

JDBC ile Veri Erişimi



JDBC API ile Veri Erişimi

- JDBC API kullanarak gerçekleştirilen veri tabanı işlemlerinde yazılan kod blokları genel olarak birbirlerine **benzer bir akışa** sahiptir
- Hepsinde veritabanı bağlantısı oluşturma, SQL'i çalıştırma, dönen sonuçlar üzerinde işlem yapma, TX varsa commit/rollback yapma, hataları ele alma ve bağlantıyı kapatma gibi **işlemler standarttır**
- **Değişen kısımlar** SQL, parametreler ve dönen sonucu işleyen kod bloğu olur

JDBC API ile Veri Erişimi

- Veritabanı bağlantı parametrelerinin belirtilmesi ve bağlantının kurulması
- *SQL sorgusunun oluşturulması*
- Oluşturulan Statement'ın derlenip çalıştırılması
- Dönen ResultSet üzerinde işlem yapan bir döngünün kurulması
- *Bu döngü içerisinde her bir kaydın işlenmesi*
- Meydana gelebilecek hataların ele alınması
- Transaction'ın sonlandırılması ve veritabanı bağlantısının kapatılması

Sadece kırmızı font ile işaretlenen kısımlar değişkenlik gösterir, diğer adımlar bütün persistence işlemlerinde standarttır

JDBC API ile Veri Erişimine Örnek

```
public Collection<Document> findDocuments() {  
    Connection c = null;  
    Statement stmt = null;  
    try {  
        c = DriverManager.getConnection(  
            "jdbc:h2:tcp://localhost/~test","sa", "");  
        c.setAutoCommit(false);  
        stmt = c.createStatement();  
        ResultSet rs = stmt  
            .executeQuery("select * from T_DOCUMENT");  
        Collection<Document> result = new ArrayList<Document>();  
        while (rs.next()) {  
            Document doc = new Document();  
            doc.setName(rs.getString("doc_name"));  
            doc.setType(rs.getInt("doc_type"));  
            result.add(doc);  
        }  
    }  
}
```



JDBC API ile Veri Erişimine Örnek

```
c.commit();  
return result;  
} catch (SQLException ex) {  
    try {  
        c.rollback();  
    } catch (Exception e) {}  
    throw new DataRetrievalFailureException(  
        "Cannot execute query", ex);  
} finally {  
    try {  
        stmt.close();  
    } catch (Exception e) {}  
    try {  
        c.close();  
    } catch (Exception e) {}  
}
```

Spring Üzerinden JDBC ile Veri Erişimi

- Spring veri erişiminde bu tekrarlayan kısımları ortadan kaldırmak için **Template Method örüntüsü tabanlı** bir kabiliyet sunar
- **JdbcTemplate** merkez sınıftır
- Utility veya helper sınıflarına benzetilebilir
- JdbcTemplate sayesinde Template Method tarafından dikte edilen **standart bir kullanım şekli** kod geneline hakim olur

JdbcTemplate ile Veri Erişimi

```
public Collection<Document> findDocuments() {  
    Collection<Document> result = jdbcTemplate.query(  
        "select * from T_DOCUMENT",  sorgu  
        new RowMapper<Document>() {  
  
        @Override  
        public Document mapRow(ResultSet rs, int rowNum)  callback  
            throws SQLException {  
                Document doc = new Document();  
                doc.setName(rs.getString("doc_name"));  
                doc.setType(rs.getInt("doc_type"));  
                return doc;  
            }  
        }  
    );  
    return result;  
}
```

