Bölüm 25

Java Veritabanı Bağlantısı (JDBC)

25.1 JDBC Nedir ?

JDBC, JavaDatabaseConnectivity(Java Veritabanı Bağlantısı) anlamına gelmektedir ve Java programlama dili ve geni veritabanları arasında veritabanı-bağımsız bağlantıyı sağlayan bir Java API'sidir.

JDBC kütüphanesi, her görev için genellikle veritabanı kullanımı ile ilişkili API'leri içerir:

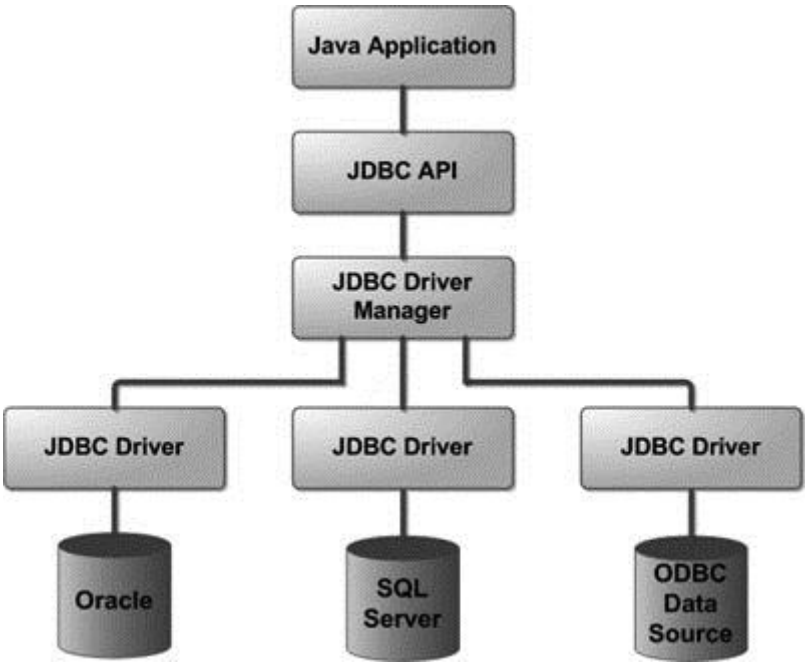
- Veritabanı bağlantısı yapma
- SQL veya MySQL ifadeleri oluşturma
- Veri tabanı içindeki SQL veya MySQL sorgularını çalıştırma
- Sonuç kayıtlarını görüntüleme ve değiştirme

25.2 JDBC Mimarisi

JDBC Mimarisi iki katmana dayanır:

- **JDBC API:** Uygulama ile JDBC Yöneticisi bağlantısını sağlar.
- **JDBC Driver API:** JDBC Yöneticisi ile Sürücü Bağlantısını destekler.

Aşağıdaki mimari diyagram, JDBC sürücüleri ve Java uygulaması ile ilgili sürücü yöneticisinin yerini göstermektedir.



25.3 Genel JDBC Bileşenleri

The JDBC API provides the following interfaces and classes:

JDBC API'si aşağıdaki interface ve sınıfları sağlamaktadır:

- **DriverManager:** Bu sınıf, veritabanı sürücülerinin listesini yönetir.
- **Driver:** Bu interface, veritabanı sunucusu ile iletişimi ele alır. Driver nesneleri ile çok nadir etkileşim kurabilirsiniz. Bunun yerine, bu türün nesnelerini yöneten DriverManager nesnesini kullanırsınız.
- **Connection:** Bu interface, bütün metotları ile veritabanına irtibat kurmak için kullanılır.
- **Statement:** SQL ifadelerini veritabanına göndermek için bu interface'ten oluşturulan nesneleri kullanırsınız.
- **ResultSet:** Statements nesnelerini kullanarak SQL sorgusunu çalıştırdıktan sonra veritabanından alınan verileri tutmak için bu nesneler kullanılır. Onu taşımanıza izin veren bir yineleyici görevi görür.
- **SQLException:** Bu sınıf, bir veritabanı uygulamasında ortaya çıkan hataları ele alır.

