**Hibernate-JPA**

**Hibernate**:

java nesnelerin saklandığı bir veri tabanı freamworküdür.

Düşük seviye sql komutlarını destekler, projede kullanılan JDBC kod miktarını azaltır.

Hibernate, ORM (ojbect relations mapping) kullanır ve geliştiricinin oluşturduğu nesnelerin veri tabanında nasıl tutulacağını ve kullanılacağını belirlemesini sağlar. Oluşturduğumuz bir class’ı veritabanındaki bir tablo ile ilişkilendirebiliriz.

**JPA:**

Nesne ilişkilendirme ve ORM için standart API’ dır. Bir arayüz tanımlar ve kullanılabilmesi için bir uygulama gerektirir. Oluşturduğumuz nesneyi veritabanında ilgili tablonun sütunlarına doğru bir şekilde kaydeder. Sql komutlarını otomatik olarak ayarlar ve kayıt işlemini gerçekleştirir.

**Spring jdbc ile veri tabanına bağlantı:**

spring.datasource.url=jdbc:mysql://localhost:3306/student\_tracker  
spring.datasource.username=springstudent  
spring.datasource.password=springstudent

application properties dosyasına eklenen bu bilgiler ile veri tabanı bağlantısı sağlanabilir, datasource komutu ile verilen bilgiler arka planda spring tarafından işlenir ve bağlantı işlemini spring gerçekleştirir.

**Spring banner kapatma:**

Uygulama her çalıştırıldığında consol ekranında spring projesinde çalıştığımızı bildiren bannerı kapatmak için application properties dosyasına

spring.main.banner-mode=off

komutunu eklemek yeterlidir.

**Spring Boot loglama seviyesini değiştirmek:**

Proje dosyası her çalıştırıldığında consol ekranına basılan spring boot loglarını sadece hataları ve uyarıları gösterecek şekilde ayarlayabiliriz. Bunun için yaplıması gereken şey application properties dosyasına aşağıdaki komutu eklemektir.

**JPA Dev Process- To Do List**

1-) Java sınıfına açıklama ekleme

2-) Veri tabanı işlemleri gerçekleştirmek için java kodu gerçekleştirme

**JPA-Hibernate**

JPA arka planda hibernate kullanarak veri tabanı işlemleri gerçekleştirir.

**Entity Class**

Entity sınıfı veri tabanında bir tablo ile ilişkilendirilmiş sınıflardır. Entity sınıfı en az bir public veya protected yapıda no-arg bir kurucu metoda sahip olmalıdır, bu minimum gerekendir normal yapıda kurucu metodlara sahip olabilir.

**@Column anotasyonu**

Bu anotasyonu kullanmak zorunlu değildir fakat kullanılmadığı takdirde entity sınıfı içerindeki değişkenlerin güncellenmesi v.b. durumlarda bu adlar ile tablodaki kolon adları otomatik değişmez, küçük projelerde sorun yaratmayabilir fakat büyük projelerde sorunlara sebep olur.

**Primary Key**

Bir veri tabanı tablosunda, satırlarında bulunan her bir değerin benzersiz olduğu kolonlar primary keydir. Springde bunu sağlamanın yolu @GeneratedValue(strategy=GenerationType.IDENTITY) anotasyonu ile sağlanır.

**ID Generation Strategies**

**GenerationType.AUTO**

Belirli bir veritabanı için uygun bir strateji seçin

**GenerationType.IDENTITY**

veritabanı kimlik sütununu kullanarak birincil anahtarları atama

**GenerationType.SEQUENCE**

bir veritabanı sırası kullanarak birincil anahtarları atama

**GenerationType.TABLE**

benzersizliği sağlamak için temel bir veritabanı tablosu kullanarak birincil anahtarları atayın

**Özel Generation strateg oluşturma**

Org.hibernate.id.IdentifierGenerator

İmplementasyonu ve public Serializable generate metodunun override edilmesiyle özel strateji belirleyebiliriz.

**Spring @Transactional**

JPA koda için otomatik olarak işlem başlatır ve bitirir.

**JPQL**

Veri tabanı sorguları çalıştırmak için kullanılan JPA özelliğidir, kullanım örneği:

TypedQuery<Student> theQuery=entityManager.createQuery(“FROM Student”, Student.class);

List<Student> students = theQuery.getResultList();

Buradaki FROM student tablo adı değil entity adıdır, JPQL varlık(entitty) adları ve varlık alanları (entity fields)’lar ile çalışır.

TypedQuery<Student> theQuery=entityManager.createQuery(“FROM Student WHERE lastName=’Doe’ OR firstName=’Daffy’ ”, Student.class);

Burada lastName ve firstName tablodaki sütun adları değil varlık adlarıdır, Student entity sınıfı içerisinde tanımlanan değişken adları ile sorgulama yapılır.

List<Student> students = theQuery.getResultList();

**Updating Objects with JPA**

Öncelikle Veri tabanı bağlantısı için kullanılan DAO sınıfının implemet ettiği interface e Update methodu eklenmeli, artından bu metod implemente edilen sınıflarda override edilmelidir. Metod içeriği ve kullanımı için örnek kodlar:

void update(Student theStudent);

ile interface metodu tanımlama.

@Override  
@Transactional  
public void update(Student theStudent) {  
 entityManager.merge(theStudent);  
}

Metodumuzu bu şekilde override etmemiz gerekir. Burada **@Transactional** anotasyonunu kullanma sebebimiz, veri tabanında bir güncelleme işlemi yapmamızdan kaynaklanır. Eğer güncelleme esnasında bir sorun olursa yapılan işlemlerin hiç yapılmamış gibi olmasını sağlar kısaca hata durumunda tabloyu eski haline geri döndürür. Metod içerisinde entityManager’in merge metodunun kullanılma sebebi ise bu metodun yapılan değişiklikleri güncellemesidir. Kendisine verdiğimiz nesneyi tabloda ilgili satıra güncellenmiş şekilde yerleştirir.

int studentId=1;  
Student myStudent=studentDAO.findById(studentId);  
myStudent.setFirstName("John");  
studentDAO.update(myStudent);

Burada öncelikle id’si 1 olan öğrencinin bilgilerini myStudent objesine çekeriz, daha sonra set metodu ile firstname’i değiştiririz. Yaptığımız bu değişikliği veri tabanına kaydetmek için update metodunu kullanırız, bu metod da entityManager’ın maerge metodunu kullanarak yapılan değişikliği veri tabanına yazar.

**Delete**

Delete işlemi için aşağıdaki kodu kullanabiliriz bu yalnızca bir satır üzerinde işlem yapmayı sağlar.

Student theStudent = entityManager.find(Student.class,id);

entityManager.remove(theStudent);

Birden fazla satır için veya İD harici farklı bir özelliğer göre silme işlemi gerçekleştirmek için aşağıdaki yöntem kullanılır

entityManager.createQuery("DELETE FROM Student").executeUpdate();

bu kod geriye sayısal bir değer döndürür, eğer istersek sayısal bir değişken ile kaç satır silindiğini elde edebiliriz. Silmeyle ilgili diğer kısımlar güncelleme ile aynıdır.

**Create Database Tables From Code**

Buraya açıklama eklenecek…