JdbcTemplate Konfigürasyonu

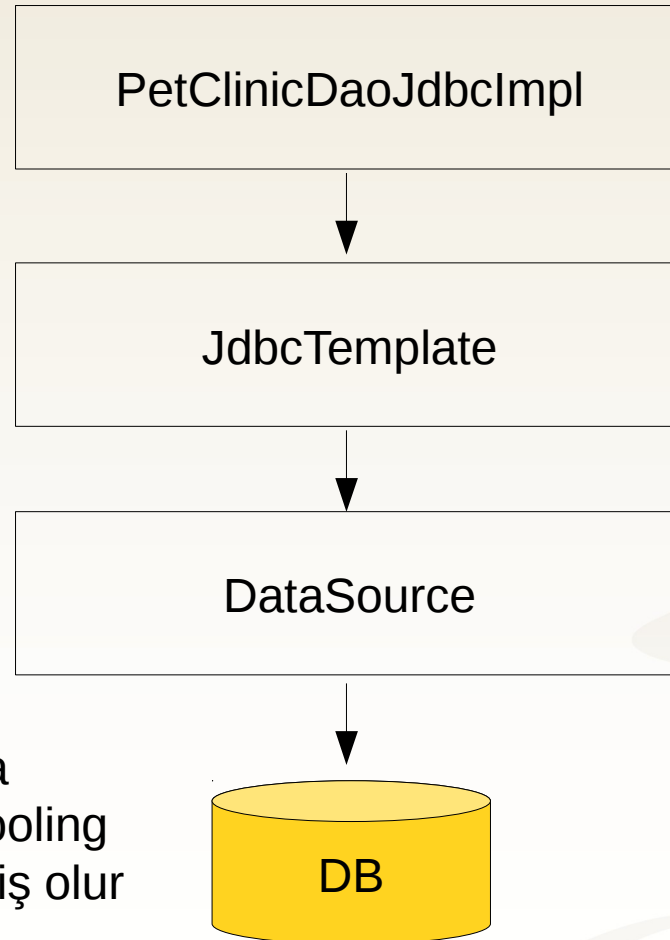


Uygulama tarafında
JdbcTemplate ile
çalışırken sadece
Callback yazmak yeterlidir

Spring veritabanı
bağlantılarını **DataSource**
nesnesinden alır

DataSource
connection factory'dir

DataSource'un kendi başına
yönetilmesi ile connection pooling
vs. uygulamadan izole edilmiş olur



JDBC ile veri erişimi
JdbcTemplate üzerine
kurulmuştur

Çalışması için
DataSource nesnesine
ihtiyaç vardır

Thread safe'dir, birden
fazla bean tarafından
erişilebilir

JdbcTemplate Kullanım Örnekleri

```
Collection<Document> result = jdbcTemplate.query(  
    "select * from T_DOCUMENT",  
    new RowMapper<Document>() {  
  
        @Override  
        public Document mapRow(ResultSet rs, int rowNum)  
            throws SQLException {  
            Document doc = new Document();  
            doc.setName(rs.getString("doc_name"));  
            doc.setType(rs.getInt("doc_type"));  
            return doc;  
        }  
    }  
);
```

JdbcTemplate Kullanım Örnekleri

```
String result = jdbcTemplate.queryForObject("select  
last_name from persons where id = ?", new Object[]{1212L},  
String.class);
```

```
List<String> result = jdbcTemplate.queryForList("select  
last_name from persons", String.class);
```

```
Map<String, Object> result =  
jdbcTemplate.queryForMap("select last_name, first_name from  
persons where id = ?", 1212L);
```

```
List<Map> result = jdbcTemplate.queryForList("select * from  
persons");
```

JdbcTemplate Kullanım Örnekleri

```
int insertCount = jdbcTemplate.update(  
    "insert into persons (first_name, last_name) values (?, ?)",  
    "Ali", "Yücel");
```

```
int updateCount = jdbcTemplate.update(  
    "update persons set last_name = ? where id = ?", "Güçlü", 1L);
```

```
int deleteCount = jdbcTemplate.update(  
    "delete from persons where id = ?", 1L);
```

```
int result = jdbcTemplate.update(  
    "call SUPPORT.REFRESH_PERSON_SUMMARY(?)", 1L);
```

