

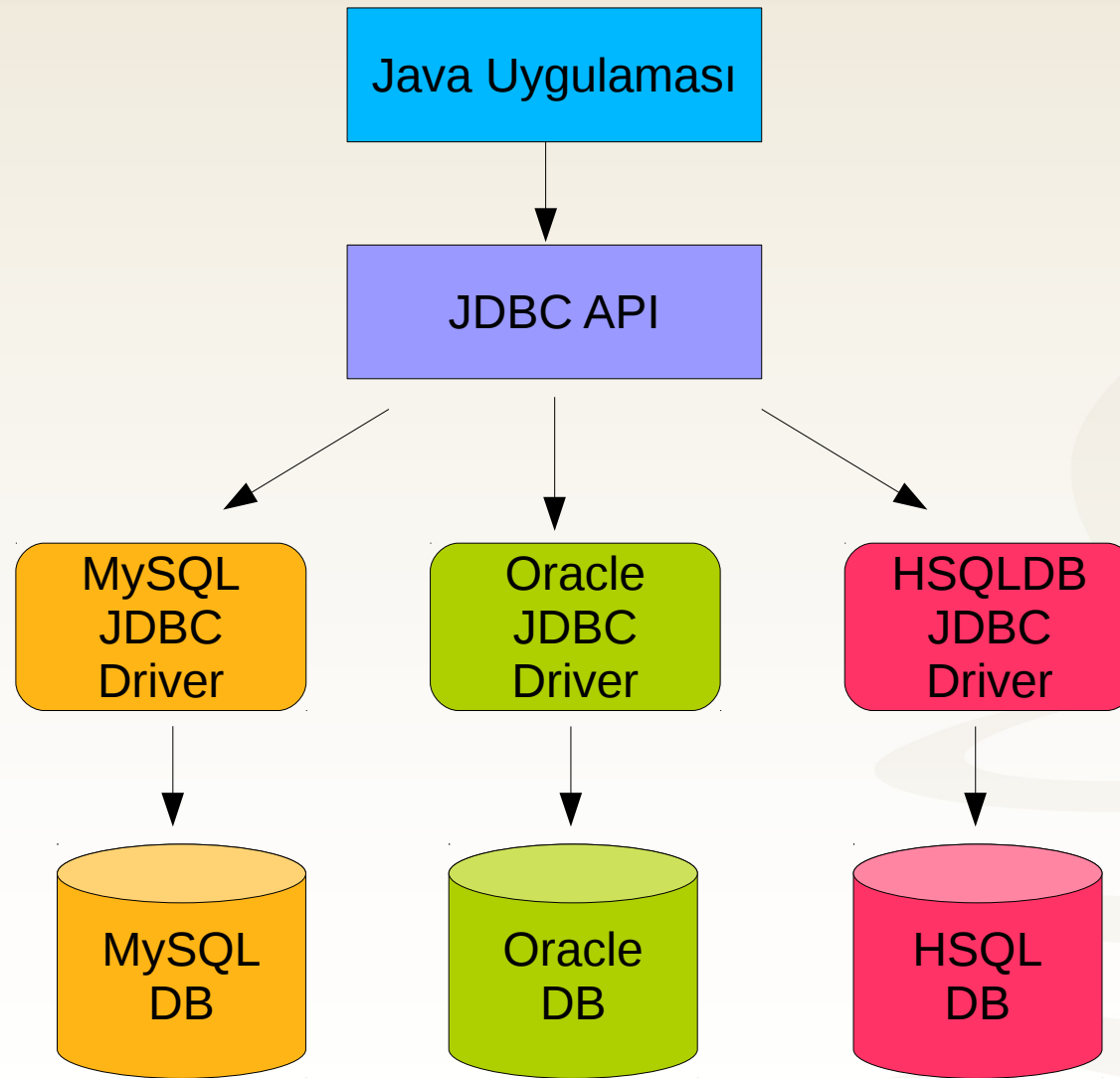
Java Database Connectivity (JDBC)



Java Database Connectivity (JDBC)

- Java uygulamalarında **ilişkisel veritabanları** (relational database) ile çalışmayı sağlar
- Farklı ilişkisel veritabanları için **ortak bir API** sunar
- JDBC vasıtası ile java uygulamaları içerisinde **SQL işlemleri** gerçekleştirilebilmektedir
- **SQL ifadeleri** ilişkisel veritabanlarına göre **değişebilir**
- Ancak veritabanına erişim, SQL ifadelerinin işletilmesi ve dönen sonuçların işlenmesi standarttır

