

Nesne Tabanlı Programlama II

Öğr. Gör. Dr. Aysun ALTIKARDEŞ

Veri Tabanı İşlemleri

Veri Tabanı Nedir?

Birbiri ile ilişkili dosyaların oluşturduğu yapıya **veritabanı** (*database*) adı verilir. Veritabanları, birbiri ile ilişkili, düzenli bilgiler topluluğudur.

Veri Tabanı Nedir?

Veritabanlarını yönetmek için tasarlanmış sistem ve yazılımlara ise **Veritabanı Yönetim Sistemi** (*DataBase Management System*, kısaca DBMS) adı verilmektedir. Veritabanı yönetim sistemlerinin kullanım sebepleri ise aşağıda sıralanmıştır:

- **Verilerin tekrarı azdır.**
- **Yanlışlıkların giderilmesi.**
- **Verilerin paylaşımı sağlanır.**
- **Bilgilerin standartlaştırılması sağlanır.**

Java Veritabanı Bağlantısı (JDBC-Java DataBase Connectivity)

JDBC, Java programları ile herhangi bir veritabanı arasında iletişimi sağlayan bir Java köprüsüdür. JDBC bize herhangi bir veritabanına Java üzerinden ulaşmamızı ve SQL ile doğrudan sorgular çalıştırarak programlama yapabilmemizi sağlar. Ayrıca veritabanı işlemlerini gerçekleştirebilmek için **java.sql** paketini program başında çağırmak (*import*) etmek gerekir.

Java Veritabanı Bağlantısı (JDBC-Java DataBase Connectivity)

JDBC, aşağıdaki tabloda da görüldüğü üzere **Connection**, **Statement**, **ResultSet** sınıf veya arayüzlerine sahiptir.

Sınıf/Arayüz	Açıklama
Connection	Bu sınıf veritabanı ile bağlantıyı gerçekleştirir.
Statement	SQL sorgularını veritabanına iletmek için kullanılır.
ResultSet	Sorgu sonuçları ile ilgili işlemler için kullanılır.

Access Veritabanı Dosyası Oluşturma

İlk olarak «**okul**» adında bir veritabanı dosyası oluşturuyoruz. Ardından «**ogretmenler**» ve «**ogrenciler**» adlı 2 tablo oluşturuyoruz.

Alan Adı	Veri Türü
ogr_okulno	Sayı
ogr_adi	Metin
ogr_soyadi	Metin
ogr_sinifi	Metin
ogr_cinsiyeti	Metin

Alan Adı	Veri Türü
ogrt_tcno	Metin
ogrt_adi	Metin
ogrt_soyadi	Metin
ogrt_cinsiyeti	Metin
ogrt_alani	Metin

Access Veritabanı Dosyası Oluşturma

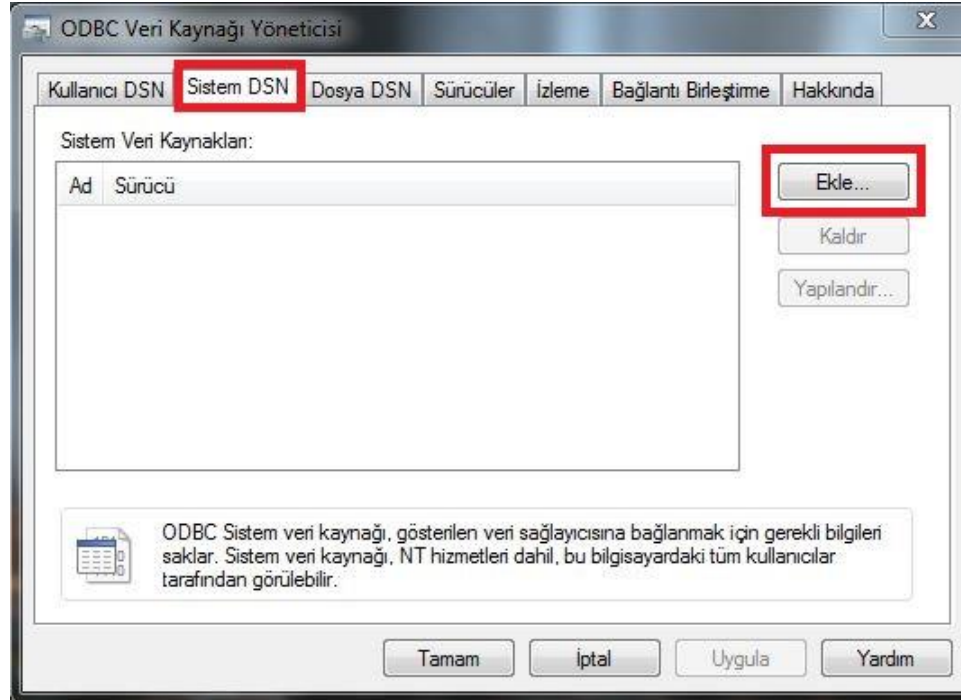
Son olarak tablolarımıza verilerimizi giriyoruz...

ogr_okulno	ogr_adi	ogr_soyadi	ogr_sinifi	ogr_cinsiyet
100	Salih	TÜRK	12/A	E
101	Ahmet	UYAR	12/A	E
102	Berna	YILDIRIM	12/B	K
103	Naim	YALÇIN	12/B	E
104	Elif	NARİN	12/B	K

ogrt_teno	ogrt_adi	ogrt_soyadi	ogrt_cinsiyet	ogrt_alani
14724835478	Merve	KAFALI	K	Fizik
57489654123	Mehmet	ŞEKER	E	Fizik
68754879124	Tufan	TÜRKOĞLU	E	Matematik
74525621152	Fatma	TOĞCU	K	Türkçe

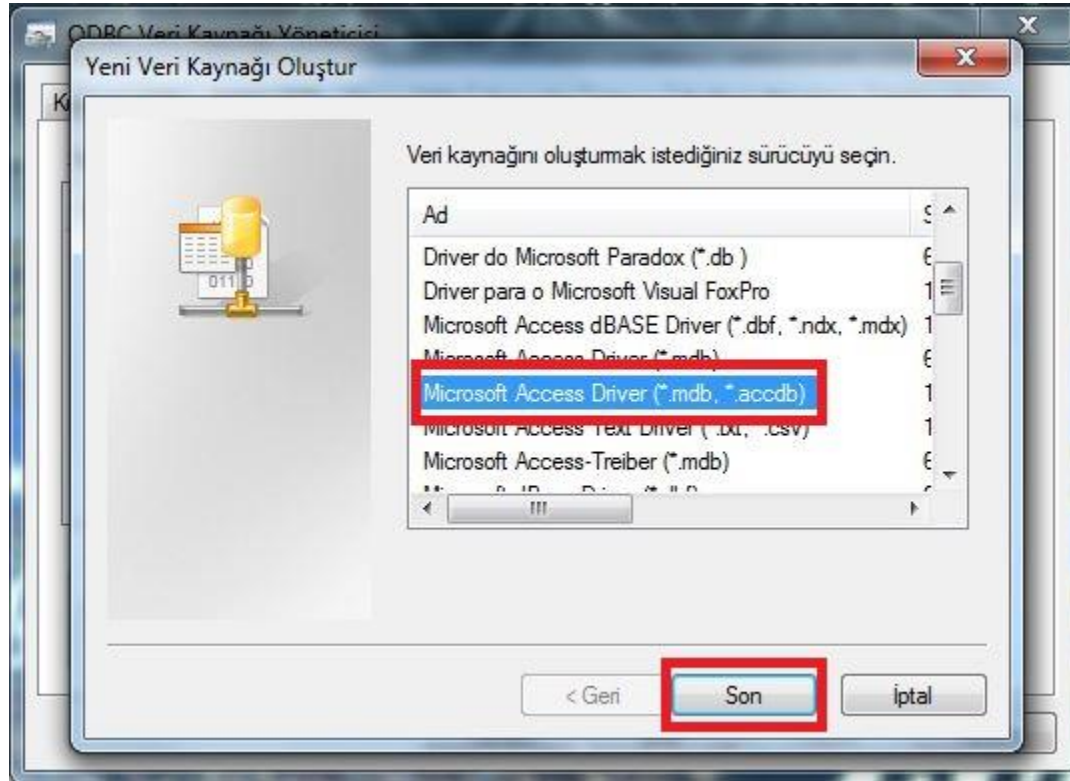
Kullanılacak Veritabanının Seçilmesi ve ODBC Bağlantısı (Access Veritabanı İçin)

İlk önce **Denetim Masası > Yönetimsel Araçlar > Veri Kaynakları (ODBC)** seçilir. ODBC Yönetici ekranından **Sistem DSN** menüsünden **Ekle** düğmesine tıklanır.



