**Spring Boot/Spring/rest**

**1.Difference between Spring and Spring boot / which one you prefer spring or spring boot?**

|  |  |
| --- | --- |
| **Spring** | **Spring Boot** |
| **Spring Framework is a widely used Java EE framework for building applications.** | **Spring Boot Framework is widely used to**  **develop REST APIs.** |
| **It aims to simplify Java EE development that makes developers more productive.** | **It aims to shorten the code length and provide the easiest way to develop Web Applications.** |
| **The primary feature of the Spring Framework is dependency injection.** | **The primary feature of Spring Boot is Autoconfiguration. It automatically configures the classes based on the requirement.** |
| **It helps to make things simpler by allowing us to develop loosely coupl ed applications.** | **It helps to create a stand-alone application with less configuration.** |
| **The developer writes a lot of code (boilerplate code) to do the minimal task.** | **It reduces boilerplate code.** |
| **To test the Spring project, we need to set up the sever explicitly.** | **Spring Boot offers embedded server such as Jetty and Tomcat, etc.** |
| **It does not provide support for an in-memory database.** | **It offers several plugins for working with an embedded and in-memory database such as H2.** |
| **Developers manually define dependencies for the Spring project in pom.xml.** | **Spring Boot comes with the concept of starter in pom.xml file that internally takes care of downloading the dependencies JARs based on Spring Boot Requirement.** |

**2.Difference between controller and rest controller?**

* **The @RestController annotation in Spring MVC is nothing but a co  
  mbination of the @Controller and the @ResponseBody annotation.**
* **The @Controller annotation indicates that a particular class serves the role of a controller.**
* **The @ResponseBody annotation tells a controller that the object returned is automatically serialized into JSON and passed back into the HttpResponse object.**

**@Controller**

**@ResponseBody**

**public class MVCController { }**

**@RestController**

**public class MVCController { }**

**3. what are the annotation used in spring boot**

1. **@SpringBootApplication**

* **It is combination of @EnableAutoConfiguration , @ComponentScan,@Configuration,**

1. **@enableAutoConfiguration**

* **Automatically configure an application based on the dependencies present in the classpath**

1. **@ComponentScan**

* **When we use @ComponentScan explicitly anywhere in the application, by default spring boot application scans all the beans (using @Component annotation) under the base package**
* **Without this annotation spring boot could not recognize the beans and we can’t add them into application context and inject the dependency**

1. **@Configuration**

* **On startup spring creates the application context. this is the object that stores all these dependencies so they can be wired into the application where they are needed**
* **Spring refers these dependencies as the beans**
* **Annotating a class with the @Configuration indicates that the class can be used by the Spring IoC container as a source of bean definitions.**

1. **@Bean**

**The @Bean annotation tells Spring that a method annotated with @Bean will return an object that should be registered as a bean in the Spring application context.**

1. **@RestController**

* **this annotation responsible for handling all the incoming request and map the url request to the service methods**
* **it is the combination of @Controller and @ResponseBody(both required in every controller class)**

1. **@Component - this is used to add/register the object/bean to the application context**
2. **@Autowired**

* **enable you to inject object dependency implicitly**
* **it is internally using setter or constructor injection**
* **it can’t use to inject string and primitive values .it works with reference only**

1. **@Value**

* **It is used to Inject property values(appliction.property) into the component**
* **Basically it is inject all the values from environment variables or spring cloud config**

10. **@Service**

**4. Purpose of controller class.**

* **Controller class used to handle the client request.**
* **It will receive the client request and invoke business class to perform business related activities.**
* **And return the response to client,**
* **It is achieved using @RestController/ @Controller with @ResponseBody**

**5. types of Request mapping?**

* **@RequestMapping *annotation is used to map web requests to Controller methods.***

**Types:**

1. **@GetMapping – used to get**
2. **@PostMapping – used to post**
3. **@PutMapping – used to update**
4. **@DeleteMapping – used to delete**
5. **@PatchMapping – used to update**

**6. difference between request mapping and get mapping**

|  |  |
| --- | --- |
| **Request mapping** | **Get Mapping** |
| **It is used in class and method level** | **It is used with method level** |
| **It is used to handle all the Http request** | **It is used to handle only Http Get methods** |
| **@RequestMapping(method = RequestMethod.GET)** | **@GetMapping** |

