1- What is JPA

Spring Boot JPA, Java uygulamalarındaki ilişkisel verileri yönetmek için bir Java spesifikasyonudur. Java nesnesi/sınıfı ve ilişkisel veritabanı arasındaki verilere erişmemizi ve bunları sürdürmemizi sağlar. JPA, Nesne-İlişki Eşleme'yi (ORM) takip eder. Bir dizi arayüzdür. Ayrıca, nesneler üzerindeki sorguları ve işlemleri veritabanına karşı işlemek için bir çalışma zamanı EntityManager API'si sağlar. Platformdan bağımsız nesne yönelimli bir sorgu dili JPQL (Java Persistent Query Language) kullanır.

Kalıcılık bağlamında, üç alanı kapsar:

Java Kalıcılık API'si

Nesne-İlişkisel meta veriler

Kalıcılık paketinde tanımlanan API'nin kendisi

JPA bir framework değildir. Herhangi bir framework tarafından uygulanabilecek bir kavramı tanımlar.

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

limit – sorgunun sonucu, ilk/üst anahtar kelimenin kullanımıyla sınırlandırılabilir

- findFirst10ByLastname
- findFirstByOrderByLastnameAsc
- findTop3ByLastname
- findTopByOrderByAgeDesc

özellik/özellikler ifadesi - sonuç, varlığın özelliğine göre filtrelenir, And, Or anahtar sözcüğü kullanılarak birden çok özellik kullanılabilir

- findByLastnameAndFirstname
- findByLastnameOrFirstname
- findByFirstname

karşılaştırma – filtreleme için kullanılan özellik belirlendikten sonra karşılaştırma modu belirtilebilir

- findByFirstnameIs
- findByFirstnameEquals
- findByStartDateBetween
- findByAgeLessThan, findByAgeLessThanEqual
- findByAgeGreaterThan, findByAgeGreaterThanEqual
- findByStartDateBefore, findByStartDateAfter

findByAgeIsNull, findByAgeIsNotNull

findByFirstnameLike, findByFirstnameNotLike

findByFirstnameStartingWith, findByFirstnameEndingWith

findByFirstnameContaining

findByLastnameNot

findByAgeIn(Collection ages), findByAgeNotIn(Collection ages)

findByActiveTrue, findByActiveFalse

findByFirstnameIgnoreCase

ordering operator - isteğe bağlı olarak yöntem adının sonunda ordering operatörünü belirtebilirsiniz

findByLastnameOrderByFirstnameAsc

findByLastnameOrderByFirstnameDesc

3-What is PagingAndSortingRepository ?

PagingAndSortingRepository: sayfalandırma ve sıralama soyutlamayı kullanarak varlıkları almak için ek yöntemler sağlamak için CrudRepository'nin bir uzantısıdır. İki yöntem sağlar:

Page findAll(Pageable pageable) – Pageable nesnesinde sağlanan disk belleği kısıtlamasını karşılayan varlıkların bir Sayfasını döndürür.

Iterable findAll(Sort sort) – verilen seçeneklere göre sıralanmış tüm varlıkları döndürür. Burada sayfalama uygulanmaz.

4-Differentiate between findById() and getOne()

- getOne()

getOne() metodu verilen id'ye ait objenin referansını döner. Bu metod içeride EntityManager.getReference() metodunu çağırır. Dökümanda belirtildiği gibi 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.

5-What is @Query used for ?

@Query annotation, bulucu sorgularını doğrudan depo yöntemlerinde bildirir. Etki alanı sınıflarında benzer @NamedQuery kullanılırken, Depo arabiriminde Spring Data JPA @Query ek açıklaması kullanılır. Bu, etki alanı sınıflarını kalıcılığa özgü bilgilerden kurtarır, ki bu iyi bir şeydir.

6-What is lazy loading in hibernate ?

Lazy Loading, Hibernate'deki 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. Lazy loading in temel amacı, gerekli nesneleri veritabanından getirmektir.

Örneğin, bir ebeveyn sınıfımız var ve bu ebeveynin bir alt sınıf koleksiyonu var. Artık Hazırda Bekletme Lazy loading i kullanabilir, bu da tüm sınıfları değil yalnızca gerekli sınıfları yükleyeceği anlamına gelir. Varlık gerektiğinde yalnızca bir kez yüklendiğinden büyük bir yükü önler. Lazy loading ,gereksiz hesaplamaları önleyerek performansı artırır ve bellek gereksinimlerini azaltır.