25.6 Veritabanı Yükleme

Sizin için en uygun olanı veri tabanı yüklemedir. Bir çok seçeneğe sahip olabilirsiniz ve en genel kullanılanları ise:

- **MySQL DB:** MySQL açık kaynaklı bir veritabanıdır. MySQL'in resmi sitesinden indirebilirsiniz. Tam Windows yüklemesini indirmenizi öneriyoruz. Ayrıca, MySQL Administrator ile birlikte MySQL Query Browser 'ı da indirip, yükleyin. Bunlar geliştirmenizi kolay hale getirilen GUI tabanlı araçlardır. Son olarak, MySQL Connector/J (MySQL JDBC sürücüsü) indirin ve uygun bir dizine açın.
- **PostgreSQL DB:** PostgreSQL açık kaynaklı bir veritabanıdır. PostgreSQL'in resmi sitesinden indirebilirsiniz. Postgres yüklemesi, pgAdmin III diye adlandırılan GUI tabanlı yönetici araçlarını içerir. JDBC sürücüleri ayrıca yüklemenin bir parçasıdır.
- **Oracle DB:** Oracle DB ticari bir veritabanıdır. Oracle yüklemesi, Enterprise Manager diye adlandırılan GUI tabanlı yönetici araçlarını içerir. JDBC sürücüleri ayrıca yüklemenin bir parçasıdır.

25.7 MySQL Database Kurulumu Yapma

Bu çalışmada, MySQL veritabanını kullanacağız. Yukarıda ki veritabanlarından herhangi birini yüklediğinizde, yönetici ID'si **root** diye ayarlıdır ve size kendi şifrenizi seçme hakkı tanır.

Root ID ve şifre kullanarak başka bir kullanıcısı ID ve şifresi oluşturabilir veya JDBC uygulaması için root ID ve şifrenizi kullanabilirsiniz.

Yönetici ID ve şifresi gerektiren, veritabanı oluşturma ve silme gibi bir çok veritabanı işlemi vardır.

25.7.1 Veritabanı Oluşturma

EMP veritabanı oluşturmak için, aşağıdaki adımları izleyin:

Adım 1:

Bir **Komut Penceresi** açın ve yükleme dizinini aşağıdaki gibi değiştirin.

```
C:\>
C:\>cd Program Files\MySQL\bin
C:\Program Files\MySQL\bin>
```

Not: mysqld.exe dosyasının yolu, MySQL'in sisteminizdeki yerine göre değişiklik gösterebilir.

Adım 2:

Aşağıdaki komutu çalıştırarak veritabanı sunucusunu başlatın, eğer hali hazırda çalışmıyorsa.

```
C:\Program Files\MySQL\bin>mysqld
C:\Program Files\MySQL\bin>
```

Adım 3:

Aşağıdaki komutu çalıştırarak **EMP** veritabanını oluşturun.

```
C:\Program Files\MySQL\bin> mysqladmin create EMP -u root -p
Enter password: ********
C:\Program Files\MySQL\bin>
```

25.7.2 Tablo Oluşturma

EMP veritabanı içinde **Employees** tablosunu oluşturmak için, aşağıdaki adımları izleyin:

Adım 1:

Bir **Komut Penceresi** açın ve yükleme dizinini aşağıdaki gibi değiştirin.

```
C:\>
C:\>cd Program Files\MySQL\bin
C:\Program Files\MySQL\bin>
```

Adım 2:

Aşağıdaki gibi veritabanına giriş yapın.

```
C:\Program Files\MySQL\bin>mysql -u root -p
Enter password: ********
mysql>
```

Adım 3:

Aşağıdaki gibi **Employee** tablosunu oluşturun.

```
mysql> use EMP;
mysql> create table Employees
    -> (
    -> id int not null,
    -> age int not null,
    -> first varchar (255),
    -> last varchar (255)
    -> );
Query OK, 0 rows affected (0.08 sec)
mysql>
```

