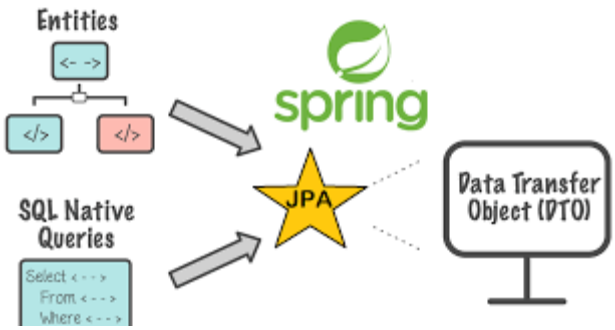
HW#4

1 – What is JPA ?

JPA , Java Persistence API (Uygulama Programlama Arayüzü) anlamına gelir. İlk olarak 11 Mayıs 2006'da piyasaya sürüldü. ORM araçlarına bazı işlevsellik ve standart sağlayan bir Java özelliğidir. Java nesneleri ve ilişkisel veritabanları arasındaki verileri incelemek, kontrol etmek ve sürdürmek için kullanılır. Nesne İlişkisel Haritalama için standart bir teknik olarak görülmektedir.
Nesne yönelimli bir model ile ilişkisel bir veritabanı sistemi arasında bir bağlantı olarak kabul edilir. Java'nın bir özelliği olduğu için JPA herhangi bir işlevi kendi başına yürütmez. Bu nedenle uygulamaya ihtiyacı vardır. Bu nedenle, Hibernate gibi veri kalıcılığı için ORM araçları JPA özelliklerini uygular. Veri kalıcılığı için **javax.persistence** paketi JPA sınıflarını ve arabirimlerini içerir.
JPA'nın aşağıdaki gibi bazı temel özellikleri;

- JPA yalnızca bir belirtimdir, bir uygulama değildir.
- Nesne-ilişkisel eşlemeyi uygulamak için arayüzleri ayarlamak için bir dizi kural ve kılavuzdur.
- Birkaç sınıfa ve arayüze ihtiyacı var.
- Basit, daha temiz ve özümsenmiş nesne-ilişkisel eşlemeyi destekler.
- Polimorfizmi ve kalıtımı destekler.
- Dinamik ve adlandırılmış sorgular JPA'ya dahil edilebilir.



Spring Data JPA, JPA tabanlı depoların uygulanmasını kolaylaştırır. JPA tabanlı veri erişim katmanları için desteği geliştirir. Veri erişim teknolojilerini kullanan Spring destekli uygulamalar oluşturmayı kolaylaştırır. Spring Data JPA, daha büyük Spring Data ailesinin bir parçasıdır.

---

2 - What is the naming convention for finder methods in the Spring data repository interface ?

Aşağıdaki tabloda JPA için desteklenen anahtar sözcükler ve bu anahtar sözcüğü içeren bir yöntemin ne anlama geldiği açıklanmaktadır:

Keyword	Sample	JPQL snippet
And	findByLastnameAndFirstname	… where x.lastname = ?1 and x.firstname = ?2
Or	findByLastnameOrFirstname	… where x.lastname = ?1 or x.firstname = ?2
Is,Equals	findByFirstname , findByFirstnameIs , findByFirstnameEquals	… where x.firstname = ?1
Between	findByStartDateBetween	… where x.startDate between ?1 and ?2
LessThan	findByAgeLessThan	… where x.age < ?1
LessThanEqual	findByAgeLessThanEqual	… where x.age <= ?1
GreaterThan	findByAgeGreaterThan	… where x.age > ?1
GreaterThanEqual	findByAgeGreaterThanEqual	… where x.age >= ?1
After	findByStartDateAfter	… where x.startDate > ?1
Before	findByStartDateBefore	… where x.startDate < ?1
NotLike	findByFirstnameNotLike	… where x.firstname not like ?1
StartingWith	findByFirstnameStartingWith	… where x.firstname like ?1 (parameter bound with appended %)
EndingWith	findByFirstnameEndingWith	… where x.firstname like ?1 (parameter bound with prepended %)
Containing	findByFirstnameContaining	… where x.firstname like ?1 (parameter bound wrapped in %)
OrderBy	findByAgeOrderByLastnameDesc	… where x.age = ?1 order by x.lastname desc
Not	findByLastnameNot	… where x.lastname <> ?1
In	findByAgeIn(Collection<Age> ages)	… where x.age in ?1
NotIn	findByAgeNotIn(Collection<Age> ages)	… where x.age not in ?1
True	findByActiveTrue()	… where x.active = true
False	findByActiveFalse()	… where x.active = false
IgnoreCase	findByFirstnameIgnoreCase	… where UPPER(x.firstame) = UPPER(?1)

---

3 - What is PagingAndSortingRepository ?

