PayCore Java Spring Bootcamp [2.3]

Homework 3#

28.01.2022

QUESTION 1: What is JPA?

Answer:

JPA means Java Persistence API. It is basically a specification that lets you do ORM when you are connecting to database which is Object-Relational Mapping. What ORM does is let you map your entity classes into sequel tables so that when you connect to the database you provide some kind of metadata on your entity classes so you don’t have to do the query and then the mapping yourself. In this phase, framework kind of handles it for you. And JPA is a way to use ORM. JPA is the API. It is the spec which lets you configure your entity classes and give it to a framework. And then framework handles the situation.

The Java programming language has many ORM tools. Examples of these ORM tools are Hibernate, EclipseLink, MyBatis. ORM tools that do the same job with different method names-methods, the lack of a certain standard causes library complexity. To eliminate this complexity, the CP Java community has determined the Java ORM library standards as JPA.

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

Answer:

We need to use camelcasing for naming finder methods. And to create finder method we can text something like this:

findByName(String name);

findByLocation(String location);

We must use “findBy” to make our method finder. And as we can see we are doing it with camelCasing.

QUESTION 3: What is PagingAndSortingRepository?

Answer:

In Crud Repository we could create, delete, update and read the entities. There is PagingAndSortingRepository that adds additional methods to ease paginated access to entities. Here is an example for PagingandSorting Repository:

public interface PagingandSortingRepository<T, ID extends Serializable> extends CRUD Repository<T, ID>{

Iterable<T> findAll(Sort sort);

Page<T> findAll(Pageble pageble);

If we want to access the second page of user by a page size of 20, we can simply do something like this:

PagingAndSortingRepository<User, Long> repository = *// … get access to a bean*

Page<User> users = repository.findAll(**new** PageRequest(1, 20));

QUESTION 4: Differentiate between findById() and getOne()?

Answer:

Both findById() and getOne() are methods. The function of these methods is retrieve an object from underlying datastore. But there is a difference between them. There are few differeneces between them.

|  |  |
| --- | --- |
| findById() | getOne() |
| 1. The findById() method is available in CrudRepository interface. | 1. The getOne() method is available in JpaRepositpry interface. |
| 2. The findById() method will return null if the record doesn’t exist in the database. | 2. The getOne() method throw EntityNotFoundException if the record doesn’t exist in the database. |
| 3. Internally findById() method use EntityManger find() method. | 3. Internally getOne() method use EntityManger getReference() method. |
| 4. Calling findById() returns an eagerly fetched entity. | 4. Calling getOne() returns a lazily fetched entity. |
| 5. The findById() methods return actual objects and entity fields will contain the value from the database. | 5. The getOne() returns a reference of the entity. All fields may contain default values. |

getOne() doesn’t go to the database. But findById goes to the database all the time and gets the object from here.

QUESTION 5: What is @Query used for?

Answer:

Derived queries are very comfortable to use as long as the queries are not too complicated. But as long as we use more than 2-3 query parameters or need to define multiple joins to other entities, you need a more flexible approach. In these situations, we use Spring Data JPA’s Query annotation. With doing that we specify a custom JPQL or native SQL Query. This annotation gives us full flexibility over the executed statement and our method name doesn’t need to follow any conventions. What we have to do is;

* To define a method in our repository interface
* Annotate method with @Query
* Provide the statement that you want to execute.

QUESTION 6: What is lazy loading in hibernate?

Answer:

Hibernate uses 2 approaches to link 2 objects together. They are lazy loading and eager loading. While we use eager loading, if we pull the object from database, the other objects which are eager will come with our object. But while we use lazy loading we only get what we need. We can say that lazy loading is a technique for fetching used for entities. It also decides whether to load a child class object while loading the parent class object.

QUESTION 7: What is SQL injection attack ? Is Hibernate open to SQL injection attack?

Answer:

SQL Injection, which is used for purposes such as deleting, changing, backing up, adding viruses to the system, damaging operations, is done by adding or injecting SQL query input from the client end of the application. For using the SQL injection, attacker must find a weakness in web application or an entry from website which has security problem.

SQL injection happens when data enters a program from an untrusted source and that data is used to dynamically generate an SQL query. The response is necessary for the attacker to understand the database architecture and access the application's secure information. With a specially crafted SQL command, the attacker can gain access to all the information in the database, obtaining a response that provides a clear idea of ​​the database structure.

There are many ways to do SQL Injection attacks. These are:

* SQL injection based on user input
* SQL injection based on cookies
* SQL injection based on HTTP headers
* Quadratic SQL injection

For hibernate:

Hibernate does not grant immunity to SQL Injection, one can misuse the api as they please.

