

Java Database Connectivity (JDBC)

JDBC RowSet API

- **java.sql.ResultSet**'i extend eder
- **Tabular veri** üzerinde daha esnek ve kolay biçimde çalışmayı sağlar
- **RowSet** nesneleri aynı zamanda **JavaBeans bileşenleridir**
- Bu sayede bütün **property'leri get/set metotları** ile erişilebilir veya değiştirilebilir
- RowSet üzerinde meydana gelen **değişikliklerden JavaBeans event notifikasyon mekanizması ile haberdar** olunabilir

JDBC RowSet API

- Bazı DBMS'ler **ResultSet** nesnelerinin **scrollable** ve **updatable** olmasını desteklemez
- **RowSet** nesneleri **default** olarak **scrollable** ve **updatable**'dir
- Böylece tabular veri üzerinde ileri/geri hareket edilebilir, yeni kayıt eklenebilir, mevcut kayıt güncellenebilir veya silinebilir

RowSet Türleri

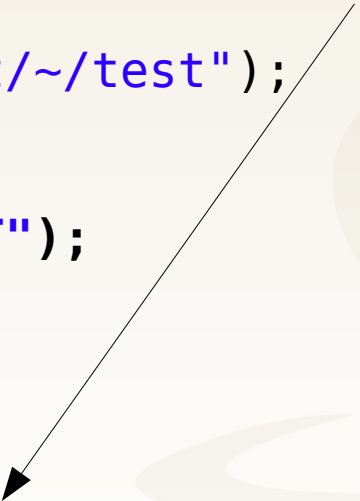
- JDBC API tarafından sağlanan **farklı RowSet gerçekleştirimleri** mevcuttur
 - JdbcRowSet
 - CachedRowSet
 - WebRowSet
 - JoinRowSet
 - FilteredRowSet
- Ayrıca RowSet nesneleri genel olarak **connected** ve **disconnected** olarak ikiye ayrılırlar

JdbcRowSet

- **Connected** bir RowSet gerçekeştirimidir
- Bir veritabanı bağlantısı açar ve ömrü boyunca bunu kullanır
- **ResultSet** nesnesine oldukça benzer
- Aslında onu **wrap eder** diyebiliriz
- Scrollable olmayan ve salt okunur bir ResultSet nesnesini **scrollable ve updatable** yapmak için kullanılabilir

RowSet Yaratılması: RowSetFactory ile

```
RowSetFactory rowSetFactory = RowSetProvider.newFactory();  
  
JdbcRowSet rowSet = rowSetFactory.createJdbcRowSet();  
  
rowSet.setUrl("jdbc:h2:tcp://localhost/~test");  
rowSet.setUsername("sa");  
rowSet.setPassword("");  
rowSet.setCommand("select * from T_PET");  
rowSet.execute();
```



createCachedRowSet
createFilteredRowSet
createJoinRowSet
createWebRowSet metotları ile diğer
RowSet gerçekleştirimlerinden de nesneler
oluşturulabilir

RowSet Yaratılması: Default No-Arg Constructor ile

```
com.sun.rowset.JdbcRowSetImpl rowSet =  
    new com.sun.rowset.JdbcRowSetImpl();  
rowSet.setUrl("jdbc:h2:tcp://localhost/~/test");  
rowSet.setUsername("sa");  
rowSet.setPassword("");  
rowSet.setCommand("select * from T_PET");  
rowSet.execute();  
  
...
```

RowSet Yaratılması: Connection Geçerek

```
try(Connection con = DriverManager
    .getConnection("jdbc:h2:tcp://localhost/~ /test", "sa", "")) {

    com.sun.rowset.JdbcRowSetImpl rowSet = new
        com.sun.rowset.JdbcRowSetImpl(con);
    rowSet.setCommand("select * from T_PET");
    rowSet.execute();

    ...

}
```


