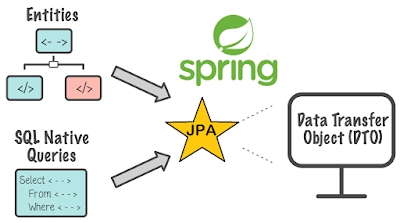
1. What is JPA?  
Answer: JPA stands for Java Persistence API. It is a Java specification used to persist data between the relational database and Java objects. It acts as a bridge between object-oriented domain models and relational databases.  Since interaction with database from Java application is very common, JPA was created to standardize this interaction.

There are many popular JPA implementations available in the Java world like Hibernate. You can further see [**Spring Data JPA using Hibernate**](https://click.linksynergy.com/deeplink?id=JVFxdTr9V80&mid=39197&murl=https%3A%2F%2Fwww.udemy.com%2Fcourse%2Fspring-data-jpa-using-hibernate%2F)to learn more about how to use Hibernate with Spring Data JPA in Java application.

[](https://click.linksynergy.com/deeplink?id=JVFxdTr9V80&mid=39197&murl=https%3A%2F%2Fwww.udemy.com%2Fcourse%2Fspring-data-jpa-using-hibernate%2F)

2. What are some advantages of using JPA?

Answer: Here are some advantages of Java Persistence API or JPA:

* JPA reduces the burden of interacting with databases.
* Annotation in JPA reduces the cost of creating a definition file.
* It is user-friendly.
* JPA providers help merge applications.

3. What is the Spring data repository?  
Answer: Spring data repository is a very important feature of JPA. It helps in reducing a lot of boilerplate code. Moreover, it decreases the chance of errors significantly. This is also the key abstraction that is provided using the Repository interface. It takes the domain class to manage as well as the id type of the domain class as Type Arguments.   
  
4. What is the naming convention for finder methods in the Spring data repository interface?  
Answer:  This is another key feature of Spring Data JPA API which makes writing query method really easy. The finder method should use a special keyword, i.e. "find", followed by the name of the variable. For example, findByLastName().   
  
  
5. Why is an interface not a class?  
Answer: Interface is not a class because it does not contain concrete methods. It can contain only abstract methods.   
  
6. Can we perform actual tasks like access, persist, and manage data with JPA?   
Answer: No, we can't because JPA is only a Java specification.   
  
7. How can we create a custom repository in Spring data JPA?   
Answer: To create a custom repository, we have to extend it to any of the following interfaces:   
a) Repository   
b) PagingAndSortingRepository   
c) CrudRepository   
d) JpaRepository   
e) QueryByExampleRepository   
  
8. What is PagingAndSortingRepository?  
Answer: PagingAndSortingRepository provides methods that are used to retrieve entities using pagination and sorting. It extends the CrudRepository interface.   
  
9. What is @Query used for?  
Answer:  Spring Data API provides many ways to define SQL query which can be executed and Query annotations one of them. The @Query is an annotation that is used to execute both JPQL and native SQL queries.   
  
10. Give an example of using @Query annotation with JPQL?   
Answer: Here is an example of @Query annotation from Spring Data Application which returns all active orders from the database:

@Query("**SELECT** **order** **FROM** Orders o **WHERE** o.Disabled= 0")

Collection<User> findAllActiveOrders();

and, here is another example, which returns matching employees from the database

@Query("select e from Employee e where se.name = ?1")

List<Employee> getEmployees(String name);

11. Can you name the different types of entity mapping ?  
Answer: one-to-one mapping, one-to-many mapping, many-to-one mapping, and many-to-many mapping.   
  
12. Define entity and name the different properties of an entity ?  
Answer: An entity is a group of states bundled (or associated) together in a single unit. It behaves like an object. It also becomes a major constituent of the object-oriented paradigm.   
  
13. What is PlatformTransactionMangaer?  
Answer: PlatformTransactionMangaer is an interface that extends TransactionManager. It is the central interface in Spring's transaction infrastructure.   
  
14. How can we enable Spring Data JPA features?  
Answer: To enable Spring data JPA features, first we have to define a configuration class and then, we can use @EnableJpaRepositoties annotation with it. This annotation will enable the features.

15. Differentiate between findById() and getOne() ?  
Answer: The findById() is available in CrudRepository while getOne() is available in JpaRepository. The findById() returns null if record does not exist while the getOne() will throw an exception called EntityNotFoundException.

And, here are some S*pring Data JPA Practice questions* for you, I haven't provided the answer to these questions but you can find them by doing some research. You can post the correct answers to these spring Data JPA question in the comments and I will pick the right one to put it here with your name.  
  
16. Are you able to participate in a given transaction in Spring while working with JPA?   
  
17. Which PlatformTransactionManager(s) can you use with JPA?  
  
18. What do you have to configure to use JPA with Spring? How does Spring Boot make this easier?

1. Datasource Configuration

We now have dependencies configured. It is not time to tell which data source to connect to. Here is my application.yml with Spring Boot data source entries.

spring:

datasource:

url: jdbc:mysql://localhost:33099/dogs

password: <ENTER \_ PASSWORD \_ HERE >

username: root

driver-class-name: "com.mysql.jdbc.Driver"

jpa:

database-platform: org.hibernate.dialect.MySQL5InnoDBDialect

hibernate:

ddl-auto: update

## Entity Bean

The first code level thing we will do is write an Entity Bean. Here is what the Oracle Documentation says about Entity Beans.import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

@Entity

public class Dog {

@Id

@GeneratedValue

private long id;

private String name;

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

## Repository Interface

The Repository represents the DAO layer, which typically does all the database operations. Thanks to Spring Data, who provides the implementations for these methods. Let’s have a look at our DogsRepoisitory, which extends the CrudRepository:

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface DogsRepository extends CrudRepository<Dog, Long> {}

## 4) Controller and Service Layer

As we have our data access layer done, we will write our controller and service layer. Notice that the DogsRepository is annotated with @Repository, which also adds it to the Spring Context. We can now Autowire the repository in Service.

import com.amitph.spring.dogs.model.DogDto;

import com.amitph.spring.dogs.repo.Dog;

import com.amitph.spring.dogs.repo.DogsRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Component;

import java.util.List;

import java.util.Optional;

@Component

public class DogsService {

@Autowired DogsRepository repository;

public void add(DogDto dto) {

repository.save(toEntity(dto));

}

public void delete(long id) {

repository.deleteById(id);

}

public List<Dog> getDogs() {

return (List<Dog>) repository.findAll();

}

public Dog getDogById(long id) {

Optional<Dog> optionalDog = repository.findById(id);

return optionalDog.orElseThrow(() -> new DogNotFoundException("Couldn't find a Dog with id: " + id));

}

private Dog toEntity(DogDto dto) {

Dog entity = new Dog();

entity.setName(dto.getName());

entity.setAge(dto.getAge());

return entity;

}

}

19. How are Spring Data repositories implemented by Spring at runtime?

These practice questions are also good for [Spring professional certification](https://www.java67.com/2019/06/core-spring-professional-50-topics-guide-java-developers.html), one of the most in-demand certifications for Java developers.   
That's all about the frequently asked **Spring Data JPA Interview Questions and Answers**. So these were the top 15 questions related to Spring Data JPA. Generally, the Spring data JPA interview questions are more or less related to the above questions listed in this article but it's not guaranteed that you will get these questions in a real interview. Instead of mugging these questions I strongly suggest you use them to learn the essential Spring Data JPA concepts better.