25.7.3 Veri Kayıtları Oluşturma

Sonuç olarak, aşağıdaki gibi Employee tablosunun içinde birkaç kayıt oluşturdunuz.

```
mysql> INSERT INTO Employees VALUES (100, 18, 'Zara', 'Ali');
Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO Employees VALUES (101, 25, 'Mahnaz', 'Fatma');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Employees VALUES (102, 30, 'Zaid', 'Khan');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Employees VALUES (103, 28, 'Sumit', 'Mittal');
Query OK, 1 row affected (0.00 sec)

mysql>
```

Şimdi, JDBC'yi denemeye hazırsınız. Sıradaki çalışma, size JDBC programlamadan bir örnek verecek.

25.8 JDBC Uygulaması Oluşturma (SELECT)

İlk olarak, mysql web sitesinden MySQL JDBC connector'ı indirin.

```
http://dev.mysql.com/downloads/connector/j/
```

Dosyaları indirip, açın, **mysql-connector-java-5.1.24-bin.jar** dosyasını bulacaksınız. **mysql-connector-java-5.1.24-bin.jar** dosyasını CLASSPATH'inize eklemeniz gerekmektedir(Eclipse içindeki src klasörüne sürükleyip bırakın, **mysql-connector-java-5.1.24-bin.jar** dosyasına sağtıklayın ve **Build Path**-> **Add to Build Path** kısmını seçin).

JDBC uygulaması oluştururken, aşağıdaki altı adım kullanılır:

- **Paketleri içe aktarma.** Veritabanı programlamada JDBC sınıflarını içeren paketler gerekmektedir. Sıklıkla, *import java.sql.** paketini kullanmak yeter.
- **JDBC sürücülerini kaydetme.** Veritabanı ile iletişim kanalları açmak için bir sürücü başlatmanız gerekmektedir.
- **Bağlantı açma.** Veritabanı ile fiziksel bağlantı sağlayan ve Connection nesnesini oluşturan *DriverManager.getConnection()* metodu gerekmektedir.
- **Sorgu çalıştırma.** Veritabanına SQL ifadesi göndermek inin, Statement türünde bir nesne gerekmektedir.
- **Sonuç takımından veri çıkartma.** Bir kere SQL sorgusu çalıştığında, tablodan kayıtları alabilirsiniz.
- **Çevreyi temizleme.** Açık bir şekilde bütün veritabanı kaynaklarını kapatmanız gerekmektedir.

Aşağıdaki örnek gelecekte kendi JDBC uygulamanızı geliştirmek için bir şablon oluşturmaktadır.

```
//STEP 1. Import required packages
import java.sql.*;

public class FirstExample {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost/EMP";

   //  Database credentials
```

```
   static final String USER = "username";
   static final String PASS = "password";

   public static void main(String[] args) {
   Connection conn = null;
   Statement stmt = null;
   try{
      //STEP 2: Register JDBC driver
      Class.forName("com.mysql.jdbc.Driver");

      //STEP 3: Open a connection
      System.out.println("Connecting to database...");
      conn = DriverManager.getConnection(DB_URL,USER,PASS);

      //STEP 4: Execute a query
      System.out.println("Creating statement...");
      stmt = conn.createStatement();
      String sql;
      sql = "SELECT id, first, last, age FROM Employees";
      ResultSet rs = stmt.executeQuery(sql);

      //STEP 5: Extract data from result set
      while(rs.next()){
         //Retrieve by column name
         int id  = rs.getInt("id");
         int age = rs.getInt("age");
         String first = rs.getString("first");
         String last = rs.getString("last");

         //Display values
         System.out.print("ID: " + id);
         System.out.print(", Age: " + age);
         System.out.print(", First: " + first);
         System.out.println(", Last: " + last);
      }
      //STEP 6: Clean-up environment
      rs.close();
      stmt.close();
      conn.close();
   }catch(SQLException se){
      //Handle errors for JDBC
      se.printStackTrace();
   }catch(Exception e){
      //Handle errors for Class.forName
      e.printStackTrace();
   }finally{
      //finally block used to close resources
      try{
         if(stmt!=null)
            stmt.close();
      }catch(SQLException se2){
      }// nothing we can do
      try{
         if(conn!=null)
            conn.close();
      }catch(SQLException se){
         se.printStackTrace();
      }//end finally try
   }//end try
   System.out.println("Goodbye!");
}//end main
}//end FirstExample
```

