Üst Düzey Programlama

JDBC

(Java Database Connectivity)

JDBC

JDBC ilişkisel veritabanlarına erişim için Java dilinde kullanılan standart bir kütüphanedir.

Bu kütüphanedeki sınıfları kullanarak, aynı söz dizilimi ile farklı veritabanlarında işlem yapabilirsiniz.

Veritabanı ile ilgili sınıflar **java.sql.*** paketinde bulunmaktadır.

JDBC ile veritabanında işlem yapabilmek için şu adımlar izlenmelidir:

1- JDBC Sürücüsünü(Driver) Yüklemek

Sürücü(driver) veritabanı ile nasıl iletişim yapacağını bilen sınıflardır.

Java sınıfları veritabanı işlemlerinde arka planda sürücü içerisindeki sınıfları kullanırlar.

Sürücüyü kullanabilmek için uygun sınıfları belleğe yüklememiz yeterlidir.

Sürücü sınıfı kendi örneğini oluşturur ve JDBC'nin sürücü yönetici(DriverManager) sınıfına kayıt eder.

1- JDBC Sürücüsünü(Driver) Yüklemek

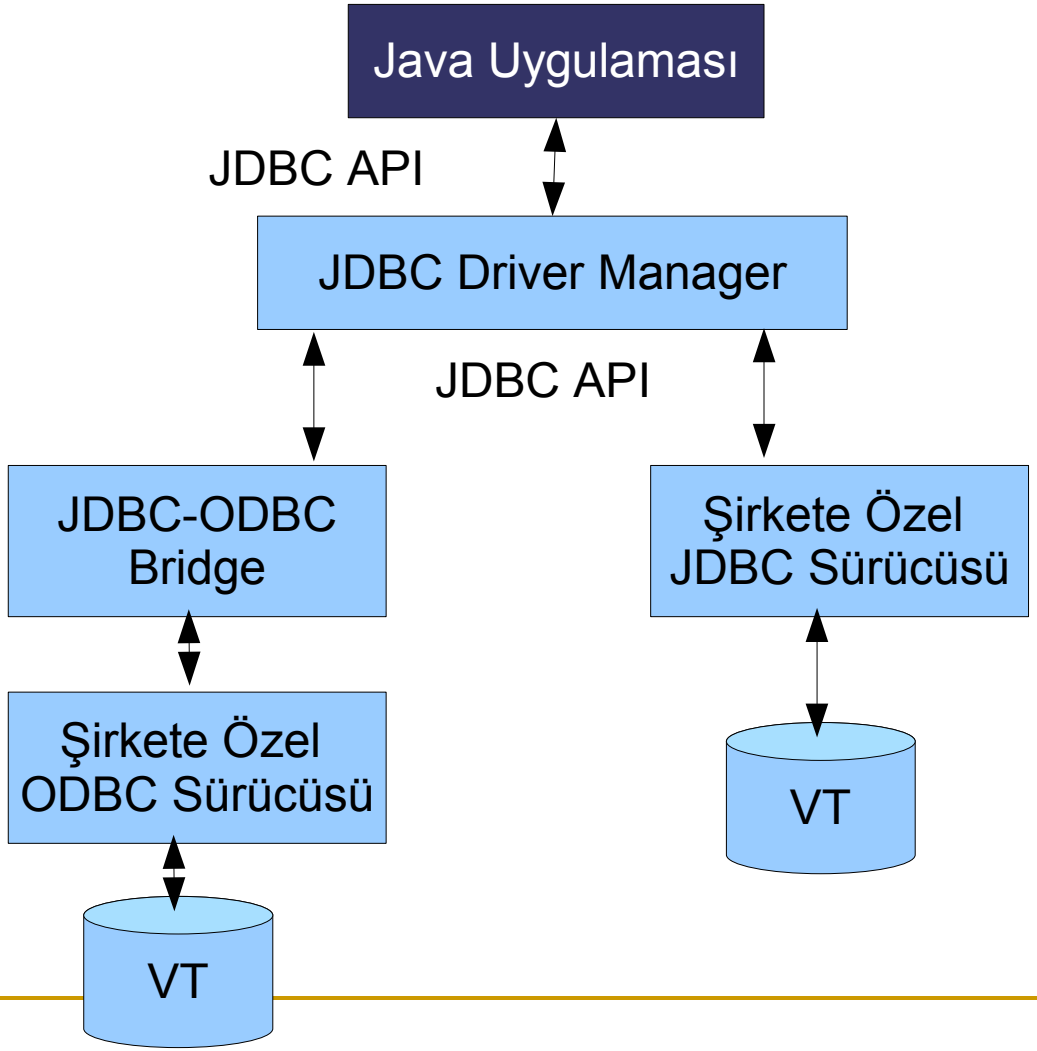
Örnek sürücü yükleme kodu.

```
try{
        Class.forName("com.mysql.jdbc.Driver");
        Class.forName("oracle.jdbc.driver.OracleDriver");
        ...
}catch(ClassNotFoundException e){
        e.printStackTrace();
}
```

