YouTube link:
https://www.youtube.com/watch?v=oHUGB0vsw60&list=PLeZr8VTNC1obmhNWF44Yz6vANbCd2ahk2&index=11

https://www.youtube.com/watch?v=rOf7fpAfNUs&list=PLeZr8VTNC1obmhNWF44Yz6vANbCd2ahk2&index=12

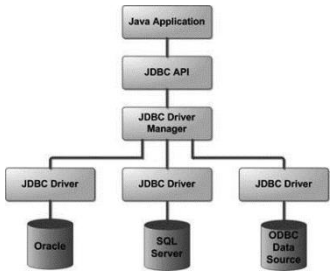
JDBC-1 02.11.20202

Java Database Connectivity (JDBC), Java programlama dilinde yazılmış uygulamaları (application) veritabanı (database) ile etkileşime girmesini sağlayan (bağlayan) bir uygulama programlama ara yüzüdür (API'dir).

JDBC ile hemen hemen tüm ilişkisel veri tabanı yönetim sistemlerine SQL sorgusu gönderilebilmektedir.

JDBC Mimarisi 2 katmana ayrılır;

1. **JDBC API**: Uygulama ve veritabanı arasında bağlantı sağlar.
2. **JDBC Driver API**: Uygulama ve kullanılan veritabanı sürücü bağlantısını destekler. Aşağıdaki diyagram bu yapının yerini göstermektedir.



Neden JDBC kullanılır? (JDBC Automation için kullanılır.**)**

Direk veri tabanına (database) bağlanıp veri (data) almak için kullanılır. Bunun için;

1. Database'e **bağlan**ılır
2. **Query** gönderip **data alınır**.
3. Bu dataları test caselerde kullanıp **assert** yapılır.

Database Testing Nasıl yapılır?

UI'daki (Front End) ve Database (Bach End) deki değerlerin aynı olup olmadığı database testing ile yapılır. Örneğin UI dan girilen dataların okunması ve bu dataların doğrulanması (Assertion).

Database testi, UI (User Interface) testinden daha hızlı. Çünkü sayfa yükleme, veri getirme gibi beklemeleri yapmıyor.

Mülakatta gelirse; Eklediğim verilerin (data) veri tabanına (database) eklenip eklenmediğini, UI'da yaptığım işlemlerin veri tabanında (database) gerçekleşip gerçekleşmediğini test etmek için direk veri tabanına bağlanarak JDBC kullandım.

Örnek-1; kayıt formuna girilen kullanıcı adı ve şifresi database de oluşmuş mu?

Örnek-2; database den girilen bir datanın UI da bulunup bulunmadığını test için,

Örnek-3; UI da yapılan değişikliğin database de olup olmadığı (yazıların değişmesi gibi).

JDBC Bileşenleri;

DriverManager: Bu sınıf, veritabanı sürücülerinin listesini yönetir.

Driver: Bu interface, veritabanı ile iletişimi ele alır. Farklı database'ler için farklı driverlar vardır.

connection: Bu interface, bütün metotları ile veritabanına (database) irtibat kurmak (bağlanmak) için kullanılır. Veri tabanlarının arka planda kullanıcı adı ve şifreleri vardır. Bağlanmak için kullanıcı adı ve şifreden yararlanılır.

connection = DriverManager.getConnection (url, username, password);

statement: SQL ifadelerini (Query) veritabanına (database) göndermek için bu interface'ten oluşturulan nesneler kullanılır.

statement = connection.createStatement (ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);

resultSet: Statements nesnelerini kullanarak SQL sorgusunu (assert) çalıştırdıktan sonra veri tabanından (database) alınan verileri tutmak için bu nesneler kullanılır. Onu taşımanıza izin veren bir yineleyici görevi görür. resultSet objesiyle veriyi (datayı) kullanırız (okumak, yazmak vb).

resultSet = statement.executeQuery ("SELECT " + string2 + " FROM dbo." + string);

SQLException: Bu sınıf, bir veritabanı uygulamasında ortaya çıkan hataları ele alır.

Bazı data (veri) alma methodları;

resultSet.next() => ilk sonraki satıra atlamak/geçiş yapmak için

resultSet.previous() =>

resultSet.getRow() => Hangi rowdasın

resultset.first() => ilk rowa git

resultset.last => son rowa git

resultset.absolute(5) => 5inci rowa git

resultSet.get**Object**("BookName") => BookName rowdaki **objeyi** ver

resultSet.get**String**("BookName") => BookName rowdaki **değeri** ver

- JDBC kurulumunu DevOps yapar, bizimde yapmamız gerekebilir. Veri tabanına bağlanıp, verileri okuyacağız.