There is nothing special about HQL (Hibernates subset of SQL) that makes it any more or less susceptible.

Functions such as createQuery(String query) and createSQLQuery(String query) create a Query object that will be executed when the call to commit() is made. If the query string is tainted you have sql injection.

Question 8: What is criteria API in hibernate?

Answer:

For retrieving data from database, hibernate provided 3 ways. Those are [HQL queries, native SQL queries](about:blank) and  hibernate criteria queries. The criteria query API lets you build nested, structured query expressions in Java, providing a compile-time syntax checking. And it also includes Query By Example (QBE) functionality. This lets you supply example objects that contain the properties you would like to retrieve instead of having to step-by-step spell out the components of the query.

QUESTION 9: What Is Erlang? Why Is It Required For Rabbitmq?

Answer:

Written in Erlang, the RabbitMQ server is built on the Open Telecom Platform framework for clustering and failover. Client libraries for interfacing with the agent are available for all major programming languages. The source code is released under the Mozilla Public License. Erlang is a general-purpose programming language and runtime environment on which RabbitMQ is built. OTP (Open Telecom Platform) is a large collection of libraries for Erlang.

In general, Erlang; It is a dynamic, functional, concurrent programming language. It was developed by Ericssion company in 1986. Erlang is part of the software operation process and this language has many design principles. You can combine this programming language with a powerfully optimized runtime environment and some middleware to handle scalability and reliability. That way we can have programming language that can act as a virtual machine as capable as an operating system. Erlang is a great programming language to control vast amounts of infrastructure like massive mobile phone networks too. The main trait that sets erlang apart from other programming languages is that it is process orianted. It is a bunch of non-memory hungry processes that communicate with each other through messages.

Question 10: What is the JPQL?

Answer:

JPQL is know as Java Persistence Query Language. It has similarities with HQL. It is using entity objects. While managing Entity objects, Session objects are used as in EntityManager or Hibernate in the JPA standard. These objects, which undertake similar tasks, offer procedures such as persist(..), merge(..), remove(..), find(…), while also providing various procedures to query entity objects.

Question 11: What are the steps to persist an entity object?

Answer:

The steps of persisting is:

1. We need to create an entity manager factory object.To create an entity manager factory object, we can text something like this:

EntityManagerFactory emf=Persistence.createEntityManagerFactory("Student\_details");

1. We need to obtain an entity manager from factory.To obtain it we text:

EntityManager em=emf.createEntityManager();

1. We need to initialize an entity manager.To do that, we can text:

em.getTransaction().begin();

1. We need to persisting a data into relational database.

em.persist(s1);

1. Then we need to close the transaction.

em.getTransaction().commit();

1. Last, we need to release the factory resources. To do that we can text:

emf.close();

em.close();

Question 12: What are the different types of entity mapping?

Answer:

There are seven different types of entity mapping.

1. Unidirectional One to One
2. Bidirectional One to One
3. Unidirectional One to Many
4. Unidirectional Many to One
5. Bidirectional Many to One
6. Unidirectional Many to Many
7. Bidirectional Many to Many

For One to One relation we can give an example with employee-adress example. We have one employee and this employee has one specific living adress. To use One to One mapping, we text @OneToOne annotation. We can use another annotations like @ManytoOne, @ManytoMany, @OneToMany.

Question 13: What are the properties of an entity?

Answer:

An entity has one or more named properties, each of which can have one or more values. Entities of the same kind do not need to have the same properties, and an entity's values for a given property do not all need to be of the same data type.

Datastore supports a variety of [data types for property values](https://cloud.google.com/appengine/docs/standard/java/datastore/entities#Properties_and_value_types). Those are:

* Integers
* Floating-point numbers
* Strings
* Dates
* Binary data
* Text string (short)
* Byte string (short)
* Byte string (long)
* Date and time
* Geographical point
* Postal address
* Telephone number
* Email address
* Google Accounts user
* Instant messaging handle
* Link
* Category
* Rating
* Datastore key
* Blobstore key
* [Embedded entity](https://cloud.google.com/appengine/docs/standard/java/datastore/entities#Embedded_entities)
* Null

Question 14: Difference between CrudRepository and JpaRepository in Spring Data JPA?

Answer:

They are both interfaces of spring data repository library. Both there are many differences between them.

|  |  |
| --- | --- |
| JPA Repository | Crud Repository |
| JPA extends Crud Repository. | Crud is the base interface. |
| JPA repository also extends the PagingAndSorting repository. It provides all the method for which are useful for implementing pagination. | Crud Repository doesn't provide methods for implementing pagination and sorting. |
| JPA also provides some extra methods related to JPA such as delete records in batch and flushing data directly to a database. | It provides only CRUD functions like findOne, saves, etc. |