3/8/24

Java Techie - Spring boot Interview - QA - 1

1. Why choose spring boot over spring framework?

* Dependency Resolution / Avoid version Conflict.
* Avoid additional Config.
* Embed Tomcat, Jetty (no need to deploy WAR files)
* provide production-ready features such as metrics, health checks

If using spring for CRUD ops with spring app, we need manual dependency Config. Spring Core, Spring Context, Spring hibernate, Spring orm, Spring web-mvc, hibernate-core, hibernate-validator, mysql-connector or any db, jackson related Config and specify version manually.
ex- If changing version of Spring-Core version 4.3.2 to 4.3.3 there will be chance to Conflict with other Spring dependencies like

Spring boot

* With do CRUD opr dependencies need
* Spring-boot-starter-data-jpa, Spring-boot-web, DB driver
* No need to mention version, Just need to specify only in <Spring-boot-starter-parent> <Version> 3.1.2 </version>. Depend on parent, other version dependency version applies

Avoid Additional Config

* 1. For Spring app-Context.xml
<datasource> - <sessionFactory> and inject datasource to session Factory and Create <hibernate properties> and Create TransactionManager inject SessionFactory to TransactionManager

* 2. Spring Rest-Servlet.xml

Create Context and mention base package to load or scan Create classes to context

* 3. web.xml

Mention dispatcher servlet.

Spring Boot

application.yml → Simply mention dataSource Related config. driver-class-name, url, username, password.

2) What all Spring boot Starter you have used?

* Spring boot Starter web - To design web MVC specific application
* Spring boot Starter data JPA - Integrated to any Persistance Layer.
* " " AOP - Aspect Oriented programming #
* Spring boot Starter web Services.
* Spring boot Starter Security.
* Spring boot Starter for Apache Kafka #
  For event driven architecture or Messaging Channel.
* " " Cloud #
  To Build microservices and if need features like service discovery, Config management, load balancing
* " " thymleaf - For Static web pages.

3) How will run your Spring boot application?

* Building Snapshot.jar
* Generate jar inside target folder from compiled classes @ main method.

Snapshot.jar

- BOOT-INF
  - Classes
  - lib
- META-INF
  - Manifest.MF → Details
- org

4). What is the propose of the @SpringBootApplication.

It includes @EnableAutoConfiguration
@ ComponentScan
@ Configuration.

@EnableAutoConfiguration.
Will configure various component to application, by dependency and classpath.

@ ComponentScan
By default it will load root package classes if @ Component mention. If to scan other the main class package by starting application include to scan

@SpringBootApplication (ScanBasePackages = "com.learning.demo")

@ SpringBootConfiguration or @ Configuration.
It allows to register extra bean in the Context or import additional config classes.

ex - Customise security Config, Swagger Config, websocket Config, JWT Config.

@ComponentScan (basePackages ={"com.swatchee.*", "com.business"})
@ Import ({SecurityConfig.class, SwaggerConfig.class})

6. What is Auto Configuration in spring boot ?

It scan the dependencies and classpath to what coe the Configuration to be auto Config.

@ConditionalOnClass (ObjectMapper.class) if found object mapper on Class path for particular class it will auto Config.

@ ConditionalOnMissingBean, @EnableConfigurationProperties

7. How can disable a specific auto-Configuration class in Spring boot ?

@SpringBootApplication (exclude = {" ". class}) or

@EnableAutoConfiguration(exclude = {" ". class}) or

Application.properties - Spring. autoconfigure. exclude = {package.classname, }

8. How can you customize the default configuration in Spring boot ?

application.yml = Server: port : 9191 or

application.properties - Server.port = 9196 or custom port num.

max port num. is 65535 for TCP/IP

9. How Spring boot run() method works internally ?

* Create Application Context
* Register bean into Context
* Kicked up embedded tomcat Container to your Jar/war.

1) Creates ConfigurableEnvironment in this

Property Sources - PropertySourceList : app.properties, app. yml loads OriginTrackedMapPropertySource

2) Creates application context checks applicationTypeContextFactory

Default ApplicationContextFactory -

or ReactiveWebServerApplication

or ServletWebServerApplication

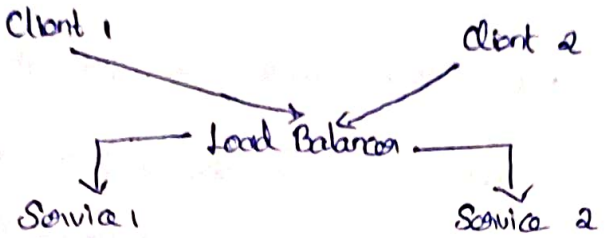
10. What is commandline runner in spring boot?

To perform some task at time of application startup. by implements CommandLineRunner - Functional Interface - run()

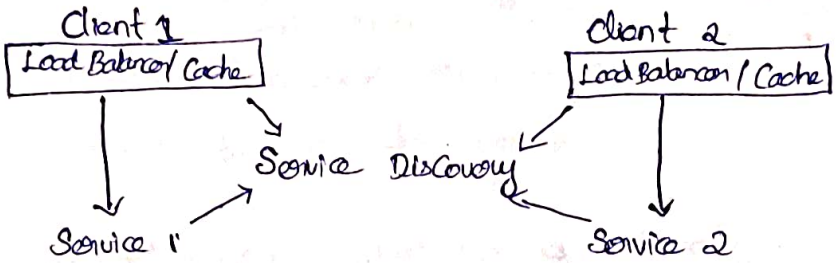
MicroServices

It is an architectural and organizational approach to software development where software is composed of small independent services that communicate over well defined APIs.

Services in Monolithic Architecture



Service Discovery



* Client gets dynamic URL from Service discovery based on Server availability & performance
Service.

* Cached URLs using eruka client
* Decouple Service from client
* Default port for eureka is 8761

Service registry project will annotated with @EnableEurekaServer

application.properties
eureka.client.register-with.eureka = false
This is for to prevent the service registry api to register itself. only the client will register with service registry.

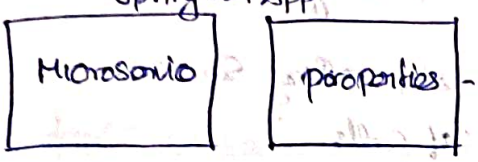
eureka.client.fetch-registry = false

To register with service Registry, other microservices are need to add dependency in their pom.xml

Spring-cloud-starter-netflix-eureka-client
@EnableDiscoveryClient

Config Server

No Config Server

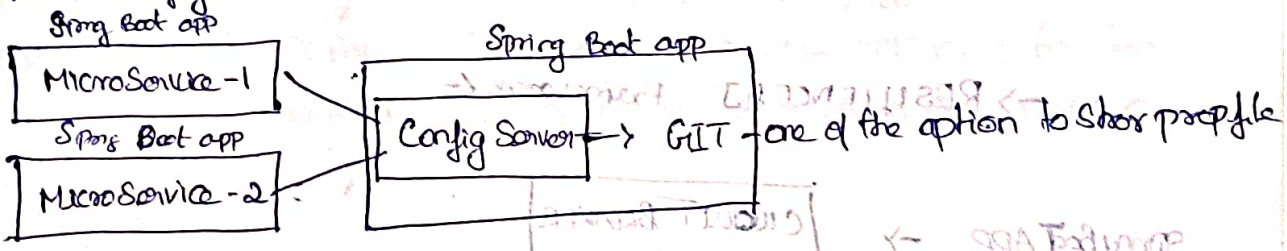


- DB related properties or other

If there are 1 or 2 microservice, it is easy to maintain properties like DB details.

If there are 100, 200, or more microservice, it is difficult to maintain.

with Config Server



→ To make Spring Boot project as config server

Dependency - Config Server in Spring Cloud Config

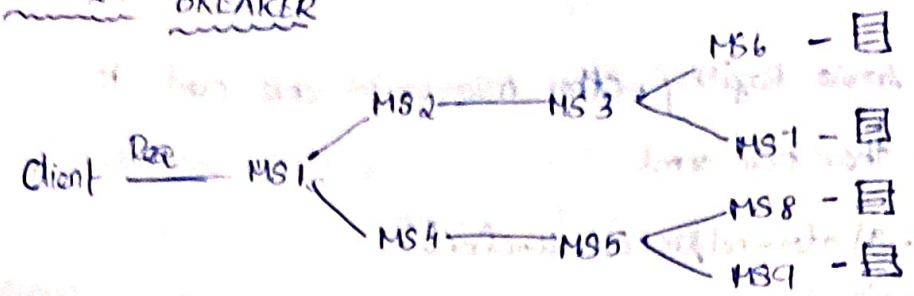
@EnableConfigServer in main method.

Default port - 8888

In application.properties

Spring.Cloud.Config.Server.git.URI = git repo url

CIRCUIT BREAKER



ex -

If user loggel in amazon ecommerce application the home page can take lot of microservice calls to fetch. If the one of the microservice call fails the online application will the chance to down if the architecture not designed properly or fault tolerance not designed properly.

CIRCUIT BREAKER - open circuit to stop calling services

RATE LIMITER - Limit number of calls.

BULK HEAD - Split traffic or memory or resource to multiple callers. Couldn't allocates all the memory or most of the memory/capacity for single caller/call

RETRY - Sets no of retry attempts

→ RESILIENCE4J Framework ←

SpringBoot APP → CIRCUIT BREAKER Interface

↑

RESILIENCE4J IMPLEMENTATION

or

HYSTRIX IMPLENTATION - By Netflix

or

SENTINAL IMPLENTATION

05-8-24

<> spring-cloud-starter-circuitbreaker-resilience4j <>

* CircuitBreakerFactory

```
@Autowired
CircuitBreakerFactory CircuitBreakerFactory;

public methods {
  CircuitBreaker cb = CircuitBreakerFactory.Create("ServiceName");
  cb.run(Supplier<T> toRun, fallback);
```

API call — methods or steps to do when the api call fails.

Default Configuration in the framework

```
public static int DEFAULT_FAILURE_RATE_THRESHOLD; // percentage 50
"        "  DEFAULT_SLOW_CALL_RATE_THRESHOLD; 100%
"        "  DEFAULT_WAIT_DURATION_IN_OPEN_STATE: 60 seconds
"        "  DEFAULT_PERMITTED_IN_HALF_OPEN_STATE; 10
"        "  DEFAULT_MINIMUM_NUMBER_OF_CALLS; 100
"        "  DEFAULT_SLIDING_WINDOW_SIZE; 100
"        "  DEFAULT_SLOW_CALL_DURATION_THRESHOLD, 60 Secs
"        "  DEFAULT_WAIT_DURATION_IN_HALF_OPEN_STATE; 0
```

API GATEWAY

Microservices without API GATEWAY

* HARD CODED URL
* CROSSCUTTING CONCERN

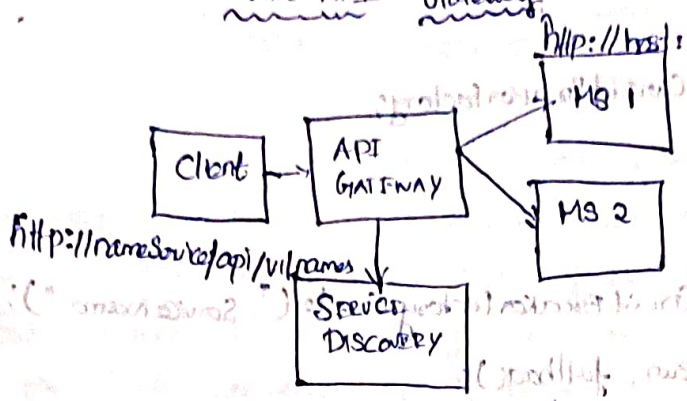
Client → MS1
Client → MS2

Some basic functionalities has to be write on multipe ms, duplication of code

* AUTHENTICATION - Need to write a login authentication or some
* authentication in multiple micro services.

* AUTHORIZATION - OAuth or JWT (need to implement in service level) without API Gateway.

With API Gateway



* STATIC / Dynamic ROUTING
* METRICS

Common API GATEWAYS IN MARKET

1. Spring Cloud Zuul :- Netflix - now open source
2. APIGEE API MANAGEMENT - Google.
3. AWS API GATEWAY - AMAZON.
4. AZURE API GATEWAY - MICROSOFT.

10/8/24

Java 8 Features

1. Lamda Expressions

It provide the implementation of an interface which has functional interface

2. Method Reference

It is used to refer method of functional interface. Easy form of lamda expressions

- Static Method Ref
- Instance Method Ref
- Ref to Constructor.

3. Functional Interface

Contains exactly one abstract method. Can contain any number of default methods.

4 Stream

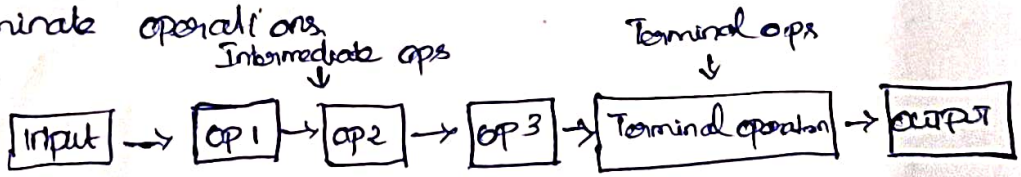
This package consists of classes, interfaces & enum to allow functional style operation on elements

Java.util.stream

operations

i. Intermediate operations

ii. Terminate operations



A sequence of elements from a source that supports aggregate functions & data processing operations

Intermediate operations

* map()

It is used to return a stream consisting of the results of applying the given functions

* filter()
* Sorted()
* flatMap()
* distinct()
* peek()
* limit()

Terminal operation

* Collect()
* forEach()
* reduce()

It is used to reduce the elements of a stream to a single value. Takes BinaryOperator as a parameter.

* Count()
* findFirst()
* allMatch()
* anyMatch()

19/8/24

4. Default Methods

* Enable to add new functionality to existing interfaces
* Also known as defender methods or virtual extension methods.

Ambiguity problem / Diamond problem.

```
public interface Interface1{
    default void eat(){
        System.out("eat meals")
    }
}

public interface Interface2{
    default void eat(){
        System.out("eat biriyani")
    }
}

public class class1 implement Interface1, Interface2{
    @override
    ~~public void funtime()~~
    public void eat(){
        Interface1.Super.eat();
    }

    main(){
        class1 classObj = new class1();
        ClassObj.eat();  meals.
    }
}
```

5. Static Methods

By default "Default" methods is available for implemented class., But static methods not available by defaulet.

* Default method is overridable but ~~default~~ Static method is not overridable.

20-8-24.

* Anonymous Inner class
  * use with concrete class.
  * use with abstract class.
  * use with interface.
  * use as method arguments.

~~Java~~ It is a class without a name & for which only a single object is created.

* Lamda Expression
  * Functional interface is required.
  * Functional interface reference can be used to refer lamda exp.

* Functions (java.util.functions)
  * It is functional interface
  * Java.util functions

apply();
Compose();
andThen();

If the functions receive single input (any) and return output (any datatype) is called functions

* Predicate. (java.util.function)

test() - SAM
default and();
default negate();
default or();
Static isEqual();
Static not();

It is functional interface which represents a predicate (boolean-valued function) of one argument.

* Supplier

* Consumer (java.util.function)

accept (T)
andThen() It doesn't return anything. Accept single argument.

* Supplier (java.util.function)

It doesn't take any arguments but produces a value of type T.

get(); It contains only one method.

* ForEach

Accept Consumer as input, returns nothing

25-8-24

* Lower Bound Generic - Super
* Upper Bound Generic - extends

:- Lower Bound generic. If the method takes Integer as argument if double passed as arg it will throw compile time exception. We will give the Number, object as arg. Greater than integer or parents of Integer

:- Upper Bound Generic (read only)

Accepts input/datatype based that defined

Lower Bound

forEach (Consumer< ? Super Integer> action

(Consumer< ? Extends Integer> action

upper Bound.

Function (T, R) BiFunction (T, T, R)

predicate (T) BiPredicate (T, T)

Consumer (T) BIConsumer (T, T)

Supplier (R)

BiFunction (T, T, R)

apply (T, T);

andThen(Function < ? Super Integer, ? extends Integer> after)

BiPredicate (T, T)

Accepts two inputs T, returns always a boolean.

Method Reference

Instead of using lamda exp for multiple times for same set of functions we use method Reference. It doesn't matches return type, access modifier but parameters must be same.