Sürücü sınıfı **CLASSPATH** değişkeni ile ulaşılabilmelidir.

Web uygulamalarında sürücü sunucunun kütüphaneleri içerisinde ya da **WEB-INF/lib** klasöründe olmalıdır.

1- JDBC Sürücüsünü(Driver) Yüklemek



2- Bağlantı Stringi oluşturmak

Bağlanmak istediğimiz veri tabanı sunucusuna özel bağlantı String leri oluşturmamız gereklidir.

Bu String ler veritabanına bağlantıda kullanılır.

Örnek:

String baglantiURL=”jdbc:mysql://127.0.0.1/test?user=root&password=a”;

3- Bağlantı Oluşturma

Sunucuya bağlanma

Örnek:

```
Connection baglanti = DriverManager.getConnection(baglantiURL);
```

Herhangi bir problemde SQLException fırlatılır.
Bu Exception değerlendirilmelidir.

4- İfade Oluşturma

Veritabanına yapacağı işleri söylemek için kullanılan(sql veya komut) sınıftır.

Örnek:

```
Statement ifade = baglanti.createStatement();
```

5- Bir sorgu ya da güncelleme icra etme

Sorgunun çalıştırılması

Örnek:

```
String sorgu = "Select * from tablo1";

ResultSet sonucKumesi = ifade.executeQuery(sorgu);
```

5- Bir sorgu ya da güncelleme icra etme

***executeQuery():** sorgu çalıştırır ve geriye ResultSet çevirir, bu küme boş olabilir fakat null olmaz.

***executeUpdate():** UPDATE,INSERT,DELETE sorguları için kullanılır. Geriye etkilenen satır sayısı çevrilir. Ayrıca CREATE TABLE, DROP TABLE, ALTER TABLE gibi DDL sorgularını da icra eder.