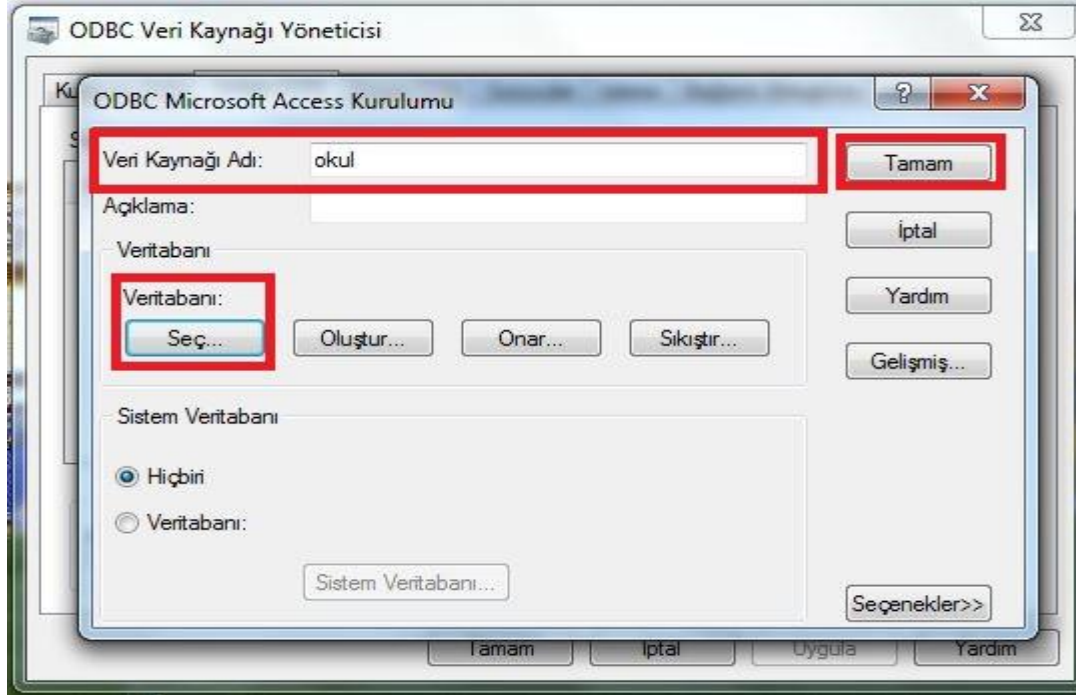
Kullanılacak Veritabanının Seçilmesi ve ODBC Bağlantısı (Access Veritabanı İçin)

Ardından gelen ekrandan Microsoft Access Driver seçeneği seçilir ve Son butonuna tıklanılır.



Kullanılacak Veritabanının Seçilmesi ve ODBC Bağlantısı (Access Veritabanı İçin)

Ardından Veri Kaynağı Adı alanına Veritabanı dosyanız için bir isim giriniz. **Bu ismi unutmamalısınız çünkü program içerisinde kullanacaksınız** 😊



Seç butonu ile kaynak veritabanı dosyanızı seçip Tamam butonuna tıklayınız.

JAVA ile Access Bağlantısı

ODBC bağlantısı tamamlandıktan sonra, «**okul_access**» adında proje oluşturulur. Ardında yine aynı adda class proje dosyasına eklenir. Daha sonra SQL işlemleri için gerekli olan «**java.sql**» kütüphanesi projeye import edilir.

```
package okul_access;

import java.sql.*;

public class okul_access {

    public static void main(String[] args)
    {
        try
        {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection bag = DriverManager.getConnection("jdbc:odbc:okul");

            Statement iletı = bag.createStatement();
            ResultSet rs = iletı.executeQuery("SELECT * FROM ogrenciler ORDER BY ogr_adi");
            System.out.println("No\t\tAdı\t\tSoyadı\t\tSınıfı\t\tCinsiyeti\t\t");
            System.out.println("-----");

            while(rs.next())
            {
                System.out.println(rs.getString("ogr_okulno")+"\t\t"+rs.getString("ogr_adi")+"\t\t"
                +rs.getString("ogr_soyadi")+"\t\t"+rs.getString("ogr_sinifi")+"\t\t"+rs.getString("ogr_cinsiyeti"));
            }
            rs.close();
            bag.close();
        }
        catch(Exception ex)
        {
            System.out.println(ex.toString());
        }
    }
}
```

JAVA ile Access Bağlantısı

Kod satırlarını açıklamak gerekirse;

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

Hangi Driver'ı kullanacağımızı seçiyoruz.

```
Connection bag = DriverManager.getConnection("jdbc:odbc:okul");
```

Bağlantımızı oluşturuyoruz. Satırın sonunda yazan «okul» a dikkat ediniz....

```
Statement iletı = bag.createStatement();
```

SQL deyimini yürütebilmek ve üretilen sonuçları alabilmek için Statement nesnesini oluşturuyoruz.

```
ResultSet rs = iletı.executeQuery("SELECT * FROM ogrenciler ORDER BY ogr_adi");
```

SQL sorgumuzu yazıp üretilen sonuçların tutulduğu ResultSet nesnesini ekliyoruz.

JAVA ile Access Bağlantısı

```
while(rs.next())
```

```
{  
System.out.println(rs.getString("ogr_okulno")+"\t\t"+rs.getString("ogr_adi")+"\t\t"  
+rs.getString("ogr_soyadi")+"\t\t"+rs.getString("ogr_sinifi")+"\t\t"+rs.getString("ogr_  
_cinsiyeti"));  
}
```

Rs nesnemizin içinde veri olduğu sürece, veritabanından gelen bu verileri ekrana yazdırmasını sağlıyoruz.

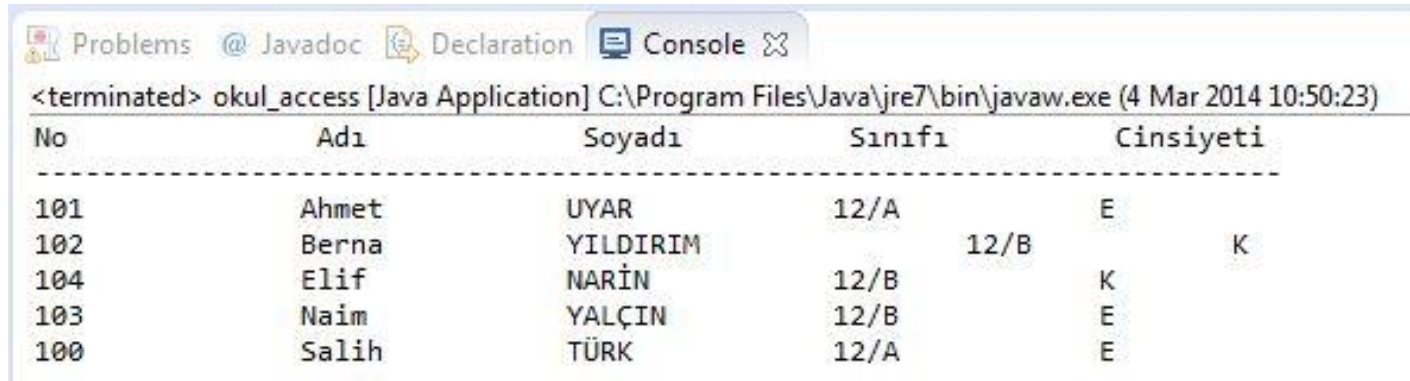
```
rs.close();
```

```
bag.close();
```

Son olarak ResultSet ve Connection nesnemizi kapatıyoruz.

JAVA ile Access Bağlantısı

Projemizin ekran çıktısı aşağıdaki gibi olur...



```
<terminated> okul_access [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (4 Mar 2014 10:50:23)
-----
No           Adı          Soyadı       Sınıfı       Cinsiyeti
-----
101         Ahmet       UYAR        12/A         E
102         Berna       YILDIRIM    12/B         K
104         Elif        NARİN       12/B         K
103         Naim        YALÇIN      12/B         E
100         Salih       TÜRK        12/A         E
```

Proje dosyasını «**KAYNAK**» klasörünün içindeki «**okul_access**» adlı klasörde bulabilirsiniz....

JAVA ile SQL Veritabanı Bağlantısı

İlk olarak «**okul**» adında bir veritabanı dosyası oluşturuyoruz. Ardından «**ogretmenler**» ve «**ogrenciler**» adlı 2 tablo oluşturuyoruz.