1. Static ref
2. Instance ref.
3. Constructor ref.
4. Reference to an instance method of arbitrary object of a particular type.
   Random

String::toLowerCase();

* It is short hand notations of lamda expressions.

Streams. Lazily Constructed
Collections - Eagerly Constructed.

Streams (Java.util.stream)

A sequence of elms that supports aggregate ops.

Seq of elm:-
Source:- Collections, Arrays, I/O resource
Aggregate ops:-
A whole formed by combining several disparate elms;- disparate = essentially different in kind
In essence, these aggregate operations are for processing the data.

Optional Java.util.
It is a public final class & used to deal with NullPointerException.

1. of

26-8-24.

OOPS

1. Inheritance

A class that inherits from another class can use the methods and fields of that class.

* Code Reusability
* Method Overriding - Achievable only through inheritance.
* To achieve abstraction

:- Single - Parent -> child.
:- Multilevel - Grand father - father - Child
* Hierarchical - Child1 ext parent, Child2 ext parent
* Multiple - not supported directly (Diamond problem)
* Hybrid - not supported directly

2. Abstraction.

* Data hiding
* Abstraction achieved by interface & abstract class
* Process of identifying only the required characteristics of an object ignoring the irrelevant details

3. Polymorphism

Having many forms.

* Compile Time. - Method overloading
* Run Time : - Method overriding

4. Encapsulation.

It is a way of hiding implementation details of a class from outside access & only exposing a public interface than can be used interact with the class.

Declare private members, for accessing private members public methods are used.

* Wrapping up a data under a single unit.

27-8-24.

Comparator And Comparable

Comparator - Functional Interface

Comparable	Comparator
1. Single Sorting sequence	Multiple Sorting Sequence
2. Affects Original class	doesn't affect original class
3. CompareTo() to Sort	Compare() to Sort
4. Java.lang	Java.util
5. Sort using Collections.Sort(list)	Collections.Sort(List, Comparator)

Marker Interface

It has 0 methods or fields.

ex- Serializable, cloneable, remote Interface.

* It is used for permission purpose

Serialization Java.io

* Conversion of a java object in to static stream (sequence) of bytes. which we can save them to a database or transfer over a network.

* Classes that are elegible for serialization need to implement a special marker interface, serializable; JVM allows special privilages

Criteria to serialize

1. class must implement Serializable interface
2. All of the fields in the class must be serializable. If a field is not serializable, it must be marked "transient"
3. Static field is not serializable (because it is belonging to class not to the reference of object)

* SerialVersionUID = 1L; or generated by ide.
  * It is created on the basis of class name, attributes name, access modifiers associated with attributes.
  * It use reflection to scrape all necessary data from objects fields, including private & final fields.
  * Static & transient field will ignore.

* private Object readResolve(){return instance}
  throws ObjectStreamException
  used when deserialization to ensure the same object is deserialized instead of creating new object. (useful in singleton scenario)

Externalization Java.io Interface Externalizable

* This is not a marker interface like Serializable. So it provides two methods
  * readExternal()
  * ~~read~~ & writeExternal()

```
@Override
writeExternal(ObjectOutput out){
    out.writeInt(id);
    out.writeString(username);
}

readExternal(ObjectInput in){
    id = in.readInt();

}
```

29-8-24.

Interview Questions

1. orElse(T) - Eagerly loading
orElseGet(Supplier) - Lazy loading

If constant value has to return when the optional is empty use orElse();

If some method or constructor called, db related call we have use orElseGet();

2. Stream() or parallelStream()

To find num > 50

List = { 1, 25, 45, 52, 99, 100, 120, 5 }

List.Stream(n -> n > 100).Count(); - "Iterate 1 by 1"

List.parallelStream(n -> n > 100).Count(); parallely done

↓ efficient way if the one num not dependent on another.

29-8-24.

3. List<String> names = Arrays.asList("Karthik", null, "akash");
names.Stream().filter(Objects::nonNull).map(String::toUpperCase)

↓

To avoid null pointer exception.

* Synchronizat, Synchronization.

To avoid parallel process. To make changes one by one. Allow only one thread by jvm to allow complete process.

Thread.run() - one by one thread will process

Thread.Start() - parallel process

* Runnable - Java.lang.

It is interface implement by a class whose Istances are intented to be a thread.

01-09-24

MultiThreading

Durga soft

Intro

1. ways to define thread

   Extendns Thread

   or Implement Runnable

2. Getting & Setting name of thread.

3. Thread Priorities

4. Methods to prevent thread

   1. yield()
   2. Join()
   3. Slep()

5. Synchronization

6. Interthread Communication.

7. Deadlock

8. Daemon Threads.

MultiTasking

executing several task simultaneously

Process Based	Thread Based.
1) Typing program in editor	1) Single Program
2) listening Song } OS	processing multiple task in Single Program. with independent or dependent task
3) Down loading files	
↑ Three diff programs	

Thread Scheduler.

It is the part of JVM, It is responsible to schedule multiple threads in runnable state. Order is maintained by thread Scheduler. Two factors priority & Time of arrival.

1. First come first Serve
2. Time slicing
3. Promptive priority scheduling

thread.run() vs start()

i run on same thread — Create new thread

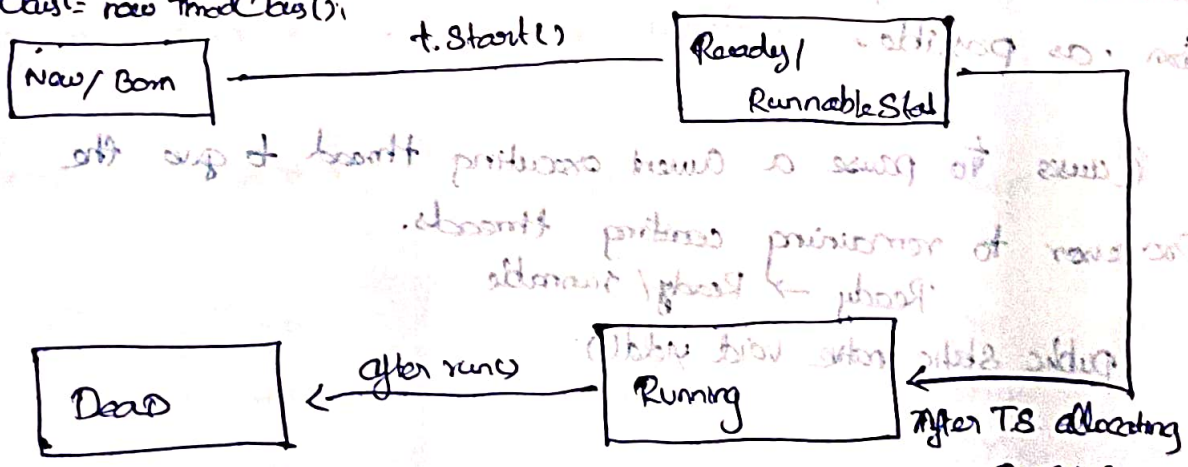
ii Same flow — Seperate flow

iii able to call run() multiple times — ii Can't call start() ~~not~~ 2nd time
Illegal State Exception()

iv 1. Register this thread with thread scheduler.
2. Perform all other mandatory activities
3. Invoke run()

Thread life cycle

ThreadClass t= new ThreadClass();

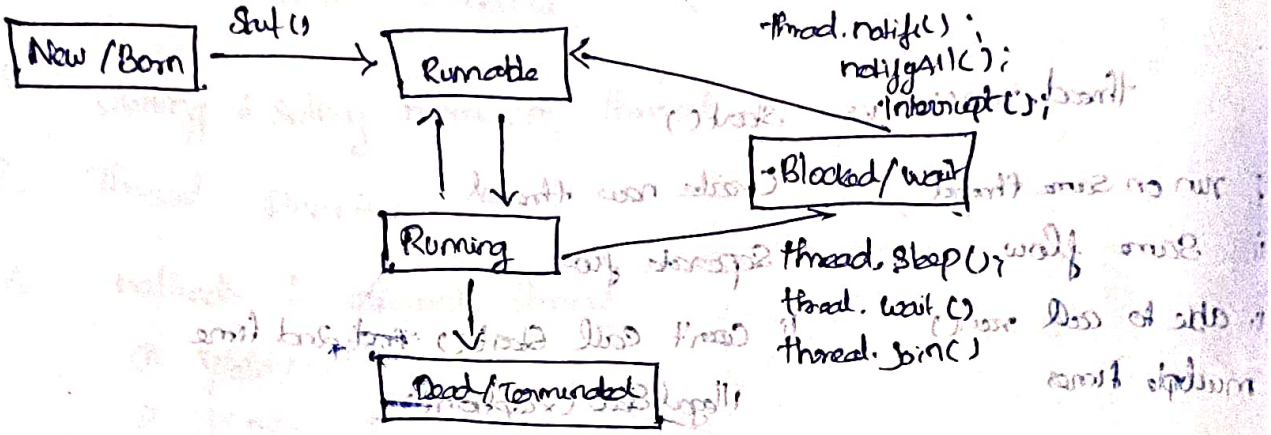


Thread has 8 types of Constructor.

Thread priorities

Min = 1, Default = 5, Max = 10;
Norm =

Default for only main thread is 5, remaing all thread default min, max priority is inherited from parent.



1. yield()

It provides mechanism to inform scheduler that the current thread is willing to relinquish its current use of processor but like to be scheduled back soon as possible.

Causes to pause a current executing thread to give the processor to remaining waiting threads.

Ready → Ready/ runnable

public static native void yield();

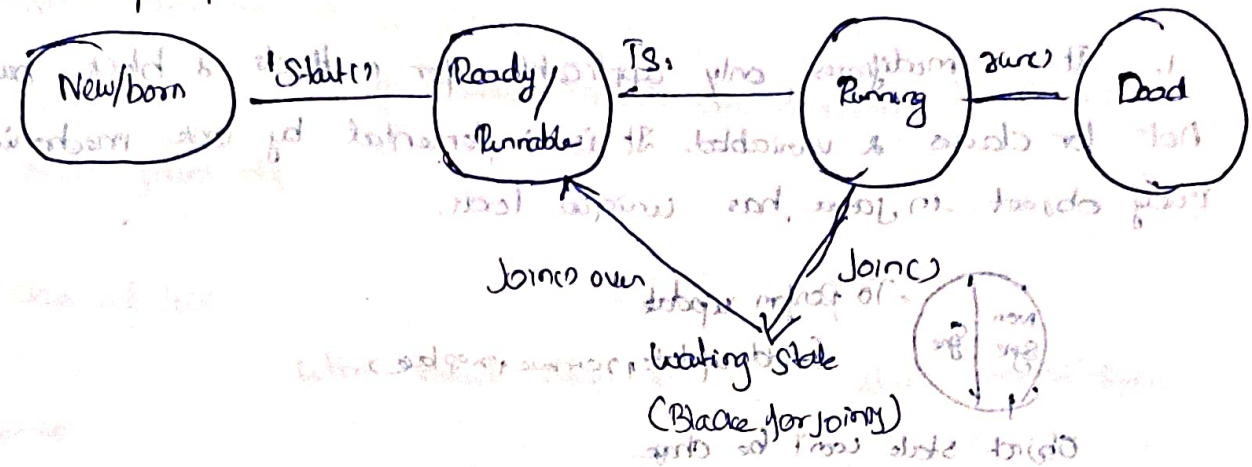
2. Join()

wait for other threads to join. wait state

join()
join(long millis)
join(long millis, int nanos)

public final void join(); throws InterruptedException.

Checked exception.



If Same thread will call join() it will dead lock.

3. Join()

public state native void sleep(long ms);

p s v s (long ms, int ns);

throws InterruptedEx

Checked

4 Interrupt()

A thread can interrupt a sleeping thread or waiting thread. It works to interrupt the Target some thread when it will goes sleep(), once Target another thread is goes to sleep, immediately interrupt will interrupt the sleeping thread. throws IE. If target thread never goes sleeping or waiting state through its life time then interrupt call will wasted.

Property	yield()	join()	sleep()
1.			
2. Overload?	NO	yes	yes
3. Final?	NO	yes	NO
4. throws IE	NO	Y	Y
5. native	Y	N	Y, N
6. Static	Y	N	Y.

Synchronized

1. It is modifiers, only applicable for methods & block but not for classes & variables. It is implemented by lock mechanism. Every object in Java has unique lock.

- To perform update (add, update, remove, replace)

Object state can't be change

Static Synchronized - Class level lock
Synchronized - Object Level lock.

Static sync m1()
Static sync m2()
Static m3()
Sync m4()
m5()

2/9/24

* Race Condition.

If multiple threads are operating simultaneously on same java obj.

* class lvl lock

When thread wants to execute() static ~~metho~~ sync method.

* object lvl lock.

To execute instance method

* wait(), notify(), notifyAll() — Object class
* Start(), sleep() ---- — Thread class

5/9/24

* Thread lock release only wait(), notify(), notifyAll() only

final void wait(); throws IE Checked Exception
native final void wait(long ms); IE
void wait(long ms, int ns); IE

native void notify();
native void notifyAll();

* Dead lock

Two threads are waiting for eachother forever,

Such type of infinite wait is called dead lock. Synchronized is only reason for dead lock.

Deadlock	Starvation
Long waiting of a thread but waiting never ends	Long waiting of a thread till waiting ends on some period. ex- low priority threads wait for other high priority threads

Daemon Thread

Threads which are executing in the background. ex- garbage Collector (Low priority thread), Attach listner, Signal dispatcher.

It provides services to user thread for background supporting tasks. It has no role in the life than to serve user threads

void setDaemon(boolean flag); - Set before thread start() or throws IllegalThreadStateException.
boolean isDaemon();

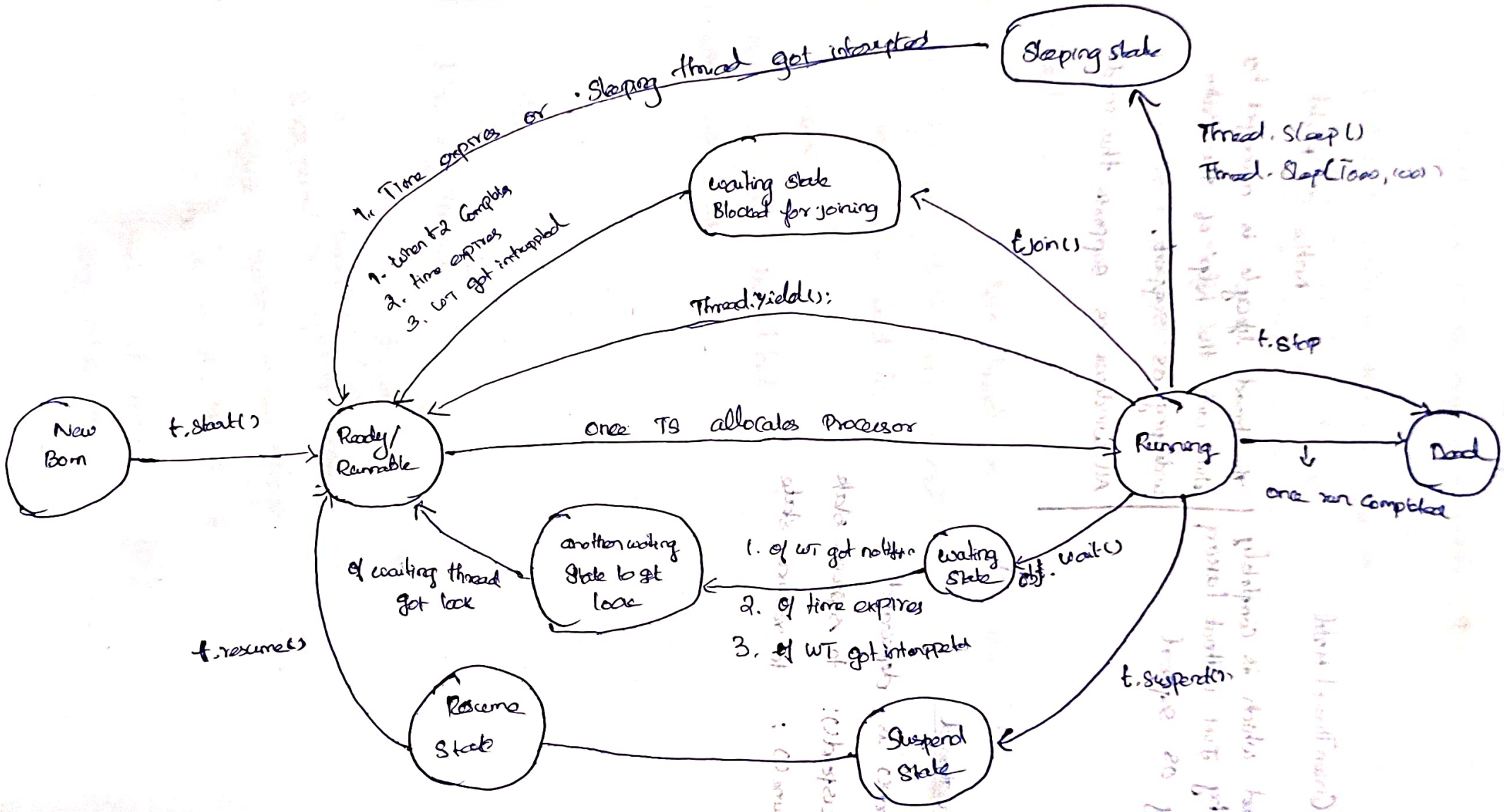
When non daemon threads ends remaing daemon threads are end.