JDBC Mimarisi



JDBC ile Çalışma Adımları

- JDBC driver'ın yüklenmesi

```
try {
    Class.forName("org.hsqldb.jdbcDriver");
} catch (ClassNotFoundException e) {
    throw new RuntimeException("Cannot load driver", e);
}
```



Uygulama içerisinde JDBC ile veritabanına bağlanmak için öncelikle veritabanına Özel driver sınıfının yüklenmesi gerekir

Her veritabanı için değişik JDBC driver sınıfları mevcuttur

Sisteme yüklenen JDBC driver sınıfları **DriverManager** tarafından yönetilir

JDBC ile Çalışma Adımları

■ Veritabanı bağlantısının oluşturulması

```

Connection connection = null;
try {

    connection =
    DriverManager.getConnection("jdbc:hsqldb:hsqldb://localhost", "sa", "");

} catch (SQLException e) {
    throw new RuntimeException("Cannot connect database", e);
}
    
```

JDBC url DB username DB password

Veritabanı bağlantısının oluşturulması için kullanılan yollardan birisi
DriverManager.getConnection() metodunu çağırma

DriverManager parametre olarak verilen JDBC url bilgisini kullanarak uygun
JDBC driver'ı tespit ederek veritabanı bağlantısını oluşturur

JDBC İle Çalışma Adımları

- Statement oluşturularak, SQL ifadesinin çalıştırılması

```
Statement stmt = null;
try {
    stmt = connection.createStatement();
    ResultSet rs = stmt.executeQuery("select id,birth_date,name from
pets");
} catch (SQLException e) {
    throw new RuntimeException("Cannot execute query", e);
} finally {
    try {
        stmt.close();
    } catch (Exception e) {
        //ignore...
    }
}
```

İşlem bitince Statement'ın kapatılması otomatik olarak ResultSet'in de kapatılmasını sağlayacaktır

JDBC ile Çalışma Adımları

■ Dönen sonuçların işlenmesi

while(rs.next()) {  ResultSet üzerinde iterate etmeyi sağlar

```
    int id = rs.getInt(1);
```

```
    Date date = rs.getDate("birth_date");
```

```
    String name = rs.getString("name");
```

```
    System.out.println("Current pet id :" + id + " name :" +  
name + " birth date :" + date);  
}
```

 Sütun index'leri 1'den başlar

 Değerlere erişirken sütun isimlerini kullanmak
Daha iyi bir pratiktir

 Sütunun değerine, SQL tipine karşılık gelen metodu kullanarak erişilir

JDBC İle Çalışma Adımları

- Olası hataların ele alınması ve bağlantıların sonlandırılması

```
try {
    //DB işlemleri try bloğu içerisinde gerçekleştirilir
} catch(SQLException e) {
    //Herhangi bir DB hatası catch bloğunda yakalanır
} finally {
    try {

        connection.close();

    } catch (Exception e) {
        //ignore...
    }
}
```

DB işlemleri sonlandığında finally bloğunda mutlaka veritabanı bağlantısı kapatılmalıdır

Try With Resources ile JDBC İşlemleri

- Java 7 ile gelen bu özellik sayesinde ResultSet, Statement ve Connection nesnelerinin **otomatik olarak kapatılmaları** sağlanabilir

```
try (Connection c = DriverManager.getConnection("jdbc:h2:mem:test","sa","")) {  
    try (CallableStatement stmt = c.prepareCall("call current_timestamp()")) {  
        try (ResultSet rs = stmt.executeQuery()) {  
            while(rs.next()) {  
                Timestamp timestamp = rs.getTimestamp(1);  
                System.out.println(timestamp);  
            }  
            // stmt kullanılarak baska islemler yapılabilir  
        }  
        // connection kullanılarak baska islemler yapılabilir  
    } catch (SQLException ex) {  
        throw new RuntimeException(ex);  
    }  
}
```

Veritabanı Hataları

- JDBC API metot çağrıları **SQLException** fırlatır
- **Checked exception**'dir
- Yakalanması veya metot imzasında belirtilmesi gerekir
- İçerisinde X/Open ANSI standardı **SQL state code** ve üretici spesifik **error code** bilgileri yer alır
- **SQLException.getSQLState()** ve **getErrorCode()** metotları ile bu bilgilere erişilebilir