**7. what is dependency injection in spring and types of dependency injection.**

* **Class may contain different dependencies, to create object of the class we need initialize dependencies also**
* **Instead of programmer spring is creating, managing the objects**
* **Spring injecting objects through constructor or setter method**
* **Object type dependencies injected using @Autowire**
* **Literals are injected through @value**

**8.what is Autowiring / what is @Autowired**

* **Autowiring means, we can wire/inject the object anywhere in the application**
* **It is done using @Autowired**
* **It can be done for object type only, not for string, primitive types**
* **It can be done in constructor/setter method level**

**9. Constructor vs setter Autowiring**

**10. what is @EnableAutoConfigure**

* **Spring Boot auto-configuration attempts to automatically configure your Spring application based on the jar dependencies that you have added.**
* **For example, If HSQLDB is on your classpath, and you have not manually configured any database connection beans, then we will auto-configure an in-memory database.**
* **You need to opt-in to auto-configuration by adding the @EnableAutoConfiguration or @SpringBootApplication annotations to one of your @Configuration classes.**
* **To exclude auto configuration for data source**
* ***@EneableAutoConfiguration(exclude={DataSourceAutoConfiguration.class})***

**10.use of @Qualifier**

* **If more than one bean of the same type is available in the container, the framework will throw NoUniqueBeanDefinitionException**, indicating that more than one bean is available for autowiring.
* **The @Qualifier annotation is used to resolve this autowiring conflict, when there are multiple beans of same type.**
* **The @Qualifier annotation can be used on any class annotated with @Component or on method annotated with @Bean. This annotation can also be applied on constructor arguments or method parameters.**

**Eg:**

**https://www.stackchief.com/blog/%20%40Qualifier%20Example%20%7C%20Spring%20Boot**

**https://www.baeldung.com/spring-qualifier-annotation**

**11.run spring boot application in external server**

[**https://dzone.com/articles/spring-boot-with-external-tomcat**](https://dzone.com/articles/spring-boot-with-external-tomcat)

**12.What is @Profile, how to create profile? What is default profile? (video required)**

* **Spring boot profiles provides the way to segregate application configuration**
* **It makes useful in configuring different environment like dev,prod,qa**

**Create Profile:**

* + **We can create profile under resources, eg : for dev env profile is application-dev.properties**
  + **Also we can create profile using java like below**
  + **Any @Component or @Configuration can be marked with @Profile to limit when it is loaded:**

***@Configuration***

***@Profile("production")***

**public class ProductionConfiguration {**

***// ...***

**}**

* **In application.properties file add spring.profile.active=dev**
* **Or else we can set the profile in command prompt**

**13.springboot starters?**

* **Spring boot has built-in starters which makes our development easier**
* **These starters adding relevant jars in our class path**
* **For example if we want to use Spring JPA for database access, we need to include the spring-boot-starter-data-jpa dependency in our build tool file of the project.**

**Starters used in our project:**

* + **org.springframework.boot:spring-boot-starter-web – includes restful, tomcat**
  + **spring-boot-starter-data-jpa – for jpa data access**
  + **org.springframework.boot:spring-boot-starter-actuator – for health chek**
  + **junit**
  + **echace**
  + **swagger**
  + **hibernate**

**14. what is actuators?**

1. **is the mechanism where we can monitor the health of our applications in production .**
2. **like how many sessions, connections, state of database resources and if we want to analyze the log**
3. **monitor and manage your application when you push it to production**

**Implementation: endpoint**

**15. what is spring boot scopes – need to study full bean scopes (video)**

* **Singleton - Singleton is the default scope for a Bean, the one that will be used if nothing else is indicated. This scope implies that Spring container will create an only shared instance of the class designated by this bean, so each time the Bean is required the same object will be injected.**
* **Prototype**
* **Request**
* **Session**
* **Global session**
* **Application**
* **Websocket**

**16.what is default scope in spring boot and behavior (eager, lazy)? how to change it using annotation?**

* **By default it is singleton**
* **And lazy loading when we are using @Autowire that time only bean will be created**

