① How to Disable/Enable the spring boot Autoconfiguration

A) By using the @EnableAutoConfiguration(exclude = {DataSourceAutoConfiguration.class})

② How to handle the exceptions in spring boot ?

@ExceptionHandler is an annotation used to handle the specific exceptions and sending the custom response to the client.

@ControllerAdvice

Creating custom Exception handler :-

1.

③ what is use of Profiles ?

A) Profiles are used to handle the environment base execution.

for example every application has many environments like

dev, TCS, Production, UAT

Each environment requires a setting specific databases like of

dev - oracle

Test - mysql

production - MongoDB ----

To work this we can add the application.properties like below

~~application-dev.properties spring.application~~

spring.profiles.active = dev

we have use @Profile("dev") to let the system let me know that

this is Bean is pickup.

④ what is the work and use of actuators in spring boot ?

A) Spring Boot is provides actuator to monitor and manage our application. Actuator is a tool which has http endpoints. when an application is pushed to production, we can choose to manage and monitor the application By using endpoints.

→ To enable this feature add the dependency

spring-boot-starter-actuator

Some actuator endpoints like.

- actuator
- autoconfig – It display the autoconfiguration report.
- health – It shows app health information
- beans – ~~create~~ display complete list of beans.
- dump – To perform thread dump
- env – Expose the properties spring configurationEnvironment.
- loggers – It show the modify configuration of logger
- trace – used to display trace information.

4) what is the use of CURDRepository interface in spring boot ?

A) curdRepository interface is provides methods for the CURD operations. This interface is extend the Repository interface. If you extending the curdRepository, there is no need to implementing your own methods.

spring boot dataJPA is a combination of Hibernate and JPA, spring flw and autoconfiguration. Data is used to perform DB operations using ORM concept. ~~JpaRepository~~ supports Basic operations and Bulk operations ~~Data JPA~~ JpaRepository is provide the all functionalities of paging and ~~sorting~~ SortingRepository and curdRepository and RepositoryInterface.

5) when ClassNotFoundException occure in java ?

A) The classNotFoundException is thrown when the ~~JVM~~ JVM trie to load a particular class and the specified class can't be found classpath The classNotFoundException is a checked exception and they must be declared in method or constructor's throws clause.

=> The forName method that reside inside the class Class.

```
try { Class loadedClass = Class.forName(CLASS_TO_LOAD)
Syso } catch (ClassNotFoundException e) { e.printStackTrace
```

① @Query annotation (or) custom query.

This annotation is used to define programmer HQL to perform multi-row select and non-select operations.

1. Define the abstract method in in Repository interface.
2. Add @Query annotation on top of the abstract method.
3. We can pass the parameters using positional parameters (?, ?1, ?2) and name (:a, :abc, :xyz) parameters.

@Query annotation gives you full flexibility over the executed statement, and your message name doesn't need to follow any conventions. The only thing you need to do is, to define a method in your Repository interface)

Ex: @Query ("select p from com.app.model.product p where p.prodId = ?1")

② How to print multiple elements count?

A) java.util.Collections.frequency() method is present in java.util.Collection class. It is used to get the frequency of a element present in the specified list of collections. More formally it returns the number of elements in the collections.

Ex:
```
public class FrequencyDemo {
    p s v m(String[] ar) {
    Integer arr[] = {10,20,10,20,30,40};
    int freq = Collections.frequency(Arrays.asList(arr), 20);
    Sy.out(freq);
    }
```

(3) Difference b/w spring 3.0 and 4.0.

A) 1. Remove deprecated packages and ~~deleted~~ methods.