Column Name	Data Type	Allow Nulls
ogr_okulno	int	<input type="checkbox"/>
ogr_adi	varchar(50)	<input checked="" type="checkbox"/>
ogr_soyadi	varchar(50)	<input checked="" type="checkbox"/>
ogr_sinifi	varchar(50)	<input checked="" type="checkbox"/>
ogr_cinsiyeti	varchar(50)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Column Name	Data Type	Allow Nulls
ogrt_tc	varchar(50)	<input type="checkbox"/>
ogrt_adi	varchar(50)	<input checked="" type="checkbox"/>
ogrt_soyadi	varchar(50)	<input checked="" type="checkbox"/>
ogrt_cinsiyeti	varchar(50)	<input checked="" type="checkbox"/>
ogrt_alani	varchar(50)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

SQL Veritabanı Oluşturulması

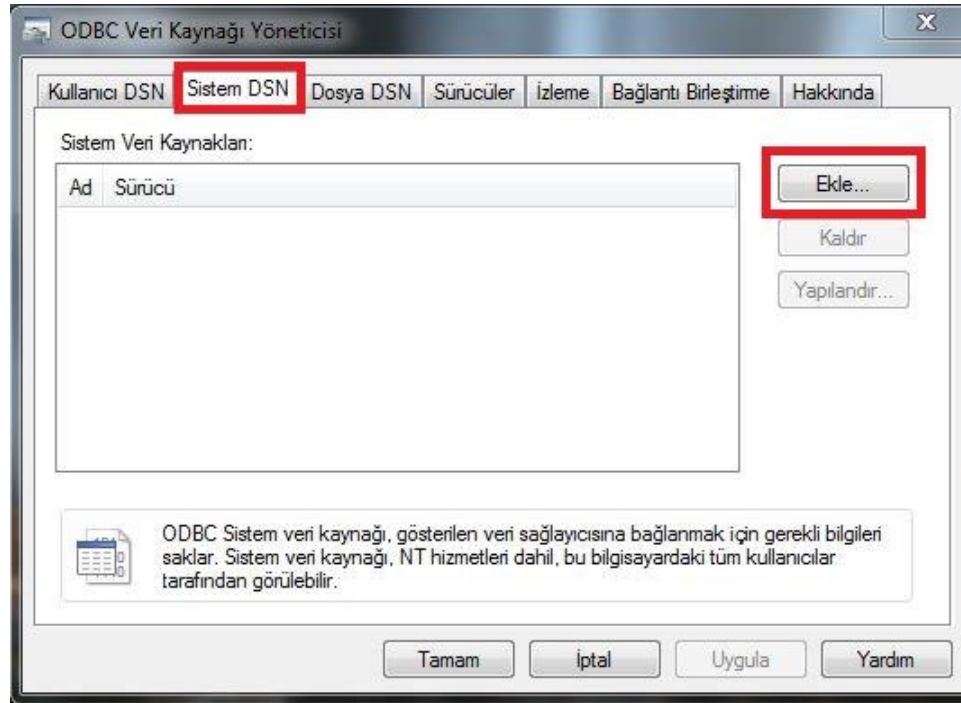
Son olarak tablolarımıza verilerimizi giriyoruz...

oqr_okulno	oqr_adi	oqr_soyadi	oqr_sinifi	oqr_cinsiyeti
100	Salih	TÜRK	12/A	E
101	Ahmet	UYAR	12/A	E
102	Berna	YILDIRIM	12/B	K
103	Naim	YALÇIN	12/B	E
104	Elif	NARİN	12/B	K
* NULL	NULL	NULL	NULL	NULL

oqrt_tc	oqrt_adi	oqrt_soyadi	oqrt_cinsiyeti	oqrt_alani
148314569	Merve	KAFALI	K	FİZİK
169147321	Fatma	TOĞCU	K	KİMYA
197342671	Tufan	TÜRKOĞLU	E	MATEMATİK
627816715	Mehmet	ŞEKER	E	FİZİK
* NULL	NULL	NULL	NULL	NULL

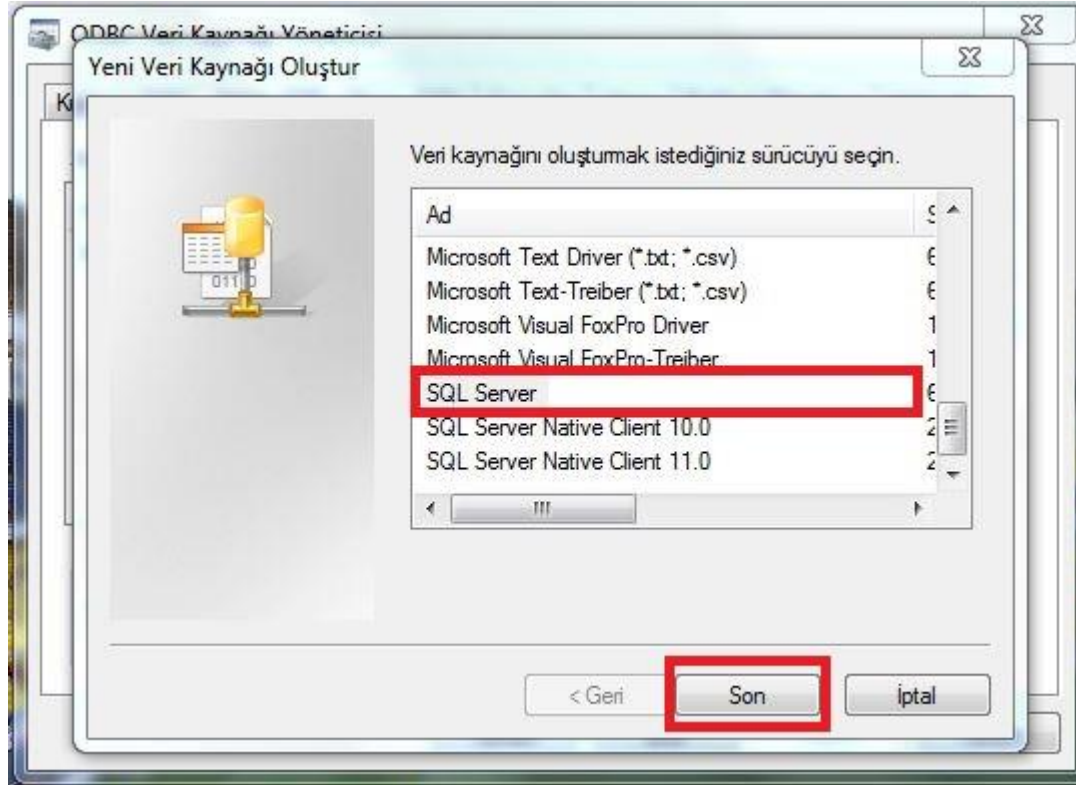
Kullanılacak Veritabanının Seçilmesi ve ODBC Bağlantısı (SQL Server Veritabanı İçin)

İlk önce **Denetim Masası > Yönetimsel Araçlar > Veri Kaynakları (ODBC)** seçilir. ODBC Yönetici ekranından **Sistem DSN** menüsünden **Ekle** düğmesine tıklanır.