***setQueryTimeOut():** Sürücünün SQLException oluşturmadan önce ne kadar süre bekleyeceğini belirtir.

***setMaxRows():** Sonuç kümesinin maksimum kaç satır olacağını belirtir. Fazlası kesilir.

VeritabaniTest01.java

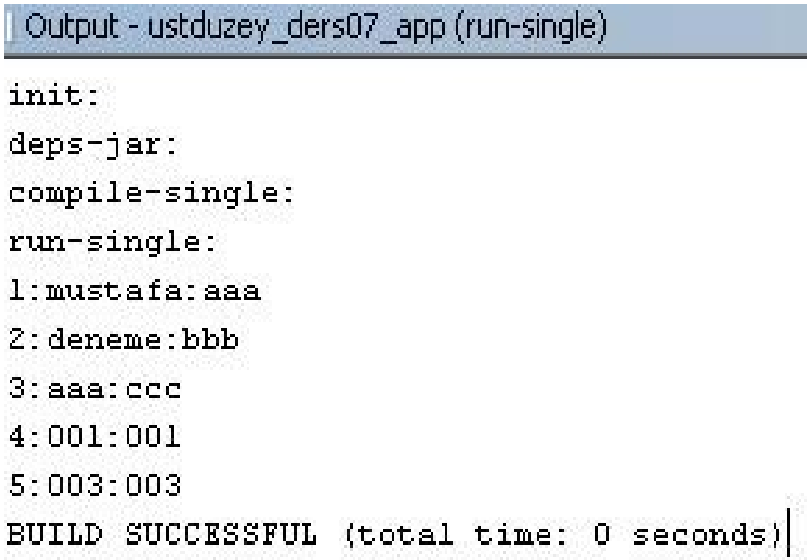
```
package ce.ders07;

import ...

public class VeritabaniTest01 {

    public static void main(String[] args) {
        String baglantiURL="jdbc:mysql://127.0.0.1/test?user=mustafa&password=mustafa";
        String surucu = "com.mysql.jdbc.Driver";
        try {
            Class.forName(surucu);
            Connection baglanti = DriverManager.getConnection(baglantiURL);
            Statement ifade = baglanti.createStatement();
            String sorgu = "select  * from tablo01";
            ResultSet sonucKumesi = ifade.executeQuery(sorgu);

            while (sonucKumesi.next()){
                System.out.print(sonucKumesi.getString(1));
                System.out.print(":");
                System.out.print(sonucKumesi.getString(2));
                System.out.print(":");
                System.out.print(sonucKumesi.getString(3));
                System.out.println("");
            }
        } catch (ClassNotFoundException e) {
            System.err.println("Sürücü Bulunamadi");
        }catch(SQLException e){
            System.err.println("Veritabanı hatası:"+e);
        }catch(Exception e){
            System.err.println("HATA:"+e);
        }
    }
```


Output - ustduzey_ders07_app (run-single)
init:
deps-jar:
compile-single:
run-single:
1:mustafa:aaa
2:deneme:bbb
3:aaa:ccc
4:001:001
5:003:003
BUILD SUCCESSFUL (total time: 0 seconds)

6- Sonuç Verilerini İşlemek

Sonuç ResultSet nesnesinin satırlarından oluşur.

Her satırda bulunan sütunlar **1 numaralı indisten** başlar.

Sonuç kümesinden belirli bir sütun bilgisini almak için ***getXXX (sütun no ya da sütun adı)*** metotları kullanılır. getString(1) , getInteger(2) vb.

Tüm sonuç kümesini dolaşmak için;

```
while(sonucKumesi.next()){
        System.out.println(sonucKumesi.getString(1));
}
```

6- Sonuç Verilerini İşlemek

***next/previous :** sonraki ve önceki kayıta gider.

***relative/absolute:** belirli bir satıra gider.

***getXXX():** sütun değerini verir.

***findColumn(sütun adı):** sütunun indis numarasını verir.

***getRow():** mevcut satır numarasını verir.

***getMetaData:** ResultSetMetaData nesnesi geriye çevirir. Bu nesne sonuç kümesi hakkında bilgi içerir.

7- Bağlantıyı Kapatmak

baglanti.close();

Bu bağlantı ile ilgili olan
diğer nesnelerde (ResultSet,Statement) kapatılır.

Bağlantı açılması ve kapatılması sistem yük getirdiği için genellikle uygulama başında açılır ve uygulama sonlandırıldığında kapatılır.

Hazırlanmış İfadelerin Kullanılması (Prepared Statement)

Aynı SQL ifadesini çok kez çalıştıracaksanız, bu sorguyu parametreli ifade haline getirerek kullanmanız uygulamanın etkinliğini arttıracaktır.