PagingAndSortingRepositoryCrudRepository , sayfalandırma ve sıralama soyutlamayı kullanarak varlıkları almak için ek yöntemler sağlamanın bir uzantısıdır . İki yöntem sağlar:

- **Page findAll(Pageable pageable)**– nesnede sağlanan disk belleği kısıtlamasını karşılayan varlıkların bir sayısını döndürür Pageable.
- **Iterable findAll(Sort sort)** – verilen seçeneklere göre sıralanmış tüm varlıkları döndürür. Burada sayfalama uygulanmaz.

---

4 - Differentiate between findById() and getOne() ?

Spring Data JPA'da getOne() metodu da findById() metodu da bir objeyi getirmek için kullanılabilir. Ancak bu iki metodun çalışma prensibi farklıdır.

- getOne()

getOne() metodu verilen id'ye ait objenin referansını döner. Bu metod içeride EntityManager.getReference() metodunu çağırır. Bu metod her zaman database'e gitmeden (lazy fetch) bir proxy döner. İstenen entity'nin database'de bulunmaması durumunda EntityNotFoundException fırlatır.

- findById()

Bu metod ise her çağırıldığında gerçekten database'e gider ve objeyi oradan getirir. Bu EAGER yüklenen işlemdir bu sebeple eğer getirmeye çalıştığımız obje DB'de yoksa null dönecektir.

Hangisini seçmeliyiz ?

Bu iki metodun arasındaki temel fark performans ile ilgili, Lazy Load olan getOne() JVM'den database'e gitmediği için performans olarak findById()'den daha iyidir.

İki metodu da kullanmak için uygun durumlar var. Örneğin objeyi getirip başka bir obje ile ilişki kurmasını sağlamak istediğimizde (OneToOne ya da ManyToOne). Ancak örneğin çağırdığımız objenin içerisinde bulunan başka bir objeye erişmeye çalıştığımızda hata alacağız bu gibi durumlarda findById() metodunu kullanmak daha iyi olacaktır.

---

5 - What is @Query used for ?

Veritabanında sorgulama işlemine denir.

---

6 - What is lazy loading in hibernate ?

"Tembel yükleme", bir varlığın yalnızca varlığa **gerçekten** *ilk* kez eriştiğinizde yükleneceği anlamına gelir .

Desen şöyle :

```
public Entity getEntity() {
   if (entity == null) {
      entity = loadEntity();
   }
   return entity;
```

}
Bu, büyük bir veri kümesindeki tüm varlıkları önceden yükleme/ön doldurma maliyetinden tasarruf sağlarken, sonuçta *hepsine* gerçekten ihtiyacınız yok .

---

7 – What is SQL injection attack ? Is Hibernate open to SQL injection attack ?

Lazy loading yüklemesi olarak da bilinen tembel yükleme, Hazırda Bekletme modundaki tüm varlıklar için kullanılan bir alma tekniğidir. Üst sınıf nesnesini yüklerken bir alt sınıf nesnesinin yüklenip yüklenmeyeceğine karar verir. Hazırda Bekletme modunda ilişki eşleme kullandığımızda, getirme tekniğini tanımlamamız gerekir. Tembel yüklemenin temel amacı, gerekli nesneleri veritabanından getirmektir.

Hibernate, devralmayı destekler; bu, ebeveyn çocuk ilişkisine sahip kayıtlar anlamına gelir. Bir ebeveynin birden fazla çocuk kaydına sahip olduğu durumu düşünün. Artık Hazırda Bekletme tembel yüklemeyi kullanabilir, bu da tüm sınıfları değil yalnızca gerekli sınıfları yükleyeceği anlamına gelir. Varlık gerektiğinde yalnızca bir kez yüklendiğinden büyük bir yükü önler. Tembel yükleme, gereksiz hesaplamaları önleyerek performansı artırır ve bellek gereksinimlerini azaltır.

---

8 - What is criteria API in hibernate ?

Hibernate Query Language (HQL) ile çalışırken, veri tabanından veri okumak için HQL sorgularını manuel olarak hazırlıyoruz. Daha iyi performans için her zaman bir sorgunun ayarlanmış sorgu olarak yazılması önerilir. HQL durumunda, ayarlanmış sorguları hazırlamamız gerekir. Hazırda Bekletme, yalnızca HQL'yi SQL'e çevirir ve ardından yürütür, ancak sorguları ayarlamaz. HQL için sorguları ayarlama sorumluluğu geliştiriciye bağlıdır. HQL sorgularını yazmak ve bunları açıkça ayarlamak yerine Hibernate Criteria API'sini kullanabiliriz.