**17.how to change default server tomcat to jetty in spring boot?**

1. Exclude Tomcat from web starter dependency, since it is added by default
2. Add the Jetty dependency

Ref: https://codeahoy.com/java/springboot/tutorial/2019/09/01/spring-boot-replace-tomcat-with-jetty-as-the-embedded-server/

**18.what is lambok?**

* Lombok is an annotation-based Java library that allows you to reduce boilerplate/ repetitive code.
* Using lambok we can avoid to writing of getter, setter, toString, hashcode and equals, no arg constructor, all arg constructor

Eg:

* + - 1. @Getter
      2. @Setter
      3. @NoArgsConstructor
      4. @AllArgsConstructor
      5. @ToString(includeFeildNames=true)
      6. **@EqualsAndHashCode()**
      7. **@Data – shortcut combines all the above required for java POJO class**
      8. **@NonNull**

**Ref : https://auth0.com/blog/a-complete-guide-to-lombok/**

**19.how application.properties will be read by spring boot application?**

**Spring boot used @** Value annotation to read from application.properties / cloud config properties

Eg: @Value("${spring.application.name}") or @Value("${spring.application.name:demoservice}") – demo service is default value

**20.what are the framework used in spring boot for database operation, where you will mention the connection details In spring boot application.**

* **For data base operation we can use Jpa and hibernate for the implementation**
* **In application.properties or cloud config data base properties are defined**

**21.how the connection happens to the data base in spring boot applications. what are the annotations required to create connections**

* **Add database connector dependency in build tool**
* <dependency>
* <groupId>com.oracle</groupId>
* <artifactId>ojdbc7</artifactId>
* <version>12.1.0.1</version>
* </dependency>
* **Setup database schema creation and table creation**
* **Provide data source details in application.properties or in cloud config**
* spring.jpa.hibernate.ddl-auto=none
* spring.datasource.url=jdbc:mysql://localhost:3306/todo\_example
* spring.datasource.username=todouser
* spring.datasource.password=YOUR\_PASSWORD
* **also we can configure the data source programmatically and create bean for getDataSource Builder**
* **restart the spring boot application**

**Annotation**

* **@SpringBootApplication/EnableAutoConfiguration,@ComponentScan,@Configuration,@Bean**

**22.how to configure multiple db using spring boot. How spring boot knows which one needs to configure first**

* **Add required spring data jpa and database dependency in build tool**
* **In cloud config/application.properties declare multiple data source**
* **Create separate package for DB1 entity , DB1 Repository and DB2 entity and DB2 Repository**
* **Create Data Source config for DB1 by annotating @Configuration, @EnableTransactionManagement,@EnableJPARepositories(basePacksges={“Repo package”})**
* **Create DataSource bean in config1 and mark it as @primary, create entity managerFactory bean with datasource where we need to mention entity package and annotate @primary and create transactionManager bean with entityManagerFactory and annotate @Primary**
* **Like same create data source config for DB2 but don’t annotate @Primary**

**Ref:** [**https://youtu.be/mIFIb\_JE47U?si=HUvGgqGH4aHnsgwW**](https://youtu.be/mIFIb_JE47U?si=HUvGgqGH4aHnsgwW)

**23.how the dependency jars/libraries added into spring boot project**

* We have to add the dependencies in the pom.xml/build.gradle file.
* These added dependencies will then get downloaded from Maven Central.
* The downloaded dependencies will get stored into the ‘.m2’ folder in the local file system.
* The Spring-Boot application can access these dependencies from  ‘.m2’ and its sub-directories.
* Example -(**.m2** -> **repository** -> **org**, etc )

**Ref:** [**https://www.geeksforgeeks.org/spring-boot-dependency-management/**](https://www.geeksforgeeks.org/spring-boot-dependency-management/)

**24.how to configure spring beans**

**1. By using @Bean annotation configuration class**

**2. By Using @ Component / @ Service**

**25.what is spring MVC**

Spring MVC is an integrated version of the Spring framework and Model View Controller. It has all the basic features of the core Spring framework like Dependency Injection and Inversion of Control.

