setAutoCommit(false); // turn off auto-commit by passing false
conn.commit( );   // to commit all changes
conn.rollback( );  // to roll back updates to the DB
```

Transaction – Παράδειγμα

```
try{
    conn.setAutoCommit(false);
    Statement stmt = conn.createStatement();
    String SQL = "INSERT INTO Employees  VALUES (106, 20,
    'Rita', 'Tez')";
    stmt.executeUpdate(SQL);
    String SQL = "INSERTED IN Employees  VALUES (107, 22,
    'Sita', 'Singh')";
    stmt.executeUpdate(SQL);
    conn.commit();  // If there is no error.
} catch(SQLException se) {
    conn.rollback(); // If there is any error.
}
```

CallableStatement

- CallableStatement χρησιμοποιείται για την κλήση μιας stored procedure της ΒΔ
- Πως δημιουργούμε ένα αντικείμενο τύπουCallableStatement?

```
CallableStatement stmt=con.prepareCall("{call myprocedure(?,?)}");
```

- Παράμετροι

Parameter	Description
IN	Μία παράμετρος της οποίας η τιμή είναι άγνωστη όταν δημιουργείται το SQL statement. Παρέχετε τιμές στις IN παραμέτρους με τις μεθόδους setXXX().
OUT	Μία παράμετρος της οποίας η τιμή καθορίζεται από τη stored procedure και επιστρέφεται. Η τιμή της OUT παραμέτρου ανακτάται μέσω των μεθόδων getXXX().
INOUT	Μία παράμετρος είσοδου και εξόδου.

CallableStatement – Παράδειγμα 1

```
CREATE PROCEDURE `users`.`getFirstName`
(IN USER_ID INT, OUT FNAME VARCHAR(255))
BEGIN
    SELECT firstname INTO FNAME
    FROM users
    WHERE users_id = USER_ID;
END
```

CallableStatement – Παράδειγμα 1

```
String query = "{call getFirstName (?, ?)}";
CallableStatement stmt = conn.prepareCall(query);
stmt.setInt(1, 1);
stmt.registerOutParameter(2, java.sql.Types.VARCHAR);
stmt.execute();
String fName = stmt.getString(2);  // Get OUT value
stmt.close();  // Always close statements
conn.close();   // Always close connection
```

CallableStatement – Παράδειγμα 2

```
CREATE PROCEDURE `getcountry` (IN COUNTRY_CODE
VARCHAR(3), OUT COUNTRY_NAME VARCHAR(255))
BEGIN
        SELECT NAME INTO COUNTRY_NAME
        FROM COUNTRY
        WHERE Code = COUNTRY_CODE;
END
```

CallableStatement – Παράδειγμα 2

```
try{
    Class.forName(" com.mysql.cj.jdbc.Driver ");
    Connection con=DriverManager.getConnection(
    "jdbc:mysql://localhost:3306/world","root","root");
    String query = "{call getcountry (?, ?)}";
    CallableStatement stmt = con.prepareCall(query);
    stmt.setString(1, "GRC");
    stmt.registerOutParameter(2, java.sql.Types.VARCHAR);
    stmt.execute();
    String countryName = stmt.getString(2);
    System.out.println("Country Name = "+countryName);
    stmt.close();  con.close();
} catch(Exception e) { System.out.println(e);}
```

CallableStatement – Παράδειγμα 3

```
CREATE PROCEDURE `getcitybycoountry`
(IN COUNTRY_CODE VARCHAR(3))
BEGIN
        SELECT *
        FROM city
        WHERE CountryCode = COUNTRY_CODE;
END
```

CallableStatement – Παράδειγμα 3

```
Class.forName(" com.mysql.cj.jdbc.Driver ");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/world","root","root");
String query = "{call getcitybycoountry (?)}";
CallableStatement stmt = con.prepareCall(query);
stmt.setString(1, "GRC");
boolean hasResults = stmt.execute();
while (hasResults) {
        ResultSet resultSet = stmt.getResultSet();
        while (resultSet.next()) {
                String cityName = resultSet.getString("name");
                System.out.println(cityName);
        }
        hasResults = stmt.getMoreResults();
}
stmt.close();  con.close();
```

DB world schema