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

SQL enjeksiyonu, birinin bilginiz olmadan veritabanınızda çalıştırılmak üzere MySQL deyimi eklemesi eylemini ifade eder. Enjeksiyon genellikle bir kullanıcıdan adı gibi bir girdi istediğinde gerçekleşir ve bir isim yerine size bilmeden veritabanınızda çalıştıracağınız bir MySQL ifadesi verirler.

Hazırda Bekletme, SQL Injection'a bağışıklık sağlamaz, api'yi istediği gibi kötüye kullanabilir. HQL (SQL'in Hazırda Bekletme alt kümesi) hakkında onu daha fazla veya daha az duyarlı hale getiren özel bir şey yoktur.

8-What is criteria API in hibernate

Ham SQL yapmadan sorgu yazmamızı sağlar ve ayrıca Hibernate'in ana özelliklerinden biri olan sorgular üzerinde nesne yönelimli kontrol sağlar. Criteria API, farklı türde filtreleme kuralları ve mantıksal koşullar uygulayabileceğimiz, programlı olarak bir ölçüt sorgu nesnesi oluşturmamıza olanak tanır.

Hibernate 5.2'den bu yana, Hibernate Ölçütleri API'si kullanımdan kaldırılmıştır ve yeni geliştirmeler JPA Ölçütleri API'sine odaklanmıştır.

Erlang çalışma zamanı, RabbitMQ tarafından kullanılan bir dizi bileşen içerir. Bu kılavuzla ilgili olarak en önemlileri şunlardır. Erlang sanal makinesi kodu yürütür. epmd, bir ana bilgisayardaki düğüm adlarını düğümler arası bir iletişim bağlantı noktasına çözümler.

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

Erlang

Erlang, genel amaçlı bir programlama dili ve çalışma zamanı ortamıdır. Erlang, eşzamanlılık, dağıtım ve hata toleransı için yerleşik desteğe sahiptir. Erlang, Ericsson'un birçok büyük telekomünikasyon sisteminde kullanılmaktadır. Erlang, http://www.erlang.org adresinde açık kaynak olarak mevcuttur.

OTP nedir?

OTP (Açık Telekom Platformu), Erlang'ın ASN.1'i derlemekten bir WWW sunucusu sağlamaya kadar her şeyi yapması için geniş bir kitaplık koleksiyonudur. "Erlang" kullanan çoğu proje aslında "Erlang/OTP" kullanıyor, yani dil ve kütüphaneler. OTP ayrıca açık kaynak kodludur.

RabbitMQ

RabbitMQ, mesaj komisyoncusu veya kuyruk yöneticisi olarak da bilinen ve Erlang ile yazılmış bir mesaj kuyruğa alma yazılımıdır. Basitçe söylenmiş; Kuyrukların tanımlandığı, uygulamaların mesaj veya mesaj aktarmak için bağlandığı yazılımdır.

10-What is the JPQL ?

JPQL, JPA belirtiminde tanımlanan Java Kalıcılık Sorgu Dilidir. İlişkisel bir veritabanında depolamak için varlıklara karşı sorgular oluşturmak için kullanılır. JPQL, SQL sözdizimine dayalı olarak geliştirilmiştir. Ancak doğrudan veritabanını etkilemez.

JPQL, SELECT yan tümcesini kullanarak bilgi veya veri alabilir, UPDATE yan tümcesini ve DELETE yan tümcesini kullanarak toplu güncellemeler yapabilir. EntityManager.createQuery() API, sorgulama dilini destekleyecektir.

Sorgu Yapısı

JPQL sözdizimi, SQL sözdizimine çok benzer. SQL benzeri sözdizimine sahip olmak bir avantajdır çünkü SQL basit bir yapılandırılmış sorgu dilidir ve birçok geliştirici bunu uygulamalarda kullanır. SQL doğrudan ilişkisel veritabanı tablolarına, kayıtlarına ve alanlarına karşı çalışırken, JPQL Java sınıfları ve örnekleriyle çalışır.

Örneğin, bir JPQL sorgusu, SQL'de olduğu gibi veritabanından alan sonucu kümesi yerine bir varlık nesnesi alabilir. JPQL sorgu yapısı aşağıdaki gibidir.

SELECT ... FROM ...

[WHERE ...]

[GROUP BY ... [HAVING ...]]

[ORDER BY ...]

The structure of JPQL DELETE and UPDATE queries is simpler as follows.
DELETE FROM ... [WHERE ...]
UPDATE ... SET ... [WHERE ...]