**26.what is spring security**

* **Spring boot provides the default form based authentication to authenticate the users**
* **To enable this feature we need to add Spring-boot-starter-security starter in out build tool , it will add the dependencies required for spring security.**
* **Whereas user name is user and password is randomly generated password. Password found in console.**
* **Most of the time we won’t rely on default user/password.**
* **We can provide own user name and password through application.properties file by adding security.basic.enabled=true, security.user.name=admin, security.user.password=root**

**27.cache mechanism/ehcache/what is cache tool used**

[**https://www.baeldung.com/spring-boot-ehcache**](https://www.baeldung.com/spring-boot-ehcache)

**28. request headers**

* **A request header is an**[**HTTP header**](https://developer.mozilla.org/en-US/docs/Glossary/header)**that can be used in an HTTP request to provide information about the request**

**Fields of request header-**

**1.Accept -** [**Media type(s)**](https://en.wikipedia.org/wiki/Media_type)**that is/are acceptable for the response. Eg: application/json**

**2. Accept-Language – acceptable languages:en-us**

**3. Access-Control-Request- Method,Access-Control-Request-Headers- Access-Control-Request-Method: GET**

**4.host - The domain name of the server (for**[**virtual hosting**](https://en.wikipedia.org/wiki/Virtual_hosting)**), and the**[**TCP port**](https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers)**number on which the server is listening. The**[**port**](https://en.wikipedia.org/wiki/Port_(computer_networking))**number may be omitted if the port is the standard port for the service requested.**

**Eg: 10.231.190.88:30323**

**5.origin - Initiates a request for**[**cross-origin resource sharing**](https://en.wikipedia.org/wiki/Cross-origin_resource_sharing)**(asks server for**[**Access-Control-\***](https://en.wikipedia.org/wiki/List_of_HTTP_header_fields#access-control-response-headers)**response fields).**

**Origin:** [**http://www.example-social-network.com**](http://www.example-social-network.com)

**6. Content-Type - The**[**MIME type**](https://en.wikipedia.org/wiki/MIME_type)**of this content - : application/json**

**29.spring boot components**

* Spring Boot Starters
* Spring Boot AutoConfigurator
* Spring Boot CLI
* Spring Boot Actuator
* Spring Initializer
* Spring Boot IDEs

**30. what is spring cloud?**

* **Spring Cloud Config is an idea to storing and serving distributed configurations across multiple applications and environments using GIT.**
* **Default port :8888**

**Steps to create Cloud config server**

1. **Create spring boot application with spring cloud config server dependency org.springframework.cloud:spring-cloud-config-server'**
2. **Add @EnableConfigServer Annotation in main class**
3. **Create the configuration repository in git and add required application.properties/yml files . File name should be <application - name>-<profile>.yml/properties.**
4. **Add spring.cloud.config.server.git.uri =”above repo url” and Server.port = 8888 , server.application.name=”app name”in application.properties file**
5. **Run the cloud config server by invoking http://localhost:8888/<application –namr>/env**

**Eg:** [**http://localhost:8888/SpringCloudDemo/local**](http://localhost:8888/SpringCloudDemo/local)

**Sample project**

**Tutorial-https://github.com/Bhuvaneswari-Vajravel/SpringCloudConfigServer**

**Steps to create Spring cloud config client**

**1. In client application add spring-cloud-starter-config starter**

org.springframework.cloud:spring-cloud-starter-config in build tool.

2.**Add** spring.config.import=configserver:http://localhost:8888 in application.properties.

Sample proj : https://github.com/Bhuvaneswari-Vajravel/SpringCloudConfigClient

**31. How to identify the service which is causing an issue in spring boot?**

**32. CORS in spring boot**

* **“CORS” stands for Cross-Origin Resource Sharing.**
* **It allows you to make requests from one website to another website in the browser, which is normally prohibited by another browser policy called the Same-Origin Policy (SOP).**
* **SOP means the resource can be accessed by the same origin. To overcome this we are using CORS.**
* **CORS providing control over the resource how it is accessed outside domain**

**33. what is interceptors?**

**You can use the Interceptor in Spring to perform operations like writing the log, do some configuration under the following situations −**

* **Before sending the request to the controller – PreHandle()**
* **Before sending the response to the client – postHandle()**