2. Java 8 support.

3. Testing improvement

4. JMS + ActiveMQ
   - @JmsListener
   - @EnableJmsListener.

5. Spring caching annotations
   - @cacheable
   - @cacheEvict
   - @caching
   - @EnableCaching.

6. Improve the ~~client~~ in Email
   - JavaMailSender
   - MimeMessageHelper
   - MimeMessagePrepare.

7. Supports profiles concept.

(4) @GetMapping

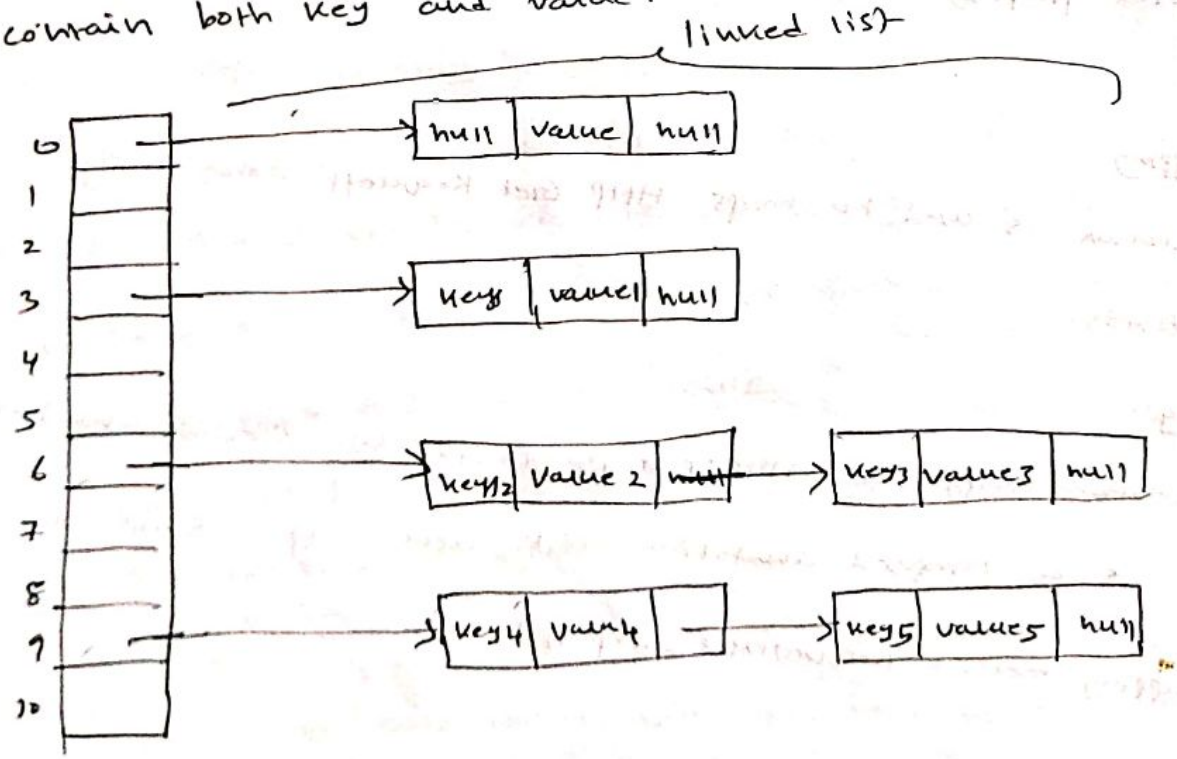
This annotation is used to maps Http Get Request onto specific handler methods.

@PostMapping.

This annotation maps the Http Post Requests onto specific handler method. It is a composed annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.POST).

(5) Explain internally HashMap working ?

1. HashMap works on the principal of hashing.
2. HashMap uses the hashcode() method to calculate a hash value. HashValue is calculated using key object. This hash value is used to find the correct ~~key whose value is to be~~ bucket where Entry object will be stored.
3. HashMap uses the equals() method to find the correct key whose value is to be retrived in case of get() and to find if that key is already exist or not in case of put() method
4. Hashing collision means more than one key having the same hash value, in that case Entry object are stored as a linked -list with in a same bucket.
5. within a bucket values are stored as Entry object which contain both key and value.



(6) How to declare a class is immutable?

1. The class must be declared as final so that child class can't be created.
2. Data members in the class must be declared as final so that we can't change the values of it after object creation.
3. A parameterized constructor
4. Getter method for all the variables in it.
5. No setters.

(7) Serialization versionId.

(8) Prototype design pattern.

(9)

(10) HashSet internal working.

A/ Now as you can see that whenever we create a HashSet, It internally create HashMap and we insert an element into HashSet using add() method, It actually call put() method on internally created HashMap object element you have specified as its key and value called as PRESENT as it's value.

If the same element is add to the HashSet it call put(e, PRESENT) method will return false and element is not added to HashSet.

remove (Object o) method returns true if the set contained the specified element.

It internally uses remove() on HashMap which returns the value of removed key i.e P.

(11) How to create docker image using Spring Boot?

Ans: Add Dockerfile in your project create src/main/docker in your project and add Dockerfile in you project.

tells docker image ← FROM: java:8-jdk-alpine

Here copy from our file to specific folder. Here copy of .jar file to Even/app folder. ← COPY ./target/demo-docker-0.01-SNAPSHORT.jar /user/app/

Set the working directory. ← WORKDIR /USR/app

Tell the docker Execute shell command the ← RUN sh -c 'touch demo-docker-0.0.1-SNAPSHORT.jar'

This allows the configure the container that will run as an Executable. ← ENTRYPOINT ["java", "-jar", "demo-docker-0.0.1-SNAPSHORT.jar"]

2-Run and maven clean and 'my bat'

2-Lets build the docker image using docke file.

$ docker build -t greeting-app.

(12) How to generate JasperReports in SPRingBoot?

1. JasperReport is an open source java reporting tool. It can generate verity of reports like pdf, Excel etc. To Enable in this feature to add one dependency in pom.xml. (jasperreports)

2. create list of employees.

3. compile the report (.jrxml or .jasper) that we have designed using jaspersoft iReport Designed tool.

```
JasperReport jasperReport = JasperCompileManager.compileReport(
        reportPath + "\\employee-report.jrxml" );
```

(14)

Exception Handling with Method Overriding :-

Rules for method overridding with exception handling.

1. If the parent class method does not declare an exception.

* If the parentclass method does not declare an exception, subclass overridden method can't be declare the checked exception but it can declare unchecked Exception.

2. If the parent class method declares an exception.

* If the parent class method declare an exception, child class overridden method can declare same, child class exception or no exception but can't declare parent exception.