---

9 - What Is Erlang? Why is it «required» for RabbitMQ ?

- RabbitMQ **Erlang** dili ile geliştirilmiştir. Open Source(Açık Kaynak) ‘dur. Ve yaygın bir biçimde kullanılmaktadır.

- RabbitMQ, bir mesaj kuyruk sistemidir.

- RabbitMQ,bir uygulamadan mesajı alıp ; diğer uygulamaya iletilebilir.

- RabbitMQ’yu kargo firması olarak düşünebilirsiniz. Kargo firmasına kargo gelir. Gelen bu kargo, kargo firmasında bekletilir. Yeri ve zamanı geldiğinde ise alıcıya teslimatı sağlanır. RabbitMQ’nun da yaptığı işlem bunun ile benzerlik gösterir. Mesaj gelir ve bu mesaj RabbitMQ tarafından kendi sistemine alınır. Yeri ve zamanı geldiğinde ise bu mesaj iletilir.

- Farklı işletim sistemlerine kurulabilir.

- RabbitMQ gibi mesaj kuyruk sistemlerine **Message Broker** adı verilir. RabbitMQ mesaj kuyruk sistemlerinden sadece bir tanesidir. **Kafka**,**MSMQ** gibi Message Broker çeşitleri de bulunmaktadır.

- Ölçeklendirilebilir,kararlı ve kolaylılığı nedeniyle RabbitMQ; en çok kullanılan kuyruk sistemidir.

- **Exchange** ve **Queue** olmak üzere iki yapısı bulunmaktadır.

- **Exchange ;** gelen mesajı alır, ilgili Route ile kuyruklara yönlendirir. **Message Broker**'a mesajlar **Publisher**(mesajı üreten kısım) tarafından gelmektedir. Mesaj geldikten sonra mesajı ilk karşılayan **Exchange** yapısıdır. **Exchange** gelen mesaj **Queue** kısmına iletilir.

- Queue kısmında ise mesajlar sıralanır. Mesajların çıkışı **"First In, First Out"** mantığında çalışmaktadır. Yani **"İlk giren mesaj, ilk çıkar."**

- Peki bu mesajları kim tüketir ? Consumer adı verilen bir uygulama sayesinde mesajlar tüketilir. **Publisher'**ı Java ile geliştirilen bir sistemde ,**Consumer** C# veya başka bir dille geliştirilebilir. Bu durumda herhangi bir sıkıntı yaşanmaz.

- **Consumer** mesajı aldıkça **Queue**'deki mesaj sayısı düşer ve **Queue**'deki sıradaki mesaj **Consumer**'a iletilir.

- RabbitMQ sayesinde; web sitesi veya mobil uygulaması farketmeksizin, anlık yapılmayan işlemlerin, asenkron şekilde gerçekleştirilmesini ve uygulamayı kullanılan kişilere gereksiz bir response time zamanına maruz bırakılmamasını sağlamış oluruz.

- Kullanıcı siteye üye olduğunda üyelik maili atıldığını düşünelim. Bunu site üzerinden yapmak yoğunluğa sebep olabilir. O nedenle RabbitMQ ile kuyruğa atıp, gecikme süresini azaltabiliriz. Aslında işlem Producer'a değil, arka tarafta çalışan uygulamaya yani Consumer'a yaptırılmış olur

---

10 – What is the JPQL ?

JPQL (Java Persistence Query Language) , JPA standardının Entity nesnelerini sorgulamak üzerine tanımladığı bir dil. JPQL, HQL (Hibernate Query Language) 'e fazlasıyla benzeşir. Bu diller SQL (Structured Query Language) diline hemen hemen benzemelerine karşın, kullandığı argümanlar veritabanı tabloları yerine Entity nesneleridir.

---

11 – What are the steps to persist an entity object ?

1. Create an *entity manager factory* object. The EntityManagerFactory interface present in java.persistence package is used to provide an entity manager. EntityManagerFactory emf=Persistence.createEntityManagerFactory("Employee_details");

2. Get an *entity manager* from the factory.

   EntityManager em=emf.createEntityManager();

3. Initialize an entity manager.

   em.getTransaction().begin();

4. Persist the data into the relational database.

   em.persist(e1);

5. Close the transaction

   em.getTransaction().commit();

6. Finally, release the factory resources.

7. emf.close();
   em.close();

---

12 – What are the different types of entity mapping ?

Bire bir eşleme : Bir varlığın örneğinin başka bir varlığın örneğiyle ilişkilendirildiği tek değerli bir ilişkiyi temsil eder. Bu tür ilişkilendirmede, kaynak varlığın bir örneği, hedef varlığın en fazla bir örneği ile eşlenebilir. Örneğin, bir kişinin yalnızca bir pasaportu olabilir ve bir kişiye bir pasaport atanır.