SQLWarning

- Veritabanı işlemleri sırasında meydana gelen uyarılar **SQLWarning** nesneleri ile ifade edilir
- **SQLException**'dan türer
- **Connection**, **Statement**, **ResultSet** nesneleri **SQLWarning** üretebilir
- Exception gibi fırlatılmaz, bu nesneler üzerinde biriken uyarılar **getWarnings()** metodu ile erişilebilir
- Ardından **SQLWarning.getNextWarning()** ile sıradaki uyarı varsa elde edilebilir

JDBC URL'in Yapısı & Örnekler

- **jdbc:vendor-name:vendor-specific-part**
- jdbc:**oracle**:<driver-type>:user/password@<database>
 - jdbc:oracle:thin:scott/tiger@myhost:1521:orcl
 - jdbc:oracle:oci:scott/tiger@myhost:1521:orcl
- jdbc:**h2**:<connection-mode>:<connection-settings>
 - jdbc:h2:[file]:[<path>]<database-name>
 - jdbc:h2:mem:<database-name>
 - jdbc:h2:tcp://<server>[:<port>]/[<path>]/<database-name>

JDBC URL'in Yapısı & Örnekler

- `jdbc:mysql://[host1][:port1][,[host2][:port2]]...[/[database]][?propertyName=propertyValue1[&propertyName2=propertyValue2]...]`
 - `jdbc:mysql://localhost:3306/test?profileSQL=true`
 - `jdbc:mysql://master,slave1,slave2/test`
- `jdbc:postgresql:[//<host>][:<port>[/]<database>]`
 - `jdbc:postgresql:testdb`
 - `jdbc:postgresql://localhost:5432/testdb`

JDBC URL'in Yapısı & Örnekler

- `jdbc:sqlserver://[server[\instance]][:port]]`
`[:property=value[:property=value]]`
 - `jdbc:sqlserver://localhost;databaseName=testdb;integratedSecurity=true`
- `jdbc:db2://<server>[:<port>]/<database>`
 - `jdbc:db2://localhost/testdb`

DataSource ve Veritabanı Bağlantıları

- **DriverManager** kullanarak veritabanı bağlantısı oluşturmaya alternatiftir
- Benzer biçimde **connection** oluşturup, connection ile ilgili ayarlar yapmaya imkan tanır
- Ancak **DataSource** nesnesi genellikle uygulama dışında bir yerde oluşturulur ve yönetilir
- Bu çoğunlukla bir uygulama sunucusu olur ve DataSource nesnesi **JNDI**'a bind edilir

DataSource ve Veritabanı Bağlantıları

- Veritabanı üreticileri **JDBC driver**'ının yanında **DataSource implementasyonları** da sunarlar
- DataSource kullanımı sayesinde uygulamanın **JDBC driver'a bağımlılığı** tamamen ortadan kalkar
- Connection pooling ve distributed transaction yönetimi gibi **kabiliyetler** mümkün hale gelir

DataSource Kullanımı

```
BasicDataSource ds = new BasicDataSource();
ds.setDriverClassName("org.hsqldb.jdbcDriver");
ds.setUrl("jdbc:hsqldb:hsqldb://localhost");
ds.setUsername("sa");
ds.setPassword("");
```

```
InitialContext context = new InitialContext();
context.bind("jdbc/PooledDataSource", ds);
```

DataSource'u kullanan
Bölümün dışında
Herhangi bir yerde
DataSource nesnesi
Oluşturularak JNDI'a
Bind edilebilir

Bu işlem uygulama
Sunucularında çoğunlukla
Dekleratif olarak, bir
Takım konfigürasyon
Dosyaları ile yapılmaktadır

```
DataSource dataSource = (DataSource)
context.lookup("jdbc/PooledDataSource");

Connection c = dataSource.getConnection();
```

Uygulama içerisinde JNDI
Context'e erişerek DataSource
Nesnesine ismi ile lookup yapılır

Daha sonra DataSource nesnesi
Üzerinden veritabanı bağlantıları
Oluşturulabilir

PreparedStatement Kullanımı

- **Statement** sınıfından extend eder
- Statement'ın sağlayamadığı **kabiliyetleri** sunar
- SQL argümanlarının **dinamik** olarak verilmesini sağlar
- SQL ifadesini DB'ye göndererek çalıştırılması için **hazır** hale getirir
- Müteakip kullanımlarda SQL ifadesinin bu **hazırlık safhası**ndan geçmesine gerek kalmaz