Hazırlanmış sorgular veritabanı sunucusunda derlenip saklandıkları için daha performanslı çalışırlar.

```
String sorgu="UPDATE tablo1 SET soyad=? where id=?";
PreparedStatement ifade = baglanti.prepareStatement(sorgu);

ifade.setString(1,"BBB");
ifade.setInt(2,3);
ifade.executeUpdate();
```

HazirlanmisIfade.java

```
package ce.ders07;

import ...

public class HazirlanmisIfade {

    public static void main(String[] args) {
        String baglantiURL="jdbc:mysql://127.0.0.1/test?user=mustafa&password=mustafa";
        String surucu = "com.mysql.jdbc.Driver";
        try {
            Class.forName(surucu);
            Connection  baglanti = DriverManager.getConnection(baglantiURL);

            String sorgu = "UPDATE tablo01 SET soyad = ? WHERE id = ? ";
            PreparedStatement ifade = baglanti.prepareStatement(sorgu);
            ifade.setString(1,"BBB");
            ifade.setInt(2,3);
            ifade.executeUpdate();

            baglanti.close();
        } catch (SQLException ex) {
            System.err.println("Veritabanı hatası:"+ex);
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        }catch(Exception e){
            System.err.println("HATA:"+e);
        }
    }
}
```

HazirlanmisIfade2.java

```
package ce.ders07;

import ...

public class HazirlanmisIfade2 {

    public static void main(String[] args) {
        String baglantiURL="jdbc:mysql://127.0.0.1/test?user=mustafa&password=mustafa";
        String surucu = "com.mysql.jdbc.Driver";
        try {
            Class.forName(surucu);
            Connection  baglanti = DriverManager.getConnection(baglantiURL);
            String sorgu = "Select * FROM tablo01 WHERE id = ? ";
            PreparedStatement ifade = baglanti.prepareStatement(sorgu);
            ifade.setInt(1,1);
            ResultSet sonuc= (ResultSet) ifade.executeQuery();
            while (sonuc.next()){
                System.out.println(sonuc.getInt(1));
                System.out.println(sonuc.getString(2));
                System.out.println(sonuc.getString(3));
            }
            baglanti.close();
        } catch (SQLException ex) {
            System.err.println("Veritabanı hatası:"+ex);
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        }catch(Exception e){
            System.err.println("HATA:"+e);
        }
    }
```

Çağrılabilir İfadeler (Callable Statements)- Stored Procedures

CallableStatement nesnedi ile veri tabanında bulunan bir stored procedure/function çalıştırılabilir. Stored procedure ve stored function veritabanında saklanılan özel yazım şekli olan prosedür ve fonksiyonlardır. Normal SQL sorgularına göre daha hızlı çalışırlar. Performans ve güvenlik için kullanılırlar.
Tipleri:

{call prosedür_adı} //parametresiz
{call prosedür_adı(?,?,...)} //parametreli
{?=call prosedür_adı} //parametresiz, geri dönüşlü
{?=call prosedür_adı(?,?,...)} //parametreli, geri dönüşlü

Çağrılabilir İfadeler (Callable Statements)- Stored Procedures

Genel kullanım şekli:

```
String prosedur="{?=call prosedur_adi(?,?)}";

CallableStatement ifade = baglanti.prepareCall(prosedur);
ifade.setString(2,"ad");
ifade.setFloat(3,120.0);
ifade.execute();

Geri Dönüş Değerini almak için örneğin;

int deger = ifade.getInt(1);
```

MySQL SQL Editor

```
1 CREATE DEFINER=`mustafa`@`%` PROCEDURE `DEGISTIR`(IN IDD INTEGER,IN YENI_SOYAD VARCHAR(45))
2 BEGIN
3      UPDATE tablo01 SET tablo01.soyad=YENI_SOYAD WHERE tablo01.id=IDD;
4 END
```

