1 – What is JPA ?

JPA veya Java Persistance API verileri kalıcı olarak saklamak, almak, bulmak, silmek ve güncellemek için kullanılan Java ORM şartnamesidir.

ORM nedir?

ORM veya Object Relational Mapping veritabanı ve programlama dili arasında ilişki kurarak, her bir alanı programlama diline uygun veri yapısıyla eşleştirmek için kullanılan bir yapıdır.

OOP tabanlı programlama dili ile veritabanı işlemleri yapan bir uygulama geliştirirken ilk olarak verilerin veritabanından alınarak programlama diline ait veri yapısı ile ifade edilmesi/dönüştürülmesi gerekir.

Bu dönüşüm sırasında veritabanındaki sütun adlarının değişmesi, veri tipinin değişmesi, uygun veri türüne dönüşüm yapılamaması gibi çeşitli hatalar ile karşılaşılır.

ORM araçları genellikle veritabanında yer alan tabloları sınıflarla ifade edererek eşlemeyi yapar.

Neden JPA

Java programlama dili Apache, Eclipse, Redhat ve çeşitli geliştiriciler tarafından geliştirilen ORM araçlarına sahiptir.

Bu ORM araçlarına örnek olarak Hibernate, EclipseLink, MyBatis verilebilir.

Aynı işi farklı metot ismi-yöntemlerle yapan ORM araçları belirli bir standartın olmayışı kütüphane karmaşıklığına neden olmaktadır.

JCP Java topluluğu bu karmaşıklığı ortadan kaldırmak için Java ORM kütüphane standartlarını JPA olarak belirlemiştir.

Bu sayede kullanılan ORM aracı ister Hibernate ister EclipseLink olsun aynı metot adları aynı sonucu verecektir.

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

limit – result of the query can be limited by usage of first/top keyword

findFirst10ByLastname

findFirstByOrderByLastnameAsc

findTop3ByLastname

findTopByOrderByAgeDesc

property/properties expression – result will be filtered based on property of entity, multiple properties can be used with usage of And, Or keyword

findByLastnameAndFirstname

findByLastnameOrFirstname

findByFirstname

comparison – comparison mode can be specified after specifying property used for filtering

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 – optionally you can specify ordering operator at the end of method name

```
    findByLastnameOrderByFirstnameAsc
    findByLastnameOrderByFirstnameDesc
```

3 - What is PagingAndSortingRepository ?

Sıralama ve sayfalama – Sorting and pagination

Spring Data Commons tarafından sağlanan PagingAndSortingRepository arayüzü kullanılır.

```
public interface PagingAndSortingRepository<T, ID> extends
CrudRepository<T, ID> {
    Iterable<T> findAll(Sort sort);
    Page<T> findAll(Pageable pageable);
}
```

Sıralama işleminde Sort sınıfı tarafında saylanan metotlar kullanılır.

Aşağıda sıralama özelliğine ait örnek kullanım yer almaktadır.

```
personRepository.findAll(Sort.by("id"))
        .forEach(System.out::println);
personRepository.findAll(Sort.by(Sort.Direction.DESC, "firstName"))
        .forEach(System.out::println);
personRepository.findAll(Sort.by("firstName").descending())
        .forEach(System.out::println);
personRepository.findAll(Sort.by("firstName", "LastName"))
        .forEach(System.out::println);
```

```
personRepository.findAll(Sort.by("firstName").and(Sort.by("lastName")))
        .forEach(System.out::println);
```

Sıralama işlemi sınıf türetilmiş metotlarla birlikte kullanılabilir.

```
Iterable<Person> findFirst3ByLastName(String lastName, Sort sort);
```

NOT: Türetilmiş metotlar ayrıca OrderByAlanAsc veya OrderByAlanDesc anahtar kelimeleri ile sıralama yapmayı sağlar.