(15) Marker interface :

Marker interface in java is an interface which does not have any method. Marker interface is used to inform the JVM that the classes implementing them will have some special behaviour. In java we have following four major marker interfaces.

1. Serializable interface
2. cloneable interface → There is a method clone() availble for object class. A class that implements cloneable interface it indicates that it is legal for clone() method to make a copy of field by field instance of that class.
3. Remote interface
4. Threadsafe interface.

(16) assert keyword :-

assert is a java keyword used to define an assert statement. An assert statement is used to declare an excepted boolean condition in a program. If the program is running with assertions enabled, then the condition checked at runtime.

If the condition is false the java runtime system throws an AssertionError.

17) Serialization version id.

The serialization at runtime associate with each Serializable class as a version number called as a SerializationUID. which is used during ~~to sender and receiver~~ deserialization to verify that the sender and receiver of serialized object have loaded classes for that object that are compitatable ~~of~~ with serialization.

For ex the client serializing the object to ~~ob~~ server

(18) what is JPA

→ Java persistant API is a standard API that provides functionality ~~for saving the~~ to access and manage persist data b/w the objects (POJO) and relational database in java EE platform (for saving the java objects to relational database table in to a convinient form).

→ It provides some annotations to map objects to database. They objects are called as Entities

→ It allow querying and retrival of data using query language such as JPQL

19) How to Handle the Exceptions in your project.

~~In my project I have using Controller based ExceptionHandler~~.

Actually handle the Exceptions in 3 ways.

1. SimpleMappingExceptionResolver.
2. Controller based ExceptionHandler
3. Global ExceptionHandler

(2). In this type of Exception handling we ~~create the~~ can write the method in side the controller class itself. So to achive this

a. Define a new method inside the controller
b. Annotated with this method @ExceptionHandler and parameter as Exception that we have to handle.
c. Handle the exception inside this method and return the specific view.

So, limitation of this approach i.e those Exceptions are thrown inside the controller. Any Exceptions thrown from outside the controller class will be left unhandled. ~~and same~~ like we can handle the many Exception handler methods as we want with each exception

(3) In this approach the implementation is almost similar to @ExceptionHandler approach only. The only diff here is that it handle the all the Exceptions thrown from any of the controllers in the application. It can be implemented as

a. Define a new class and annotated @AdviceController
b. Keep adding new method for each exception using @ExceptionHandler annotation.

So, class which is annotated with @ControllerAdvice is considered as global Exception handler because methods inside it can handle the exception thrown from any controller in the application.

Example :-

```
@controller
public class LoginController {

@RequestMapping(value="show", method=RequestMethod.POST)
public String showpage(Model mod) {
 map.addAttribute("user", new user());
 return "/login";
}
@RequestMapping(value="save", method=RequestMethod.POST)
public String savepage(@ModelAttribute User user, Model model) {

 String name = user.getName();
 String password = user.getPassword();
 if("kb".equals(name) && "123".equals(password) {
  model.addAttribute("user", user);
  return "/home";
 }
 else{ // user not given to proper credential
  throw new InvalidUserException("InvalidUser", "user not valid
                                  for this site");
 }
}
}
@
```

```
@RequestMapping(value="/getGenericExcep", method=RequestMethod.GET)
public String getGenericException(){
    throw new Exception();
}

@ExceptionHandler(InvalidUserException.class)
public String handleInvalidException(){
    return "userNotFound";
}
}
```

Define a controller with @ControllerAdvice annotation.

```
@ControllerAdvice
public class ExceptionHandlerController{

@ExceptionHandler(Exception.class)
public String handleGenericException(){
    return "genericException";
}
}
```

This controller we have written HandlerMethod for genericException. This handler Handle the all Exceptions thrown from ~~any~~ any controller in the application. Here throw the loginController and getting handle the Exception in that method.

---

1) DataIntegrityViolationException → Exception thrown when an attempt to insert or update data results in violation of an integrity constraint.

2) NonTransientDataAccessException

3)

(20) How to communicate two controllers in Spring MVC.

(21) what is done? (in agile)

Done on this level means "the product owner reviewed and accepted the user story. once accepted the "done" user story will contribute to the team velocity. you must meet all of the defined criteria on the user story isit done. user story DOD. Ex: unit tests passed

(22) How to invoke linux GUI using putty?

we can actually invoke any linux GUI in windows by installing x-ming sw. In this must be Enable the X11 forwarding.

(21) What is Serialization.

Serialization in Java allows us converting an java object to stream of bytes that we can send over the network or save it as file or store in DB for later usage. Deserialization is process of converting object stream to actual java object

→ If you want a class object to be serializable to implement class as serializable interface. Serializable in java is a marker interface and has no methods to implement. That tell the JVM that the objects of this class is ready for being written to and read from a persistant storage or over the n/w.

(22). why service ~~discovery~~ registry in microservice?

In a microservice application the set of running service instances changes dynamically. Instances have dynamically assigned ~~ips~~ network locations - consequently, in order for to make a request to service it must be use a service discovery mechanism. A key part of the service discovery is the service registry.

(23) what are the class loaders in java? explain.