CagrilirIfade01.java

```
package ce.ders07;

import ...

public class CagrilirIfade01 {

      public static void main(String[] args) {
        String baglantiURL="jdbc:mysql://127.0.0.1/test";
        String surucu = "com.mysql.jdbc.Driver";
        try {
            Class.forName(surucu);
            Connection  baglanti = DriverManager.getConnection(baglantiURL,"mustafa","mustafa");
            String sorgu = "{call DEGISTIR(?,?)}";
            CallableStatement ifade = baglanti.prepareCall(sorgu);
            ifade.setInt(1,5);
            ifade.setString(2,"deneme");
            ifade.execute();

            baglanti.close();
        } catch (SQLException ex) {
            System.err.println("Veritabanı hatası:"+ex);
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        }catch(Exception e){
            System.err.println("HATA:"+e);
        }
    }
}
```

MySQL SQL Editor

```
1 CREATE DEFINER=`mustafa`@`%` FUNCTION `BIRLESTIR`(SS CHAR(20)) RETURNS char(50) CHARSET utf8
2 BEGIN
3     RETURN concat("merhaba ,",ss);
4 END
```

CagrilirIfade2.java

```
package ce.ders07;

import ...

public class CagrilirIfade2 {
    public static void main(String[] args) {
        String baglantiURL="jdbc:mysql://127.0.0.1/test";
        String surucu = "com.mysql.jdbc.Driver";
        try {
            Class.forName(surucu);
            Connection  baglanti = DriverManager.getConnection(baglantiURL,"mustafa","mustafa");
            String sorgu = "{?=call BIRLESTIR(?)}";
            CallableStatement ifade = baglanti.prepareCall(sorgu);
            ifade.setString(2,"mustafa");
            ResultSet sonuc = (ResultSet) ifade.executeQuery();
            while (sonuc.next()){
                System.out.println("SONUC = " + sonuc.getString(1));
            }
            baglanti.close();
        } catch (SQLException ex) {
            System.err.println("Veritabanı hatası:"+ex);
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        }catch(Exception e){
            System.err.println("HATA:"+e);
        }
    }
}
```

MySQL SQL Editor

```
1 CREATE DEFINER=`mustafa`@`%` PROCEDURE `BBB`()
2 BEGIN
3   SELECT * FROM tablo01;
4 END
```

CagrilirIfade3.java

```
package ce.ders07;

import ...

public class CagrilirIfade3 {

    public static void main(String[] args) {
        String baglantiURL="jdbc:mysql://127.0.0.1/test";
        String surucu = "com.mysql.jdbc.Driver";
        try {
            Class.forName(surucu);
            Connection  baglanti = DriverManager.getConnection(baglantiURL,"mustafa","mustafa");
            String sorgu = "{call BBB()}";
            CallableStatement ifade = baglanti.prepareCall(sorgu);

            ResultSet sonuc = (ResultSet) ifade.executeQuery();
            while (sonuc.next()){
                System.out.println("ID = " + sonuc.getString(1));
            }
            baglanti.close();
        } catch (SQLException ex) {
            System.err.println("Veritabanı hatası:"+ex);
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        }catch(Exception e){
            System.err.println("HATA:"+e);
        }
    }
}
```

Veritabanı Hareket(Transaction) Kullanımı

Bir veritabanı güncellendiğinde değişiklikler kalıcı olarak yazılır.
Bu varsayılan özelliği program kodumuz ile değiştirebiliriz.

Otomatik kayıt işlemini durdururuz.

Değişiklikler ve güncellemeleri yaparız ve daha sonra değişikliklerin veritabanına kalıcı olarak yazılmasını ya da yapılan tüm değişikliklerin iptal edilmesini söyleriz.

Bu işleme ***hareket yönetimi (transaction management)*** adı verilir.

Veritabanı Hareket(Transaction) Kullanımı

Birden fazla veri tabanı işlemlerinin ya hepsinin yapılması ya da hiçbirinin yapılmaması için transaction mekanizması kullanılır.

Veritabanı bütünlüğü korunur.

Veritabanının varsayılan bağlantısı veritabanı güncellemelerini kendiliğinden gerçekleştirir (autocommit). Her çalıştırılan ifade(statement) kendiliğinden veritabanında değişikliğe sebep olur.