Sayfalama işlemi için PageRequest sınıf tarafından sağlanan metotlar kullanılır.

```
PageRequest of(int page, int size);
PageRequest of(int page, int size, Sort sort);
```

Aşağıda sıralama özelliğine ait örnek kullanım yer almaktadır.

```
personRepository.findAll(PageRequest.of(0, 2))
        .forEach(System.out::println);
personRepository.findAll(PageRequest.of(0, 2, Sort.by("firstName")))
        .forEach(System.out::println);

Page<Person> persons = null;
Pageable pageable = PageRequest.of(0, 2, Sort.by("firstName"));
while (true) {
    persons = personRepository.findAll(pageable);
    int number = persons.getNumber();
    int numberOfElements = persons.getNumberOfElements();
    int size = persons.getSize();
    long totalElements = persons.getTotalElements();
    int totalPages = persons.getTotalPages();
    System.out.printf("Sayfa %s,%n"
```

```
            + "Sayfadaki eleman sayısı: %s,%n"
            + "Boyut: %s,%n"
            + "Toplam eleman sayısı: %s,%n"
            + "Toplam Sayfa sayısı: %s%n",
            number, numberOfElements, size, totalElements, totalPages);
    persons.forEach(System.out::println);
    //persons.map(p -> p.getFirstName().concat("
").concat(p.getLastName()))
    //        .forEach(System.out::println);
    if (!persons.hasNext()) {
        break;
    }
    pageable = persons.nextPageable();
}
```

Sıralama işlemi için kullanılan arayüz türetilmiş metotlar ile birlikte kullanılabilir.

```
Iterable<Person> findFirst3ByLastName(String lastName, Pageable pageable);
```

Page arayüzü Slice arayüzünü genişleterek eleman sayısı, toplam sayfa sayısı bilgisini sağlar.

Page ve Slice arayüzü Streamable arayüzünü genişlettiğinden dolayı Streamable arayüzünde tanımlı Stream işlemlerini yapmayı sağlar.

NOT: Spring Data sayfalama işlemi sırasında eleman sayısı için ek sorgu kullanır.

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.

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.

findById kullandığımızda

```
    @PostMapping("/findById/{id}")
    public void findByIdCreate(@PathVariable Long id) {
        Employee employee = new Employee();
        employee.setName("MyEmployee");
        Instant start = Instant.now();
        Department department = departmentRepository.findById(id).get();
        Instant end = Instant.now();
        System.out.println(Duration.between(start, end));
        employee.setDepartment(department);
        employeeRepository.save(employee);
```

```
    }
```

insert yapmadan önce select sorgusu ile Departmanı getiriyoruz arkasından insert metodu 0.004S'de çalışıyor.

```
Hibernate: select department0_.id as id1_0_0_, department0_.name as name2_0_0_ from Department department0_ where department0_.id=?
PT0.004S
Hibernate: insert into Employee (department_id, name) values (?, ?)
```

getOne kullandığımızda

```
    @PostMapping("/getOne/{id}")
    public void getOneCreate(@PathVariable Long id) {
        Employee employee = new Employee();
        employee.setName("MyEmployee");
        Instant start = Instant.now();
        Department department = departmentRepository.getOne(id);
        Instant end = Instant.now();
        System.out.println(Duration.between(start, end));
        employee.setDepartment(department);
        employeeRepository.save(employee);
    }
```

ise Departman'ın referansını getirdiği için yani aslında database'e gitmediğimiz için işlem çok daha hızlı gerçekleşir.

```
PT0.002S
Hibernate: insert into Employee (department_id, name) values (?, ?)
```

5 - What is @Query used for ?

Türkçe karşılığı “sorgu” olan Query, basitçe, bilgi için yapılan bir istektir. Bilginin bir veritabanından alınmasıdır. Bu veri manipülasyonu için oldukça kullanışlıdır – özellikle veriyi eklemek, silmek ve değiştirmek için.Ancak, rastgele bir ‘istek’ yazılmaz. Sorgunuzu bir takım ön tanımlı kodlar üzerine yazarsınız, böylece veritabanı talimatı anlar. Bu koda sorgu dili denir.