RowSet Yaratılması: ResultSet Wrap Ederek

```
try(Connection con = DriverManager
        .getConnection("jdbc:h2:tcp://localhost/~:/test", "sa",
"")) {
    Statement stmt = con.createStatement(
        ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_UPDATABLE);

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

    com.sun.rowset.JdbcRowSetImpl rowSet = new
com.sun.rowset.JdbcRowSetImpl(rs);

    ...
}
```

RowSet ile Veri Üzerinde Navigasyon

ResultSet metodudur, cursor'ın bir sonraki kayda iterate edilmesini sağlar

```
while(rowSet.next()) {  
    String name = rowSet.getString("NAME");  
    System.out.println("Pet name : " + name);  
}
```

`rowSet.beforeFirst();` → Cursor'ı ilk satırın öncesine götürür

`rowSet.afterLast();` → Cursor'ı son satırın sonrasına götürür

`rowSet.first();` → Cursor'ı ilk satıra götürür

`rowSet.last();` → Cursor'ı son satıra götürür

`rowSet.previous();` → Cursor'ı halihazırdaki satırdan bir önceki satıra götürür

`rowSet.absolute(1);` → Cursor'ı ilk satıra götürür

`rowSet.absolute(10);` → Cursor'ı onuncu satıra götürür

`rowSet.absolute(-1);` → Cursor'ı son satıra götürür

`rowSet.absolute(-2);` → Cursor'ı sondan bir önceki satıra götürür

RowSet ile Kayıtların Güncellenmesi

```
rowSet.setCommand("select * from T_PET");  
  
rowSet.execute();  
  
rowSet.absolute(3);  
  
rowSet.updateString("NAME", "My Pet 1");  
  
rowSet.updateDate("BIRTH_DATE",  
                  new java.sql.Date(new Date().getTime()));  
  
rowSet.updateRow();
```

RowSet ile Yeni Kayıt Eklenmesi

```
rowSet.setCommand("select * from T_PET");  
  
rowSet.execute();  
  
rowSet.moveToInsertRow();  
  
rowSet.updateLong("ID", 100L);  
rowSet.updateDate("BIRTH_DATE",  
                    new java.sql.Date(new Date().getTime()));  
rowSet.updateString("NAME", "My Pet");  
rowSet.updateInt("TYPE_ID", 1);  
rowSet.updateInt("OWNER_ID", 10);  
rowSet.updateInt("VERSION", 0);  
  
rowSet.insertRow();
```

RowSet ile Kayıtların Silinmesi

```
rowSet.setCommand("select * from T_PET");  
rowSet.execute();  
  
rowSet.last();  
  
rowSet.deleteRow();
```

RowSet ve Listener Notifikasyonu

- RowSet üzerinde yapılan navigasyonlar, kayıtlar üzerinde yapılan değişiklikler, ekleme ve silmeler ile ilgili **RowSet event**'leri fırlatılır
- **RowSetListener** arayüzünü implement eden nesneler ile bu event'ler handle edilebilir

```
public interface RowSetListener extends java.util.EventListener {  
    void rowSetChanged(RowSetEvent event);  
    void rowChanged(RowSetEvent event);  
    void cursorMoved(RowSetEvent event);  
}
```

→ RowSetEvent içerisinde
üzerinde işlem yapılan RowSet
nesnesine erişilebilir

RowSet ve Listener Notifikasyonu

```
rowSet.addRowSetListener(new RowSetListener() {
```

```
@Override
```

```
public void rowSetChanged(RowSetEvent event) {  
    System.out.println("RowSet changed");  
}
```

Örneğin execute() metodu çağrılıp sorgu çalıştığında düşer

```
@Override
```

```
public void rowChanged(RowSetEvent event) {  
    System.out.println("Row changed");  
}
```

Kayıt ekleme, silme, güncelleme durumlarında çağrılır

```
@Override
```

```
public void cursorMoved(RowSetEvent event) {  
    System.out.println("Cursor moved");  
}
```

```
});
```

