JDBC - JdbcTemplate - Hibernate Farkları

Jdbc Nedir?

JDBC, Java Veritabanı Bağlantısı'nın kısaltmasıdır. ODBC (Açık Veritabanı Bağlantısı) için bir gelişmedir. JDBC, verileri ön uçtan arka uca taşımak için geliştirilmiş standart bir API özelliğidir. Bu API, Java ile yazılmış sınıflardan ve arayüzlerden oluşur. Temel olarak Java programınızla veritabanları arasında bir arayüz (Java'da kullandığımız değil) veya kanal görevi görür, yani ikisi arasında bir bağlantı kurar, böylece bir programcı Java kodundan veri gönderebilir ve gelecekte kullanmak üzere veritabanında saklayabilir.

Jdbc ile insert etmek istersek sorguyu string olarak tutup, daha sonra örnekteki gibi bağlantı açıp sql stringini execute edip daha sonra bağlantıyı kapatarak işlemi yapabiliriz. Diğer CRUD işlemleri de aynı şekilde sorguyu yazarak gerçekleştirilebilir.

```
public class JDBCExample {
   static final String DB_URL = "jdbc:mysql://localhost/";
   static final String USER = "guest";
   static final String PASS = "guest123";

   public static void main(String[] args) {
      // Open a connection
      try(Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
         Statement stmt = conn.createStatement();
      ) {
         String sql = "CREATE DATABASE STUDENTS";
         stmt.executeUpdate(sql);
         System.out.println("Database created successfully...");
      } catch (SQLException e) {
         e.printStackTrace();
      }
   }
}
```

JdbcTemplate Nedir?

org.springframework.jdbc.core.JdbcTemplate sınıfı, JDBC çekirdek paketindeki merkezi sınıftır. JDBC kullanımını basitleştirir ve yaygın hatalardan kaçınmaya yardımcı olur. Temel JDBC iş akışını yürütür ve uygulama kodunu SQL'i sağlamak ve sonuçları çıkarmak için bırakır. Bu sınıf, SQL sorgularını veya güncellemelerini yürütür, ResultSets üzerinde yineleme başlatır ve JDBC istisnalarını yakalar ve bunları org.springframework.dao paketinde tanımlanan genel, daha bilgilendirici istisna hiyerarşisine çevirir.

Jdbc'den farklı olarak her seferinde bağlantı açıp kapatmaya gerek kalmaz, properties dosyası içerisinde gerekli bilgiler verilerek bağlantı sağlanmış olur. Yazılan sql sorgu stringleri hep jdbc template tarafından execute edilirken bağlantı otomatik olarak sağlanır.

```
public class StudentJDBCTemplate implements StudentDAO {
   private DataSource dataSource;
   private JdbcTemplate jdbcTemplateObject;

   public void setDataSource(DataSource dataSource) {
      this.dataSource = dataSource;
      this.jdbcTemplateObject = new JdbcTemplate(dataSource);
   }
   public List<Student> listStudents() {
      String SQL = "select * from Student";
      List <Student> students = jdbcTemplateObject.query(SQL, new StudentMapper());
      return students;
   }
}
```

Hibernate Nedir?

Hibernate, açık kaynak GNU Kısıtlı Genel Kamu Lisansı (LGPL) kapsamında lisanslanan ve indirmesi ücretsiz olan yüksek performanslı bir Nesne/İlişkisel kalıcılık ve sorgulama hizmetidir. Hibernate, yalnızca Java sınıflarından veritabanı tablolarına (ve Java veri türlerinden SQL veri türlerine) eşlemeyle ilgilenmekle kalmaz, aynı zamanda veri sorgulama ve alma olanakları da sağlar. Bu öğretici, basit ve kolay adımlarla veritabanı tabanlı web uygulamalarınızı geliştirmek için Hibernate'i nasıl kullanacağınızı öğretecektir.

Hibernate xml formatında gerekli maplemeleri yaparak daha sonrasında hazır sorguları çağırabileceğimiz bir araç oluşturur. Basit CRUD işlemleri otomatik olarak oluşturulmakla birlikte istenildiği takdirde ekstra sorgular yazarak sorgularımızı xml dosyası içerisinden yönetip kullanabiliriz.

```
<hibernate-mapping>
   <class name = "Employee" table = "EMPLOYEE">

      <meta attribute = "class-description">
         This class contains the employee detail.
      </meta>

      <id name = "id" type = "int" column = "id">
         <generator class="native"/>
      </id>

      <property name = "firstName" column = "first_name" type = "string"/>
      <property name = "lastName" column = "last_name" type = "string"/>
      <property name = "salary" column = "salary" type = "int"/>

   </class>
</hibernate-mapping>
```

```java
/* Method to CREATE an employee in the database */
public Integer addEmployee(String fname, String lname, int salary){
   Session session = factory.openSession();
   Transaction tx = null;
   Integer employeeID = null;

   try {
      tx = session.beginTransaction();
      Employee employee = new Employee(fname, lname, salary);
      employeeID = (Integer) session.save(employee);
      tx.commit();
   } catch (HibernateException e) {
      if (tx!=null) tx.rollback();
      e.printStackTrace();
   } finally {
      session.close();
   }
   return employeeID;
}
```