JdbcTemplate Kullanım Örnekleri

```
KeyHolder keyHolder = new GeneratedKeyHolder();
```

```
PreparedStatementCreator psc = new PreparedStatementCreator() {  
    @Override  
    public PreparedStatement createPreparedStatement(  
        Connection con) throws SQLException {  
        PreparedStatement stmt = con.prepareStatement(  
            "insert into persons(first_name,last_name) values(?,?)");  
        stmt.setString(1, person.getFirstName());  
        stmt.setString(2, person.getLastName());  
        return stmt;  
    }  
};  
  
int insertCount = jdbcTemplate.update(psc, keyHolder);  
  
Long id = (Long) keyHolder.getKey();
```

JdbcTemplate Kullanım Örnekleri

```
int[] batchUpdateCount = jdbcTemplate.batchUpdate(  
    "update t_item set active = true",  
    "delete from persons");
```

```
List<Object[]> args = new ArrayList<Object[]>();  
args.add(new Object[]{"Kenan", "Sevindik", 1L});  
args.add(new Object[]{"Muammer", "Yücel", 2L});
```

```
int[] batchUpdateCount = jdbcTemplate.batchUpdate(  
    "update persons set first_name = ?, last_name = ? where id = ?",  
    args);
```

JdbcTemplate Kullanım Örnekleri

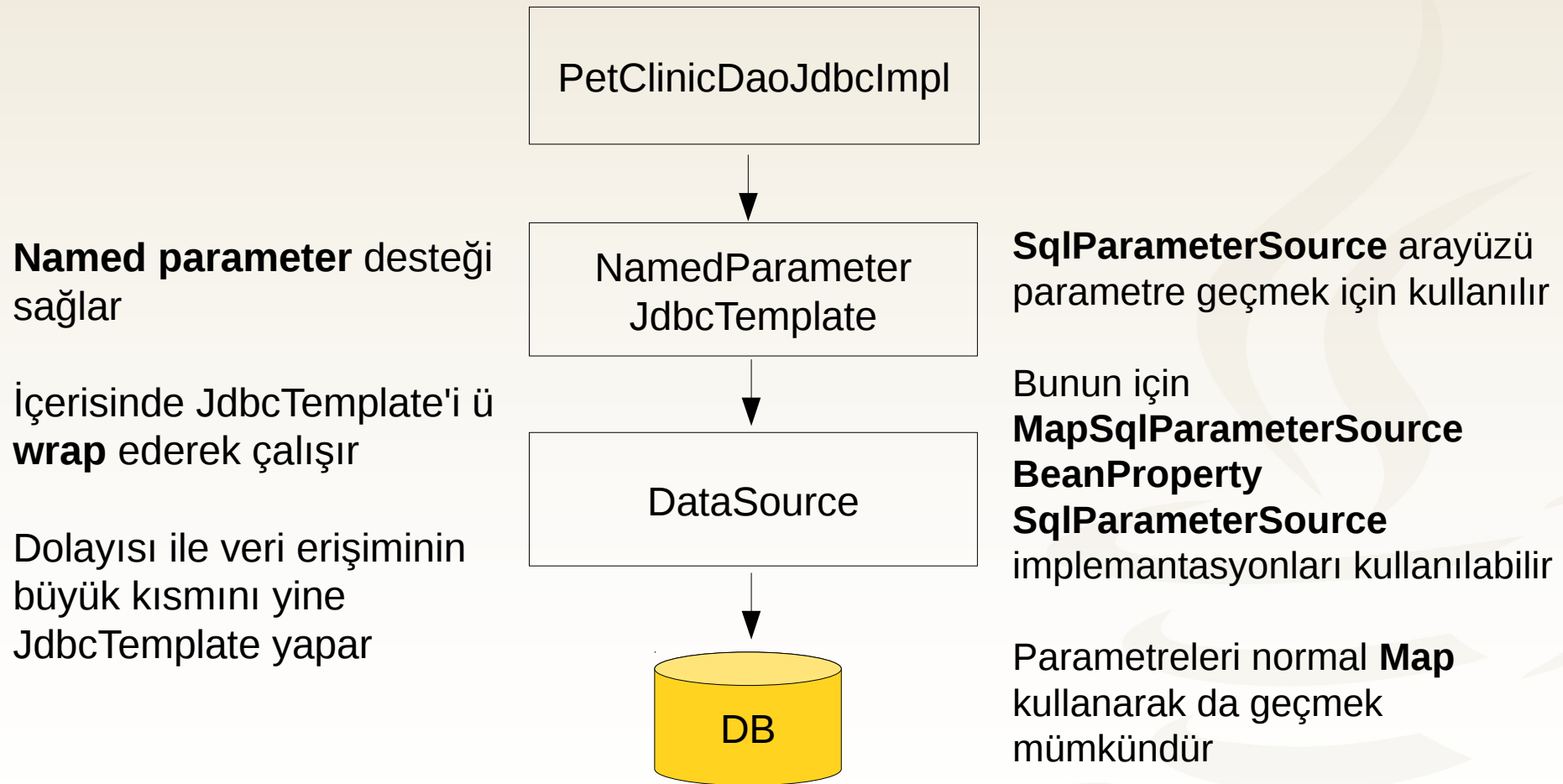
```
jdbcTemplate.execute("create table mytable (id integer, name  
varchar(100))");
```

```
Collection<PetType> result = jdbcTemplate.execute(new  
StatementCallback<Collection<PetType>>(){  
    @Override  
    public Collection<PetType> doInStatement(Statement stmt)  
        throws SQLException, DataAccessException {  
        ResultSet resultSet = stmt.executeQuery("select * from  
t_pet_type");  
        List<PetType> types = new ArrayList<>();  
        while(resultSet.next()) {  
            PetType type = new PetType();  
            type.setId(resultSet.getLong("ID"));  
            type.setName(resultSet.getString("NAME"));  
            types.add(type);  
        }  
        return types;  
    }  
});
```