Cursor pozisyonunu değiştiren metotlar, first, last, absolute gibi metotlar çağrıldığında çalışır

CachedRowSet

- **JdbcRowSet**'in sahip olduğu **bütün kabiliyetlere sahiptir**
- Ayrıca **disconnected** durumda veri üzerinde **güncelleme yapmaya** imkan tanır
- Veriyi DB'ye yazmak için kendisi **veritabanı bağlantısı açar**
- DataSource ile arasındaki **conflict'leri tespit eder ve çözümleyebilir**

CachedRowSet Örneği

```
cachedRowSet.setCommand("select ID,NAME,BIRTH_DATE from T_PET");
cachedRowSet.setKeyColumns(new int[]{1});
cachedRowSet.execute();
```

```
cachedRowSet.last();
```

```
cachedRowSet.deleteRow();
```

```
cachedRowSet.first();
```

```
cachedRowSet.updateString("NAME", "My Pet 123");
cachedRowSet.updateRow();
```

```
cachedRowSet.moveToInsertRow();
```

```
cachedRowSet.updateLong("ID", 100L);
cachedRowSet.updateDate("BIRTH_DATE", new java.sql.Date(new Date().getTime()));
cachedRowSet.updateString("NAME", "My Pet");
```

```
cachedRowSet.insertRow();
```

```
cachedRowSet.moveToCurrentRow();
```

```
cachedRowSet.acceptChanges();
```

Her bir kaydı CachedRowSet içerisinde benzersiz biçimde ayırtırmaya yarayacak sütunlar tanımlanmalıdır

insertRow() sonrası çağrılmalıdır

Ekleme, güncelleme, silme işlemlerinin DB'ye yansıtılması için acceptChanges() metodunun çağrılması gerekir

WebRowSet

- **CachedRowSet**'den türer
- Kendisini, DB metadata'yı ve veriyi **XML dokümanı** şeklinde yazabilir
- WebRowSet nesnesine karşılık gelen bir **XML dokümanını yükleyebilir**

WebRowSet Örneği: RowSet'i XML'e Dönüştürme

```
RowSetFactory rowSetFactory = RowSetProvider.newFactory();
```

```
WebRowSet webRowSet = rowSetFactory.createWebRowSet();  
webRowSet.setUrl("jdbc:h2:tcp://localhost/~/test");  
webRowSet.setUsername("sa");  
webRowSet.setPassword("");
```

```
webRowSet.setCommand("select ID,BIRTH_DATE,NAME,TYPE_ID,OWNER_ID,VERSION  
from T_PET");  
webRowSet.setKeyColumns(new int[] { 1 });
```

```
webRowSet.execute();
```

```
FileOutputStream fos = new FileOutputStream("/tmp/out.xml");
```

```
webRowSet.writeXml(fos);
```

WebRowSet'den Elde Edilen XML'in Yapısı

```
<webRowSet>
  <properties>
    <command>select * from T_PET</command>
    <table-name>T_PET</table-name>
    <url>jdbc:h2:tcp://localhost/~test</url>
    ...
  </properties>
  <metadata>
    <column-count>6</column-count>
    <column-definition>
      <column-index>1</column-index>
      <column-name>ID</column-name>
      <column-type>-5</column-type>
      ...
    </column-definition>
    ...
  </metadata>
  <data>
    <currentRow>
      <columnValue>1</columnValue>
      ...
    </currentRow>
    ...
  </data>
</webRowSet>
```

RowSet property'leri ile ilgili bölüm

ResultSetMetadata ile ilgili bölüm

Data bölümü

WebRowSet Örneği: XML'den RowSet'e Dönüştürme

```
RowSetFactory rowSetFactory = RowSetProvider.newFactory();
```

```
WebRowSet webRowSet = rowSetFactory.createWebRowSet();  
webRowSet.setUsername("sa");  
webRowSet.setPassword("");
```

```
FileInputStream fin = new FileInputStream("/tmp/out.xml");
```

```
webRowSet.readXml(fin);
```

```
while(webRowSet.next()) {  
    String name = webRowSet.getString("NAME");  
    System.out.println("Pet :" + name);  
}
```

```
webRowSet.last();
```

```
webRowSet.updateString("NAME", "bobo");
```

```
webRowSet.updateRow();
```

```
webRowSet.acceptChanges();
```