Hareket (Transaction) yönetimi için bu özellik iptal edilmelidir. Bu işlem için bağlantı (connection) nesnesinin *setAutoCommit(false)* metodu çalıştırılır.

Veritabanı Hareket(Transaction) Kullanımı

İşlemler sonucunda değişikliklerin veritabanına gerçekleştirilmesi için bağlantı(connection) nesnesinin ***commit()*** metodu, hata olduğunda tüm işlemlerin olmaması içinde ***rollback()*** metodu çalıştırılır.

Veritabanı Hareket(Transaction) Kullanımı

Genel kullanım:

```
Connection baglanti = DriverManager.getConnection(url);
boolean otomatikCommit = baglanti.getAutoCommit();
Statement ifade;

try{
        baglanti.setAutoCommit(false);
        ifade = connection.createStatement();
        ifade.execute();
        ifade.execute();
        ...
        baglanti.commit();
}catch(SQLException e){
        baglanti.rollback();
}finally{
        ifade.close();
        baglanti.setAutoCommit(otomatikCommit);
}
```

TransactionOrnek1.java

```
package ce.ders07;
import ...

public class TransactionOrnek1 {
        public static void main(String[] args) {
         String baglantiURL="jdbc:mysql://127.0.0.1/test";
         String surucu = "com.mysql.jdbc.Driver";
         Statement ifade;
         Connection baglanti = null;
         try {
             Class.forName(surucu);
             baglanti = DriverManager.getConnection(baglantiURL,"mustafa","mustafa");
             boolean autoCommit = baglanti.getAutoCommit();
             baglanti.setAutoCommit(false);
             ifade = baglanti.createStatement();
             ifade.executeUpdate("update tablo01 set soyad=111 where id=1");
             ifade.executeUpdate("update tablo01 set soyad=222 where id=2");
             ifade.executeUpdate("update tablo01 set soyad=333 where id=3");

             baglanti.commit();
             //baglanti.rollback();
         } catch (ClassNotFoundException e) {
             System.err.println("Sürücü Bulunamadi");
         }catch(SQLException e){
             try {
                 baglanti.rollback();
             } catch (SQLException ex) {
                System.err.println("Veritabanı hatası:"+e);
             }
             System.err.println("Veritabanı hatası:"+e);
         }catch(Exception e){
             System.err.println("HATA:"+e);
         }
```

TransactionOrnek2.java

```
package ce.ders07;
import ...

public class TransactionOrnek2 {
    public static void main(String[] args) {
        String baglantiURL="jdbc:mysql://127.0.0.1/test";
        String surucu = "com.mysql.jdbc.Driver";
        Statement ifade;
        Connection baglanti = null;
        try {
            Class.forName(surucu);
            baglanti = DriverManager.getConnection(baglantiURL,"mustafa","mustafa");
            boolean autoCommit = baglanti.getAutoCommit();
            baglanti.setAutoCommit(false);
            ifade = baglanti.createStatement();
            ifade.executeUpdate("update tablo01 set soyad=lll where id=1");

            Savepoint save1 = baglanti.setSavepoint("save1");
            ifade.executeUpdate("update tablo01 set soyad=222 where id=2");

            Savepoint save2 = baglanti.setSavepoint("save2");
            ifade.executeUpdate("update tablo01 set soyad=333 where id=3");

            baglanti.rollback(save2);
            baglanti.commit();

        } catch (ClassNotFoundException e) {
            System.err.println("Sürücü Bulunamadi");
        }catch(SQLException e){
            try { baglanti.rollback();
            } catch (SQLException ex) {System.err.println("Veritabanı hatası:"+e);}
            System.err.println("Veritabanı hatası:"+e);
        }catch(Exception e){System.err.println("HATA:"+e);}
```

sorguCalistir.html

```
<html>
    <head>
        <meta http-equiv="text/html;charset=utf8">
        <title></title>
    </head>
    <body>
        <form action="TablolTest01" method="post">
            <table>
                <tr>
                    <td>Sürücü</td>
                    <td><input type="text" name="surucu" size="30"></td>
                </tr>
                 <tr>
                    <td>URL</td>
                    <td><input type="text" name="url" size="30"></td>
                </tr>
                 <tr>
                    <td>Sorgu</td>
                    <td><textarea rows="5" cols="30" name="sorgu" ></textarea></td>
                </tr>
            </table>
            <input type="submit" value="Gönder">
        </form>
    </body>
</html>
```