7/9/24.

Green Thread Model	Native OS Model
* The thread which is completely managed by JVM without taking underlying OS support	The thread lifecycle is managed by JVM with the help of the native underlying OS support.. All windows OS supports this model

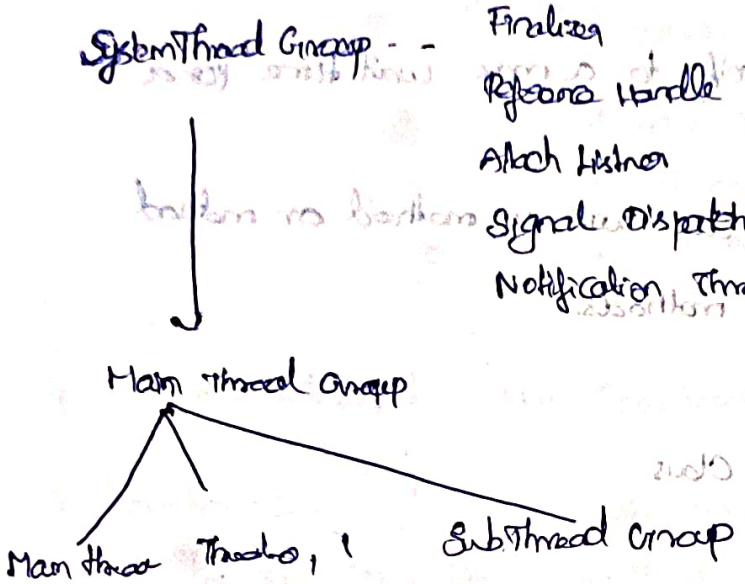
<u>Stop a thread</u>

* t.stop() is deprecated.
* t.suspend(): to suspend state
* t.resume() : to resume state


Sleeping state
1. Time expires or
Sleeping thread got interrupted
Thread.Sleep()
Thread.Sleep(1000,100)
Waiting state
Blocked for joining
t.Join()
1. when t2 Completes
2. time expires
3. WT got interrupted
Thread.yield();
t.stop
New Born
t.start()
Ready/ Runnable
Once TS allocates Processor
Running
Dead
one run Completed
another waiting State to get lock
1. If WT got notify
Waiting State
obj.wait()
If waiting thread got lock
2. If time expires
3. If WT got interrupted
t.resume()
t.suspend()
Resume State
Suspend State

* Thread Group

Contains group of threads, in addition contains sub thread groups.



1. ThreadGroup Tg = new ThreadGroup("Grp name");

2 ThreadGroup tg = new ThreadGroup (ThreadGroup parentGroup, String grpName);

String getName();

int getMaxPriority();

void setMaxPriority(int p);

ThreadGroup getParent();

void list();

int activeCount();

int activeGroupCount();

int enumerate(Thread[] t) - To copy all active threads of this thread grp in given thread arr, subthreadgrp also consider

int enumerate (ThreadGroup[] g);

boolean isDaemon();

void setDaemon(boolean b)

void interrupt()

void destroy()

Java.util.Concurrent.locks introduced in 1.5 V

1. We are not having any flexibility to try for a lock without waiting in - Synchronized.

2. There is no way to specify to a max wait time for a lock. - In Synchronized

3. Synchronized keyword is use with in method or method level, couldn't use across methods.

* Interface Lock
* ReentrantLock - Implementation Class

Interface Lock

1. void lock();, wait until to get lock.

2. boolean tryLock();, if available get lock else do other ops, wont wait

3. boolean tryLock(long time, TimeUnit unit); TimeUnit.Hour,

4. void lockInterruptibly();

Acquires a lock if it is available & return immediately. If lock is not available it will wait. when waiting thread is interrupted then wont get a lock.

5. void unlock();

Throws IllegalMonitorStateException() if the thread is not owner of the lock. only the owner will unlock.

ReentrantLock (C)

* Direct child class of object

ThreadHoldCount = 1	t.lock();
2	t.lock()
3	t.lock();
2	t.unlock();
1	t.unlock()
0	t.unlock() - Finally unlocked

1. ReentrantLock l = new ReentrantLock();

2. ReentrantLock l = new ReentrantLock (boolean fairness);

If true then this thread will first get a chance to lock among multiple threads. (longest waiting thread get chance first) It follows first come first serve.

Thread pools (Executor Framework) Java 1.5 V
(java.util.concurrent)

Creating a new thread for every thread for creating a performance memory problems. To overcome this we should go for threadPool. ThreadPool is a pool of already created threads to ~~created~~ ready to do our jobs

ExecutorService service = ~~new FixedThreadPool~~
Executors.newFixedThreadPool(3);

Service.Submit(Job); - Runnable
Service.shutdown();

8/9/24

1.0V

Callable & Runnable — 1.5 Version

java.util.Concurrent — Java.lang

* Runnable job thread won't return anything after completing its job.
* In Callable job thread return some result after execution

Future = public object call() throws exception

* Future is used to retrive a result from callable job.

Thread Locals

* Thread local class provides thread local variable.
* It maintain values per thread basis.
* Each thread local obj maintains a separate value like

userId, transId etc.

---

1. Object get();
2. Object InitialValue();
3. Object.set (Object newvalue);
4. void remove()

9/9/24.

Memory Management.

* STACK
* HEAP.

STACK

* Store Temporary variables & separate memory block for method
* Store primitive data types
* Store reference of the heap objects
* Strong ref, weak ref, Soft ref.
* Each thread has its own stack memory.
* Variables with in a scope is only visible and as soon as any variable goes out of the scope, its ~~gb declared~~ deleted from the stack (LIFO order)
* StackOverflowError Java.lang.

Strong Reference.

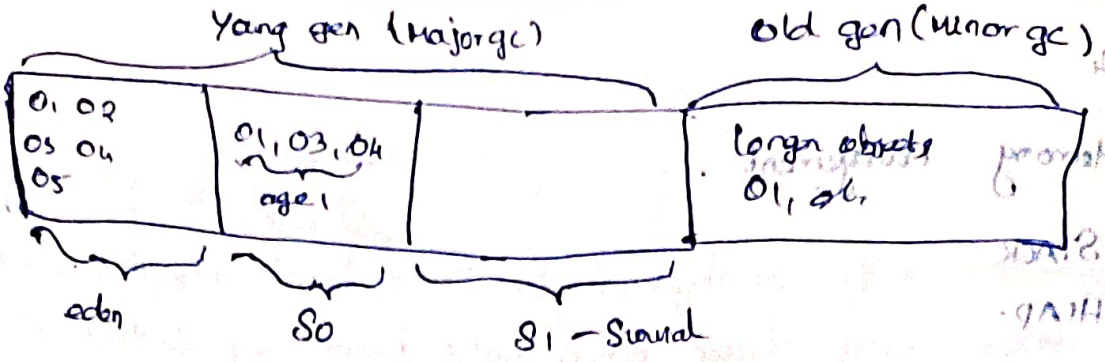
Person obj = new Person();

weak Reference

WeakReference<Person> wRef = new WeakReference<Person>();

Soft Reference

don't delete ref immediately, delet if its most necessary to free up space.



Ob1, ob2, ob3, ob4, ob5 - Created

Minor gc - after gc(); 2, 3, cleared gc(); Mark & Sweep algo
ob1, ob3, ob4

Heap:-

* Heap memory is shared with all the threads.
* No order of allocation of memory
* GC is used to delete unreferenced objects from heap.
  - Mark and Sweep algo

Types of GC
* Single GC
* parallel GC
* CMS (Concurrent Mark Sweep)
* G1

* It also contain String pool.
* OutofMemoryError java.lang
* Young Gen (Minor GC)
  Eden
  Survivor

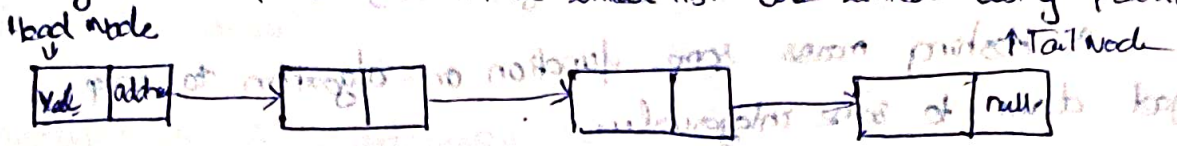
Old (Tenured Gen (Major GC)

Permanent Gen. before Java 7, now MetaSpace

11-9-24

* Linked List

It is linear data structure, not stored in contiguous memory location. Elements in a linked list are linked using pointers



~~Insert, update~~

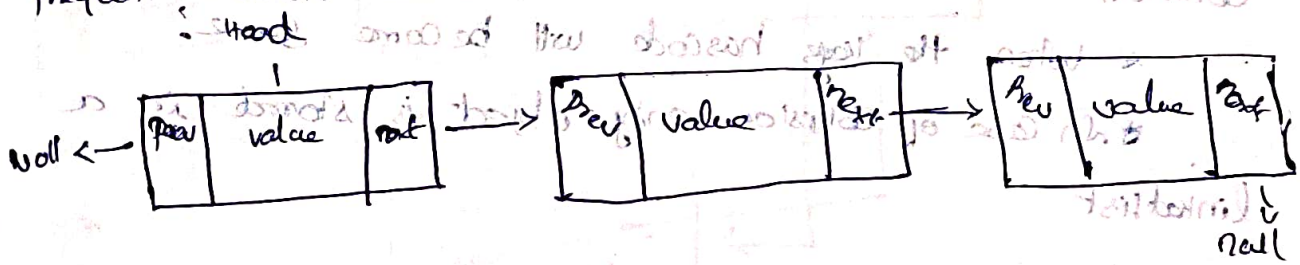
Insert - O(N), on first elm - O(1)

Delete at First - O(1), else O(N)

Search - O(n)

* Doubly Linked List

Advantage of doubly linked list is that it allows for efficient traversal of list in both sides. It suitable for frequent insertions and deletions.



TC

Insert at begining or end - O(1) or Random O(n)

Delete at beg or end - O(1) " " O(n)

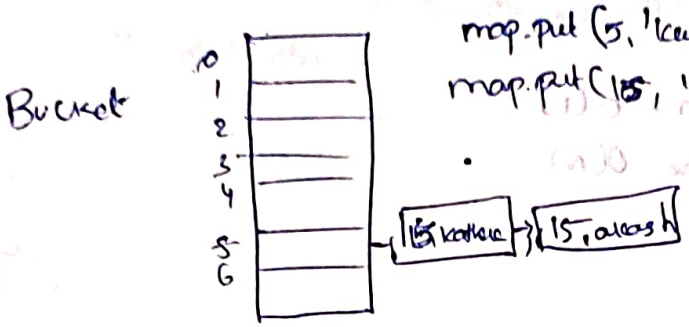
Search - O(n) or Search on 1st - O(1)

Access - O(n)

12-09-24

* Internal working of Hashmap

* It works on the principle of Hashing
* Hashing means some function or algorithm to map object data to some integer value.



* Store a entry is hashtable, key will be hashing

* Treeify_Threshold = 8
* Untreeify_Threshold = 6

Collision

* When the keys hascode will become same.
* In case of collision entry object is stored in a linked list

After Improvement

* If bucket reaches the maximum threshold colliding keys, entries will be stored in a balanced binary Tree.

O(n) to O(log n)

Bucket

	Hash	Key	Value	next
0				
...				
15				

Nodes

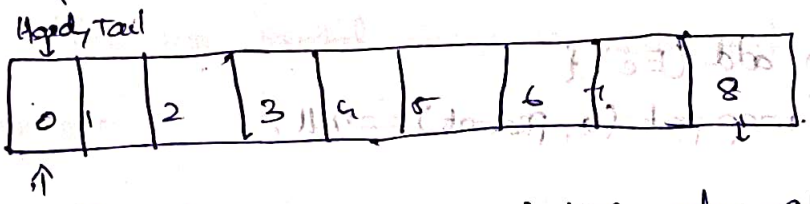
Default bucket size 16

1. Bucket - Arr of nodes
2. Initial Capacity - 16
3. Load Factor - 0.75 or 75% . 16×0.75=12, if 13 get value it increases another 16 nodes - doubles
4. Hashing -

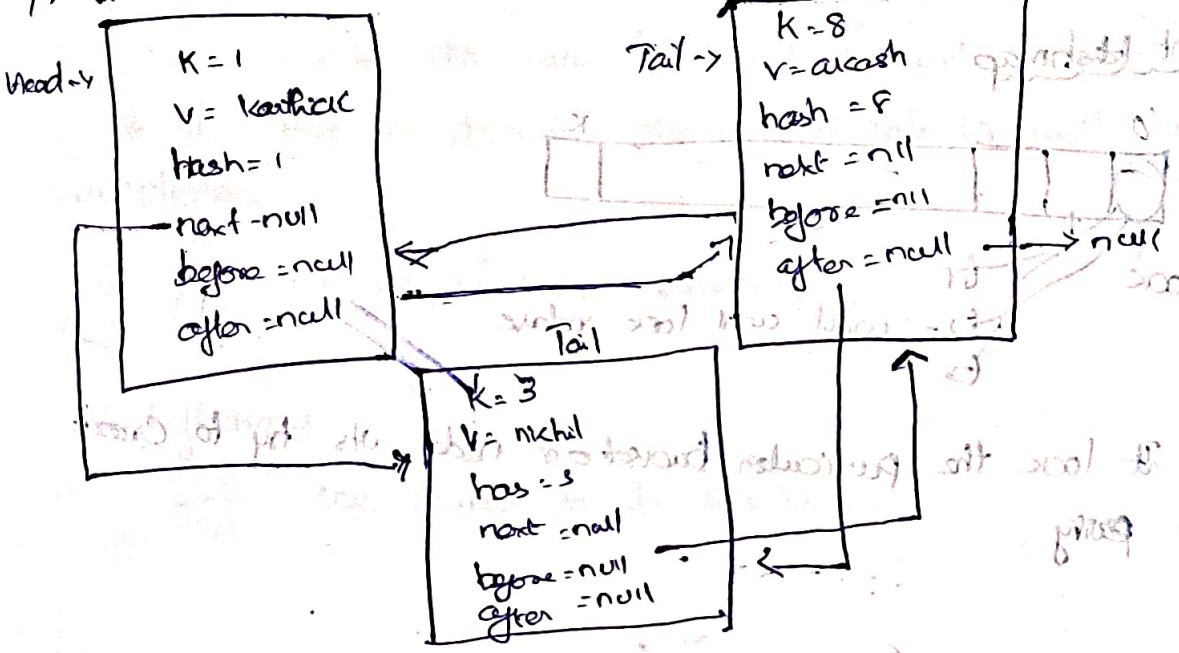
Keys are ordered

LinkedHashMap & LinkedHashSet

LinkedHashMap - extends HashMap implement map



1. K=1, V= kartheik



LinkedHashSet extends HashSet implement Set

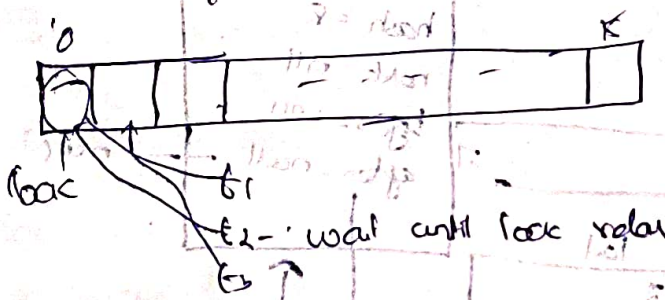
* Stores values as key
* Inserts value dummy
* other working parameters its same as linked Hashmap

* HashSet

* Doesn't maintain insertion order.
* Not thread safe
* It internal implention is like hashmap

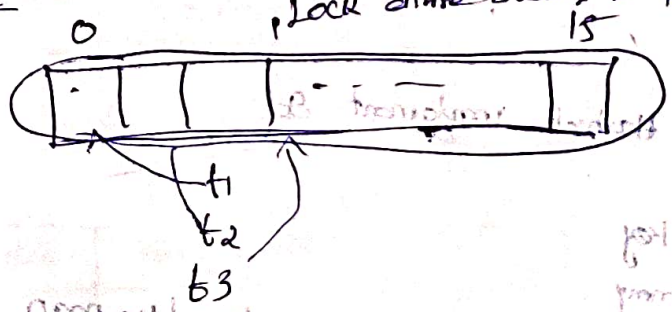
```
public HashSet(){
    map = new HashMap<>();
}

public boolean add(E e){
    return map.put(e, present) == null;
}
```

Concurrent Hashmap



It lock the particular bucket or node. its try to create a new entry

HashTable



* lock all buckets to put value in single node

other threads are wait until object lock

* Low Performance Issue

Interview Questions.