XML dosyadan yüklenen veri
iterate edilebilir veya
güncellenebilir

JoinRowSet

- **WebRowSet**'den türer
- Veritabanına bağlantı kurmadan **SQL JOIN** yapmayı sağlar

JoinRowSet Örneği

```
RowSetFactory rowSetFactory = RowSetProvider.newFactory();

CachedRowSet cachedPetsRowSet = rowSetFactory.createCachedRowSet();
cachedPetsRowSet.setUrl("jdbc:h2:tcp://localhost/~:/test");
cachedPetsRowSet.setUsername("sa");
cachedPetsRowSet.setPassword("");

cachedPetsRowSet.setCommand(
    "select ID,BIRTH_DATE,NAME,TYPE_ID,OWNER_ID,VERSION from T_PET");
cachedPetsRowSet.setKeyColumns(new int[] { 1 });

cachedPetsRowSet.execute();

CachedRowSet cachedVisitsRowSet = rowSetFactory.createCachedRowSet();
cachedVisitsRowSet.setUrl("jdbc:h2:tcp://localhost/~:/test");
cachedVisitsRowSet.setUsername("sa");
cachedVisitsRowSet.setPassword("");

cachedVisitsRowSet.setCommand(
    "select ID,VISIT_DATE,DESCRIPTION,PET_ID from T_VISIT");
cachedVisitsRowSet.setKeyColumns(new int[] { 1 });

cachedVisitsRowSet.execute();
```

JoinRowSet Örneği

```
JoinRowSet joinRowSet = rowSetFactory.createJoinRowSet();

joinRowSet.addRowSet(cachedPetsRowSet, "ID");
joinRowSet.addRowSet(cachedVisitsRowSet, "PET_ID");
joinRowSet.setJoinType(JoinRowSet.INNER_JOIN);

while(joinRowSet.next()) {
    String name = joinRowSet.getString("NAME");
    String desc = joinRowSet.getString("DESCRIPTION");
    Long petId = joinRowSet.getLong("ID");

    Long visitId = joinRowSet.getLong(7);

    System.out.println(String.format(
        "Pet ID:%d Pet Name:%s Visit ID:%d Visit Desc:%s",
        petId, name, visitId, desc));
}
```

Join'de yer alacak RowSet instance'ları match column(s) belirterek eklenir

Join Tipi belirtilebilir

Aynı isimde birden fazla sütun varsa ilk sütundan sonrakilere index ile erişilebilir

FilteredRowSet

- **WebRowSet**'den türer
- Veri üzerinde **filtreleme yapmaya** imkan tanır
- Böylece **sadece istenen verinin görüntülenmesi** sağlanabilir
- Filtreleme için herhangi bir **SQL ifadesi** yazılmasına veya **DB'ye bağlı olmaya gerek yoktur**

FilteredRowSet Örneği: Predicate Yazılması

```
public class DateRangeFilter implements javax.sql.rowset.Predicate {  
  
    private Date before, after;  
    private String columnName;  
    private int column;  
  
    public DateRangeFilter(Date before, Date after,  
                           String columnName, int column) {  
        this.before = before;  
        this.after = after;  
        this.columnName = columnName;  
        this.column = column;  
    }  
  
    @Override  
    public boolean evaluate(RowSet rs) {  
        try {  
            Date date = rs.getDate(columnName);  
            return this.evaluate(date, columnName);  
        } catch (SQLException ex) {  
            return false;  
        }  
    }  
    ...  
}
```