Veritabanı yönetimin standartı Structured Query Language (SQL) – Yapılandırılmış Sorgu Dili’dir. Unutmayın, SQL MySQL’den farklıdır: ilki bir sorgu dili, ikincisi ise bu dili kullanan yazılımdır. Veritabanı yazılımları arasında en popüler tercihin SQL olduğu doğru olmasına rağmen tek sorgu dili SQL değildir. AQL, Datalog ve DMX gibi başka birçok sorgu dili bulunmaktadır.

Ne olursa olsun, bu diller veritabanı iletişimini oldukça kolay hale getirir.

6 - What is lazy loading in hibernate ?

ORM metodolojide veritabanı tabloları, java entityler ile eşleştirilmekte. Böylece tablo nesne ilişkisi kurulmaktadır. Bunların yanında tablolar arasında kurulan ilişkiler, tablolar arası ilişkiyi desteklecek şekilde nesneler arasında da kurulmalıdır. Bu ilişkinin kurulması ORM dünyasında oldukça keyifli ve sistematiktir.

Motivasyon nesnelere erişim, READ aşamasında aşağı doğru bir eğim göstermektedir. Nesneler arası ilişkilerin motivasyon başlangıç noktası ise bu nesnelere erişimin nasıl olacağı yönündedir. Bu noktada tasarım ne kadar iyiyse motivasyon da o kadar iyidir.

Lazy Loading

Bir senaryo üzerinden konuşalım. Elimde customer entity ve customer ile ilişkili address entityleri olsun. Projemin herhangi bir yerinde customer tablosuna ait bilgileri istediğimde address bilgilerinin gelmemesini istiyorum. Ya da belirli koşullar altında address bilgilerini görüntülemek istiyorum. Bu durumda kullandığımız yapının adı lazy loadingdir. Yani asıl nesne, entity çağrısı yapıldığında ilişkili nesnenin gelmemesi durumu.

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

Sql Injection genel olarak sorguları manipüle ederek database üzerinde CRUD işlemleri yapabilmek için izlenen saldırı çeşididir.Bu saldırıyı yapmak için izlenen yollar genellikle querystring üzerinden veya sitede yer alan herhangi bir input üzerinden olabilir.Asp.Net MVC projelerimizde aşağıda yer alan çözümleri uygulayarak kolay bir şekilde çözebiliriz.

Sql Injection korunma yöntemleri ise ;

- ORM (Entity Framework , Hibernate)
- XSS Defence
- Data Validation

Bir örnek üzerinden açıklamaya çalışıp nasıl korunabiliriz bakalım ;

Özellikle herhangi bir Object Relational Mapping (ORM) aracı kullanmıyorsanız özellikle input veya querystring üzerinden gelen ifadeyi direkt sql sorgusuna dahil ediyorsanız çok büyük bir açığınız var demektir.

Entity Framework (EF) kullandığınız takdirde manuel bir sql sorgusu yazmayacağınız ve objeler üzerinden sorgulamalar yapacağınız için dışarıdan gelen herhangi bir Injection denemesi duvara çarpacaktır.

Aşağıda ki gibi bir formunuz var ve basit bir giriş işlemi ;

sql injection korunma

Gelen verileri ise aşağıdaki gibi bir şekilde işleme sokuyorsunuz.Dönen veri var ise giriş işlemi başarılı gibi düşünüyoruz ;

```
String email = request.getParameter("email");
String password = request.getParameter("password");

String sql = "select * from users where (email ='" + email +"' and password ='" + password + "')";
```

view raw

Password değeri istediğimiz şekilde gelirse hiç bir zaman sorun olmayacak.Fakat buna aşağıda ki gibi bir sql injection denemesi yapalım ;

sql injection korunma

Hoop sistemin içindesiniz. Sonda yer alan 1=1 true döndüreceği için önceki şartın artık bir anlamı kalmıyor ve login işlemi başarıyla sonlanarak Sql Injection gerçekleşiyor.

SQL Injection Korunma

Bu tarz saldırılara karşı savunma yapmak için başta Entity Framework ve Linq süper ikilisini projenizde kullanmaya gayret edin.EF ve Linq sizin yerinize bu saldırıları native olarak ekarte etmektedir.