Actually, class loader is a part of the Java Runtime Environment that ~~does~~ that dynamically loads java classes in to JVM. Usually classes are loaded only on demand. The java Runtime system doesn't to know about files and and file system because of classloaders.

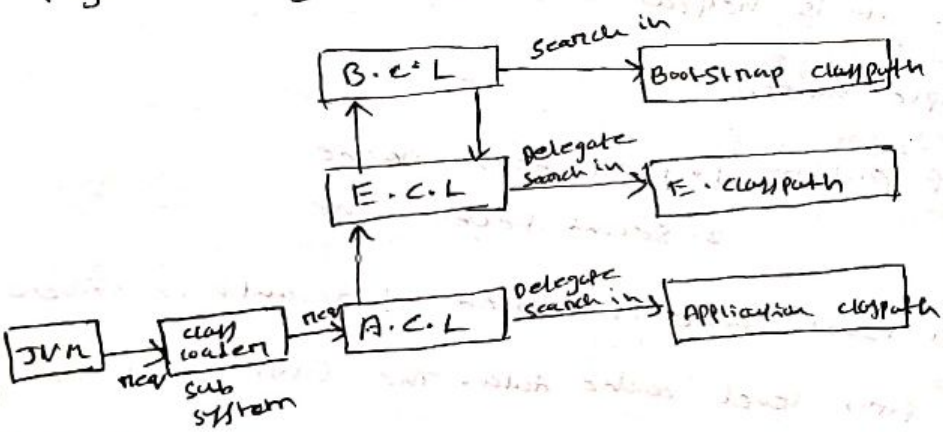
The java classloader is following the deligation principle.

In java contains 3 types of class loaders

1. Bootstrap class loader : It ~~loads~~ is a parent of all the class loaders. It does not have any parent. It loads standard JDK class from jre/lib/rt.jar and other core classes.

2. Extension class loader :- It delegate the request to its parent if the loading unsuccessful, it loads classes from jre/lib/ext directory or any other directory java.ext.dirs.

3. System (or) Application class loader : It loads application specific classes from the class path environment variables. It can be set while invoking program using -cp or classpath command options.



(25) what is difference b/w @RestController and @Controller annotation?

Ans 1. The @RestController annotation in Spring MVC is nothing but a combination of the @Controller and @ResponseBody annotation. By using this annotation we can develop RestfulWebservices in Spring framework easily.

2. @Controller annotation is a stereotype annotation By using this on class level and it is used to combination with a @RequestMapping annotation used on request handling methods.

Simply: @Controller is to create a map of model object and find a view.

- @controller is specialization of @component annotation while @RestController is a specialization of @controller annotation

```
@Target (value = TYPE)
@Retention (value = RUNTIME)
@Documented
@controller
@ResponseBody
public @interface RestController
```

```
@Target (value = TYPE)
@Retention (value = RUNTIME)
@Documented
@component
public @interface controller.
```

(15) can you explain cache in Hibernate ?

A) Hibernate caching is used to improve the performance of the application by pooling the object in the cache and reduce the n/w calls to the db. It is helpful when we have to fetch the common data in multiple times.

There are two types of caching 1. First Level cache
2. Second Level cache.

First level cache :- first level cache is cabled by default in Hibernate session object holds the first level cache data. The first level cache does not available to entire application. An application can use many session object.

2nd level cache :- SessionFactory object holds the second level cache data. The data stored in the second level cache will be available to entire application. we need to enable explicitly

- EH cache
- JBoss cache
- OSo cache

> Different vendors have provided the implementation of second level cache.

1. create the ~~Persistant~~ maven project

2. add the cache dependencies in pom.xml

3. ~~write~~ we can add cache properties in cfg.xml file.

like-<property name = "cache.use_second_level_cache"> true </property>

<property name = "cache.region.factory_class"> org.hibernate.

cache.ehcache.EhCacheRegionFactory </property>

4. on top of the model class we can annotate two annotations

@Cacheable, @Cache (usage = CacheConcurrencyStrategy.READ-ONLY)

(26) where you get the user stories in your company ?

A) I get the user from ~~Agile~~ Jira software.

once the user stories (created) prepared by BA.

~~In our company my~~ In our company my scrum master is assigning

the user stories to me.

28) How many ways clone the java object and Explain?

A clone is an exact copy of the original. In java it essentially means the the ability create an object with similar state as the original object.

In java, if a class needs to support cloning it has to following things.

1. A class must implement Cloneable interface.
2. you must override clone() method from object class method.

Ex:

```
class student implements Cloneable {
    int sid;
    String sname;
    Student(int sid, String sname) {
        this.sid = sid;
        this.sname = sname;
    }
    public object clone() throws CloneNotSupportedException {
        return super.clone();
    }
    public static void main(String args[]) {
        try {
            student s1 = new student(1, "siva");
            Student s2 = (Student) s1.clone();
            System.out.println(s1.id + "," + s1.name);
            System.out.println(s2.id + "," + s2.name);
        }
        catch (CloneNotSupportedException e) {
        }
    }
}
```

1. Shallow copy

shallow clone is default implementation in java. In overridden clone method.

2. Deep copy :- In deep copy, we create a clone which is independent of original object and making a changes in the cloned object should not effected original object.