PreparedStatement Kullanımı

```
PreparedStatement pstmt = null;  
try {  
    pstmt = connection.prepareStatement("update pets set name =  
? where id = ?");  
    pstmt.setString(1, "boncuk");  
    pstmt.setInt(2, 1);
```

```
    int updateCount= pstmt.executeUpdate();
```

```
    pstmt.setString(1, "maviş");  
    pstmt.setInt(2, 3);
```

SQL ifadesi farklı parametreler
ile tekrar tekrar çalıştırılabilir

```
    updateCount = pstmt.executeUpdate();  
} catch (SQLException e) {  
    throw new RuntimeException("Cannot perform update", e);  
} finally {  
    try {  
        pstmt.close();  
    } catch (Exception e) {  
        //ignore...  
    }  
}
```

PreparedStatement nesnesi kapatılana değin
SQL ifadesi hazır biçimde hafızada tutulur

CallableStatement Kullanımı

- **PreparedStatement** sınıfından extend eder
- **Stored procedure** ve stored function'lara erişmek için kullanılır
- Çağırılacak prosedür ve fonksiyonlara **input parametre** geçmeyi ve dönen **output değerleri** ele almayı sağlar

CallableStatement Kullanımı

PrepareCall metodu ile CallableStatement nesnesi oluşturulur

Input parametreler index'lerine göre set edilir

```
CallableStatement cstmt = connection.prepareCall("call upper(?)");  
cstmt.setString(1, "harezmi");  
cstmt.execute();  
ResultSet rs = cstmt.getResultSet();  
if(rs.next()) {  
    System.out.println(rs.getString(1));  
}
```

Eğer output parametreler söz konusu olursa bunlarda fonksiyon çalıştırılmadan evvel cstmt.registerOutParameter() metodu ile tanıtılmalıdır

Daha sonra bu output parametrelere tipine göre getXXX() metodu ile erişilebilir

JDBC İle Transaction Yönetimi

- Veritabanı transaction'ı yapılan **işlemlerin bir bütün halinde** DB'ye yansıtılmasını sağlar
- **Veri bütünlüğünü** korumaya yardımcı olur
- Birden fazla SQL işlemini **tek bir atomik unit** olarak ele almayı sağlar
- Atomik unit içerisinde gerçekleştirilen işlemlerin ya hepsi beraber **başarılı** sonlanacaktır
- Ya da içlerinden herhangi birisinin başarısız olması, bütün atomik unit'i **başarısız** kılacaktır

JDBC ile Transaction Yönetimi

```
try {  
    connection.setAutoCommit(false);  
    stmt.executeUpdate("update pets set name = 'maviş' where id =  
5");  
    stmt.executeUpdate("delete from vets where id = 2");  
    connection.commit();  
} catch (SQLException e) {  
    try {  
        connection.rollback();  
    } catch (SQLException e2) {  
        //ignore...  
    }  
    throw new RuntimeException("Cannot perform operation", e);  
}
```

Autocommit false, DB transaction'ı başlatır

Veritabanı işlemleri transaction commit edilene değin kalıcı hale gelmez

Herhangi bir hata söz konusu olduğunda veritabanı transaction'ı rollback edilerek o ana kadar yapılan SQL işlemlerinin göz ardı edilmesi sağlanabilir


Transaction Isolation Düzeyinin Değiştirilmesi

- JDBC Connection'ının transaction isolation seviyesi veritabanının desteklediği isolation düzeylerinden birisi ile değiştirilebilir

```
connection.setTransactionIsolation(  
    Connection.TRANSACTION_REPEATABLE_READ);  
connection.setAutoCommit(false);
```

...

```
connection.commit();
```



TRANSACTION_NONE
TRANSACTION_READ_UNCOMMITTED
TRANSACTION_READ_COMMITTED
TRANSACTION_REPEATABLE_READ
TRANSACTION_SERIALIZABLE

