JDBC (Java Database Connectivity)

JDBC (Java Database Connectivity) uygulamamızın veritabanı ile iletişime geçmesini sağlayan bir Java kütüphanesidir. JDBC ile herhangi bir veritabanına bağlanılabilir.

JDBC kullanırken izleyeceğimiz adımlar şunlar.

1. JDBC driver'ı yükleyeceğiz.

```
private static final String DB_URL = "jdbc:mysql://127.0.0.1:3306/test";

private static final String USER = "root";
private static final String PASS = "";
```

2. Veritabanı bağlantı sınıfını ekleriz.

```
Class.forName("com.mysql.cj.jdbc.Driver");
```

3. DriverManager sınıfı içerisindeki getConnection method iler veritabanı bağlantısı sağlarız.

```
DriverManager.getConnection(DB_URL, USER, PASS);
```

4. Veritabanı sorgularımızı Statement,PreparedStatement ile oluşturacağız.Veri ekleme,güncelleme slime işlemlerinden sonra executeUpdate method kullanılır.

```
PreparedStatement prepareStatement = null;
try {
    prepareStatement = connection.prepareStatement( sql: "INSERT INTO USER (ID, NAME, EMAIL, PHONE_NUMBER) VALUES (?,?,?,?)");
    prepareStatement.setLong( parameterIndex: 1, user.getId());
    prepareStatement.setString( parameterIndex: 2, user.getName());
    prepareStatement.setString( parameterIndex: 3, user.getEmail());
    prepareStatement.setString( parameterIndex: 4, user.getPhoneNumber());

    int executeUpdate = prepareStatement.executeUpdate();
```

5. ResultSet oluşturacağız.ResultSet veri çekme işleminden sonra verileri listelemek için kullanacağız.

6. Son olarak bağlantıyı kapatırız.

JDBC kullanıldığında ileride tablolarımız çoğaldığında bağımlılıklarımız arttığında karmaşıklaşabilir. Connection açma kapama gibi işlemleri yönetmesi zorlaşacaktır. Burada yapılan bir hata bir çok yeri etkileyecektir. Böyle durumlarda Hibarnate kullanmak daha yararlı. Aşağıda Hibarnate ilee ilgili yazı var.

```java
private static final String INSERT_USER = "INSERT INTO USER (ID, NAME, EMAIL, PHONE_NUMBER)
VALUES (?,?,?,?);";
public void save(User user) {
   Connection connection = connect();
   if (connection != null) {
      PreparedStatement prepareStatement = null;
      try {
         prepareStatement = connection.prepareStatement(INSERT_USER);
         prepareStatement.setLong(1, user.getId());
         prepareStatement.setString(2, user.getName());
         prepareStatement.setString(3, user.getEmail());
         prepareStatement.setString(4, user.getPhoneNumber());

         int executeUpdate = prepareStatement.executeUpdate();
         System.out.println("result: " + executeUpdate);
      } catch (SQLException e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
      } finally {
         try {
            connection.close();
            prepareStatement.close();
         } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
         }
      }
   } else {
      System.out.println("Connection oluşturululamadı!");
   }
}
```

```java
private static final String SELECT_ALL_USER = "SELECT * FROM USER";

public List<User> findAll() {
   List<User> userList = new ArrayList<User>();
   Connection connection = connect();
   try {
      PreparedStatement prepareStatement = connection.prepareStatement(SELECT_ALL_USER);
      ResultSet result = prepareStatement.executeQuery();
      while (result.next()) {
         Long id = result.getLong("id");
         String name = result.getString("name");
         String email = result.getString("email");
         String phoneNumber = result.getString("phone_number");

         userList.add(prepareUser(id, name, email, phoneNumber));
      }
   } catch (SQLException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
   }
```

```
    return userList;
}
```

HIBARNATE

Hibarnate bir ORM kütüphanesidir. Nesne modellerine göre veritabanı şeması eşler. Veritabanı işlemlerini(veri ekleme,slime,güncelleme gibi) kolaylaştırır.Hibarnate bizi SQL yazmaktan kurtarır.Hibarnate ile kayıt eklerken ve listeden veri çekerken kullanılması gereken kod parçacığı aşağıdaki gibidir.

```
userRepository.save(user);
```

```
userRepository.findById(id);
```

Hibarnate ile işlem yapmak bu kadar kolay.Yukarıda JDBC ile işlem yapmak istediğimde bir sürü işlem yapıyordum Hibarnate bizi bu işlem adımlarından kurtarıyor ve zaman kazanmamızı sağlıyor. Hibarnate modellerimizi anlayarak HQL yazmamızı sağlar.

JDBC de farklı veritabanı için farklı kodlar yazılması gerek ancak hibernate de aynı kod küçük değişiklikler ile bir çok veritabanında çalışabilir.

JDBC de ilişki kurmak zordur.Hibarnate de manyToOne,oneToOne,oneToMany,manyToMany anatasyonları ile ilişki kurulabilir.

Kaynakça:

https://www.geeksforgeeks.org/difference-between-jdbc-and-hibernate-in-java/

https://www.yusufsezer.com.tr/java-jdbc/

https://blog.burakkutbay.com/hibernate-nedir-hibernate-dersleri.html/