29) Why interfaces and what is the purpose of interfaces ?

A) 1. An interface in java is provide the specification of method prototype (signature). whenever you need to guide the programmer or make a contract specifying how the methods and fields of a type should can define interface.

2. In java doesn't support the multiple inheritance using interfaces you can achive multiple inheritance.

3. one interface can inherited by multiple classes.

4. An interface is providing abstraction. an abstraction is a process of hiding implementation details from the user, only functionality will be provided to the user.

Science all the methods of interface are abstract and user doesn't know how a method written Except method signature. thats why using interfaces you can achive abstraction.

5. Loose coupling By using interface is loose coupling.

(30) Difference b/w arrayList and linkedlist

ArrayList	Linked List
1. ArrayList is implementation class of List interface	1. LinkedList is implementation class of List interface.
2. It is not synchronized	2. It is not synchronized.
3. It follows Resizable array data structure	3. It's follows Double linked list ds
4. ArrayList search operation is pretty fast compared to the linkedList search operation. Reason:- Arraylist maintains indexbased system for its elements as it uses array ds implicitly which makes it faster for searching an element in the list.	4. Linked list element deletion is faster than Array list. Reason:- Linked list each element maintain two pointers(addr which points to both neighbour elements in list
5. ArrayList maintain low memory consumption.	6. Linked list maintain high memory consumption.

(31) How to resolve the client issues? can you explain?

In our company L1 support team is there. once they will get any ticket from clients.

3) HashSet

1. HashSet implements Set interface
2. It not allow duplicates
3. HashSet used to stored distinct objects.
4. Hash set allow we can store single null value
5. By using iterator to iterate the HashSet
6. can store objects of objects as its elements
7. use HashSet when you need uniques of the data.

~~Fast - fail and fail - safe~~

HashMap

1. HashMap implements Map interface
2. It is also not allow duplicates.
3. HashMap is used to storing key, value pair.
4. HashMap allow we can store single null key.
5. HashMap way to convert ~~him~~ into Set for iteration.
6. key value

4) Fail - fast and fail - safe :-

Iterator is based on fail-fast principle. It throws concurrentModificationException if a collection is modified during iteration over the collection.

An Enumeration is based on fail-safe principle. It's does not throws any exception if a collection is modified during traversal.

fail-fast iterator does not clone the original collection. fail-safe iterator creates a copy of the original collection of objects

*ConcurrentHashMap

ConcurrentHashMap Extends abstractMap in java. It was introduced in Java 1.5 onwards. It provides concurrency in a collection based on a HashMap.

2. All methods are in thread-safe methods in concurrentHashMap.
3. Internally there is a Hashtable backing in a - ConcurrentHashMap. This table supports concurrent methods. update data retrieval data. ...
4. It also support sequential and bulk operations.

Userdefined Exceptions:

If an Exception is created by programmer or user is called of user defined Exception. If user defined Exceptions are created when none of the predifined Exceptions are not matching our application.

Ex:

```
class SmallAge Extends RuntimeExceptions ← Exception
{
 SmallAge(){
 }
 SmallAge(String str){
 }
}
```

```
public class sample {
ps v m (String[] args){
int age = Integer.parseInt(args[0]);

try {
   if(age <20){
     throw new SmallAge("your age less than 20")
    }
   else{
     syout-("your age is matthan 20")
   }
}
catch(SmallAge s){
   s.printStackTrace();
}}}
```

VI) Spring Boot

1. Avoid the Boilerplate code.
Group the dependencies together in a single unit
like: spring-boot-starter-web.
2. Provided default servers
3. Spring boot is auto configure
4. Reduces the dev time and increase productivity

Spring

1. Specify each and every dependency seperately. This take more time.
2. Have to provide
3. is not provided autoconfigure.
4. Take significantly more time.

I-

Dependency injection:-

Dependency is Normally But if one object is dependent on another one object then it is called of dependency injection. and we can Establish the dependency b/w objects in two ways

1. setter injection
2. constructor injection.

How to work Spring Boot auto-configuration works?

Ans: Spring Boot auto-configuration automatically configure a spring application based on the dependencies present on the classpath.

To Enable auto configuration in Spring Boot we use @EnableAutoConfiguration

The @EnableAutoConfiguration annotation enable the autoconfiguration of spring.

SpringBoot provides various Autoconfiguration classes in spring-boot-autoconfigure-{version}.jar which is responsible for registering the various components.

① What is Assert in JUnit ?

A) ~~Assert is a api~~

Assert is a API by using to Validate any application logic we use Assert API method, which are also called as unitTesting methods.

All these methods are static methods. These throws the Error in fail, These methods tell is Test pass or Fail only.

After Executing logic it will return some o/p it can compare to Excepted results. If valid then Pass otherwise false

Ex methods :-

assertEquals()
assertNotEquals()
assertTrue()
assertFalse()
assertSame(), assertNotSame().

② How can you update you workspace/local with remote ?

A) By using pull and rebase operations.

③ RowMapper (I)

To fetch data from db using select query, JdbcTemplate has provide special method like.

a) queryForObject(single row)

b) query(Multiple row)

These methods will fetch data from db table in ResultSetFormat (having rows) after executing SQL query.