Named Parameter Desteği

- JdbcTemplate default olarak **pozisyonel parametreleri** destekler
- Parametreler SQL ifadesi içerisinde ? **işareti** ile belirtilir
- Değerler de bir **Object array** ile verilir
- Spring veri erişiminin **isimlendirilmiş parametre desteği** de mevcuttur
- İsimlendirilmiş parametreler SQL ifade içerisinde **:paramName** şeklinde gösterilir

NamedParameterJdbc Template Konfigürasyonu



NamedParameterJdbc Template Kullanım Örneği

```
String sql = "select count(*) from persons where first_name = :firstName";
```

```
Map namedParameters = new HashMap();  
namedParameters.put("firstName", "Kenan");
```

```
int count = namedParameterJdbcTemplate  
            .queryForObject(  
                sql,  
                namedParameters,  
                Integer.class);
```

Sorgu ve isimleri ile eşleştirilmiş parametreler

Sorgu return değeri tipi

IN Clause'una Değişken Sayıda Parametre Geçilmesi

```
String sql = "select * from users where id in (:idList)";
```

```
Map namedParams = Collections.singletonMap(  
    "idList",  
    Arrays.asList(new Integer[]{1,2,3}));
```

```
RowMapper<User> rowMapper = new RowMapper<User>() {  
    public User mapRow(ResultSet rs, int rowNum) throws  
    SQLException {  
        User user = new User();  
        user.setFirstName(rs.getString("first_name"));  
        user.setLastName(rs.getString("last_name"));  
        return user;  
    }  
};
```

```
List<User> result=namedParameterJdbcTemplate.query(sql,  
namedParams, rowMapper);
```

İletişim

- **Harezmi** Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- <http://www.java-egitimleri.com>
- info@java-egitimleri.com