1. Types of design patterns

1. Factory
2. Singleton.
3. Builder

2. Singleton.

There are several ways to break and fix singleton design pattern.

1. Serialization. - breaks

To resolve this issue "readResolve()" will override in pojo. At the time of deserialize the readResolve() will call implicitly.

```
readResolve(){ return instance;}
```

2. Reflections

use Enums to fix this issue.

3. Exception hierarchy in inheritance

```
Class parent{
    public void add(){throws exception.
}
Class Child extends parent{
    @override
    public void add(){ }
```

No need mandatory to throw exc.

If parent class shouldn't throws exception but child class overridden method throws exception means compile time exception occurs. Parent class has to definitly throws exception.

5. Can we override static and private method? → No

* Static methods belongs to class not for instance.
* private scope method only accecible with in class

6. Final, Finally, Finalize.

```
protected void finalize() {
    Sysout("finalize method called");
}

System.gc();
```

7. equals and hashCode method

:- hashcode

To generate unique or reference number of an object.

:- equals

used to compare object based on its content

8. Exception

Object
↓
Throwable
↓
Error
- Out of Memory exp
- StackOverFlow

Exception
↓
UNChecked
- Null pointer
- Index Out of bound
- Arithmetic
- Illegal Argument

Checked
- Class Not Found Exp
- IO Exp
- SQLExp
- TimeNotFound
- NoSuchMethod Exp

9. Why String is immutable?

- String pool is possible only because String is immutable in java.
- String is thread safe because of immutable.
- It's hasCode is already cached

10. StringBuffer and StringBuilder (Mutable)

	String	Buffer	Builder
Mutability	Immutable	Mutable (Synchronized)	Mutable (Non synchronized)
Thread Safe	~~non~~ thread safe	Thread Safe	Non thread Safe
performance	very high	Moderate	very high.

11. How to write own immutable class

- Make your class final
- Declare all instance variable with private and final
- Say no to setter methods
- Initialize all variables in constructor.

Return clone object, return new ArrayList

12. DB fail fast & fail safe Iterator

· Fail fast

• A iterator which will fail fast when we do any modification while iterating a Collection is called fail fast iterator (eg - ArrayList, Hashmap, Vector). throw ConcurrentModificationException

Fail safe

* Iterator which allows us to modify in midde while iterating a Collection is called non-fail-Fast iterator.

ex- CopyOnWriteArrayList, CopyOnWriteArraySet, ConcurrentHashMap.

13. Iterable & Iterator:

Iterable → part of Collection framework, provide a way to obtain an ~~Integer~~ Iterator for that Collection.

(but not functional Interface)

* Iterable has single method, iterator() returns Iterator for Collection.

* For-Each loop can be use only when the class is implemented iterable

* Generic

Iterator - Interface

It is used for iterating over a Collection. It provides methods to traverse the elements & manipulate the Collection

* Traversal Methods : hasNext(), next(), remove()

* Stateful - It maintains state, so once we retrieve an element, you cannot it again without resetting the iterator.

* Singlethreaded - Does not support parallel processing

11. TreeMap

```
Map<String, String> treemap = new TreeMap<>();
    t.put("a", "acde");
    t.put("d", "czab");
    t.put("c", "a3La");
```

"a".compareTo("d").

a<d -1 a>d +1 else 0

a, d

"c".compareTo("d")

a, c, d

print - a, c, d

15. What is lamda expression?

It basically express interfaces of functional interfaces, in other word it provides a clear and consise way to represent method of functional approach using an expression.

11. Stream API?

It introduced in java 8, and it is used to process collections of objects with functional style of coding using lamda exp.

12 Stream?

It is a sequence of objects that support various methods which can be pipelined to produce the desired result

23/09/24

What is servlet

* It is used to create a web application (resides at server side and generates a dynamic web page)

Many ways to describe

* It is an api that provides many interface and classes including documentation

* It is an interface that must be implated for creating any servlets

* servlet class that extends the capability of the servers & responds to the incoming request.

The servlet technology is similar to other extension such as Common Gateway Interface (CGI) and PHP

CGI

This technology enable the web server to call an external program and pass HTTP request information to the external program to process request. For each process it ~~request~~ starts a new process

Ds-Adv

* For each request it starts a process, and web server is limited to start process

* It is platform dependent.

Advantage of servlets

* Web containers creates threads for handling multiple request to the servlet.

* Threads are share a common memory area, light weight, cost of communication between the threads are low.

* Better performance

* portability

* Robust, Jvm manages servlets

* secure

:- What is IoC Container. (Inversion of control)

IoC Container is responsible to instantiate, Configure and assemble the objects. It gets information from the XML file and works accordingly.

* To initiate the application class
* To Configure object
* To assemble the dependencies between objects
* It is a design principle.

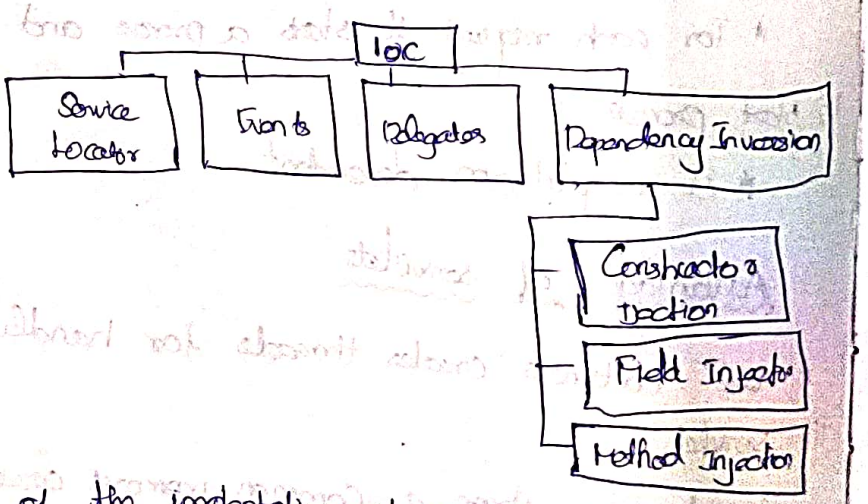
* Two types of IoC Containers
  1. Bean Factory..
  2. Application Context

* Org.springframework.beans.factory.BeanFactory
* Org.springframework.context.ApplicationContext

Application Context interface is built on top of the BeanFactory interface. It has some extra functionality such as simple integration, message resource handling (I18N), event propagation, app layer specific Context. eg- WebApplicationContext

Dependency Injection

1. Constructor Injection
2. Setter Injection
3. Field Injection.



* It is ~~impl~~ one of the implentation of IoC

Spring.6

```
main(){
ApplicationContext Context = new ClasspathXmlApplicationContext
                                    ("Spring.xml");
Dev obj = (Dev)Context..getBean("dev");
obj.build();
```

```
<beans>   For object creation
  <bean  id="dev"  class="com.telusko.Dev">
  <bean>
<beans>
```

ORM

It is a technique used in creating a "bridge" between object-oriented programs and in most cases with relational databases.

25/9/24

1. To return custom response status

```
public ResponseEntity<List<products>> getAllproducts(){
    return new ResponseEntity<>(Service.getAllproduct(), HttpStatus.OK)
}
```

* OWASP - open web application Security Project.
* Spring Filters

26/9/24

* CSRF - Cross-site Request Forgery
* CORS - Cross origin Resource Sharing is an http header based mechanism that allows a server to indicates any origins (domain, scheme, or port) other that its own from which a browser should permit loading resources

:- Spring Boot Interview Questions - 1

1. What is spring boot starter parent? <parent>
   * It is project starter <\parent>
   * It provides default configuration for our application
   * It is a special starter that provides useful maven defaults. It also provides dependency management section so we can omit version tag for dependencies using in pom.
   * Inherits dependency management from spring-boot-dependencies

2. How you secure a spring boot Application? tell me some ways...
   1. JWT, OAUTH

3. What is method level security in spring boot?

4. What are the annotations used for writing JUNIT TestCases?

5. What is the use of @SpringBootTest?

6. Do you know what is the order of precedence in Springboot Configuration?

7. What is spring profile / profiling?

8. Can a static block throw exception? yes
   But we must handle with in the static block with try, catch, Not allowed to use throws.

9. How do you handle exception in your project?
   Custom exceptions do you have used? name son of them?

10. What is Java generics?
    Class Test<Type> {
    }
    Accepts any kind of Non primitive data type/ wrapperclass/ user defined class. eg - ArrayList, HashSet, HashMap

11. What is Inner class?

The purpose of nested classes is to group classes that belongs together which makes code more readable and maintainable.

```
Class OuterClass {
    int id;

    class InnerClass {
        int rollno = 5;
    }
}

OuterClass oObj = new OuterClass();
OuterClass.InnerClass iObj = oObj.~~create~~ new InnerClass();
```

12. How does Garbage Collection handle Circular Reference?

13. How does you investigate and fix a memory leak in java application?

14. Can deadlock occur with single Thread? Yes

15. What is java classloaders?

It is a subsystem / an integral part of ~~JRE~~ JRE that dynamically load java classes ~~it~~ into JVM. JVM system doesn't need to know about files and file systems because of classloaders.

1. Bootstrap class Loader (primordial ClassLoader)

- The Bootstrap ClassLoader is a machine code responsible for initiating the JVM's operation.

- In java version up to 8, it loads core java files from rt.jar. From Java 9, it loads core java files from the Java Runtime Image (JRT). <JAVA_HOME>/lib directory

2 platform Class Loader (Extension ClassLoader)

- Before Java 9, there was an extension ClassLoader, but from Java 9 onwards it called platform class Loader.
- It loads platform specific extensions from the JDK's module system.

* It loads files from JRT or from any other module specified by the system property java.platform or -module-path (<JAVA_HOME>/lib/ext), java.ext.dirs

3. System ClassLoader (Application classloader)

- Also known as the application ClassLoader. It loads classes from the application's classpath.
- It is child of platform class loader.
- Classes are loaded from directories specified by the environment variable CLASSPATH, the -classpath or -cp command line option which includes directories, JAR,

Principles of Functionality of ClassLoader

1. Delegation Model

* Follows delegation hierarchy algorithm
* Delegation hierarchy starts from Application ClassLoader, moves to platform classloader, to Bootstrap ClassLoader

2. visibility principle

* Classes are loaded by parent ClassLoader are visible to child Classloaders, but not vice versa
* This ensures encapsulation and prevents Conflicts between Classes loaded by different ClassLoaders.

3.) Uniqueness property

* ClassLoaders ensure that classes are loaded only once to maintain uniqueness
* If a parent class loader can't find a class, only then the current instance attempts to load it.

2. How to secure springboot application?

It is a framework that provides authentication, & authorization and protection against common attacks

Springboot principle

Spring security defines the notion of a principal, which is the currently logged in user. When a user authenticates successfully, the principal is stored in spring's security context, which is thread-bound, thus making it available to rest of our service

Types of Authentication

* basic - Credentials are sent as plain text.
* form-based -
* token-based
* OAuth 2

LDAP - Light weight Directory Access protocol

* Token based

* web server responds with a token, JWT, which contains information info about the user and their rights.

* The web's browser stores the token, typically in a cookie, or local storage and send back to server with subsequent request

* Token validity is maintained

pros

* Stateless - Server doesn't maintain session information, simplifying server architecture and reducing server load.

* Scalable - Token based authentication is scalable, as token distributed across multiple servers, allowing horizontal scaling.

* Cross platform - Token can used across multiple platforms and devices

* SSO (Single sign on) - Token based authentication can be used to implement single sign-on for multiple web applications, making it easier for user to access multiple applications with a single set of credentials

Cons:-

* Complexity
* Storage
* Security risks - Token theft, replay Attacks
* Token expiration

* OAuth2 Authentication.

It is an authentication protocol that allows users to grant third-party applications access to their resources on a web server without revealing their credentials.

* secure
* Scalable
* Flexible
* Centralized

Back to question-2

before spring boot 3.0 we use implements WebSecurityConfigurerAdapter

After Springboot 3.0 onwards create a bean for

"UserDetailsService"

* JWT - RFC 7519

Header - Algorithm & TokenType

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Payload: Data

```
{
  "name": "Basant",
  "iat": 1516239022
}
```

Verify Signature

```
HMACSHA256(
    base64UrlEncode(header) + "." + base64UrlEncode(payload),
    your-256-bit-secret )
```

Add 3 dependency for JWT

```
<groupId> io.jsonwebtoken</groupId>
<artifactId> jjwt-api</artifactId>
<  ...  > jjwt-impl <  ...  >
<  ...  > jjwt-Jackson <  ...  >
```

Create bean for Authentication Manager

```
@Bean
public AuthenticationManager authenticationManager(AuthenticationConfiguration config)
    return config.getAuthenticationManager();
}
```

```
@postMapping("/login")
public String login(@RequestBody Users user) {
    return Service.verify(user);
}
@Autowired
AuthenticationManager authManager;
public String verify(Users user){
    Authentication authenticate = authManager.authenticate(new
                UsernamePasswordAuthenticationToken(user.getUsername, user.pwd));
if (authentication.isAuthenticated())
    return "Success";
} return "fail";


public String generateToken(String userName) {

    Map<String, Object> claims = new HashMap<>();

    return Jwts.builder().claims
            .add(claims)
            .subject(userName)
            .issuedAt(new Date(Sys.CurrentTimeMill()).
            .expiration(new Date(Sys.CurrentTimeMill() + 60 x 60 x 30))
            .and()
            .signWithKey SignWith(getKey())
            .Compact();
}
String SecretKey = "";
KeyGenerator KeyGen = KeyGenerator.getInstance("HmacSHA256");
SecretKey SK = KeyGen.generateKey();
SecretKey = Base64.getEncoder().encodeToString(SK.getEncoded());
```

Junit

Mockito

```
@InjectMocks
EmployeeServiceImpl employeeServiceImpl;

@Mock
EmployeeCrudRepo repo;
```

Mock is used to test classes like repository layer with some methods hitting dB instead of hitting DB every time, just return some dummy or constant value using

```
when(repo.findById(2)).thenReturn(new employee(2, "Kathir"... ));
assertEquals(emp.getName(), "Kathir");
```

```
@SpringBootTest
class UserRepositoryImplTest {

    @Test
    public void getEmpByIdTest(){

    }
}
```

3. What is Method level Security
   - Security Filter chain
     URL Based Security

     - requestMatchers(HttpMethod.GET, "/special").hasRole("ADMIN")
     - requestMatchers(HttpMethod.GET, "/basic").hasAnyRole("ADMIN", "USER");

   - Method level Security

```
@PreAuthorize("hasRole('ADMIN')") -> Spring Expression language
@GetMapping("/special")
    public String special(){
        return "special";
    }
```

2 Annotations

1 @EnableGlobalMethodSecurity (prePostEnabled = true) - Class level

2 @preAuthorize() for each method

@postAuthorize() - Check authorization after the method is called useful if you need to check the return value's content before returning.

1st		2nd
URL Based Security	if passes	Method level security

@preFilter / @postFilter

- Allows you to filter a collection & only pass certain items in to the method.