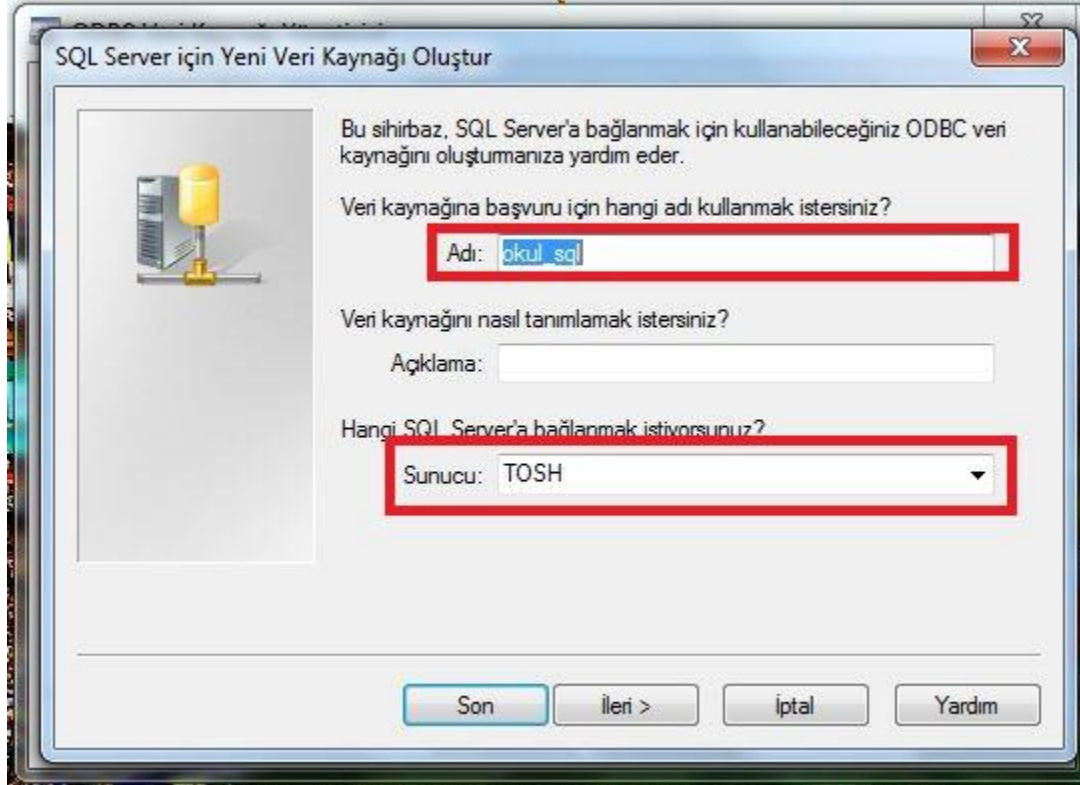
Kullanılacak Veritabanının Seçilmesi ve ODBC Bağlantısı (SQL Server Veritabanı İçin)

Ardından gelen ekrandan SQL Server seçeneği seçilir ve Son butonuna tıklanılır.



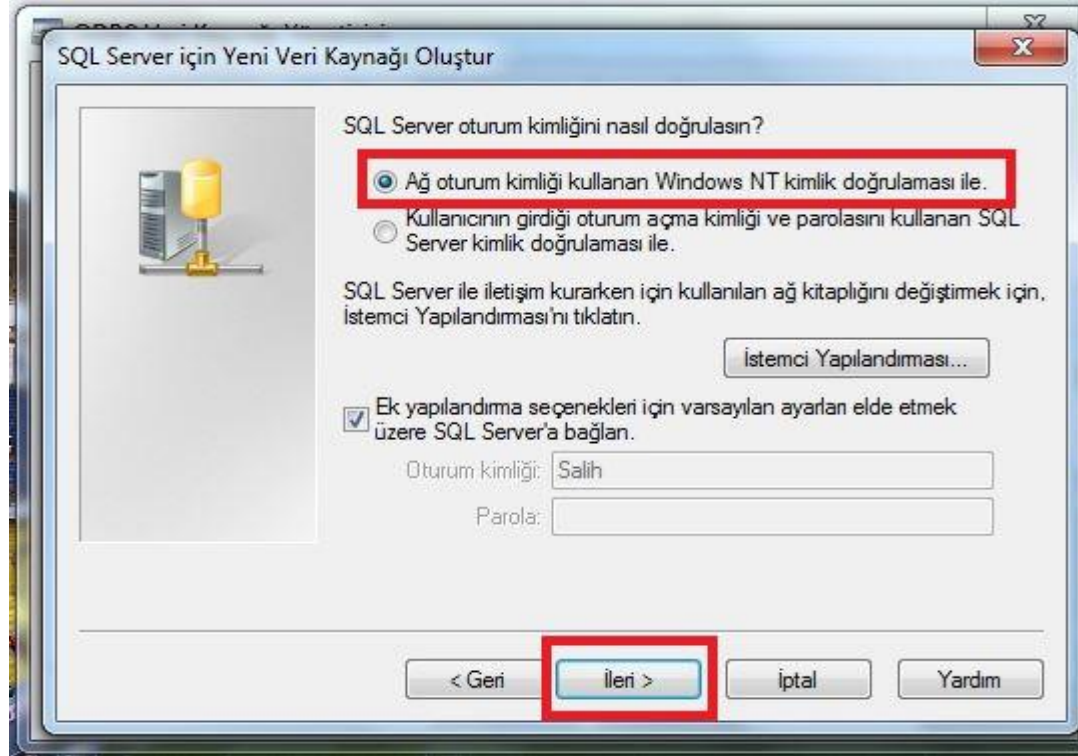
Kullanılacak Veritabanının Seçilmesi ve ODBC Bağlantısı (SQL Server Veritabanı İçin)

Sıradaki sayfada bağlantımıza herhangi bir isim veriyoruz ve kullanacağımız Server'ı seçiyoruz.



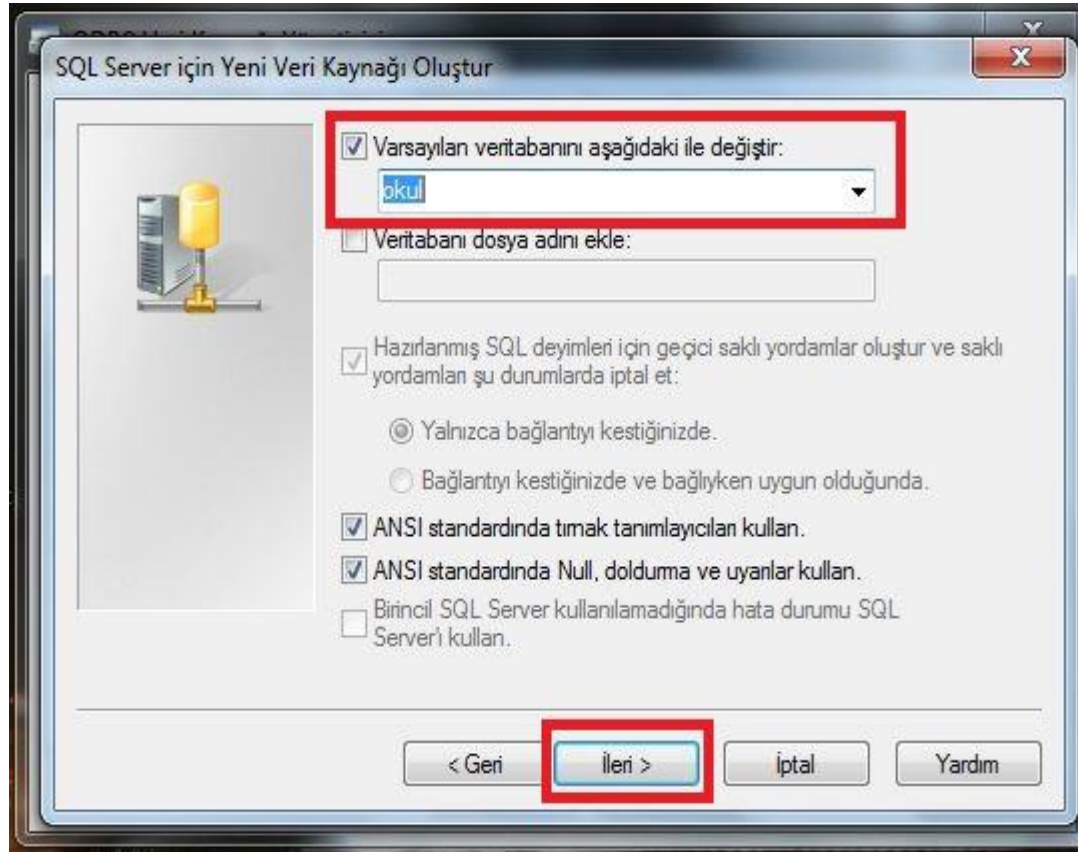
Kullanılacak Veritabanının Seçilmesi ve ODBC Bağlantısı (SQL Server Veritabanı İçin)

Bir sonraki seçenekte Server'a Windows oturumu ile mi yoksa şifre ile mi erişeceğimizi soruyor. Biz Windows oturumunu tercih ediyoruz.



Kullanılacak Veritabanının Seçilmesi ve ODBC Bağlantısı (SQL Server Veritabanı İçin)

Sıradaki adımda hangi veritabanına bağlanacağımızı seçiyoruz.