Bire Çok eşleme : Bu tür eşleme, bir varlığın diğer varlıkların bir koleksiyonuyla ilişkilendirildiği koleksiyon değerli ilişki kategorisine girer. Bu tür ilişkilendirmede, bir varlığın örneği, başka bir varlığın herhangi bir sayıda örneği ile eşlenebilir. Örneğin, bir Kişi örneğinin birden fazla Banka Hesabı örneği olabilir, ancak bir banka hesabı örneğinin hesap sahibi olarak en fazla bir kişi örneği olabilir. Bir takımı yöneten tek bir yönetici varken, birden fazla takımı yöneten çalışanlar var".

Çoktan bire eşleme : Bu eşleme türü, bir varlık koleksiyonunun benzer varlıkla ilişkilendirilebileceği tek değerli bir ilişkiyi temsil eder. İlişkisel veritabanında, bir varlığın birden fazla satırı, başka bir varlığın aynı satırına başvurabilir. Bire çok eşleme ile ilgilidir, ancak fark perspektiften kaynaklanmaktadır. Örnek olarak, herhangi bir sayıda kredi kartı bir müşteriye ait olabilir ve herhangi bir kredi kartı olmayan bazı müşteriler olabilir, ancak bir sistemdeki her kredi kartının bir çalışanla ilişkilendirilmesi gerekir (toplam katılım gibi). Tek bir kredi kartı birden fazla müşteriye ait olamazken.

Çoktan çoğa eşleme : Çoktan Çoka eşleme, herhangi bir sayıda varlığın diğer varlıkların bir koleksiyonuyla ilişkilendirilebileceği, koleksiyon değerli bir ilişkiyi temsil eder. İlişkisel veritabanında, bir varlığın birden fazla satırı, başka bir varlığın birden fazla satırına başvurabilir. Bir kişinin birden

fazla yeteneği olabilir. Bir beceriyi birden fazla kişi elde edebilir. Bir müşteri herhangi bir sayıda ürünü satın alabilir ve bir ürün birçok müşteri tarafından satın alınabilir.

---

13 - What are the properties of an entity ?

Persistability : Bir nesne, veritabanında depolanıyorsa ve herhangi bir zamanda erişilebilirse kalıcı olarak adlandırılır.

Persistent Identity : Java'da her varlık benzersizdir ve bir nesne kimliğini temsil eder. Benzer şekilde, nesne kimliği bir veritabanında depolandığında, kalıcı kimlik olarak temsil edilir. Bu nesne kimliği, veritabanındaki birincil anahtarla aynıdır.

Transactionality: Varlık oluşturma, silme, güncelleme gibi çeşitli işlemleri gerçekleştirebilir. Her işlem veritabanında bazı değişiklikler yapar. Veritabanında yapılan her türlü değişikliğin atomik olarak başarılı veya başarısız olmasını sağlar.

Granularity :: Varlıklar, ilkel, ilkel sarmalayıcılar veya tek boyutlu duruma sahip yerleşik nesneler olmamalıdır.

---

14 - Difference between CrudRepository and JpaRepository in Spring Data JPA?

JpaRepository NAME uzanır PagingAndSortingRepository NAME sırayla uzanan CrudRepository NAME .

Başlıca işlevleri:

- CrudRepository NAME , temel olarak CRUD işlevlerini sağlar.
- PagingAndSortingRepositoryNAME , sayfa numaralandırma ve sıralama yapmak için yöntemler sağlar.
- JpaRepository NAME , kalıcılık bağlamını temizleme ve bir toplu işteki kayıtları silme gibi JPA ile ilgili bazı yöntemler sunar.

nedeniyle, JpaRepositoryCrudRepository ve PagingAndSortingRepository işlevlerinin tümüne sahip olur. Öyleyse JpaRepository ve PagingAndSortingRepository tarafından sağlanan işlevlere sahip olmak için depoya ihtiyacınız yoksa, CrudRepository öğesini kullanın.

Kaynak:

1. https://www.geeksforgeeks.org/java-jpa-vs-hibernate/
2. https://www.javaguides.net/2018/11/spring-data-jpa-query-creation-from-method-names.html
3. https://www.programmersought.com/article/43494177214/
4. https://stackoverflow.com/questions/2192242/what-is-lazy-loading-in-hibernate
5. https://ruveydakardelcetin.medium.com/rabbitmq-nedir-ni%C3%A7in-kullanmal%C4%B1y%C4%B1z-rabbitmq-kurulumu-a3037c0af971
6. https://kodedu.com/2012/10/jpql-sorgulama-dili/