Tablo1Test01.java

```
import ...

public class Tablo1Test01 extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        String surucu = request.getParameter("surucu");
        out.println(surucu);
        if (surucu==null || surucu.trim().equals("")){
            surucu = "com.mysql.jdbc.Driver";
        }
        String url = request.getParameter("url");
        if (url==null || url.trim().equals("")){
            url="jdbc:mysql://127.0.0.1/test?user=mustafa&password=mustafa";
        }
        String sorgu = request.getParameter("sorgu");
        if (sorgu==null || sorgu.trim().equals("")){
            sorgu="select * from tablo01";
        }

        out.println("<html>");
        out.println("<head>");
        out.println("<title>JDBC Sorgusu</title>");
        out.println("</head>");
        out.println("<body>");
        tabloGoster(surucu,url,sorgu,out);
        out.println("</body>");
        out.println("</html>");

        out.close();
    }
```

Tablo1Test01.java

```
private void tabloGoster(String surucu,String url,String sorgu,PrintWriter out){
    try {
        Class.forName(surucu);
        Connection baglanti = DriverManager.getConnection(url);
        DatabaseMetaData dbBilgi = baglanti.getMetaData();
        out.println("<UL>");
        String urunAdi = dbBilgi.getDatabaseProductName();
        String urunVersiyonu = dbBilgi.getDatabaseProductVersion();
        out.println("<LI><B>Veritabani:"+urunAdi+"</B>");
        out.println("<LI><B>Versiyonu:"+urunVersiyonu+"</B>");
        Statement ifade = baglanti.createStatement();
        ResultSet sonucKumesi = ifade.executeQuery(sorgu);
        out.println("<table border=1>");
        ResultSetMetaData rsBilgi = sonucKumesi.getMetaData();
        int sutunSayisi = rsBilgi.getColumnCount();
        out.println("<tr>");
        for (int i = 1; i <= sutunSayisi; i++) {
            out.println("<th>"+rsBilgi.getColumnName(i));
        }
        out.println();
        while(sonucKumesi.next()){
            out.println("<tr>");
            for (int i = 1; i <= sutunSayisi; i++) {
                out.print("<td>"+sonucKumesi.getString(i));
            }
            out.println();
        }
        out.println("</table>");
        baglanti.close();
    } catch (Exception e) {
        out.println("HATA:"+e);
    }
}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XML
    <servlet>
        <servlet-name>Tablo1Test01</servlet-name>
        <servlet-class>ce.ders07.Tablo1Test01</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Tablo1Test01</servlet-name>
        <url-pattern>/Tablo1Test01</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>
            index.jsp
        </welcome-file>
    </welcome-file-list>
    <context-param>
        <param-name>javax.servlet.jsp.jstl.sql.dataSource</param-name>
        <param-value>jdbc:mysql://127.0.0.1/test,com.mysql.jdbc.Driver,mustafa,mustafa</param-value>
    </context-param>
</web-app>
```

JSTL etiketleri bu parametreyi kullanir

listele.jsp

```jsp
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>


<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <sql:query var="sonuc">
            select * from tablo01
        </sql:query>
        <table border="1">
            <c:forEach var="satir" items="${sonuc.rows}">
                <tr>
                <td><c:out value="${satir.id}"/></td>
                <td><c:out value="${satir.ad}"/></td>
                <td><c:out value="${satir.soyad}"/></td>
                </tr>
            </c:forEach>
        </table>
    </body>
</html>
```

Üst Düzey Programlama

JDBC

(Java Database Connectivity)