JAVA ile SQL Bağlantısı

ODBC bağlantısı tamamlandıktan sonra, «**okul_sqlserver**» adında proje oluşturulur. Ardında yine aynı adda class proje dosyasına eklenir. Daha sonra SQL işlemleri için gerekli olan «**java.sql**» kütüphanesi projeye import edilir.

```
okul_sqlserver ▶ src ▶ okul_sqlserver ▶ okul_sqlserver ▶ main(String[]): void
package okul_sqlserver;

import java.sql.*;

public class okul_sqlserver {

    public static void main(String[] args)
    {

        try
        {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection bag = DriverManager.getConnection("jdbc:odbc:okul_sql");

            Statement iletisi = bag.createStatement();
            ResultSet rs = iletisi.executeQuery("SELECT * FROM ogretmenler WHERE ogretmen_alani='FİZİK'");
            System.out.println("TC\tAdı\tSoyadı\tCinsiyeti\tAlanı");
            System.out.println("-----");

            while(rs.next())
            {
                System.out.println(rs.getString("ogretmen_tc")+"\t"+rs.getString("ogretmen_adi")+"\t"+
rs.getString("ogretmen_soyadi")+"\t"+rs.getString("ogretmen_cinsiyeti")+"\t"+rs.getString("ogretmen_alani"));
            }
            rs.close();
            bag.close();
        }
        catch(Exception ex)
        {
            System.out.println(ex.toString());
        }
    }
}
```

JAVA ile SQL Bağlantısı

Kod satırlarını açıklamak gerekirse;

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

Hangi Driver'ı kullanacağımızı seçiyoruz.

```
Connection bag = DriverManager.getConnection("jdbc:odbc:okul_sql");
```

Bağlantımızı oluşturuyoruz. Satırın sonunda yazan «okul_sql» e dikkat ediniz....

```
Statement ileti = bag.createStatement();
```

SQL deyimini yürütebilmek ve üretilen sonuçları alabilmek için Statement nesnesini oluşturuyoruz.

```
ResultSet rs = ileti.executeQuery("SELECT * FROM ogretmenler WHERE  
ogrt_alani='FİZİK'");
```

SQL komutumuzu yazıp üretilen sonuçların tutulduğu ResultSet nesnesini ekliyoruz.

JAVA ile SQL Bağlantısı

```
while(rs.next())  
{  
System.out.println(rs.getString("ogrt_tc")+"\t\t"+rs.getString("ogrt_adi")+"\t\t"  
+rs.getString("ogrt_soyadi")+"\t\t"+rs.getString("ogrt_cinsiyeti")+"\t\t"+rs.getString(  
"ogrt_alani"));  
}
```

Rs nesnemizin içinde veri olduğu sürece, veritabanından gelen bu verileri ekrana yazdırmasını sağlıyoruz.

```
rs.close();  
bag.close();
```

Son olarak ResultSet ve Connection nesnemizi kapatıyoruz.

JAVA ile SQL Bağlantısı

Projemizin ekran çıktısı aşağıdaki gibi olur...



```
<terminated> okul_sqlserver [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (4 Mar 2014 11:34:01)
TC          Adı          Soyadı       Cinsiyeti    Alanı
-----
148314569   Merve       KAFALI      K            FİZİK
627816715   Mehmet     ŞEKER      E            FİZİK
```

*Proje dosyasını «**KAYNAK**» klasörünün içindeki «**okul_sqlserver**» adlı klasörde bulabilirsiniz....*

JAVA ile MySQL Baęlantısı

Öncelikle Mysql ile bağlanmak için gerekli olanlar;

1 - Lokalde çalışacaksak sistemimizde MySQL kurulu olmalıdır. Uzak sunucuda çalışacaksak, veritabanlarının bulunduğu sunucunun IP adresi gerekmektedir. Uzak sunucuda çalışmak için, sunucunun size izin vermesi gerekebilir. Biz lokalde yani kendi bilgisayarımızda bulunan veritabanından veri çekeceğiz.

JAVA ile MySQL Bağlantısı

2 - Java ile veritabanı bağlantısı yapabilmek için driver gereklidir. MySQL için MySQL Connector gereklidir. Gerekli Driver'ı

<http://dev.mysql.com/downloads/connector/j/>

adresinden indirebilirsiniz.

JAVA ile MySQL Bağlantısı

Karşımıza aşağıdaki ekran çıkacaktır. Çıkan 2 dosyadan her hangi birini indirebiliriz. Sadece sıkıştırma biçimleri farklı olduğu için farklı boyuttalar. Aslında ikisi de aynı dosyaları içermektedir.



Generally Available (GA) Releases

Connector/J 5.1.29

Select Platform:
Platform Independent

Looking for previous GA versions?

Platform Independent (Architecture Independent), Compressed TAR Archive (mysql-connector-java-5.1.29.tar.gz)	5.1.29	3.4M	Download
Platform Independent (Architecture Independent), ZIP Archive (mysql-connector-java-5.1.29.zip)	5.1.29	3.7M	Download

MD5: 58d44ae9d20fe86bba321640ea781c53 | Signature

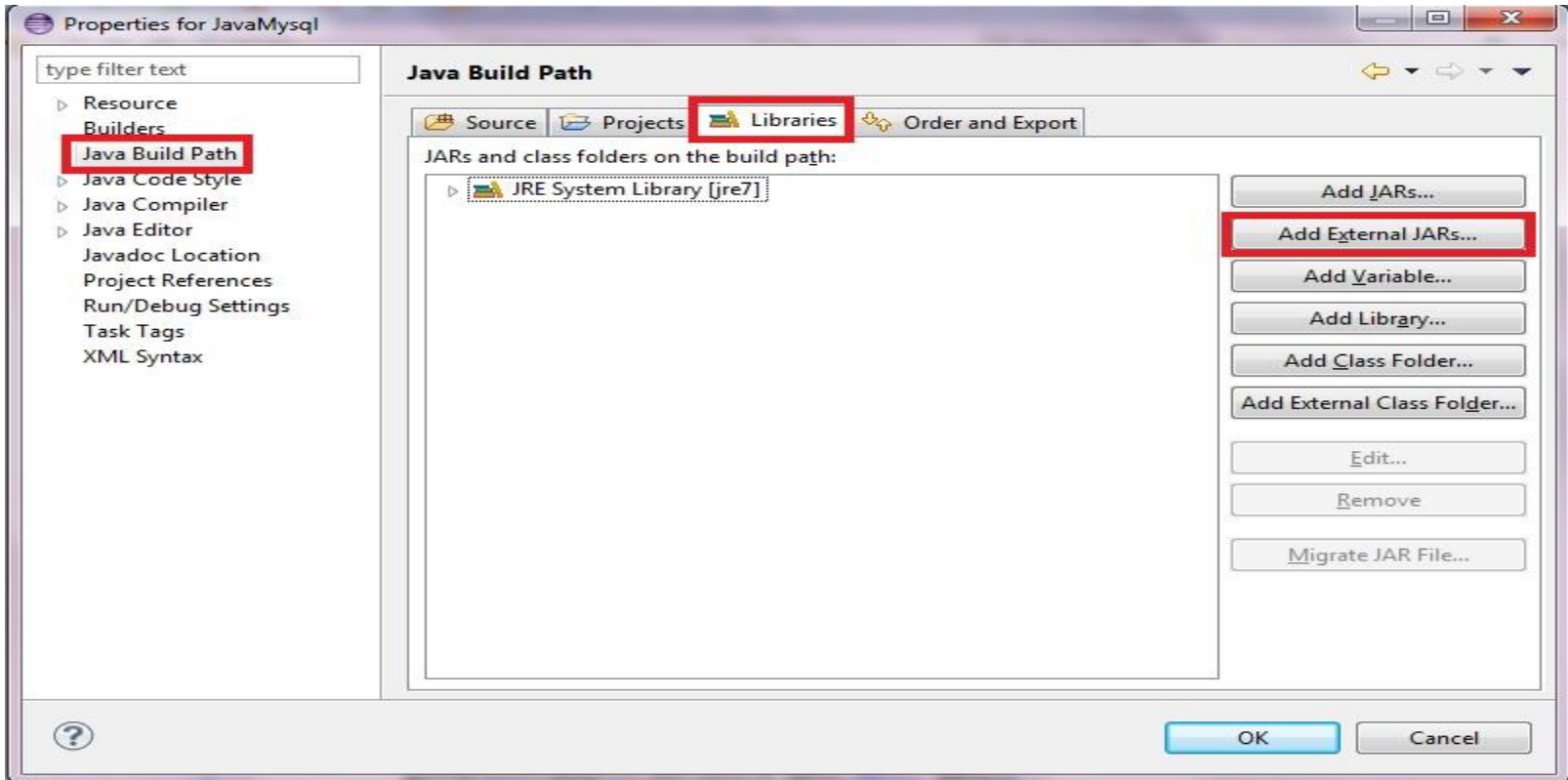
MD5: baeb34fe2dc21079de1bbcfe75ffe882 | Signature

We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

İndirdikten sonra yeni bir **Java Projesi** oluşturacağız ve **mysql-connector-java-5.1.29-bin.jar** dosyasını projeye ekleyeceğiz. Bu işlem için projemize sağ tıklayıp **Properties**'i seçelim.