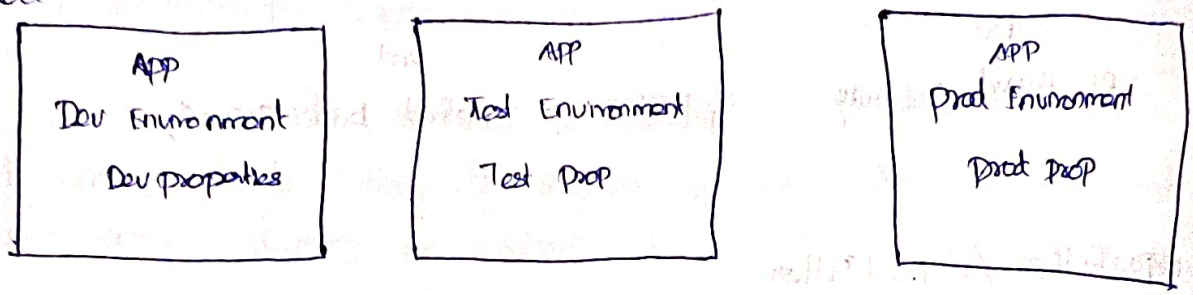
6. order of precedence :-

1. CLA
2. Java system properties (System.getProperties())
3. OS environment variables.
4. @PropertySource annotations on your @Configuration classes.
5. Application.prop outside of your packaged jar or app.yml
6. Application.prop inside of your packaged jar or app.yml
7. Default properties (SpringApplication.setDefaultProperties)

Spring Boot uses a very particular PropertySource order that is designed to allow sensible overriding of values, properties are considered in the following above order

1. What is Spring profile?

Spring profile provide a way to segregate parts of your application Configuration and make it to available only in certain environments. Any @Components, @Configuration or @Configuration properties marked with @profile to limit when it is loaded.



ex- 1st way

application.properties
Spring.profiles.active = Dev
Spring.datasource.url = jdbc: postgres://localhost:5732/esp-dev

Create new file
application-Dev.properties

= jdbc: postgres://dev:5732/cmp-dev-qa

2nd way

Open debug Configuration -> New configuration - profile - list of profile

3rd way
CLA / Terminal

1. Maven build to get a Snapshot.jar when we run on an environment. Deploying in linux ~~mvn~~ jar or war files

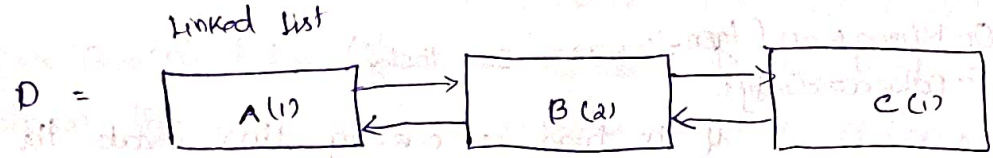
Java -jar -DSpring.profiles.active=qa spring-profile-0.0.1-SNAPSHOT.jar
(Project Name)

12. How does GC handles Circular Reference? Ref - Baeldung.com

* Reference Counting

This algorithm associate a ref Count with each object. These algorithms Consider an object to be alive as long as the number of reference to that object is greater than Zero. Runtime Stores ref Count in header.

In naive impl, each new reference to an object should trigger an atomic ref Count increment, for dereference also trigger atomic decrement.



If D it become unreachable eventhough linkedlist also unreachable now but the ref Count for its Components is more than one. With naive reference Counting imple, the runtime can't Consider this linked list as garbage.

* Tracing GCs

Tracing Collector will determine the objects' reachability by tracing them from a set of root objects, Known as GC roots. If an object is reachable from a root object, either directly or indirectly, then it will be Considered alive

11/6/24

Spring boot Interview Q&A

1. Can we use @EnableAutoConfiguration, @ComponentScan @Configuration instead of @SpringBootApplication. - yes

2. @ConditionalOnClass({DatabaseClass, ..., Class})
public class DatabaseAutoConfig{
}
If the mention class is present in classpath then load the particular class

@ConditionalOnMissingBean(type="com.ra...sql.DataSource")
public class DatasourceConfig{
}
If the bean is missing then create the bean or load the below class

@EnableConfigurationProperties(DatasourceProperties.class)
If the datasource related properties are found, then only it load the below class
ex- Spring.datasource.url, Spn...username, Spnn.datasource.password.

3. How can you exclude auto Config classes?
@SpringBootApplication(exclude={Datasource.class, JWT.class})
or
App.properties.
Spring.AutoConfigure.exclude = org.springframework.boot.AutoConfigure.NOP,
Com.practice project.Config.OAuth2Config

Spring Boot Q&A - part 2

1. Can you explain purpose of Stereotype annotation?

It is used to mark a java class as a bean. Spring framework pick it up and configure it in the application Context as Spring bean. And also define the role of the class

@Component, @Service, @Repository, @Controller, @RestController

2. What is dependency injection

It is a design pattern and it is used to achieve loose coupling between the classes by removing the direct dependency between them. It is used to implement Inversion of Control.

3. Types of Dependency Injection

1. Field level DI
2. Setter Injection
3. Constructor Injection

What are the different between this three above DI

1. Constructor Injection

* If we need a dependency mandatory while creating bean of parent we must go for Constructor DI, while startup
* Immutable in nature = Not option to get the bean to modify
* we can't resolve circular dependency

2. Setter Injection

* Mutable in nature
* If Not mandatory to create a bean, we go for setter injection. Create bean only if we needed.
* optional
* Resolve circular dependency by @Lazy

4. Use of @postConstruct

It is used on method that needs to be executed after dependency injection is done to perform any initialization.

Ex- Connections initialize, logger obj creation, kafka config

5. How can we dynamically load values in a spring boot app?

```
@value("${discount.offer.price}")
private int discount;
```

```
app.properties
discount.offer.price = 20
```

or

```
@Autowired
Environment environment;

environment.getProperty("discount.offer.price")
```

or. Environment variables in debug config tab

6. Difference between yml & properties files.

* Syntax & structure
* Hierarchy - yml ✓, prop ✗
* List and arrays - yml ✓ prop ✗
* Complex Data types ✓ ✗
* Readability ✓ ✗

7. If same values or configured or different values are configured which value will read first yml or properties file?

More precendence than yml

application.properties - 1st propertiesPropertySourceLoader.java

- yaml or yml - 2nd yamlPropertySourceLoader.java

PropertySourceLoader - Interface

8. How to load external properties in spring boot?

Spring.Config.import in properties file
= file:/usr/kk/test.properties

* external properties get more priority the internal

9. How to map or bind Config properties to java obj?

@Configuration
@ConfigurationProperties(prefix = "spring.datasource") ← Database related properties
public class Config {

String username;

String password;

part-3 SpringBoot Q&A

1. How will resolve bean dependency ambiguity? byType

@Autowired Honda or Suzuki
@Qualifier("RequiredClassName") Honda implCar
private Car car; Suzuki implCar

2. Can we resolve dependency ambiguity without @Qualifier? Yes byName

using @Resource (name="RequiredClassName") bean alias name
↓
From jakarta.annotation from Java

@primary - class level

3. Bean Scope

Bean Scope refers to the life cycle and visibility of a spring bean within the IoC Container. or How long a bean instance will alive, when it will be Created or destroyed

* Singleton
* Prototype
* Request
* Session
* Application
* WebSocket

Request, Session, Application → Web based application

WebSocket → Stand alone or web based app
↳ thes two will introduced Spring 4.0 onwards

@Singleton @Scope ("Singleton")

Spring will create a single bean instance for all request in application context. Spring IOC container will get the instance from inMemory rather that creating new instance every time.

@prototype @Scope ("prototype")

A new instance of the bean is created every time it is requested from the Spring Container.

Request - only for web applications

It creates a new instance of the bean for each HTTP request, After processing the request, the bean instance is typically discarded.

```
@Scope(value = webApplicationContext.SCOPE_REQUEST,
        proxyMode = ScopedProxyMode.TARGET_class)
```
↓ Mandatory attr

Session @Scope (value = webApplicationContext.SCOPE_SESSION, proxyMode = "")

It creates single instance of the bean per user session. The instance persists throught the user's session and is typically discarded when the session ends.

Application

It is used in web-based Spring Boot Applications to Create a single instance of the bean for the entire web application Context.

webSocket

It Creates a single instance of the bean per WebSocket session.

4. Can we mention or Create Custom Scope ? Yes

By ~~extending~~ Implementing Scope interface

main overriding methods are get(), remove()

@Scope ("threadScope") - Class level

To say this is the Scope we Created to Spring

```
Scope scope = new CustomThreadScope()
Context.getBeanFactory().registerScope("threadScope", scope);
```

5. UseCases for when to choose Singleton & prototype

Singleton	prototype
* Database	* User session
* Service layer	* Thread safety
* Application Config	* Heavy initialization

6. Can we inject prototype bean in singleton bean?

Yes; but prototype bean will act as singleton. To overcome we need to do different setup

```
@Component
public Class Singleton {
    @Autowired
    private prototypeScope prototype;

    public prototypeScope getprototype() {
        return prototypeScope
    }

    @Autowired
    ApplicationContext Context;

    public prototypeScope getScope() {
        return Context.getBean(prototypeScope.class)
    } -> Returns prototype object

                or

    @Autowired
    Privat ObjectFactory<prototypeScope> objFact;

    public prototypeScope getobj() {
        return prototypeBeanObjectFactory.getObject();
    }

                or

    @Lookup
    public prototypeScope getInstance() {
        return null;
    }
```

7. @Lookup

It is one of the injection (like @inject, @Resource, @Autowired) annotations used at the method level. This annotation tells spring to overwrite the method, redirecting to the bean factory to return a bean matching the return type of the method.

8. Difference Between Spring singleton & plain singleton?

1. Spring singleton is bean Instance is Created for application Context. If we Created two application Context the 2 instance will Created for singleton Scope

2. Plain singleton Scope within the JVM.

9. Purpose of BeanPostProcessor interface in spring?

To do custom action on before and after bean initialization.

```
Class 'UserPostProcess' implements BeanPostProcessor{
    @Override
    public object postProcessBeforeInitialization(Object bean, String beanName)
    {
        inside we write logic
        return bean
    }
```

by default its Default

```
Default Object postProcessAfterInitialization(object bean, String beanName)
    return bean
```

3-10-24

SpringBoot Q&A part-4

1. Different between @pathvariable & @Requestparam

@pathvariable - Mandatory to give value otherwise 404 error occurs

localhost:8080/products/filter/electronics or something

```
@GetMapping("/filter/{ProductType}")
public ResponseEntity<?> getproducts(@pathvariable String productType){
}
return ResponseEntity.ok(products);
```

optional

@Requestparam - optional localhost:8080/products/filter?productType=medicine

```
@GetMapping("/filter")
Public ResponseEntity<?> getproducts(@Requestparam(value="productType",
                                                   Required=false,
                                                   defaultvalue="Electronics"){
List<products> prod = productType!=null ? Service.getproductByType(productType) :
                                          Service.getAllproducts();
ResponseEntity.ok(prod);
}
```

2. @RestController & @Controller

@Controller - Always expect a return types as html, JSP or Something
If there is not Html file name like return value error will occur

```
@GetMapping("/welcome")
public String welcomeMsg(){
    return "welcome"; if welcome.html presents ✓, else error will occur
}
```

explicitly Return Type by

```
@GetMapping(value="/greeting", Produces = MediaType.TEXT_PLAIN)
@ResponseBody
public String welcome(){
    return "welcome";
}
```

@RestController

Combination of @Controller & @ResponseBody

3. Can we perform update action in post Http method ?

We can perform update in post Http method and insert in put HttpMethod. It will violate principle of rest service. It will violate the <u>Idempotent</u> Nature.

× Idempotent

Idempotency is a property of operations or API requests that ensures repeating the operations multiple times produces the same result as executing it once.

GET, HEAD, OPTIONS, TRACE, PUT, DELETE

But Not safe

4. How can we perform Content Negotiation (XML/JSON) in rest endpoint ?

Return both xml or Json, or return any mediatype

```
@GetMapping(Produces ={"application/Json", "application/xml"})
```

Json is already defined of Rest full service by Jackson

For xml binding we need to add dependency in Pom.xml

4-10-24

need to add jackson xml dependency in pom.xml

app.properties

spring.mvc. Contentnegotiation. favor-path-extension = true
· Content negotiation. favor-parameter = true
· Contentnegotiation. defaultContentType = application/xml

5. Status Code you have observed in your application?

* 400 - Bad Request. If the endpoint expecting some type of inputs but are giving some input

* 404 - Resource not found

* 401 - Authentication error (unAuthorized)

* 403 - Good Authentication is done, but authorized to access a specific endpoint

* 405 - Method not allowed i.e if we try to send a request for getMapping by post-Type this error will occur

* 415 - unSupported Media Type

* 500 - Generic error. That server encountered an unexpected Condition. unable to fullfill our request

* 502 - Bad gateway

* 200 - ok

* 201 - Request fullfilled and a new resource has been Created

* 204 - No Content

* 402 - Successfully Deleted.

6. How can you Customize status Code of your endpoints?

```
@ ResponseStatus( value = HttpStatus.CREATED)
public String print(){
    return "ok";
}
```

4-10-24

need to add jackson xml dependency in pom.xml

app.properties

Spring.mvc. Contentnegotiation. favor-path-extension = true

· Contentnegotiation. favor-parameter = true

· Contentnegotiation. defaultContentType = application/xml

5. Status Code you have observed in your application?

* 400 - Bad Request. If the endpoint expecting some type of inputs but are giving some input

* 404 - Resource not found.

* 401 - Authentication error (unAuthorized)

* 403 - Good Authentication is done, but authorized to access a specific endpoint

* 405 - Method not allowed. If we try to send a request for getMapping by post-Type this error will occur.

* 415 - unSupported Media Type

* 500 - Generic error. That server encountered an unexpected Condition unable to fullfill our request

* 502 - Bad gateway

* 200 - ok

* 201 - Request fullfilled and a new resource has been Created

* 204 - No Content

* 402 - Successfully Deleted.

6. How Can You Customize Status Code of your endpoints?

```
@ ResponseStatus(value = HttpStatus.CREATED)
public String print(){
    return "a";
}
```

201

7. How can you enable Cross origin?

Cross origin is allowing particular server request from other domains / Port / application to access our endpoints eg Angular: Accessing back end from angular has different port number like 4346

@ CrossOrigin (origins = "http://localhost:4356")
@ RestController

Java based Config

implement WebMvcConfigurer

@Override
void CorsMapping (CorsRegistry registry) {
registry.addMapping("/api/**")
.allowedOrigins("http://localhost:4356")
.allowedMethods("GET", "PUT", "POST", "DELETE")
.allowedHeaders("origin", "Content-Type", "Accept", "Authorization")
.allowCredentials(true)
.maxAge(4000);

8. How will upload a file in spring?

@RequestParam("file") MultiPartFile file,

9. How do you maintain versioning for your REST API?

by RequestParam, URI, RequestHeader

10. How will you document your Rest API?

Swagger / Open-api Springdoc - openapi - UI
Add dependency in pom.xml
To get proper documentation of our endpoints & Quick Test

@Hidden - Method level annotation - To hide the particular rest endpoint to be documented or exposed.

@operation (description = "About this endpoint");

11. How will you consume restful API ?

* Rest Template
* Feign Client
* WebClient
* Advance Rest Client - SpringBoot 3.2 Introduced.

* Feign client - By Netflix
It is declarative web client. It makes writing web service clients easier. To use feign client - Create an interface and annotate it. Spring cloud integrates Ribbon and Eureka to provide load balanced http client. It provides service discovery features. Creating proxy for a particular remote service by interface

* Need to add dependency in pom.xml
* @EnableFeignClient in main method
* @FeignClient(url="....", name="USER-CLIENT")

```
Interface UserClient {

    @GetMapping("/users")
    public List<Users> getUsers();
```

* WebClient

It is used for reactive programming to achieve asynchronous and non blocking consumer implementation.