These Resultset data can be converted ~~from db~~ to objects format using RowMapper (I).

- RowMapper will not get data from db. If only converts ResultSet rows to model class objects.

- RowMapper will not get data from db. If only converts ResultSet rows to model class objects.

→ It is having mapRow()

(4) Diff b/w DELETE and TRUNCATE commands?

DELETE command is used to remove rows from the table and where clause can be used for conditional set of parameters. Commit and Rollback can be performed after delete statement.

TRUNCATE removes all rows from the table, Truncate operation cannot be rolled back.

(5) How to fetch common records (rows) from two tables

select * from T1 intersect select * from T2;

(or Column) select Name from tab1 union select name from table2;

~~public class~~

Bubblesort program:-

```
public class BubbleSortExample {
static void bubbleSort(int[] arr) {
int n = arr.length;
int temp = 0;
for(int i=0; i<=n; i++) {
for(int j=0; j<(n-1); j++) {
```

~~if(arr[j]~~

```
if( arr[j-1] > arr[j]){
  //swap elements
    temp = arr[j-1];
    arr[j-1] = arr[j];
    arr[j] = temp;
  }
 }
}
}
p s v m(String[] ar){
int arr[] = {3,6,7,12,2,18,4}
System.out("Arrays Before sort");
for(int i=0; i<arr.length; i++){
System.out(arr[i] + " ");
}
System.out();
bubblesort(arr); // Sorting array elements
System.out("Array after Bubblesort");
for(int i=0; i<arr.length; i++){
 System.out(arr[i]+" ");
}
} // main method
} //main class
```

(1) What are the design patterns are used in your project?

A) In my project I had used MVC, Singleton, Factory, Iterator and Template pattern.

MVC pattern:- I had used at presentation so model seperated by controllers.

Singleton pattern:- To share common data access throughout the application.

Factory pattern:- Between each layer for decoupling and centralized object creation to avoid code compilation.

Iterator pattern:- For providing tight encapsulation through middle tier object.

Observer pattern:- For sending notifications to subscribed used.

Template pattern:- For consistance vocabulary and naming convension for business objects.

(2) Why Spring Boot?

1. It is very easy to develop spring based applications with Java or Groovy.
2. It reduce lots of development time and increase productivity.
3. It avoids writing boiler plate code, and it supports the auto configuration.
4. It provides Embedded servers like tomcat, Jetty, Undertow.

1) What is volatile keyword.

2) co-variant

3) Garbage collection

4) what is concurrency

5) diff b/w Hashtable & HashMap & Concurrent HashMap.

6) what is Java ? How Java Platform independent.

---

1) what is singleton pattern.

2) what is cache.

---

1) what is scopes and types

2) what is Dependency injection.

Scopes :- Here scope indicates time period of data available in memory.

1. Singleton - Single bean object instance per Spring Ioc container.

2. Prototype - It produce new instance for each and every time bean is requested.

3. Request - It is a single instance created and available for complete life cycle of an HTTP request.
It only valid web-aware in spring ApplicationContext.

4. Session - Single instance will be created and available during complete lifecycle of HTTP session.

5. application - A single instance will be created and available during complete life cycle of servlet-context
It is only valid in web-aware Spring applicationcontext.

Singleton :

```
public class ConnectionSingleton {
    private ConnectionSingleton() {
    }
    private static SessionFactory sessionFactory;
    static {
        Configuration con = new Configuration();
        con.configure();
        sessionFactory = con.buildSessionFactory();
    }
    public SessionFactory ConnectionMethod() {
        return sessionFactory;
    }
}
```

Singleton design pattern :-

This design pattern solves the problem of code which has to be executed only one time in the life cycle of project

1. create a private constructor
2. create a static block
3. create private static variable where only one time object has to be created
4. write a static public method().

```
public class Singleton {
    private static SessionFactory sessionFactory;
    private singleton() {
    }
    static {
        Configuration configuration = new Configuration();
```

```
configuration.configure();
SessionFactory = configuration.buildSessionFactory();
}
public static SessionFactory create() {
return sessionFactory;
}
}
```

1) Volatile keyword in java.

A) Volatile keyword is used to modify the value of a variable by different threads. It is also used to make class thread safe. It means that multiple threads can use a method and instance of the class at same time without any problem. Volatile keyword can be used for only variables.

Ex: class Test {
static int var=5;
}

In above Ex two threads are working in same class. Both threads run on diff process and Each thread is copy of variable. If any thread is modifying the value, the will not reflect the original one in main memory. It leads data inconsistancy because another thread is not aware of ~~original~~ modified value.

2) Garbage collection in java?

A) Garbage collection is the process by which java programs perform automatic memory management. java programs compile to byte code that runs on JVM. The garbage collector finds these unused objects and deletes them to free up memory. from heap memory

There are many ways
- By nulling the reference
- By assigning a reference to another
- By anonymous object etc.

(3) What is Cookie

Ans: Cookie is small piece of code stored in client-side which servers use when communicating with clients.