RowSet içerisinde cursor navigate ettirilken çağrılır

FilteredRowSet Örneği: Predicate Yazılması

```
public class DateRangeFilter implements javax.sql.rowset.Predicate {  
    @Override  
    public boolean evaluate(Object value, int column)  
        throws SQLException {  
        if (column == this.column) {  
            Date date = (Date) value;  
            if (date.before(before)) return false;  
            else if (date.after(after)) return false;  
        }  
        return true;  
    }  
  
    @Override  
    public boolean evaluate(Object value, String columnName)  
        throws SQLException {  
        if (columnName == this.columnName) {  
            Date date = (Date) value;  
            if (date.before(before)) return false;  
            else if (date.after(after)) return false;  
        }  
        return true;  
    }  
}
```

RowSet'e yeni bir kayıt eklerken çağrılır

FilteredObject Örneği: Predicate'in Kullanımı

```
RowSetFactory rowSetFactory = RowSetProvider.newFactory();

FilteredRowSet filteredRowSet = rowSetFactory.createFilteredRowSet();
filteredRowSet.setUrl("jdbc:h2:tcp://localhost/~/test");
filteredRowSet.setUsername("sa");
filteredRowSet.setPassword("");

filteredRowSet.setCommand(
    "select ID,BIRTH_DATE,NAME,TYPE_ID,OWNER_ID,VERSION from T_PET");
filteredRowSet.setKeyColumns(new int[] { 1 });

filteredRowSet.execute();

filteredRowSet.setFilter(new DateRangeFilter(
    dateFormat.parse("01-01-2007"),
    dateFormat.parse("31-12-2008"), "BIRTH_DATE", 2));

while(filteredRowSet.next()) {
    String name = filteredRowSet.getString("NAME");
    Date birthDate = filteredRowSet.getDate("BIRTH_DATE");
    System.out.println(String.format("Pet : %s birth date : %s",
                                     name,birthDate));
}
```

devre dışı bırakmak için setFilter(null) yapılabilir

SyncProvider & SyncResolver JAVA Eğitimleri ile Çakışmaların Çözülmesi

- CachedRowSet nesneleri üzerinden kayıtlarda **yapılan değişiklikler** DB'de meydana gelmiş değişikliklerle **çakışabilirler**
- RowSet, **SyncProvider** vasıtası ile bu çakışmaları tespit etmeye çalışır
- **WebRowSet** hariç varsayılan durumda **RIOptimisticProvider** SyncProvider gerçekleştirimi kullanılır
- WebRowSet ise default olarak **RIXMLProvider** kullanır

SyncProvider & SyncResolver ile Çakışmaların Çözülmesi

- RIOptimisticProvider ile çalışıldığı vakit herhangi bir çakışma durumunda **SyncProviderException** fırlatılır
- RIXMLProvider ise herhangi bir **çakışma kontrol etmeden** değişiklikleri veritabanına yazar
- SyncProviderException üzerinden de **SyncResolver** nesnesi elde edilerek çakışmanın çözülmesi sağlanabilir

SyncProvider & SyncResolver ile Çakışmaların Çözülmesi

```
try {
    cachedRowSet.acceptChanges();
} catch (SyncProviderException ex) {
    SyncResolver syncResolver = ex.getSyncResolver();
    while (syncResolver.nextConflict()) {
        if (syncResolver.getStatus() == SyncResolver.UPDATE_ROW_CONFLICT) {

            int row = syncResolver.getRow();

            cachedRowSet.absolute(row);
            int colCount = cachedRowSet.getMetaData().getColumnCount();
            for(int colIndex = 1; colIndex <= colCount; colIndex++) {
                Object conflictedValue =
                    syncResolver.getConflictValue(colIndex);

                if(conflictedValue == null) continue;

                Object value = cachedRowSet.getObject(colIndex);
                //resolve conflict based on the available values...

                Object resolvedValue = value;
                syncResolver.setResolvedValue(colIndex, resolvedValue);
            }
        }
    }
}
```

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)