FirstExample 'I çalıştırdığınızda, aşağıdaki sonuç üretilecektir.

```
C:\>java FirstExample
Connecting to database...
Creating statement...
ID: 100, Age: 18, First: Zara, Last: Ali
ID: 101, Age: 25, First: Mahnaz, Last: Fatma
ID: 102, Age: 30, First: Zaid, Last: Khan
ID: 103, Age: 28, First: Sumit, Last: Mittal
C:\>
```

25.9 JDBC Uygulaması Oluşturma (INSERT)

Bu çalışma, JDBC uygulamasında tablo nasıl kayıt girileceğini göstermektedir. Yeni bir JDBC INSERT uygulaması için aşağıdaki adımlar gereklidir.

- Paketleri içe aktarma
- JDBC sürücülerini kaydetme
- Bağlantı açma
- Sorgu çalıştırma
- Çevreyi temizleme

```
//STEP 1. Import required packages
import java.sql.*;

public class JDBCExample {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost/EMP";

   //  Database credentials
   static final String USER = "username";
   static final String PASS = "password";

   public static void main(String[] args) {
   Connection conn = null;
   Statement stmt = null;
   try{
      //STEP 2: Register JDBC driver
      Class.forName("com.mysql.jdbc.Driver");

      //STEP 3: Open a connection
      System.out.println("Connecting to a selected database...");
      conn = DriverManager.getConnection(DB_URL, USER, PASS);
      System.out.println("Connected database successfully...");
```

```
      //STEP 4: Execute a query
      System.out.println("Inserting records into the table...");
      stmt = conn.createStatement();

      String sql = "INSERT INTO Employees VALUES (104, 18,'Ahmet', 'Yildiz')";
      stmt.executeUpdate(sql);
      sql = "INSERT INTO Employees VALUES (105,25,'Zuhal', 'Ak')";
      stmt.executeUpdate(sql);
      sql = "INSERT INTO Employees VALUES (106,30,'Mert', 'Akin')";
      stmt.executeUpdate(sql);
      sql = "INSERT INTO Employees VALUES(107,28,'Selim', 'Man')";
      stmt.executeUpdate(sql);
      System.out.println("Inserted records into the table...");

   }catch(SQLException se){
      //Handle errors for JDBC
      se.printStackTrace();
   }catch(Exception e){
      //Handle errors for Class.forName
      e.printStackTrace();
   }finally{
      //finally block used to close resources
      try{
         if(stmt!=null)
            conn.close();
      }catch(SQLException se){
      }// do nothing
      try{
         if(conn!=null)
            conn.close();
      }catch(SQLException se){
         se.printStackTrace();
      }//end finally try
   }//end try
   System.out.println("Goodbye!");
}//end main
}//end JDBCExample
```

JDBCExample'ı çalıştırdığınızda, aşağıdaki sonuç üretilecektir:

```
C:\>java JDBCExample
Connecting to a selected database...
Connected database successfully...
Inserting records into the table...
Inserted records into the table...
Goodbye!
C:\>
```

25.10 JDBC Uygulaması Oluşturma (UPDATE)

JDBC UPDATE uygulaması oluşturmak için aşağıdaki adımlar gerekmektedir:

- **Paketleri içe aktarma**
- **JDBC sürücülerini kaydetme**

- **Bağlantı açma**
- **Sorgu çalıştırma**
- **Çevreyi temizleme**

```
//STEP 1. Import required packages
import java.sql.*;

public class JDBCupdate {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost/EMP";

   //  Database credentials
   static final String USER = "username";
   static final String PASS = "password";

   public static void main(String[] args) {
   Connection conn = null;
   Statement stmt = null;
   try{
      //STEP 2: Register JDBC driver
      Class.forName("com.mysql.jdbc.Driver");

      //STEP 3: Open a connection
      System.out.println("Connecting to a selected database...");
      conn = DriverManager.getConnection(DB_URL, USER, PASS);
      System.out.println("Connected database successfully...");

      //STEP 4: Execute a query
      System.out.println("Creating statement...");
      stmt = conn.createStatement();
      String sql = "UPDATE Employees SET age = 30 WHERE id in (100, 101)";
      stmt.executeUpdate(sql);

      // Now you can extract all the records
      // to see the updated records
      sql = "SELECT id, first, last, age FROM Employees";
      ResultSet rs = stmt.executeQuery(sql);

      while(rs.next()){
         //Retrieve by column name
         int id  = rs.getInt("id");
         int age = rs.getInt("age");
         String first = rs.getString("first");
         String last = rs.getString("last");

         //Display values
         System.out.print("ID: " + id);
         System.out.print(", Age: " + age);
         System.out.print(", First: " + first);
         System.out.println(", Last: " + last);
      }
      rs.close();
   }catch(SQLException se){
      //Handle errors for JDBC
      se.printStackTrace();
   }catch(Exception e){
      //Handle errors for Class.forName
      e.printStackTrace();
   }finally{
      //finally block used to close resources
```

```
      try{
         if(stmt!=null)
            conn.close();
      }catch(SQLException se){
      }// do nothing
      try{
         if(conn!=null)
            conn.close();
      }catch(SQLException se){
         se.printStackTrace();
      }//end finally try
   }//end try
   System.out.println("Goodbye!");
}//end main
}//end JDBCupdate
```

JDBCupdate'i çalıştırdığınızda, aşağıdaki sonuç üretilecektir:

```
C:\>java JDBCupdate
Connecting to a selected database...
Connected database successfully...
Creating statement...
ID: 100, Age: 30, First: Zara, Last: Ali
ID: 101, Age: 30, First: Mahnaz, Last: Fatma
ID: 102, Age: 30, First: Zaid, Last: Khan
ID: 103, Age: 28, First: Sumit, Last: Mittal
Goodbye!
C:\>
```

25.11 JDBC Uygulaması Oluşturma (DELETE)

JDBC DELETE uygulaması oluşturmak için aşağıdaki adımlar gerekmektedir:

- **Paketleri içe aktarma**
- **JDBC sürücülerini kaydetme**
- **Bağlantı açma**
- **Sorgu çalıştırma**
- **Çevreyi temizleme**

```
//STEP 1. Import required packages
import java.sql.*;

public class JDBCdelete {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost/EMP";

   //  Database credentials
   static final String USER = "username";
   static final String PASS = "password";

   public static void main(String[] args) {
   Connection conn = null;
```

```
   Statement stmt = null;
   try{
      //STEP 2: Register JDBC driver
      Class.forName("com.mysql.jdbc.Driver");

      //STEP 3: Open a connection
      System.out.println("Connecting to a selected database...");
      conn = DriverManager.getConnection(DB_URL, USER, PASS);
      System.out.println("Connected database successfully...");

      //STEP 4: Execute a query
      System.out.println("Creating statement...");
      stmt = conn.createStatement();
      String sql = "DELETE FROM Employees WHERE id = 101";
      stmt.executeUpdate(sql);

      // Now you can extract all the records
      // to see the remaining records
      sql = "SELECT id, first, last, age FROM Employees";
      ResultSet rs = stmt.executeQuery(sql);

      while(rs.next()){
         //Retrieve by column name
         int id  = rs.getInt("id");
         int age = rs.getInt("age");
         String first = rs.getString("first");
         String last = rs.getString("last");

         //Display values
         System.out.print("ID: " + id);
         System.out.print(", Age: " + age);
         System.out.print(", First: " + first);
         System.out.println(", Last: " + last);
      }
      rs.close();
   }catch(SQLException se){
      //Handle errors for JDBC
      se.printStackTrace();
   }catch(Exception e){
      //Handle errors for Class.forName
      e.printStackTrace();
   }finally{
      //finally block used to close resources
      try{
         if(stmt!=null)
            conn.close();
      }catch(SQLException se){
      }// do nothing
      try{
         if(conn!=null)
            conn.close();
      }catch(SQLException se){
         se.printStackTrace();
      }//end finally try
   }//end try
   System.out.println("Goodbye!");
}//end main
}//end JDBCdelete
```