Database'e Bağlanmak İçin Adımlar;

1. **Connect to MySQL Database**;

Connection Url = jdbc:mysql://+**HOST**+":"+**PORT**+"/**DATABASENAME**"; HOST = 107.182.225.121
PORT = 3306
DB Name = LibraryMgmt
username = techpro
password = tchpr2020

connection data;

```
String url = "jdbc:mysql://107.182.225.121:3306/LibraryMgmt";
String username = "techpro";
String password = "tchpr2020";
```

2. **Query yazmak için ihtiyaç olan statement object'i yazma**;

Stament stament= connection.**createStatement**(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
statement.**executeQuery**("SELECT * FROM Book;");

Kod yazmak için; IntelliJ 'de Cucumber (veya TestNG, veya JUnit) projesi açılır.

1. Adım: pom.xml'de dependencies içine www.mvnrepository.com'da Microsoft JDBC Driver For SQL Server kütüphanesinden kopyalanan dependency yapıştırılır ve import edilir.

```
<dependency>
  <groupId>com.microsoft.sqlserver</groupId>
  <artifactId>mssql-jdbc</artifactId>
  <version>8.2.2.jre8</version>
</dependency>
```

2. Adım: test'in altindaki java klasöründe database_stepdefinitions adli package oluşturulur.

3. Adım: runner package'nin içine DbRunner adli Class oluşturulur.

```
package runners;

import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(
        plugin = {"html:target/default-cucumber-reports",
                "json:target/json-reports/cucumber.json",
                "junit:target/xml-report/cucumber.xml"},
        features = "src/test/resources/features",
        glue = "database_stepdefinitions",
        tags = "@dbokuma",
        dryRun = false
)

public class DbRunner {

}
```

4. Adım: resources altindaki features klasörü içine dbokuma.feature dosyası oluşturulur. İçine istenilen senaryo yazılır.

```
dbokuma.feature
1  @dbokuma
2  Feature: Veri Okuma
3    Scenario: kullanici hotel tablosundaki verileri okur
4      Given kullanici veritabanina baglanir
5      And kullanici "tHOTEL" tablosundaki "Name" verilerini alir
6      And kullanici "Name" sutunundaki verileri okur
```

5. Adım: DbRunner 'a gidilir, dryRun = true yapılıp çalıştırılır.

6. Adım: Önerilen adımları console'dan alıp database_stepdefinitions package'nin içinde oluşturacağımız DbReadStepDef Class'ına yapıştırırız.

```
package database_stepdefinitions;

import io.cucumber.java.en.Given;

public class DbReadStepDef {
    @Given("kullanici veritabanina baglanir")
    public void kullanici_veritabanina_baglanir() {

    }

    @Given("kullanici {string} tablosundaki {string} verilerini alir")
    public void kullanici_tablosundaki_verilerini_alir(String string, String string2) {

    }

    @Given("kullanici {string} sutunundaki verileri okur")
    public void kullanici_sutunundaki_verileri_okur(String string) {

    }
}
```

7. Adım: Java'yı bağlanacağımız database 'in (koalapalace sitesi) **URL**'ine ihtiyacımız var.

String **url** =
"jdbc:sqlserver://184.168.194.58:1433;databaseName=kaolapalacedb;user=Ahmet_User;password=Ahmet123!";

sqlserver => Hangi Server'ı kullanacaksak (MySQL vb.) onu yazarız, değişiklik gösterebilir.

//184.168.194.58:1433 => bağlanacağımız (koalapalace sitesi) database 'in adresi.

databaseName=kaolapalacedb => database 'in ismi.

user=Ahmet_User; password=Ahmet123! => Bağlanmak isteyen kullanıcı ismi ve şifresi.

8. Adım: İhtiyaç duyulan **connection**, **statement** ve **resultSet** adlı 3 objeyi java.sql kütüphanesinden import edip kullanacağız.

```
public class DbReadStepDef {

    String url = "jdbc:sqlserver://184.168.194.58:1433;databaseName=kaolapalacedb;user=Ahmet_Use
    String username="Ahmet_User";
    String password="Ahmet123!";

    Connection connection; // Veritabanına bağlanmak için kullanacağız.
    Statement statement;   // Query'leri çalıştırmak ve verileri almak için kullanacağız.
    ResultSet resultSet;   // Aldığımız verileri resultset'in içine ekleyeceğiz.
```

9. Adım: DbReadStepDef Class'ına yapıştırılan önerilerin içine nesneler oluştururuz. Oluşan hata tipini method imzası bölümüne SQLException throws ederek yok edebiliriz. (Bazı sitelerde güvenlik önlemi alıyorlar, bunu geçmek için koalapalace sitesinde **FROM**'dan sonra **dbo.** isimli bir kod eklemek gerekir.) Tüm verileri okumak için ise while-loop oluşturulur.

https://www.kaolapalace.com/admin/HotelAdmin

manager2

Man1ager2!

API => Application Programming Interface

(Kaynaklar: https://medium.com/@gokhanyavas)

https://www.kaolapalace-qa-environment.com/admin/HotelAdmin/Create

JDBC-2 Database Utilities (DBUtilities) 02.11.20202