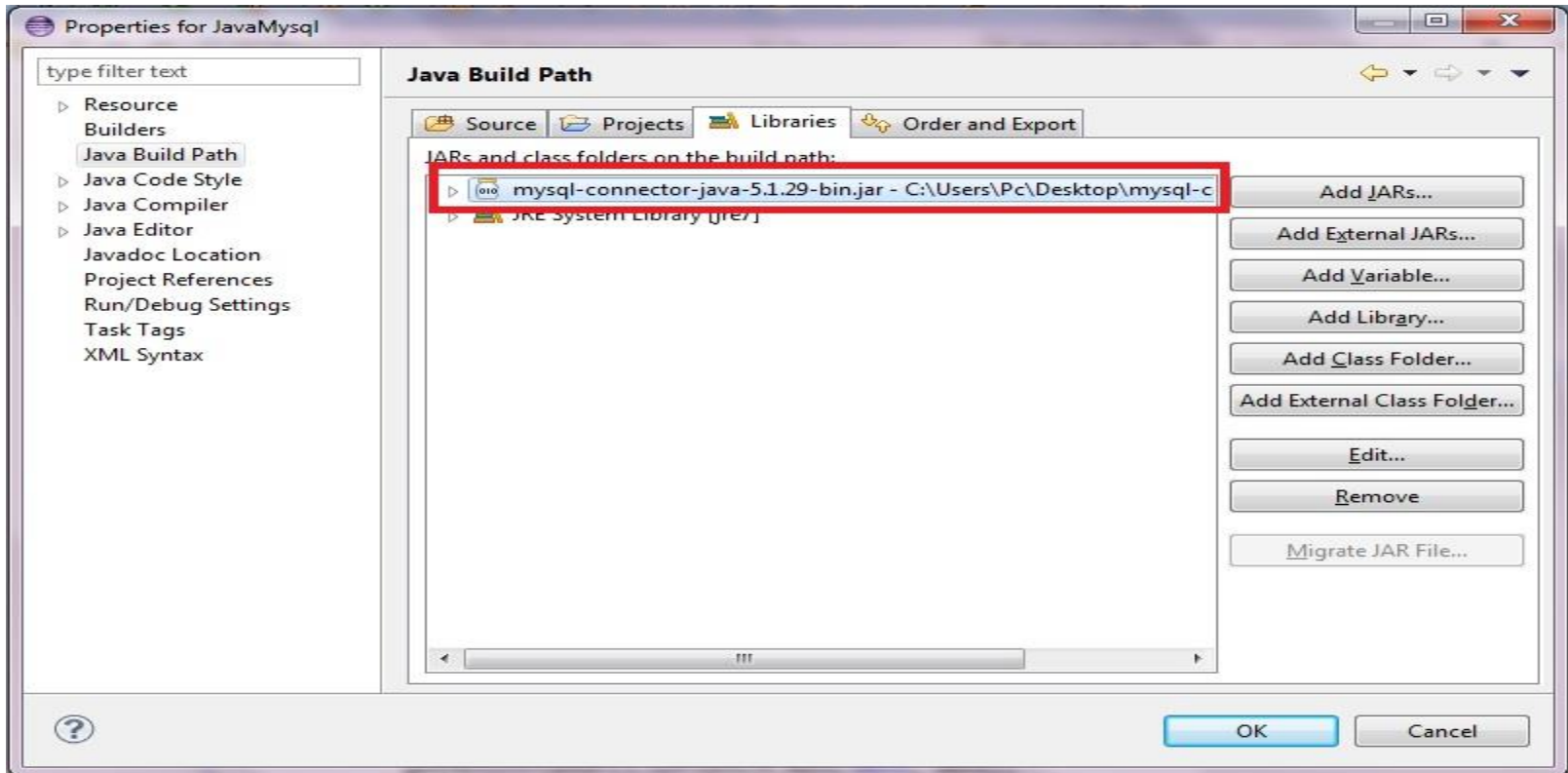
JAVA ile MySQL Bağlantısı

Properties ekranına geldikten sonra sol menüden **Java Build Path**'i ardından **Libraries** sekmesini açıyoruz ve **Add External Jars** butonuna basıyoruz.



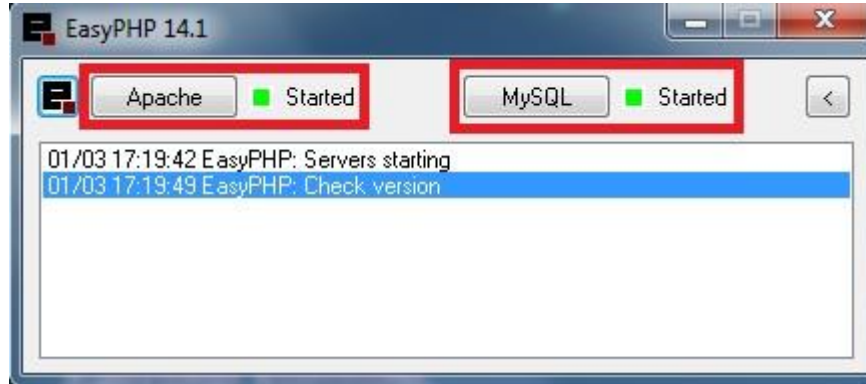
JAVA ile MySQL Bağlantısı

Add External JARs butonuna bastıktan sonra *mysql-connector-java-5.1.29-bin.jar* dosyamızı seçip projemize ekliyoruz.



JAVA ile MySQL Bağlantısı

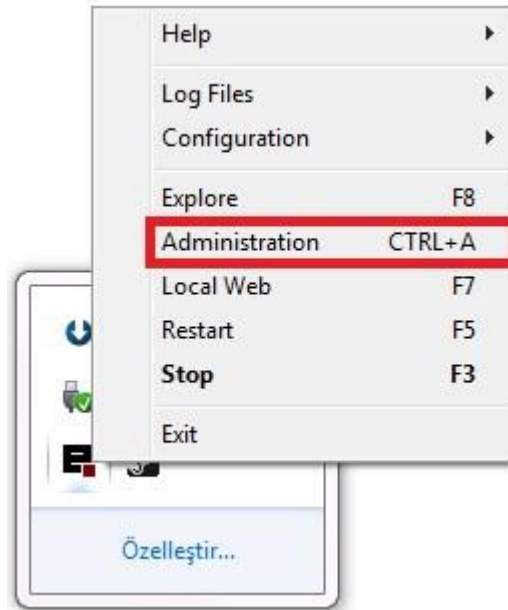
Başta belirttiğim gibi veritabanına bağlanabilmemiz için sunucumuzda MySQL kurulu olmalıdır. Bunu kolay yoldan kurabilmemiz için bilgisayarımıza herhangi bir sunucu programı (Easyphp,XAMPP,Wamp Server vb.) kurabiliriz. Ben Easyphp kurdum.



Gördüğünüz gibi hem sunucu hem de MySQL sorunsuz bir şekilde çalışıyor.

JAVA ile MySQL Bağlantısı

Kurulumu bitirip programı çalıştırdıktan sonra MySQL sayfamıza girip bir veritabanı oluşturmamız gerekiyor. Bunun için sağ altta simge durumunda ki EasyPHP simgesine sağ tıklayıp Administration yazısına tıklamamız gerekiyor.



JAVA ile MySQL Bağlantısı

Administration yazısına tıkladıktan sonra tarayıcıda açılan sayfada ki resimde ki kısımdan open diyerek veritabanı oluşturup düzenleyebileceğimiz MySQL sayfasına erişebiliriz.