5-10-24

Spring Boot Interview Question - part - 5

1. How will you handle Exception in your project?

@RestControllerAdvice
Class GlobalExceptionHandler {

}

* ProblemDetail Introduced in Spring 3.0 to send a response, instead of MessageVO (Custom class created for return a exception Details)

return ProblemDetail.forStatusAndDetail(HttpStatus.BadRequest, exception.getMessage());

it provides URI type, title, Status, detail, URI instance, properties

2. How will you validate or sanitize your input payload.

Dependency - SpringBoot-Starter-validation

ViewVO
@NotNull
private int viewId;

Annotation from jakarta.validation.constraints.*; 3.0

Thus validation is useful for restapi either we validate in client side

@NotBlank
private String viewName;

@Min, @Max, @BigInt @BigDecimalMin, @NotEmpty, @Email, @DecimalMin @DecimalMax
@AssertFalse, true,

Throws "MethodArgumentNotValidation";

@NotNull(message = " viewId must not be null")

3. How can you define custom bean validation?

4. How can you enable a specific environment with using profiles?

application.Properties
app.active.env = prod

@Bean
@ConditionalOnProperty (prefix = "app.active", name = "env", havingValue = "prod")
//@Profile ("prod")

5. Difference between @profile & @Conditional

1. @Profile is related to only profile level change
2. @Conditional is related to generic, bean creation for Some Condition, db level Config, Kafka Config etc.

6. What is AOP? Aspect Oriented programming

It increase modularity by allowing the Separation of Cross cutting concerns. Cross-Cutting Concerns are aspects of a program that affect multiple parts of the application, such as logging, security, notification, transaction management.

<u>Aspect</u> - Mark a class as aspect

Key unit of modularity in AOP, representing a Concern that cuts across multiple classes.

<u>Advice</u>

This action can taken by an aspect at a particular join point.

7. PointCut & Joint point

<u>Jointpoint</u>

Jointpoint is nothing, the target method where we want to implement the particular aspect. Like LoggingAspect, TransactionAspect, validation Aspect

<u>Point Cut</u>

The expression that we tell to the Aop who is your jointpoint

@pointcut("execution(public void com.example.Service.*.*(..))")

8. Types of Advice?

* Before - Aspect executes before jointpoint executes.
* After - Aspect executes after jointpoint executes.
* AfterReturning - Aspects executes when jointpoint throws error.
* AfterThrowing - Aspects executes only if jointpoint throws error.
* Around - Advice the runs before and after execution.

For around advice (Jointpoint JP) arg will not work Instead of use (proceedingJointpoint PJP)

SpringBoot - Persistance Layer - Q&A - part-6

1. How does your application interact with database & which framework are you using?

JPARepository

2. Hibernate Naming Conversion

privateint UserId - Hibernate will create column name as user_id

To avoid this naming Convention

spring.jpa.hibernate.naming.physical-strategy = org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl.

3. Key benefits of Spring Data JPA?

Abstraction level - JPA is higher level API that provides an abstraction layer over JDBC, allowing you to interact with the abstract database through objects and their relationship rather than SQL queries directly.

* Supports Query method generation by method Naming Conventions

4. Difference Between Hibernate, JPA, Spring Data JPA

Hibernate	JPA	Spring Data JPA
• ORM Tool • Table Mapping • Caching / transaction • It's like impl of JPA	Specification gives APIs & methods Does not contains any impl like interface	It is just a layer of abstraction on top of JPA framework

5. How can you connect multiple databases or datasources in a single application?

Entity - Repository - datasource - Mysql
↓
Provides
Entity Manager
Transaction Manager

Entity - Repository - datasource - Oracle
↓
Entity Manager
Transaction Manager

Get dataSource properties from application.properties

```
public class EmployeeDataSourceConfig {
    @Primary
    @Bean (name = "employeeProperties)
    @ConfigurationProperties ("spring.dataSource.employee")
    public DataSourceProperties dataSourceProperties () {
        return new DataSourceProperties();
    }
```

Add build dataSource object.

```
@Primary
@Bean (name = "employeeDatasource")
@ConfigurationProperties ("spring.dataSource.employee")
public DataSource dataSource (@Qualifier ("employeeProperties")
                                DataSourceProperties dataSourceProperties) {
    return properties.initializeDataSourceBuilder().build();
}
```

Create EntityManager and next TransactionManager

6. What are the different ways to define Custom Queries in Spring Data JPA?

1. Method Syntax = findByAge(), findByAgeEquals(), findByAgeBetween();
2. JPQL - "Select u From UserVO u, where u.age = :age"
3. SQL = "Select * from user_info where age = ?1", nativeQuery = true)
   List<UserVO> findAgeUsingSql(int age) - Index based

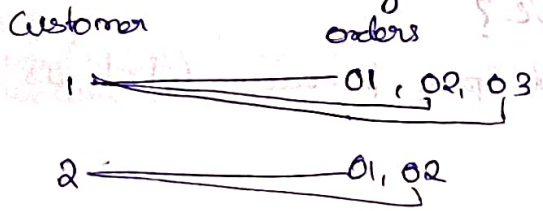
Method Syntax

Aggregation function is not supported in method Naming Convention. like findByAgeAvg();

7. How will you define entity relationship or association mapping in Spring Data JPA?

ex- ecommerce

one to many or many to one



Cust

Id	Name	age

order FK

Oid	Name	Qty	Cust_Id

```
public class Customer {
  Id
  int id;
  String name;
  int age;
  List
  @OneToMany(mappedBy = "Customer", CasCade = CasCadeType.All)
  List<Order> orders;
}
```

every customer transaction add order transaction

```
public class Order {
  int Oid;
  String name
  int Qty
  @ManyToone
  @JoinColumn (name = "Cust_Id")
  Customer Customer
}
```

8. Can we perform Join Query in Spring Data JPA? Yes

If it is microservice architecture this is not good to keep a mapping between entity, also we won't need any Join query. Each service will have own DB configuration.

By using @Query either SQL or JPQL or HQL

'Select e.name, Count(*o) from Customer e Join c.Order o
group by o.id';

9. How will you implement pagination & Sorting in Spring Data JPA?

Pagination OverAllRecord – 1000

PageNum or offset – 0,1,2,3. --- etc 0 = 1 to 100 records, 1 = 101 to 200 reco

PageSize – 100

10. Key features of Spring MVC?

1. It is sub framework of Spring framework which is used to build a web application.

2. It is build on the top of Servlet API

3. It follows the Model-View-Controller design pattern.

4. It implements all the basic features of a core Spring framework like inversion of Control, dependency injection.

8-10-24

Hibernate

latest version - 6.6 Stable

It is a java framework that simplifies the developement of java development application to interact with the database. It is a open source, lightweight, ORM tool.

1. open source and light weight.
2. Fast performance - Inbuild cache, First & second level cache. First level cache is default enabled
3. Database Independent query - HQL
4. Automatic Table Creation
5. Simplifies Complex join
6. provides query statistics & database status

@Embeddable → Create new column or map column with the particular object -

```
@Entity
Class User {
    int userId;
    int age;
    @OneToMany(mappedBy="Student")
    int UserName username;
}
```

```
@Embeddable
Class UserName {
    String firstName;
    String lastName;
    String middleName;
}
```

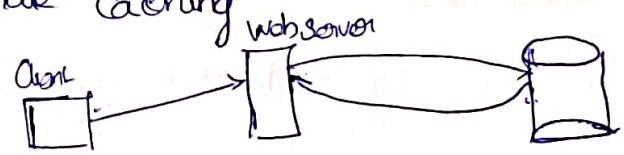
9/10/24

* Fetch Eager / Lazy

By default lazy.

@OneToMany(mappedBy="alien", fetch=FetchType.EAGER)

* Hibernate Caching



* First level cache,
* Second level cache.

* First level cache - For single session of same user.

If a user firing query for first time "hitting the DB" records are retrieved & stored in first level cache. If the same user again firing the same query (with same session) record will retrieved from "First level cache". (Session scope). By default First level cache is enabled, no way to disable it.

* Second level cache

It is sessionFactory scoped, means its shared by all sessions created with same session factory

* If an instance is already present in the first level cache, it's retrieved from there.

* If an instance isn't found in the first-level cache, and the corresponding is cached in second level cache, then the data is fetched from there

Otherwise the necessary data are loaded from the database and an instance is assembled and returned.

For second level cache provider we have go for third party API.

EhCache - one of the cache provider
osCache, Swarm;

@Cachable - For entity class
@Cache - Strategy of caching

Pom.xml

Dependency

1. net.sf.ehcache
2. org-hibernate

hibernate.cfg.xml

```
<property name="hibernate.cache.use_second_level_cache">true</property>
<p> name="hibernate.cache.region.factory_class">  org.hibernate.cache.NoCacheRegionFactory</p>
name="
        use_query-cache > true <
```

```
Configuration con = new Configuration().configure().addAnnotatedClass(Alien.class);
ServiceRegistry sr = new ServiceRegistryBuilder().applySettings(          );
SessionFactory sf = con.buildSessionFactory(sr);
Session s = sf.openSession();
s.beginTransaction();

Query q = s.CreateQuery("from alien where aid=:id");
q.setCacheable(true);
```

HQL - Hibernate Query language

```
Select rollno from Student;
       |          |
     property     class
```

Q.list() - Return list of object

Q.uniqueResult(); Returns single record

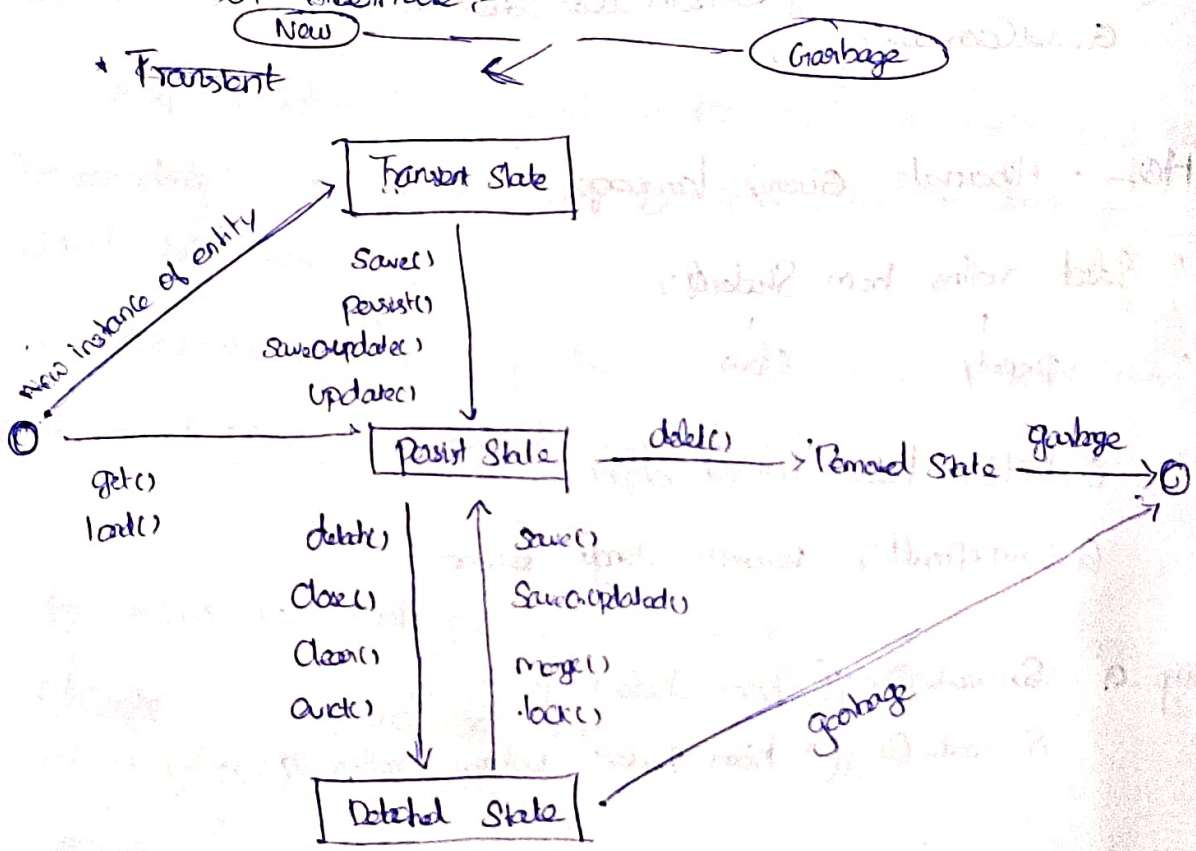
```
Query a = s.CreateQuery("from student");
          s.CreateQuery("from student where rollno = ?");
```

```
int b = 10;

Query q = session.CreateQuery("Select sum(marks) from student s
                                where s.marks > :b");
Q.SetParameter("b", b);
```

```
SQLQuery q = Session.CreateSQLQuery("Select name, from students
                                     where marks > 60");
q.addEntity(Student.class);  -> map to entity class
List<Student> stu = q.list();
```

```
SQLQuery Q = Session.CreateSQLQuery("Select name, marks from student where
                                     marks > 60");
~~Q.SetResultTransaction~~
Q.SetResultTransformer(Criteria.ALIAS_TO_ENTITY_MAP);
```

* States In Hibernate:-



1) Transient-

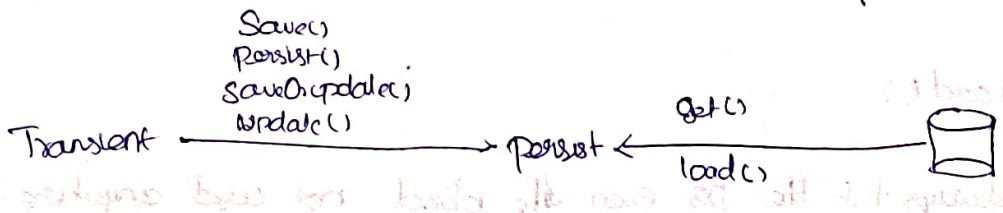
* First State
* When initiate we new() then object is in transient state
* We can modify object
* It is not connected to any hibernate session, and state is not connected to database table

2) Persist State

* One the object is connected with the hibernate Session then the object moves into persistent state

* Save the entity object
* load the entity object

For the above methods, each object represents one row in the table, Therefore if we make changes in data hibernate will detect these changes and update in DB.



3) Detached State

For convert persist to detached, we either have to close the session or we have to clear its cache. once these steps are done then any changes in the data will not be affected in DB.

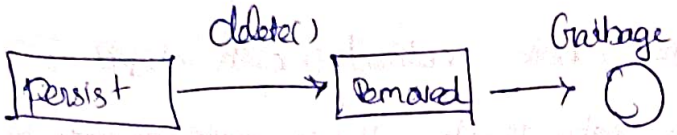
Session . detach(e)
. evict (e)
. Close (e)
. Clear (e)

* When ever needed detached object can be reconnected to new hibernate session by using

: merge()
update()
load()
Save()
refresh()
update()

4) Removed state

* In Hibernate lifecycle it is last state
* When the entity object is removed from DB the is known to be removed state.
* Changes in data will not affect DB.



Get() vs load()

* get() always hit the DB. even the object not used anywhere after fetch.
* load() doesn't hit the DB if the fetched object not used anywhere after fetch. Basically it gives proxy object or dummy obj.
* get() will throw NullPointerException() if the id or something not present in DB.
* load() will throw ObjectNotFoundException();

Hibernate with JPA

```
Alien a = new Alien();
a.setAid(6);
a.setAname("KK");
a.setTech("Java");

EntityManagerFactory emf = persistence.CreateEntityManagerFactory("pu");
EntityManager em = emf.CreateEntityManager();
```

```
em.getTransaction().begin();
em.persist(a);
em.getTransaction().commit();
```

11/10/21.

Hibernate

ORM Features

* By mapping between logical business model & physical storage model, ORM implements domain model pattern

* It navigates object relational transparency.

* ORM provides one way to translate Query to appropriate database syntax.

* It provides Concurrency Support.

* It provides a host of services.

* It provides Cache management

* Surrogate key, identifier, and other key features can be automated in ORM.

Hibernate CORE API'S

1. Configuration. (org.hibernate.cfg)

* Loads mapping & Config file in to memory & makes them available to hibernate engine

2. SessionFactory (org.hibernate.SessionFactory)

* Hibernate implements SessionFactory Interface

* Session Factory is one per DBMS

* Thread safe.

* Not a Singleton.

* Provide Second level Cache

3. Session.

* light weight
* It encapsulates Connection & first level cache
* Not thread safe
*

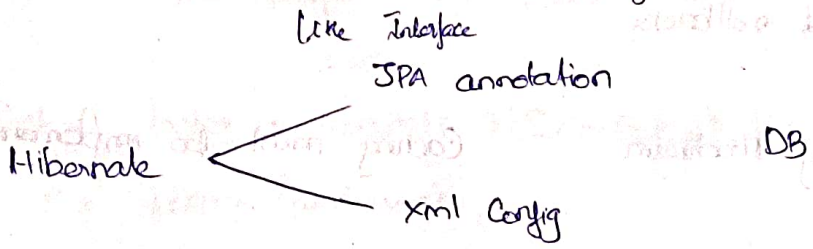
4. Transaction.