JSP:

```
Cookie usernameCookie = new Cookie("username", request.getParameter
                                   ("username"));
usernameCookie.setMaxAge("60*60*10");
response.addCookie("usernameCookie");
```

Spring:

```
Public void readCookie( @CookieValue(value = "one",
                        defaultValue = "Unknown") String
                        favColor)
```

9511955157

(4) Covariant return type in java?

~~Covariant returntype is a method that~~

Covariant return type of a method is one that can be replaced by a "narrower" type when the method is overridden by in subclass.

Before Java 5, any method that overrides the method of parent class would have same return type.

From Java 5 onwards a child class is override a method of parent class and the child class method can return an object that its child class object return by parent class method.

```
Ex class A{
    A m1(){
        return new A();
    }
class B extends A{
```

return Arrays.equals(ch1, ch2);

}

}

ii) Why wait(), notify(), notifyAll() methods are available in object class not in thread class.

A) Every object in Java has only one lock or monitor and wait and notify() and notifyAll() are used for monitor scenario thats why there are part of an object class. These methods are controls locks and locks are associated on objects not the threads. Thats the reason is this methods are available in the objects rather than Thread class.

wait(): This method is used to suspend the execution of a thread until it received a notification

notify(): - This method is used to send a notification to one of the waiting threads.

notifyAll(): - This method is used to send a notification to all the waiting threads.

ii) Dependency Injection.

A) A dependency injection is a design pattern. It is used to implement Inversion of control (IoC) in spring framework. As per this pattern we do not create an objects in an application by using new operator. In this way create an object is not tightly coupled with another another object. A container is responsible for creating an object and wiring the object.. The container can call injecting the code and wire the objects at the configuration at runtime.

Springframework supports 3 types dependency injection.

1. Setter injection (SI):

Setter injection create an object using default constructor and provide data using set method then it is called as SDI.

Ex:

Pojo class:

```
public class Employee {
Private int empId;
Private int empName;
// def const, set, get
}
```

Configuration

```
@configuration
Public class AppConfig {
@Bean
Public Employee eob() {
Employee e = new Employee();
e.setEmpId(10);
e.setEmpName("AA");
return e;
}}
```

Test class:

```
public class Test {
psvm(String[] ar) {
ApplicationContext ac = new AnnotationConfigApplicationContext(AppConfig.class) {
Employee e = ac.getBean("eob", Employee.class)
syout(e);
}}
```

2. Constructor injection

Constructor dependency injection creating object and providing data by using parameterized constructor is known as CDI.

```
@configuration {
public class AppConfig {
@Bean
public Employee eob() {
Employee e = new Employee(10, "AA");
return e;
}
```

11) @PostConstruct, @PreDestroy annotations.

A) These annotations are used to implement lifecycle methods via annotations.

@PostConstruct: It means execute method after creation object with data. It is equals to init method.

@PostDestroy: It means execute method before destroying object from container. It is equals to destroy method.

Ex
```
@PostConstruct and PreDestroy
Public void sample(){
---
}
```

12) What are the types of auto-wiring in spring.

A) By using @Autowire annotation we can used to we make a dependency between the objects.

In spring framework 5 types of autowiring modes support.

- no - Default, no autowiring, set it manually via "ref" attribute.
- ByName - In @Autowire annotation we can make the dependency b/w the objects by using "byName" and "byType" means. we can Establish dependency b/w the object based on "property name".

and byType means we can Establish the dependency b/w the object based on "datatype" of the property.

- constructor - byType mode in constructor argument
- autodetect - If a default constructor is found, use "autowired by constructor" otherwise by type".

(13) @Qualifier : There may be a situation we are creating more than one bean of same type and want to only one of them with property. In such cases we use @Qualifier annotation along with @Autowire annotation.

Ex: public class Student{
--
--
}

public class Profile{
@Autowired
@Qualifier("student4")
private Student student;
}

@Scope :- This annotation is used to specify the scope of the Bean like singleton or prototype. If we create the container object by using BeanFactory Interface the container will create the Spring Bean object when we call getBean() even if the scope is singleton or prototype by default scope is singleton.

@Required :- The @Required annotation in Spring is a method level annotation applied to the setter method of a bean property and they making the setter-injection is mandatory. This annotation indicates that the required bean property must be injected with a value at the configuration time.

14) what is Aggregation and composition.

Aggregation :- It creates loose relation b/w parent and child incase of Hy-A relation. on performing any (non-select) operation in parent it is not applicable for child. on child again coe should write code to do operation [By default every HAS-A is aggregation] i.e no cascading.

composition :- It create strong relation b/w any parent and child class object. In case HAS-A relation. on performing an (non-select) operation is same applicable in child object also. Do not write code in child object (i.e cascading).

15) How to create the db connection using JDBC.

A) There are five steps.

1. Registering the Driver with DriverManager.
2. Establishing the Connection with the Database server.
3. Create the statement object using the established connection.
4. Exeute the queries by using statement object.
5. close connection.

```
Driver d = new Driver();
        DriverManager.registerDriver(d);
Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost
                                    :1521:xe", "scott", "tiger");
Statement s = con.createStatement();
s.executeUpdate("insert in to student values (50, 'ab', 50.3);
```

1) How to configure weblogic server in spring boot.