MODULES ? recommended modules



MySQL Administration : PhpMyAdmin 4.1.4 ?

open

JAVA ile MySQL Bağlantısı

Örneğin Market adında bir veritabanı oluşturalım. Yeni bir veritabanı oluşturmak için gerekli adımlar;



The screenshot shows the phpMyAdmin interface for a MySQL server (Sunucu: 127.0.0.1). The 'Veritabanları' (Databases) tab is active. In the left sidebar, the 'Yeni' (New) button is highlighted with a red box, labeled '1. Yeni diyoruz'. The main area shows the 'Veritabanı oluştur' (Create database) form. The 'Veritabanı adı' (Database name) field contains 'Market' and is highlighted with a red box, labeled '2. Veritabanı adımızı yazıyoruz'. The 'Karakter seti' (Character set) dropdown is set to 'Karşılaştırma'. The 'Oluştur' (Create) button is highlighted with a red box, labeled '3. Oluştur diyoruz.'. Below the form, a warning message states: 'Not: Buradaki veritabanı istatistiklerini etkinleştirmek web sunucusu ile MySQL sunucusu arasında yüksek trafiğe yol açabilir.' (Note: Enabling statistics for the database here may cause high traffic between the web server and the MySQL server.) A table below shows existing databases: 'asdasd' with character set 'latin1_swedish_ci' and 'manhattan' with 'utf8_turkish_ci'. Both are marked as 'Kopya edildi' (Copied) and have 'Yetkileri kontrol et' (Check permissions) links. The table summary shows 'Toplam: 2' (Total: 2) databases. At the bottom, there are options for 'Tümünü Seç' (Select all), 'Seçilileri: Kaldır' (Remove selected), and 'İstatistikler etkin' (Statistics enabled).

1. Yeni diyoruz

2. Veritabanı adımızı yazıyoruz

3. Oluştur diyoruz.

Not: Buradaki veritabanı istatistiklerini etkinleştirmek web sunucusu ile MySQL sunucusu arasında yüksek trafiğe yol açabilir.

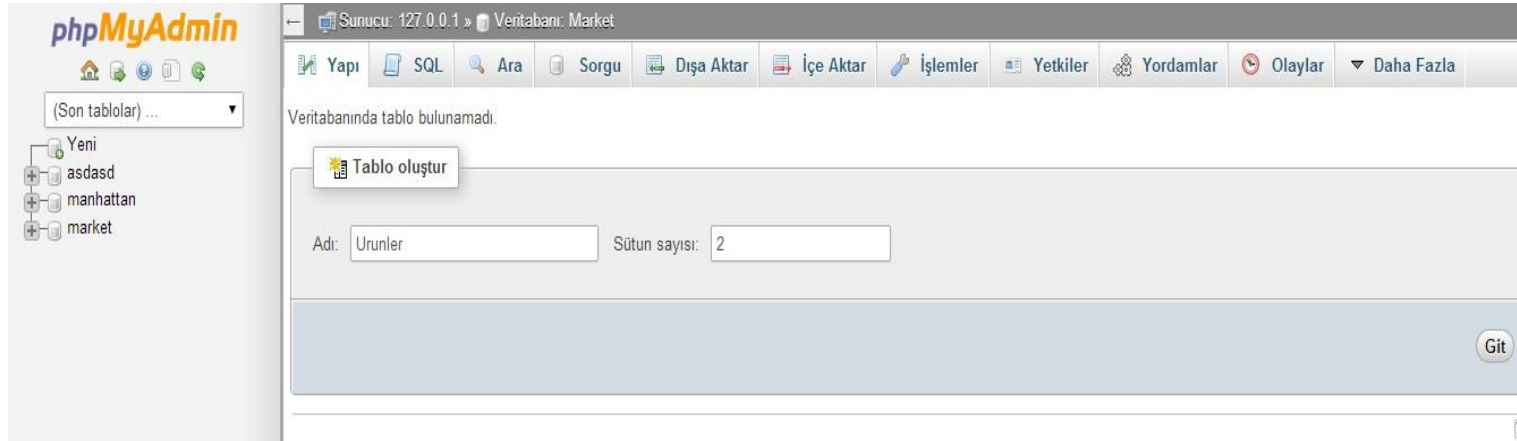
Veritabanı	Karşılaştırma	Master kopya etme
asdasd	latin1_swedish_ci	✓ Kopya edildi Yetkileri kontrol et
manhattan	utf8_turkish_ci	✓ Kopya edildi Yetkileri kontrol et
Toplam: 2	latin1_swedish_ci	

↑ Tümünü Seç Seçilileri: [Kaldır](#)

• İstatistikler etkin

JAVA ile MySQL Bağlantısı

Veritabanı oluşturduktan sonra veritabanımıza tıklıyoruz ve karşımıza tablo oluşturma ekranı çıkıyor. Oluşturacağımız tablo adı **Urunler** olduğu için Adı kısmına **Urunler** yazıyoruz ve **UrunID** ve **UrunAdi** alanlarından oluşacağı için Sütun sayısı kısmına 2 yazıyoruz ve git diyoruz.



JAVA ile MySQL Bağlantısı

Git dedikten sonra karşımıza gelen ekranda ki resimde işaretli yerleri görüldüğü şekilde dolduruyoruz ve kaydet diyoruz.

Adı	Türü	Uzunluk/Değerler	Varsayılan	Karşılaştırma	Öznitelikler	Boş	İndeks	A_1
UrunID	INT	11	Yok			<input type="checkbox"/>	PRIMARY	<input checked="" type="checkbox"/>
UruAdi	VARCHAR	255	Yok			<input type="checkbox"/>	---	<input type="checkbox"/>

Ekle dedikten sonra karşımıza şöyle bir ekran gelecektir. Gelen ekrandaki urunler yazısına tıklıyoruz.

Tablo	Eylem	Satır	Türü	Karşılaştırma	Boyut	Ek Yük
<input type="checkbox"/> urunler	Gözet Yapı Ara Ekle Boşalt Kaldır	0	MyISAM	latin1_swedish_ci	1 KiB	-
1 tablo	Toplam	0	MyISAM	latin1_swedish_ci	1 KiB	0 B

JAVA ile MySQL Baęlantısı

Tıkladıktan sonra size **MySQL boş bir sonuç kümesi döndürdü (yani sıfır satır)**. Gibi bir uyarı verecektir. Bu tablomuzda henüz herhangi bir kayıt olmadığı için çıkan uyarıdır. Tablomuzda kayıt eklemek için üst menüde bulunan Ekle yazısına tıklıyoruz.

JAVA ile MySQL Bağlantısı

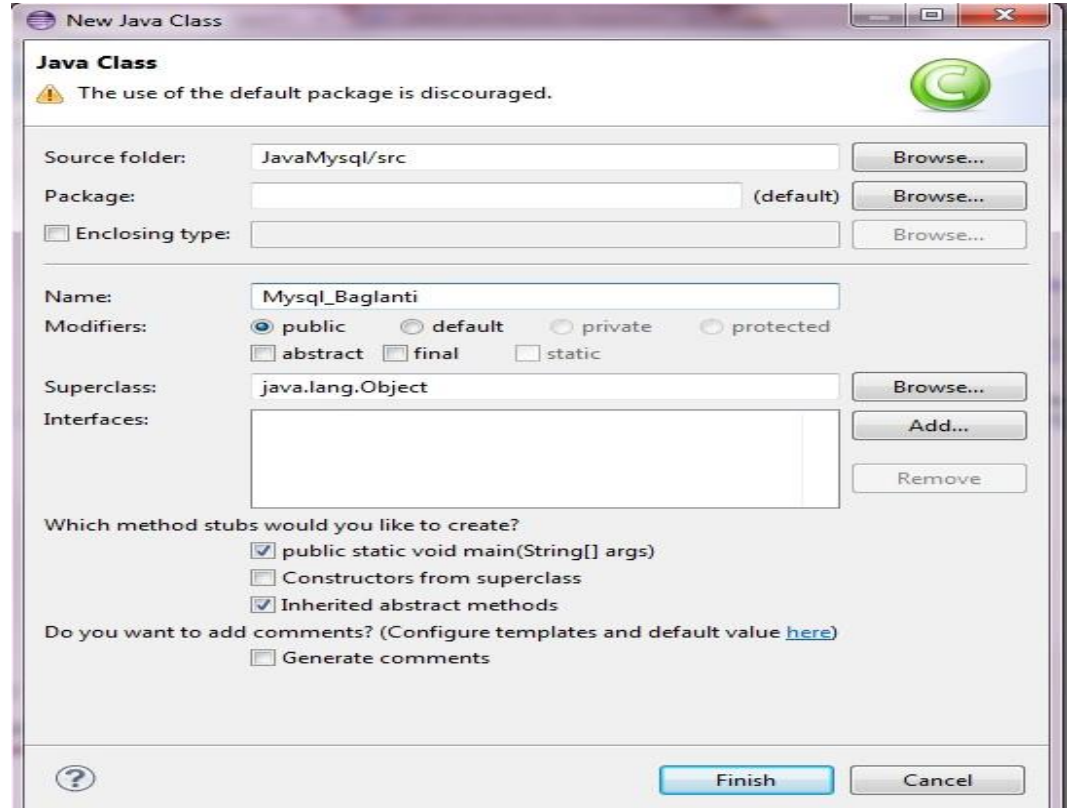
Ekle dedikten sonra karşımıza çıkan ekranı resimde ki gibi dolduruyoruz ve ilk Git butonuna değil ikinci Git butonuna tıklıyoruz.