JDBCdelete'i çalıştırdığınızda, aşağıdaki sonuç üretilecektir:

```
C:\>java JDBCdelete
Connecting to a selected database...
Connected database successfully...
Creating statement...
```

```
ID: 100, Age: 30, First: Zara, Last: Ali
ID: 102, Age: 30, First: Zaid, Last: Khan
ID: 103, Age: 28, First: Sumit, Last: Mittal
Goodbye!
C:\>
```

25.12 PreparedStatement Nesnesi

PreparedStatement interface'i Statement interface'inden genişletilen , size bir çift avantajla birlikte ek işlevsellik sağlayan genel bir Statement nesnesidir. Bu ifade, argümanları dinamik olarak temin ederken size bir esneklik kazandırır.

```
//STEP 1. Import required packages
import java.sql.*;

public class JDBCdelete {
         // JDBC driver name and database URL
         static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
         static final String DB_URL = "jdbc:mysql://localhost/EMP";

         // Database credentials
         static final String USER = "username";
         static final String PASS = "password";

         public static void main(String[] args) {
                 Connection conn = null;
                 PreparedStatement pstmt = null;
                 try {
                   // STEP 2: Register JDBC driver
                   Class.forName("com.mysql.jdbc.Driver");

                   // STEP 3: Open a connection
                   System.out.println("Connecting to a selected database...");
                   conn = DriverManager.getConnection(DB_URL, USER, PASS);
                   System.out.println("Connected database successfully...");

                   // STEP 4: Execute a query
                   System.out.println("Inserting record into the table...");

                   String sql = "INSERT INTO ogrenci VALUES (?, ?, ?, ?)";
                   pstmt = conn.prepareStatement(sql);

                   pstmt.setInt(1, 112);
                   pstmt.setInt(2, 32);
                   pstmt.setString(3, "Halil");
                   pstmt.setString(4, "Guven");
                   pstmt.executeUpdate();

                   System.out.println("Inserted record into the table...");
```

```
                } catch (SQLException se) {
                        // Handle errors for JDBC
                        se.printStackTrace();
                } catch (Exception e) {
                        // Handle errors for Class.forName
                        e.printStackTrace();
                } finally {
                        // finally block used to close resources
                        try {
                                if (pstmt != null)
                                        conn.close();
                        } catch (SQLException se) {
                        }// do nothing
                        try {
                                if (conn != null)
                                        conn.close();
                        } catch (SQLException se) {
                                se.printStackTrace();
                        }// end finally try
                }// end try
                System.out.println("Goodbye!");
        }// end main
}// end JDBCprepared
```

JDBC de ki bütün parametler, parametre işareti olarak bilinen **?** ile gösterilir. SQL ifadesini çalıştırmadan önce,her parametre için değerleri tedarik etmelisiniz.

setXXX() metodu değerleri parametrelere bağlar, burada **XXX** Java daki herhangi bir veri tipi olabilir. Eğer değerleri temin etmeyi unuttuysanız, bir SQLException alırsınız.

Her parametre işareti, kendisinin sıralı pozisyonu ile anılır. Birinci işaret pozisyon 1 ile, sıradaki pozisyon 2 ile ve benzeri. **Bu metod sıfır ile başlayan Java dizi indexinden farklıdır.**

Her Statement nesnesinin metotları veritabanı ile etkileşim kurmak içindir (a)execute (b) executeQuery(), ve (c) executeUpdate() metotları ayrıca PreparedStatement nesnesi ile birlikte çalışır.