* By Default AutoCommit is in disable mode

Hibernate Q & A

1. What is Hibernate & why we use?
2. How we map object to Database Table

It provides JPA implementation hence we can use JPA annotations as well as xml Config.



3. Why Hibernate?

* Hibernate eliminates all the boiler-plate code that come with JDBC

Boilerplate codes.

4.

Hibernate	JDBC
1. It uses a session object, which manages the connection & transactions automatically	· You need to explicity manage connection, statements, and resultsets
2. Define our Data model using XML or annotations, Hibernate handles mapping between Java objects & database tables	· Manually map resultsets sets to Java objects, requires more repetitive code
3. save(), update(), delete(), find()	we have to write SQL statements & handle exceptions & resultset for each operations
4. HQL - allowing for more object oriented querying	Requires writing raw SQL
5. Transaction Management is often integrated into session handling. Simplifying commits & rollbacks	Need to manage transaction manually, add more complexity.
6. In built Caching Mechanism	Caching must be implemented manually

* Hibernate throws JDBCException or Hibernate exceptions which are <u>unchecked</u>. No need to write try and catch

4. Important Interfaces used in Hibernate?

* SessionFactory

* Used to retrive Session objects for database operations.
* We need to initialize that once and can cache it to reuse.

* One sessionFactory obj per database connection. Like 1 for mySql, 1 - for oracle.

* Session

Its factory for transaction, it's used for connecting application with persistan store like hibernate framework / DB. It is used to get a physical connection with the database, It also provides methods for CRUD operations.

* Transaction

This specifies single / atomic units of work

```
SessionFactory factory = metadata.getSessionFactoryBuilder().build();
Session S = factory.openSession();
Transaction T = S.beginTransaction();

    S.Save(obj);
    T.Comit();
    factory.close();
    Session.close();
```

5. Imp Annotations used in Hibernate

* Jakarta.persistence.Entity : used with model classes to specify that they are entity beans.

* @Table --used with entity beans to define corresponding table name in database.

* @Access - used to define the access type either field or property. By Default value is field
If we want hibernate to use getter/setter methods then we need to set property eg:- @Access(value = AccessType.PROPERTY)

@Id - used to define primary key in entity bean

@EmbeddedId - used to define composite primary key in the entity bean.
Must implement Serializable, equals() and hashCode();

@ Column - used to define the column name in database table.

@ GeneratedValue - used to define strategy to be used for generation of primary key. Used in conjuction with Jakarta.persistence.GenerationType.enum
eg:- GenerationType.IDENTITY

@ OneToOne - Used to define mapping between two entity beans.

* @ cascade - ~~@Cascade~~ @OneToMany(mappedBy = "Customer", Cascade = CascadeType.ALL)

CascadeType. PERSIST - when parent entity is persisted, the child entities are also persisted
MERGE
REMOVE
REFRESH
DETACH
ALL - Applies all cascade operations

Cascading in hibernate refers to the ability to propagate certain operations like (persist, merge, etc) from one entity to its related entities automatically.

@primaryKeyJoinColumn

This is used in hibernate / JPA in the context of inheritance strategies, particularly when we're dealing with joined inheritance. This annotations specifies foreign key column that refers a primary key of the parent entity in a subclass.

```
@Entity
@Inheritance(Strategy = InheritanceType.JOINED)
public abstract Class Vehicle {
    @id
    Private long id;

    Private String brand;
}
```

```
pub
@Entity
@PrimaryKeyJoinColumn(name="vehicle_id")
public Class Car extends Vehicle {
    private String FuelType;
}
```

```
@(Inheritance(Strategy = InheritanceType.SINGLE_TABLE)
@Inheritance (Strategy = InheritanceType.TABLE_PER_CLASS)
```

6. Association Mapping

3 types of Association mapping Hibernate

OneToOne	OneToMany	ManyToMany

```
@OneToOne (targetEntity = CodeDecode.class)
@OneToMany (targetEntity =
```

```
@ManyToMany
@Entity
Public class Student {

Pvt Long id

@ManyToMany(mappedBy = "Students")
Pvt. Set<Course> Courses = new HashSet<>();

}


Public Class Course {

    Pvt long id;

    pvt String title;

    @ManyToMany
    @JoinTable ( name = "Student_Course",
                 JoinColums = @JoinColumn(name = "Course_id"),
                 InverseJoinColum = @JoinColumn(name = "Student_id")
    Pvt Set<Student> Students = new HashSet<>();


}
```

7. Hibernat Config file & Mapping file

hibernateOracle.cfg.xml , hibernatePostgres.cfg.xml

```
<hibernate-Configuration>
    <Session-factory>
        <property name = "hbm2ddl.auto"> update</property>
        <property name ="dialect"> org.hibernate.dialect.Oracle9Dialect</p
        <property name = "Connection.url" >                    <propert>
        <p name = "Connection.username">                       <p>
        <p name = "Connection.password">                       <p>
        <p name = "Connection.drive_class>                     <p>
    <Mapping- resource = "employee.hbm.xml">
```

Cls_name_hbm.xml

<hibernate-mapping> root elm

<class name = "com.KK.employee" table = "empTable">

<id name = "emp-id">

<generator class = "assigned" or "autogenerated"> </generator>

</id>

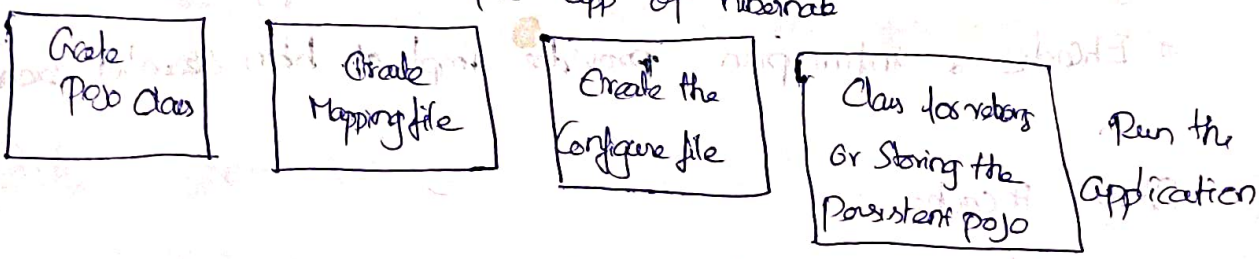
<property name = "firstName"> </property>

"lastName" etc

</class>

</hibernate-mapping>

8. Steps to create Sample app of hibernate



9. openSession() and getCurrentSession()

getCurrentSession() method returns the session bound to the Context. Since this session belongs to the Context of hibernate, One the SessionFactory is closed, this session object get Closed.

OpenSession() - To open new Session

We should close this session object once we done with all database operations.. For multi-threaded envi we should open a new session.

10. Hibernate

First level ↓ Second level Query level

* First level cache is associated with session obj
* By default enabled, no way to disable it.
* Still hibernate provides methods through which we can delete selected objects from the cache or clear the cache completely
* Cache object will not be visible to other session.
* It implement first level cache through persistent context.

Second Lvl Cache

* By default disabled, we can enable throw configuration.
* EhCache & Infinispan provide impl of hbm second lvl cache.

Setup · Ehcache

Add hibernate EhCache dependency in pom.xml or add corresponding jars

* Config file

hibernate.cache.ehcache.EhCacheRegionFactory
use_second_level_cache
use_query_cache

"net.sf.ehcache.configurationResourceName">/ehcache.ehcache.cache.xml<

13-10-24

Spring Boot Interview Questions

1. How does Springboot supports Internationalization in microservices

* It supports internationalization (i18n) in microservices through a combination of features that allows developers to create application that can adapt to different languages & regions.

:- Message Source Configuration

Spring uses MessageSource interface to manage messages in different languages. Typically done through properties file

Default prop file - messages.properties for default language

language specific - messages_en.properties, messages_fr.properties

Config in Spring Boot :

Spring.messages.basename = messages

And Need Java base Configuration

2. Bean Life Cycle

1. Bean definition

Before a bean is created, it must be defined in Spring Context. @Component, @Service, @Repository or xml file.

2. Instantiation.

When the ~~conte~~ application context is initialized, spring creates an instance of the bean using Class Contructor.

3. Populating properties

After instantiation, Spring populates bean's properties by injecting any dependencies defined in the configuration. This is done using setter or constructor injection.

4. BeanNameAware Interface

if the bean implements the BeanNameAware interface, Spring call the SetBeanName() method passing the bean's ID only one method

5. BeanFactoryAware Interface

If bean implents this interface, Spring calls the SetBeanFactory() method providing a reference to the BeanFactory that created the bean.

This can be useful for beans that need to access or manipulate their surrounding Spring container or require additional beans from the factory.

Use Case

* Accessing other beans, A bean can retrieve other beans from bean factory.

* It allows a bean with bean factory for tasks like refreshing, destroying, or checking the availaty of other beans.

6. Application Context Aware (I)

It is Part of Spring framework & allows a bean to be aware of its owning ApplicationContext.

This is useful for bean that need to access the spring application Context to perform operations like retrieving other beans, accessing environment properties, or even publishing events.

7. pre-initialization (Bean post-processors)

* To customize bean instantiation, configuration, & lifecycle management.

* They are invoked during bean creation process, after and before of bean initialization.

BeanPostprocessor - postProcessBeforeInitialization(Object bean, String beanName)
postProcessAfterInitialization(Object bean, String beanName)

8. Initialization

InitializingBean Interface - afterPropertiesSet().

9. post Initalization

allowing further modifications after initialization

11. Destruction (optional)

When the application Context is closed, spring will destroy the bean. If the bean implements the DisposableBean interface, destroy() method will be called.

If a custom destroy method will defined using @postpreDestroy annotation or through XML config that method also called.

3. Can we create custom bean scope ? yes

* Create custom threadlocalScope class
  inside this class create inner class extends ThreadLocal

* RegisterScope

```
ConfigApplicationContext . context = new AnnotationConfigApplicationContext();
Scope scope = new ThreadLocalScope();
context.getBeanFactory().registerScope("threadScope", scope);
```

* Create pojo class user ex

* Config class that return new instance that thread scope

```
@Bean
@Scope("threadScope")
public user getUser() {
    User u = new User();
        u.set();

    return u;
}
```

4. Bean FactoryI vs ApplicationContextI

BF	AP-Con
* Lazy loader	Eagaly loaded
* Slower for first time calling	
* Basic DI	Supports (Aop, Internationalization, event etc
* Event handling not supported	Event handling supported
* Internationalization not supported	Internationalization supported
* Basic Bean life cycle	Advaced (@post Construct)

* Bean Factory is for light weight Containers & do not require adv features

* Application Context for most applications, as it provides a richer set of capabilities, powerful & flexible.

5. How do Pagination & sorting works in JPA ?

EntityManager / JPA

```
TypedQuery<Entity> Query = EntityManager.CreateQuery("Select e From Entity e", Entity.Class);
Query.SetFirstResult(page * size);  Starting position
Query.SetMaxResult(size);  Max number of results to return
return Query.getResultList();
```

<u>using JPA</u> JPA repository or PagingAndSortingRepository
Use page Interface - getTotalElements(), getTotalPage(), getContent();

```
public Page<User> ListEmployee(int pageNumber, int pageSize, String field) {
    return empRepo.findAll(PageRequest.of(pageNumber, pageSize).withSort(Direction.DESC, field));
```

Pageable pb;

6. How does spring boot handles complex transactions & scenarios?

@Transactional in spring used to define the transactional boundries for method or class.

1. Propagations

We can define how transactions should behave when invoked by other methods

1. Propagation.REQUIRED
2. Propagation.REQUIRES_NEW
3. Propagation.NESTED
4. Propagation.SUPPORTS
5. Propagation.NOT_SUPPORTED
6. Propagation.MANDATORY
7. Propagation.NEVER

1. Required (Default)

If a transaction exists, the method will run in that same transaction, if not, a new transaction will be created.

Use case: Where we want a method to participate in an existing transaction.

2. REQUIRES_NEW @Transactional (propagation = Propagation.REQUIRES_NEW)

Always create a new transaction, suspending any existing transaction

Use case: Use when we want to ensure that methods runs in its own transaction, regardless of outer transaction

3. NESTED

If there is an existing transaction, a nested transaction is created. If no transaction exists, it behaves like "REQUIRED".

Use case: When we want to roll back part of a transaction without affecting the whole transaction.

4. SUPPORTS

If a transaction exists, the method runs within that transaction, if not, it runs non transactionally.

use case :- Suitable for methods that can function in a transactional context but do not require one.

If called with in a transaction, it will participate, otherwise, it won't create transaction.

5. NOT_SUPPORTED

If a transaction exists, it is suspended, and the method runs without transaction.

use case : When we want to ensure that the method does not participate in any transaction

If called during active transaction, the trans will suspended while logTransaction executes.

6. MANDATORY

The method must runs within an existing transaction. if no transaction exists, an "exception" is thrown

use case : use when a method must be called with in the context of existing transaction.

If audit transaction is called without an active transaction, it throws an exception.

IllegalTransactionStateException()

7. NEVER

The method must not run within a transaction. If an existing transaction exists an "exception" is thrown.

Use Case: when you need to ensure that the method never runs within a transaction

Propagation Type	Existing Transaction	New Transaction
REQUIRED	Join	Create
NESTED	Create nested	Create
Supports	Join	none
Not Supported	Suspend	none
Mandatory	Join	exception
NEVER	exception	none
~~NOT REQUIRED~~ REQUIRES_NEW	Suspend	Create

2. ISOLATION

It defines how transaction integrity is visible to other transactions. It controls the level of visibity that one transactions has to changes made by other current transactions

@Transactional(propagation = Propagation.REQUIRED, Isolation = Isolation.READ_COMMITTED)

* DEFAULT

uses the default isolation level of the underlying database

• READ_UNCOMMITTED

Allows reading uncommited changes from other transaction. - Dirty Reads

• READ_COMMITTED

Ensures that only Committed changes are visible :(Prevent dirty reads).

• REPEATABLE_READ

It guarantees that if a row is read multiple times, it will have the same value

* SERIALIZABLE

Most Isolation level, Complete Isolation from other transactions

1. READ_UNCOMMITTED

Begin

update account set balance = balance - 100 where ac_id = 1 Condition; not Committed

Select balance from account where ac_id = 1 ; read uncommited changes

like others

Q. REPEATABLE_READ

Select * from ac; 100 rows

insert into ac ? ; now 101 rows

Commit

Select * from ac; 100 rows will be returned

3. Rollback Rules

We can specify which exception should be trigger a rollback, including checked and unchecked exceptions.

4. Time Out - In Seconds

We can set timeout for the transaction, after which it will be rolled back if not completed

```
@Transactional(propagation = Propagation.REQUIRES_NEW, Isolation = Isolation.READ_Comm
public void performTransaction() {  , timeout = 30, rollbackFor
                                                    {Exception.class})
}
```

7. How to clone a repo?

branch

git clone -b dev "uri"

- Shallow clone - git clone --depth 1 "https:_ .git"

8. Difference between git rebase and git merge

Git Merge

- Combines changes from two ~~new~~ branches and creates a new commit (merge commit) in target branch

- preserves history of both branch

```
git checkout main
git merge feature-branch
```

Git Rebase

Moves or Combines a sequence of commits from one branch on to another branch. It effectively "reapplies" commits on top of the target branch - git checkout feature-branch
git rebase main

Difference	Merge	Rebase
Commit History	Retains original branch structure & creates a merge commit	Rewrites history. Creates new commit on top of the target branch
Conflict	Conflicts are resolved once in merge commit	must be resolved for each individual commit
Commit Id	original Id maintains	New Id generated
History	Can lead to more complex history with multiple branches	Result in clean, linear history
usage	preserves history	prefered a clean history

Q. What is Maven build lifecycle?

Maven uses build lifecycle to manage the process involved in building and deploying.

default, clean, site

- Default lifecycle - mvn install

validate - checks if the project is correct & all necessary info is available, it verifies the structure and integrity of the project

Compile - Compiles the source code.

Test - Runs unit tests using suitable framework. This phase only compiles code and test without packaging

~~Verif~~ -
Package - JAR or war file

Verify - Checks on the results of previous phase, checks for code coverage

Install - Installs the package artifact into local maven repo, making it available for other projects on the same machine

Deploy - Copies the final package to the remote repo for sharing with other developers & projects.