11-What are the steps to persist an entity object

Varlık yöneticisi, API'yi uygular ve hepsini tek bir arabirimde kapsüller.
Varlık yöneticisi, bir varlığı okumak, silmek ve yazmak için kullanılır.
Bir varlık tarafından başvurulan bir nesne, varlık yöneticisi tarafından yönetilir.

1)Creating an entity manager factory object

Java.persistence paketinde bulunan EntityManagerFactory arabirimi, bir varlık yöneticisi sağlamak için kullanılır.

```
EntityManagerFactory emf=Persistence.createEntityManagerFactory("Student_details");
```

Persistence - Persistence, EntityManagerFactory arabirimi elde etmek için kullanılan bir önyükleme sınıfıdır.
createEntityManagerFactory() yöntemi - Bu yöntemin rolü, adlandırılmış kalıcılık birimi için bir *EntityManagerFactory* oluşturmak ve döndürmektir. Bu nedenle, bu yöntem Persistence.xml dosyasında geçirilen kalıcılık biriminin adını içerir.

2)Obtaining an entity manager from factory.

```
EntityManager em=emf.createEntityManager();
```

EntityManager - EntityManager bir arayüzdür
createEntityManager() yöntemi - Uygulama tarafından yönetilen yeni EntityManager oluşturur

3) Intializing an entity manager.

```
em.getTransaction().begin();
```

getTransaction() yöntemi - Bu yöntem, kaynak düzeyindeki EntityTransaction nesnesini döndürür.
start() yöntemi - Bu yöntem, işlemi başlatmak için kullanılır.

4)Persisting a data into relational database.

```
em.persist(s1);
```

persist() - Bu yöntem, bir örneği yönetilen ve kalıcı hale getirmek için kullanılır. Bu yöntem içinde bir varlık örneği geçirilir.

5) Closing the transaction

```
em.getTransaction().commit();
```

6)Releasing the factory resources.

```
emf.close();
```

em.close();

close() – Bu yöntem fabrika ayarlarını serbest bırakmak için kullanılır.

12-What are the different types of entity mapping ?

One-to-Many

Many-to-One

One-to-One

Many-to-Many

13-What are the properties of an entity

Genel olarak, varlık, tek bir birimde bir araya getirilmiş bir durum grubudur. Davranış eklendiğinde, bir varlık bir nesne gibi davranır ve nesne yönelimli paradigmanın ana bileşeni haline gelir. Dolayısıyla bir varlık, Java Kalıcılık Kitaplığı'nda uygulama tanımlı bir nesnedir.

Entity Özellikleri

Bunlar, bir nesnenin sahip olması gereken bir varlığın özellikleridir:

Kalıcılık(Persistability) - Bir nesne, veritabanında depolanıyorsa ve herhangi bir zamanda erişilebilirse kalıcı olarak adlandırılır.

Kalıcı Kimlik(Persistant Identity) - Java'da her varlık benzersizdir ve bir nesne kimliği olarak temsil edilir. Benzer şekilde, nesne kimliği bir veritabanında depolandığında kalıcı kimlik olarak temsil edilir. Bu nesne kimliği, veritabanındaki birincil anahtara eşdeğerdir.

İşlemsellik(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.

Granuality - Varlıklar, ilkel, ilkel sarmalayıcılar veya tek boyutlu duruma sahip yerleşik nesneler olmamalıdır.

Entity(Varlık) Meta Verileri

Her varlık, onun bilgilerini temsil eden bazı meta verilerle ilişkilendirilir. Veritabanı yerine, bu meta veriler sınıfın içinde veya dışında bulunur. Bu meta veriler aşağıdaki biçimlerde olabilir: -

Açıklama - Java'da ek açıklamalar, meta verileri temsil eden etiketlerin biçimidir. Bu meta veriler sınıfın içinde kalır.

XML - Bu formda, meta veriler XML dosyasında sınıfın dışında kalır.

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

JpaRepository, CrudRepository'yi genişleten PagingAndSortingRepository'yi genişletir.CrudRepository temel olarak CRUD işlemlerini sağlar.

PagingAndSortingRepository, kayıtların sayfalandırılması ve sıralanması için yöntemler sağlar.JpaRepository, kalıcılık bağlamını temizleme ve toplu olarak kayıtların silinmesi gibi JPA ile ilgili yöntemler sağlar.

Miras yapıları nedeniyle JpaRepository, CrudRepository ve PagingAndSortingRepository'nin tüm davranışlarına sahip olacaktır. Bu nedenle, JpaRepository ve PagingAndSortingRepository tarafından sağlanan işlevlere sahip olmak için depoya ihtiyacınız yoksa, CrudRepository kullanın.