Bir diğer husus ise Sql Injection ve XSS açığı birbiriyle ilişkilidir. Sql Injection saldırılarını önlemek için sitede yer alan XSS açıklarını kapatmak gerekir.Yazdığım yazı Asp.Net Web Güvenliği : XSS Attacks

Bir diğer madde ise Data Validation . Giren girdileri belli formatlara uygun olarak girmeye zorlayarak minimize edebilirsiniz.

8 - What is criteria API in hibernate ?

Hazırda Bekletme, bazı soyutlama katmanları sağlayan bir çerçevedir, yani programcının uygulamalar hakkında endişelenmesine gerek yoktur, Hazırda Bekletme, veritabanıyla bağlantı kurma, CRUD işlemlerini gerçekleştirmek için sorgular yazma vb. gibi uygulamaları sizin için dahili olarak yapar.

Hazırda Bekletme, RDBMS tablolarında bulunan verileri elde etmek için Criteria API aracılığıyla bir yöntemi takip eder. Bu, sonuç kümesini istediğiniz gibi filtrelemenize yardımcı olacaktır. yani bir SQL için “WHERE” yan tümcesini tam olarak nasıl yazdığımız, burada Criteria Query aracılığıyla ele alınabileceği şekilde. Mantıksal işlem, sayfalandırma kavramları, sıralama kavramları, toplama kavramları da Criteria Query ile desteklenir.

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

RabbitMQ bir mesaj kuyruğu sistemidir. Benzerleri Apache Kafka, Msmq, Microsoft Azure Service Bus, Kestrel, ActiveMQ olarak sıralanabilir. Amacı herhangi bir kaynaktan alınan bir mesajın, bir başka kaynağa sırası geldiği anda iletilmesidir. Mantık olarak Redis Pub/Sub’a benzemektedir. Ama burada yapılacak işler bir sıraya alınmaktadır. Yani iletimin yapılacağı kaynak ayağa kalkana kadar, tüm işlemler bir quee’de sıralanabilir. Fakat aynı

durum Redis Pub’Sub için geçerli değildir. RabbitMQ çoklu işletim sistemine destek vermesi ve açık kaynak kodlu olması da en büyük tercih sebeplerinden birisidir.

Peki neden kullanılmalıdır: Bazı işlemlerin anlık yapılmasına ihtiyaç yoktur. Örnek vermek istenir ise sisteme yeni bir haber girildiğinde, ya da var olan bir haberin güncellenmesi anında cache’in düşürülmesi, bir başka örnek de upload edilen”Gif” dosyalarının scale işleminin yapılması gibi düşünülebilir. Hatta zaman ayarlı message ve otomatik mailler de yine RabbitMq’ya güzel bir örnek olabilir. Sıraya alınan bu işlemlerin asenkron bir şekilde yapılması, hem çalışan uygulamanın boş yere bekletilmemesinden hem de sunucu üzerindeki işlem maliyetinin minimuma indirilmesinden dolayı RabbitMQ iyi bir tercih sebebi olabilir. Ayrıca scalable olmasından dolayı da değişen trafikli yapılarda ayrıca tercih edilebilir.

Peki Erlang nedir : Ericsson’da çalışan Joe Armstrong, Mike Williams ve Robert Virding in tarafından geliştirilen aynı zamanda message broker olan RabbitMQ için muhteşem bir platform olan bir programlama dilidir.

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 ?

Steps to persist an Entity using JPA/Hibernate

Add the hibernate dependency. Add the hibernate-entitymanager dependency in the pom. ...

Add the JDBC driver. ...

Add JPA and database config in META-INF/persistence. ...

Create EntityManagerFactory. ...

Mapping the Entity class. ...

Use EntityManager to perform CRUD on Entity. ...

Test the code.

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 ?

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

Their main functions are:

CrudRepository mainly provides CRUD functions.

PagingAndSortingRepository provides methods to do pagination and sorting records.

JpaRepository provides some JPA-related methods such as flushing the persistence context and deleting records in a batch.

Because of the inheritance mentioned above, JpaRepository will have all the functions of CrudRepository and PagingAndSortingRepository. So if you don't need the repository to have the functions provided by JpaRepository and PagingAndSortingRepository , use CrudRepository.