JDBC SavePoint

- Mevcut transaction içerisindeki belirli bir bölüm işaretlenebilir
- Transaction rollback yapılırken savepoint belirtilirse sadece bu bölüm rollback yapılır, geri kalan bölüme dokunulmaz

```
connection.setAutoCommit(false);
```

```
...
```

```
Savepoint savepoint = connection.setSavepoint("my-savepoint");
```

```
...
```

```
connection.rollback(savepoint);
```

```
...
```

```
connection.commit();
```

Scrollable ResultSet

- ResultSet üzerinde **ileri**, **geri** veya **absolute** pozisyon belirterek hareket edilebilir
- JDBC sürücüsünün **desteklemesi** gerekir
- **Statement oluşturulurken** scrollable ResultSet istendiği belirtilmelidir

```
Statement stmt = connection.createStatement(
    ResultSet.TYPE_SCROLL_INSENSITIVE,
    ResultSet.CONCUR_READ_ONLY);
```

```
ResultSet rs = stmt.executeQuery("select * from T_PET");
```

TYPE_FORWARD_ONLY (default)
 TYPE_SCROLL_INSENSITIVE
 TYPE_SCROLL_SENSITIVE (DB'deki değişiklikleri anında yansıtır)

Scrollable ResultSet

```
rs.next();
rs.previous();
rs.first();
rs.absolute(1);
rs.absolute(10);
rs.relative(1);
rs.relative(-1);
rs.beforeFirst();
rs.last();
rs.absolute(-1);
rs.absolute(-2);
rs.afterLast();
rs.isBeforeFirst();
rs.isAfterLast();
rs.isFirst();
rs.isLast();
```

Hepsi boolean sonuç döner

Updatable ResultSet

- Mevcut **ResultSet** üzerinden ilgili sütunlar doğrudan **güncellenebilir**, yeni kayıt **eklenebilir** veya mevcut bir kayıt **silinebilir**
- JDBC sürücüsünün **desteklemesi** gerekir
- **Statement** oluşturulurken **updatable ResultSet** istendiği belirtilmelidir

```
Statement stmt = connection.createStatement(
    ResultSet.TYPE_SCROLL_INSENSITIVE,
    ResultSet.CONCUR_UPDATABLE);
```

```
ResultSet rs = stmt.executeQuery("select * from T_PET");
```

CONCUR_READ_ONLY
CONCUR_UPDATABLE

Updatable ResultSet

```
rs.first();
rs.updateString("NAME", "111");
rs.updateRow();

rs.last();
rs.updateDate("BIRTH_DATE", new java.sql.Date(123));
rs.updateRow();

rs.moveToInsertRow();
rs.updateLong("ID", 101);
rs.updateString("NAME", "My Pet");
rs.updateInt("VERSION", 0);
rs.updateDate("BIRTH_DATE", new java.sql.Date(123));
rs.insertRow();
rs.moveToCurrentRow();

rs.absolute(10);
rs.deleteRow();
```

Batch Insert & Update Kabiliyeti

- Tek bir network iletişiminde **birden fazla** kayıt üzerinde **ekleme, silme** veya **güncelleme** yapmaya imkan tanır
- JDBC sürücüsünün **desteklemesi** gerekir
- **PreparedStatement** üzerinden gerçekleştirilir

Batch Insert & Update Kabiliyeti

```
connection.setAutoCommit(false);
```

```
PreparedStatement pstmt = connection.prepareStatement(  
    "insert into T_PET(ID,VERSION,NAME,BIRTH_DATE) values (?,?,,?)");
```

```
Object[][] data = new Object[][]{  
    {110,0,"Bobo",new java.sql.Date(123)},  
    {111,0,"Coco",new java.sql.Date(456)},  
    {112,0,"Dodo",new java.sql.Date(789)}  
};
```

```
for (Object[] row : data) {  
    pstmt.setObject(1, row[0]);  
    pstmt.setObject(2, row[1]);  
    pstmt.setObject(3, row[2]);  
    pstmt.setObject(4, row[3]);
```

```
    pstmt.addBatch();  
}
```

```
int[] updateCounts = pstmt.executeBatch();
```

```
connection.commit();
```

Batch işlem sonucu üzerinde
işlem yapılan satır sayısı döner

İletişim



www.harezmi.com.tr

www.java-egitimleri.com



info@harezmi.com.tr

info@java-egitimleri.com



[@HarezmiBilisim](https://twitter.com/HarezmiBilisim)

[@JavaEgitimleri](https://twitter.com/JavaEgitimleri)