1. By default spring Boot is created jar. on the Eclipse open pom.xml and change packing method into war.

2. your project does not have servlet initializer class create it and override the configure method.

```
public class ServletInitializer extends SpringBootServletInitializer {
@override
protected SpringApplicationBuilder configure(SpringApplicationBuilder application){
return application.sources(SpringBootWeblogicApplic.class);
}
}
```

3. add dependency in pom.xml (tomcat)

In more than one application class in your project, please tell the spring Boot which class will be used to you application

```
<properties>
    <start-class> origo.siva . . .  </start-class>
</properties>
```

4. After that create 2 files such as weblogic.xml & dispatcherservlet-servlet.xml files in src/main/webapp/WEB-INF folder

5. use maven to create war file. (in target folder)

6. Deploying to weblogic.

Scanned with CamScanner

17) How to implement serverside validation in Spring MVC ?

A) Actually in our project implementing the validations in client(UI) side that it is take care about our front end development team. But I know about the validations how to implement serverside validations.

Spring MVC flw has provided validator(I) which has validate method and supports() to check the input data and returns errors back to controller and to UI

supports() — check for possible model class objects.

validate() — Reads model Attribute of input and returns errors object to controller and there to UI

```
@component
public class EmployeeValidator implements Validator {

@override
public boolean supports (Class<?> clazz){

return Employee.class.equals(clazz);
}
@override
public void validate (Object target, Errors errors){

Employee e=(Employe) target;

if( !pattern.matches ("[A-Z]{2,5}", e.getEmpName())) {

errors.rejectValue("empname", null, "Please Enter Valid Name");
}
```

```
@Controller
public class EmployeeController {

  @Autowire
  Private EmployeeValidator validator;

  Public ModelAndView readform(@ModelAttribute Employee employee,
                                                    Errors errors) {

    validator.validate(employee, errors);
    ModelAndView m=new ModelAndView();
    if( errors.hasErrors()){  //errors exist
      m.setViewName("Register");
    }
    else{
      m.setViewName("Data");
      m.addObject("emp", employee);
    }
    return m;
  }
}
```

(18) How to configure Dispatcher servlet in spring MVC?

Spring framework is providing one class AbstractAnnotationConfigDispatcherServletInitializer by extending this class to configure dispatcher servlet and we java base configuration.

AACDS extends AbstractDispatcherServletInitializer.

This class is having 3 methods to override those methods are

(AppConfig.class)

getRootConfigClasses() — for root application context configuration (ex: ↑)

getServletConfigClasses() — for Dispatcher servlet application context configuration.

getServletMappings() — for map the URL patterns

instanceof :- In java instanceof operator is used to test whether the object is an instance of the specified type (class or variable) The instanceof in java is also known as comparision operator b/c it compare the instance with type. It returns true or false.

* Difference b/w HashMap and ConcurrentHashMap. ?

HashMap	ConcurrentHashMap
1. HashMap is not synchronized in internally and it is not thread safe. we can make a HashMap synchronized externally using collections.synchronizedMap() method.	1. ConcurrentHashMap is internally synchronized and hence it is thread safe.
2. when you make HashMap synchronized externally collections.synchronized(), all the operations will synchronized.	2. all the operations in ConcurrentHM are not synchronized. only modifying operations like add and delete are synchronized. and read operations are not synchronized.
3. Introduced in JDK 1.2	3. Introduced in JDK 1.5
4. Fail-fast	4. fail safe
5. HashMap is not syncronized it is suitable for single threaded application	6. It suitable for multithreaded environment applications.

Builder designpattern :-

The builder designpattern is alternative way to construct complex objects. This should be used only when you want to build different immutable using same object building process.

Let us take one employee ~~immutable~~ object created it will not modify the state of object. simply create immutable object. In this class having 3 fileds like name, Dob, Phone no. In normal process create constructor pass 3 value

→ In some cases 2 fields mandatory one is optional, in this case we need write one more construction,

→ In some cases i want add one more field like email now a problem raise.

In such case without loosing immutability, implementing construction, Field we can consume additional field while retaining immutability of emp class.

→ create emp class (immutable)

```
public class Emp {
Private final String ename; // mandatory
Private final String dob; // mandatory
Prting final String mobile; // optional
Private final String email; // optional

// constru
// getters, toString
}
```

→ create empBuilder

```
public static class empBuilder {

Private final String ename;
Private final String dob;
Private String mobile;
private String email;
```

```
public EmpBuilder(String ename, String dob){
    this.ename = ename;
    this.dob = dob;
}
public EmpBuilder phone(String phone){
    this.phone = phone;
return this;
}
public EmpBuilder email(String email){
    this.email = email;
    return this;
}
// finally construct emp object
public Emp emp(){
    Emp e= new Emp(this);
    return e;
}
```

Test class :-

```
main
Emp e=new Emp.EmpBuilder("siva","20/1/21").phone("2341").ema("--");
    sysout(e);

Emp e1= new Emp.EmpBuilder("ravi","21/1/21").phone("3411");
    sysout(e1);
```

O/P :-

siva, 20/1/21, 2341, ---

ravi, 21/1/21, 3411, null.