The screenshot shows a database management interface with a menu bar at the top containing: Gözet, Yapı, SQL, Ara, Ekle, Dışa Aktar, İçe Aktar, and Yetkile. Below the menu bar, there are two identical forms for adding data. Each form has a header row with columns: Sütun, Türü, İşlev, Boş, and Değer. The first form has 'UrunID' with type 'int(11)' and 'UruAdi' with type 'varchar(255)'. The 'UruAdi' field contains the text 'Kola'. The second form has 'UrunID' with type 'int(11)' and 'UruAdi' with type 'varchar(255)'. The 'UruAdi' field contains the text 'Meyve Suyu'. Both forms have a 'Git' button at the bottom right. There is also a checkbox labeled 'Yoksay' (Ignore) between the two forms.

JAVA ile MySQL Bağlantısı

Git dedikten sonra yine Ekle diyerek Kola ve Meyve Suyu yerine Peynir ve Zeytin yazarak yine Git diyoruz böylelikle tablomuza 4 tane kayıt ekliyoruz.

Artık MySQL ile olan kısımları bitirdik bu nedenle Java Projemize geçebiliriz. Öncelikle oluşturduğumuz Java Projesine Mysql_Baglanti adında yeni bir class ekliyoruz.



JAVA ile MySQL Bağlantısı

Öncelikle projemize gerekli olan kütüphaneleri import ediyoruz.

A screenshot of an IDE window showing a Java file named '*Mysql_Bagl...'. The code includes four import statements for JDBC classes: java.sql.DriverManager, com.mysql.jdbc.Connection, java.sql.ResultSet, and com.mysql.jdbc.Statement. Below the imports is a public class named 'Mysql_Baglanti' with a public static void main method that takes a String[] args parameter. The main method body is currently empty.

```
import java.sql.DriverManager;
import com.mysql.jdbc.Connection;
import java.sql.ResultSet;
import com.mysql.jdbc.Statement;

public class Mysql_Baglanti {

    public static void main(String[] args) {

    }

}
```

JAVA ile MySQL Bağlantısı

Daha sonra Class dosyamızda Baglan adında bir fonksiyon oluşturuyoruz ve aşağıda ki kodları yazıyoruz.



```
public static void Baglan(){
    try
    {
        String connectionString = "jdbc:mysql:///market";
        Class.forName("com.mysql.jdbc.Driver");
        Connection baglanti = (Connection) DriverManager.getConnection(connectionString,"root","");
        try {
            String SQL = "SELECT * FROM urunler";
            Statement durum = (Statement) baglanti.createStatement();
            ResultSet rs = (ResultSet) durum.executeQuery(SQL);

            while (rs.next()) {
                System.out.println(rs.getString("UrunID") + " " + rs.getString("UruAdi"));
            }
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

JAVA ile MySQL Bağlantısı

Kodları açıklamak gerekirse ;

String connectionString = "jdbc:mysql:///market"; satırında bağlanacağımız sunucuyu ve veritabanı adımızı belirtiyoruz.

Class.forName("com.mysql.jdbc.Driver");
satırında kullanacağımız driverı yani indirdiğimiz Driverı belirtiyoruz.

Connection baglanti = (Connection)
DriverManager.getConnection(connectionString,"root","");

*Satırında bağlantımızı kuruyoruz. Parametre olarak 3 parametre alır. Bunlar : **Bağlanılacak Sunucu, Kullanıcı Adı, Şifre** şeklindedir. Localhostta %99 olarak kullanıcı adı **root**tur ve şifre yoktur.*

JAVA ile MySQL Bağlantısı

```
String SQL = "SELECT * FROM urunler";
```

Satırında çalıştıracığımız SQL sorgusunu yazıyoruz.

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

Satırında çalıştırdığımız sorgudan sonuçları alabilmemiz için gerekli nesneyi tanımlıyoruz.

```
ResultSet rs = (ResultSet) durum.executeQuery(SQL);
```

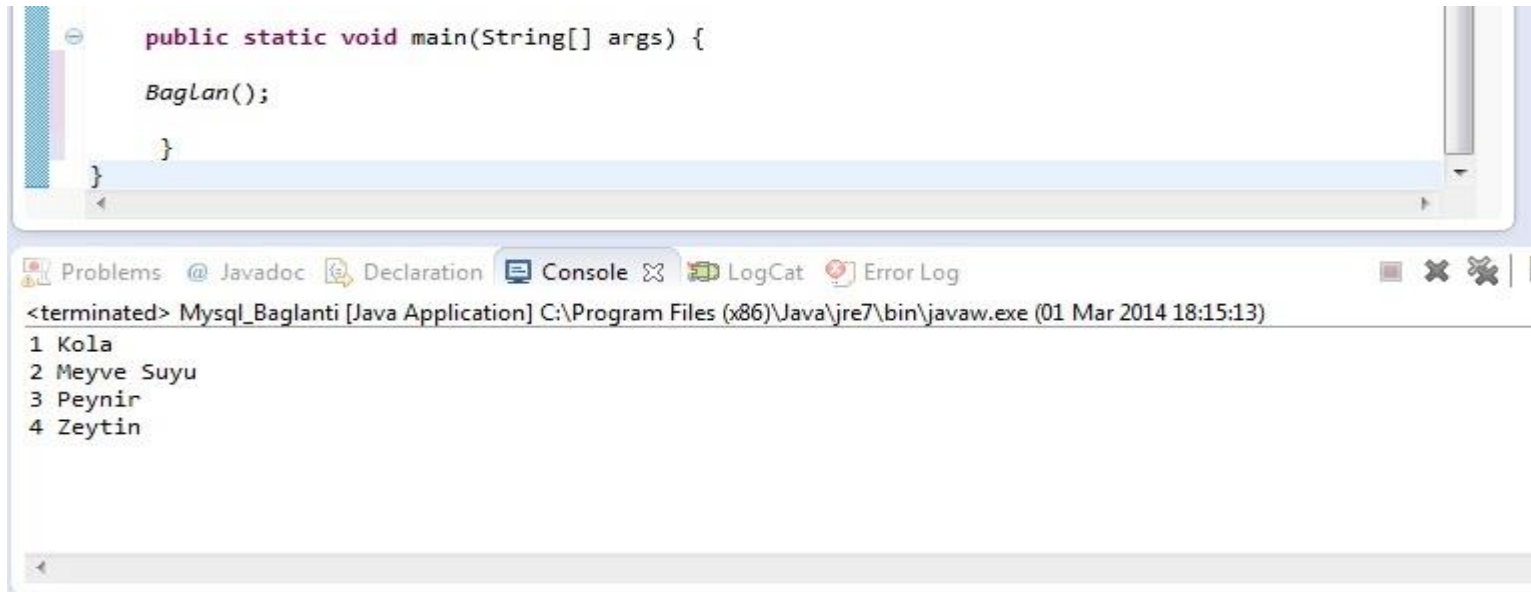
Satırında SQL sorgumuzu çalıştırıyoruz ve sonuçları ResultSet nesnemize aktarıyoruz.

```
while (rs.next()) {  
System.out.println(rs.getString("UrunID") + " " + rs.getString("UruAdi"));}
```

Son olarak while döngüsü ile kayıtlarımızı ekrana yazdırıyoruz.

JAVA ile MySQL Bağlantısı

Tüm işlemlerimizi Try-Catch bloğunda yaparak oluşabilecek bir hatada gerekli hata mesajımızı düzenli bir şekilde alıyoruz. Son Olarak Main metodumuzda oluşturduğumuz Baglan(); fonksiyonunu çağırıyoruz ve projemizi çalıştırarak kayıtları listeliyoruz.



```
public static void main(String[] args) {  
    Baglan();  
}
```

Problems @ Javadoc Declaration Console LogCat Error Log

<terminated> Mysql_Baglanti [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (01 Mar 2014 18:15:13)

- 1 Kola
- 2 Meyve Suyu
- 3 Peynir
- 4 Zeytin

Proje dosyasını «**KAYNAK**» klasörünün içinde bulunan «**JavaMysql**» adlı klasörden temin edebilirsiniz.