2. mvn clean Clean lifecycle

3. Site lifecycle mvn site

19-10-24

Spring boot - 3

1. Expression Language (SPEL)

```
Person p = new Person("Alice", 30);
ExpressionParser parser = new SpelExpressionParser();
StandardEvaluationContext Context = new StandardEvaluationC
                                    ontext(p);

String name = parser.parseException("name").getValue(Context,
                                    String.Class);

String result = parse.parseExpression("age > 18 ? 'Adult' : 'Minor'")
                .getValue(Context, String.class);
```

2. Annotation Based Configuration

1. Key annotations

@Configuration
@Bean
@Component, @Service, @Repository, @Controller

3. Improved Support for RestFul Web Services

Spring 3 provides comprehensive support for building RestFul web services through the @RequestMapping annotations and RestTemplate for making HTTP Request

Features - Supports Content negotiation, HTTP method mapping, and various media Type.

4. Need atleast - Java 17

5. Improved observability & micrometer and micrometer frame logging, metrics, and traces.

6. GraalVM Native image Support - a new way to deploy & run java application.

GraalVM offers two ways to run java application: on the the Hotspot JVM with Graal Just-in-time (JIT) or Ahead-of-time (AOT) Compiled.

7. Moving Java 11 to Java 17 - we need to change
- Javax import to Jakarta.
- Change hibernate dialect properties. & use hibernate 6

8. API - Changes

@RequestMapping annotation doesn't work without @Controller, or @RestController

- we need to upgrade to Apache HttpClient 5 if we using restTemplate.
- Http method is a class and no longer as enum.
- we need to upgrade log4j2 version.

23/10/24.

1. Can we call non static methods or non static instance variable from static methods?

We cannot directly call non static methods or access non static instance variables from a static method without creating an instance of the class. This is because static methods belong to the class itself, not to any specific instance of the class.

Compilation error occur :- Cannot make a static reference to the non static field. method.

2. Static purpose of static methods?

* UTILITY Functions :- Static methods often used to create utility or helper functions that perform common tasks without needing to create an instance of the class. Ex- Math.max() or Math.Sqrt()

* Shared state :-

It can access static variables, which are shared across all the instances of a class. It can be useful for maintaining the global state or counting instances.

* Static Constructor - X Create single object, useful for memory management instead of creating new object every time.

Cannot make a static constructor

3. Why we have to store password in char[] instead of String?

* Mutability : Char[] is mutable, so we can modify its Content. Setting all Chars to "0" once we done.

* String Immutability: String objects are immutable, once created, their values cannot be changed. When we store a password in String, it remains in memory until it is garbage collected.

* Reduced Expose : It can be interned and stored in the String Pool. a

4. Which Collection should you use if you want avoid duplicates and maintains insertion order? LinkedHashSet

5. Does Treeset allows Null values?

TreeSet extends AbstractSet<E> implements NavigableSet<E>, Cloneable, Serializable

NavigableSet<E> extends SortedSet<E>

```
SortedSet<E> {
    Comparator();
}
```

It internally implements Comparator, null values are allowed. "NullPointerException" occurs while compare.

6. Which method is executed First?

HashCode(), equals();
1 2

7. Do both methods are always executed? - No

If same hashcode will created then equals methods check the values also same.

8. what is IdentityHashMap & WeakHashMap ?

* Identity Hash Map

It uses reference equality (==) instead of object equality (equals()) to compare keys. Two keys are considered equal only if they are same object.

* Key & values are held with strong references, meaning they are not eligible for garbage collection as long as the HashMap exists

* Null values : Allows one null key & multiple null values
* Usage - Suitable for general purpose mapping when we need to maintain strong reference to key & values

* Weak HashMap

It uses weak reference for its keys.

- Weak reference - keys are held with weak reference, they can be garbage collected if there are no strong reference to them. This makes it suitable for cache-like scenarios

- Null values - Allows one null key & multiple null values

- useful for caching scenarios.

Ex

```
WeakHashMap<String, String> weakMap = new WeakHashMap<>();
String str1 = new String("one");
weakMap.put(str1, "value1");   all the values are present
str1 = null;
System.gc();   after printing the map there is no objects present
```

9. use of reduce()?

It is used to perform a reduction on the element of a stream, Combining them into a single result.

* Aggregation of data - Summing numbers, Concatenating String, or find maximum/minimum values

* Immutable Result :- The result of the reduce operations is typically an immutable value.

10. Enhance made in HashMap in java 8?

1. Linked List to Btree Balanced tree when the thresholds limit exceeds. The default number of entries in a bucket is 8. This transformation improves the performance of get & put from O(n) to O(log n).

2. ForEach & Compare, Compute added to HashMap

3. HashMap supports spliterator

11. Give scenario where we should only a Comparator?

To use when where we need to compare & sort of two objects.

12. CrudRepository & JpaRepository

1. CrudRepository → Base Interface

It is basic interface that provides (CRUD) operation.

Save(), findById(), findAll(), Count, deleteById, delete, deleteAll()

* JPARepository

This extends CrudRepository and provides additional JPA-specific methods, such as Batch operations of pagination.

findAll (Sort sort), findAll (Pageable pageable), deleteInBatch (Iterable<T> entities);

Includes methods for flushing, batch operations, more complex queries.

JPARepository<T, ID> extends ListCrudRepository<T, ID>, ListPagingAndSortingRepository<T, ID>, QueryByExampleExecutor<T>

Interface ListPagingAndSortingRepository<T, ID> extends PagingAndSortingRepo<

13 Iterable and Iterator

Interface Iterable is a part of Collection frame work. But serve different purposes.

Interface Iterable<T>

Iterator ();
ForEach ();
Spliterator();

1. Usage

* Any class that implements Iterable can be used in enhanced for-loops (for-each-loops)

* ArrayList, HashSet, etc that implements classes

Iterator - Interface

It provides some methods for iterating over a Collection. It allows us to traverse the elements of a Collection & remove elements during the iteration.

* hasNext();
* next();
* remove();

14. How or which logging tool are you using? Log4J

* Java offers several logging frameworks, including log4J, Java.util.logging (JUL), tiny log, logback, and more.

* Spring boot includes SL4J by default.

Simple Logging Facade for Java; Serves as a unified interface for these logging frameworks; with just "1" dependency, we can utilize any of these logging frameworks.

* Different levels of logging To categorize log messages based on their severity & Importance

1. TRACE - Most detailed level, deep insights into flow such as method calls & variable values - For debugging and diagnosis

2. DEBUG - It Contains the messages that is useful for debugging purposes. They provide more insights of the application.

3. INFO Default Gives a high level overview of the application state and major events, useful for normal tracking

4. Error - To log the errors include exceptions, failed operations, or other issues.

5. WARN

Every logging framework comes with three elements

- Logger - Capture the message
- Formatter - Align or style
- Handler - Console, file, mail etc

We LoggerFactory class of org.slf4j and the Logger interface is the entry point to slf4j API

log.level.root = DEBUG

Logger logger = LoggerFactory.getLogger(Controller.class);

- Date and Time
- INFO - log level
- processId
- thread Name
- Logger Name
- Log message.

Spring Logging.output.ansi.enabled = true

logging.file.name = logger.log

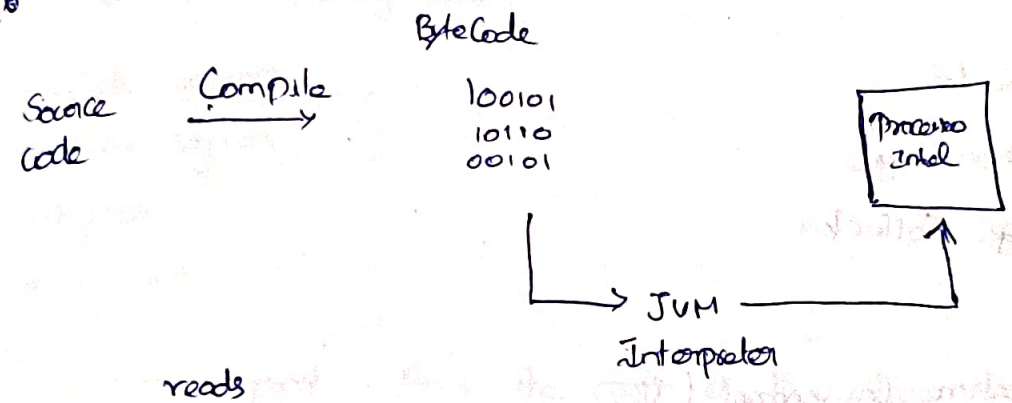
15. Difference between returning a responseEntity & Custom objects?

	ResponseEntity	Custom Objects
Control over status	Full control over HTTP Status Code	Limited to default (usually 200)
Custom Headers	Can include custom headers	Cannot include headers
Complexity	Slightly more complex to implement	Simple & straightforward
Use Case	need for flexibility in response	when only the response body is needed.

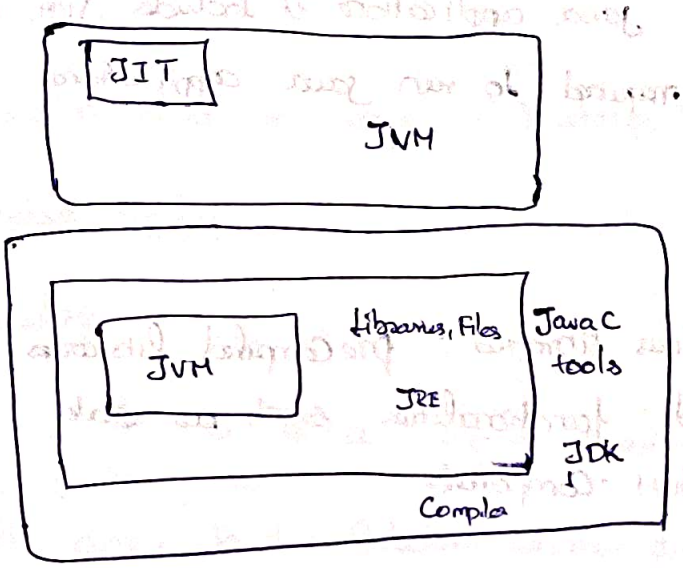
16. How you do indexing in spring boot ?

```
@Entity
@Table (name = "my_entity", indexes = {
        @index (name = "idx_my_field", Colum list = "fieldName")
    })
```

26-10-24.



JVM - reads Run the code line by line and process the code / run it. will make JVM bit slow, So JIT introduced



JVM

It enables a computer to run java programs and programs written in other languages that are also compiler to java byteCode. It provides a run time environment and is responsible for executing the compiled java byteCode.

Loading - It loads class file (.class)

Executing - JVM executes the byte code using either an interpreter (which interprets byte code line by line) or Just in Time Compiler Compiles byte code in to native machine code

Components

* Class loader
* Execution engine
* Garbage Collector

2. Java Runtime Environment (JRE)

It is a subset of JDK provides the necessary to run java applications. It includes JVM, core libraries, other components required to run java applications.

Key Components:-

* JVM
* Java class libraries: precompiled libraries that provide standard functionalities such as data structures, networking and GUI components.
* JNI

We can run java applications without JDK.

3. JDK

It includes complete development environment, it includes JRE as well as development tools such as compilers and debuggers.

Components.

* JRE
* J compiler
* JavaDoc, Jconsole, JdB - debugging

4. About Java Memory model.

* Code segment
* Data segment
* Registers
* Counters

* Code segment - Stores the byte of the class that the thread executing
* Data Segment - Stores static & global variable.

5. User session data is stored on HashMap, how would you ensure thread safety?

SynchronizedMap

* It is a wrapper around regular map that synchronizes access to it. Collections.SynchronizedMap();

Locking Mechanism

* It locks the entire map, can become a bottleneck

Iteration. Need Manual synchronized block to avoid Concurrent Modification.

```
Synchronized (map) {
    for (        ) {
    }
}
```

Features	SynchronizedMap	ConcurrentHashMap
Locking	Locking entire map	Segment lock
Performance	Slower under high	Faster, especially with many threads
	Must manually synci	Fail safe iterators
Use case	Simple use case with low latency	High concurrency scenarios
Null handling	Allows null key/value	Does not allow null key/value

6. Life Cycle of Spring IoC Containers

1. Initialization - Create Context - Load Configuration
2. Bean Creation - Instantiate bean - Inject Dependencies
3. Post processing - Invoke - Beanpost processors.
4. Initialization - afterPropertiesSet() -> Invoke Custom init methods
5. Using - Beans are ready for use.
6. ShutDown - Close context() - invoke destroy() - if DisposableBeanI - destroy()
7. Cleanup - Releases resources

7. How can we modify bean life cycle using hooks?

1. Using @postConstruct , @preDestroy
2. Implementing InitializingBean & DisposableBean
   afterpropertiesset() destroy()
3. Custom initializing
   @Bean(initMethod = "customInit", destroyMethod = "customDestroy")
4. Using BeanpostProcessor Interface implements the
   postProcessBeforeInitialization();
   postProcessAfterInitialization();
5. Application Listeners
   ContextRefreshedEvent(E), ContextClosedEvent

8. How would you ensure certain bean is load last?

@DependsOn({"beanA", "beanB"}) used with Class, Interface, method.

9. If two profiles are active; which property will load.

Spring manages the configuration by merging the properties defined in each profile.

1. Spring.profiles.active = dev, test.

The last one loaded takes precedence. this means the properties from last profile will override those from earlier profiles

Order of precedence

* app-profile.properties
* app-profile.yml
* app-properties
* application.yml

yml is loaded first. if there were properties file present, with same property then second loaded prop file will override the property.

10. How does Spring data JPA optimize repository performance?

1. Dynamic Query Generation.

Generate query based on method naming Convention.

2. pagination & Sorting. using pageable Interface, Sorting.

3. Query Caching.

JPA provides like hibernate uses 1st level Cache associated with Entity Manager.

* Second-level-cache

with appropriate Config, we can enable Second level Caching.

4. Batch processing

This reduce round trips to database.

5. Fetch strategies

* Eager and lazy loading

use lazy loading to defer the load of entities until they are used or needed.

* Custom-Fetch Joins - JOIN FETCH

6. Specification API

7. optimistic locking

21/10/24

1. Types of garbage Collector:

1. Serial garbage Collector

use one thread to do garbage Collection, Stoping other threads while it works. It's simple and good for small applications but can cause long pauses

2. parallel Garbage Collector:

uses multiple threads for garbage collection, which helps improve overall throughput. It's better for larger, multithreaded applications but can still have noticable pauses.

3. Concurrent - Mark - Sweep (CMS)

Aims to reduce pause time by doing most of its work alongside the running application. It's good for application needs low latency but can be more complex to manage.

4. G1 (Garbage Collector)

* Splits heap in to regions
* Designed for large heap size & manage pause time effectively

5. Z Garbage Collector (ZGC) & Shenandoah.

* very low pause times, handle large heap size
* Mostly works Concurrently with application.

2. Types of GC Algo.

* Mark Sweep
* Mark Compact
* Generational COPYING

28-10-24

1. How does spring boot applications start?

Starter-POM

META-INF/ Spring.factories

1. Enable

2. Disable

Based on @ Conditional and @ Configuration - Info in spring.

When we Gradling project with what are all the dependencies we mentioned, It enable the components for us to build a application.

@ Configuration

@ ConditionalOnBean (DataSource.class)

@ ConditionalOnClass(JPARepository.class)

@ ConditionalOnMissingBean {JPARepositoryFactoryBean.class, JPARepositoryConfig.class}

@ConditionalOnProperty {prefix = "Spring.data.JPA.Repositories", name = "enabled" havingValue = true, MatchIfMissing = true}

@ Import (JPARepositories AutoConfigurationRegistrar.class)

@AutoConfigureAfter ("Hibernate JPAAutoConfiguration.class)

Public class JPARepositoryAutoConfiguration {

}

1. Bootstrap the application Context → Check app type

It creates and refreshes the app context, which is the central interface to the spring IOC. This Context hold all the bean and manage their life cycle.

2. Load app prop.

It load properties from various sources, such as app.prop, or app.yml.

3. Configure Auto-Configuration.

If we have a database driver in the classpath, Spring boot will configure a data source for you.

4. Create & Register Beans

It scans the specified packages for spring Components (@Component, @Service, etc) and register them in application Context.

5. Start the embedded server.

6. Application Events.

ApplicationStartedEvent, ApplicationReadyEvent, ApplicationFailedEvent.

7. Command line Argument.

1. Create app context - check app type - register all the annotated class. - Create embedded server container and add the context.