**afterCompletion() – called after completing the request and response**

**For example, you can use an interceptor to add the request header before sending the request to the controller and add the response header before sending the response to the client.**

**34. what is filter in spring boot?**

**Filters as the name suggest used to perform filtering on either the request to a resource or on the response from a resource, or both. *Spring Boot* provides few options to *register custom filters* in the *Spring Boot application*.With the help of filter, we can perform the following operations.**

1. **Perform some request processing before the request is handed over to the controller.**
2. **Processing response before it reaches the client.**

**35.what is swagger**

* **swagger is used to document the Rest API webservices**
* **if we done any changes in it will automatically change in documentation**
* **we don’t need to manage manually.**

**Steps to create swagger**

* + - 1. **add swagger dependency in build tool**
      2. **create configuration file and annotate with @EnableSwagger2**
      3. **write a method where we have to mention the controller base package and it will generate the documentation**

**36. What is loosely coupling**

**Loosely coupling in java**

* **In simple words, loose coupling means they are mostly independent. If the only knowledge that class A has about class B, is what class B has exposed through its interface, then class A and class B are said to be loosely coupled**
* **Example: Imagine you have created two classes, A and B, in your program. If you change class A volume, then you are not forced to change class B. This is called loose coupling in Java.**
* **When class A requires changes in class B, then you have tight coupling.**

### **Tightly Coupling**

* **When two classes are highly dependent on each other, it is called tightly coupling.**
* **It occurs when a class takes too many responsibilities or where a change in one class requires changes in the other class.**
* **Example: Imagine you have created two classes A and B, in your program. Class A is called volume, and class B evaluates the volume of a cylinder. If you make any changes in the volume, then the same changes will reflect in class B. Hence, we can say both the classes are highly dependent on each other and are tightly coupled.**

**37. different between Query param and path param**

* **Path Param is basically used to identify a specific resource or resources whereas Query Parameter is used to sort/filter those resources.**
* **Hence Path Param is mandatory, query param is optional to pass value**
* **Let's consider an example where you want identify the employee on the basis of employeeID, and in that case, you will be using the URI param.**
* **GET /employee/{employeeID}**
* **Take another example where you want to filter the employee on the basis of designation, and in that case, you will be using Query Parameter.**
* **GET /employee?designation=SSE**

**Ex:**

**@RestController**

**@RequestMapping(path = "/spring-mvc-basics ")**

**public class PrecautionsController**

**{**

**@GetMapping("/foos/{id}")**

**public String getFooById(@PathVariable(“id”) String id) { return "ID: " + id; }**

**/\*\***

**Then we can map based on the path:**

**http://localhost:8080/spring-mvc-basics/foos/abc**

**----**

**ID: abc \*\*/**

**@GetMapping("/foos")**

**@ResponseBody**

**Plic String getFooByIdUsingQueryParam(@RequestParam String id) {**

**return "ID: " + id;**

**}**

**/\*\***

**which would give us the same response, just a different URI:**

**http://localhost:8080/spring-mvc-basics/foos?id=abc**

**----**

**ID: abc**

**\*\*/**

**38.Can we replace @Component with @Controller , @Service, @Repository**

**39. What are the spring modules used in project**

**what Is spring AOP?**

* **Spring AOP enables Aspect-Oriented Programming in spring applications.**
* **In AOP, aspects enable the modularization of concerns such as transaction management, logging or security that cut across multiple types and objects (often termed crosscutting concerns).**
* **It entails to break down application logic into different parts.**
* **AOP provides the way to dynamically add the cross-cutting concern before, after or around the actual logic using simple pluggable configurations. It makes easy to maintain code in the present and future as well.**
* **You can add/remove concerns without recompiling complete source code simply by changing configuration files (if you are applying aspects suing XML configuration).**

[**https://howtodoinjava.com/spring-aop-tutorial/**](https://howtodoinjava.com/spring-aop-tutorial/)

[**https://medium.com/@bushra.saifi/spring-aop-18ebdcf669a1**](https://medium.com/@bushra.saifi/spring-aop-18ebdcf669a1)

**how you will manage the transaction management in springboot with multiple database**

**bean factory and application context**

**how to get the list of employees, who has age is more than 25? (default method)**

**how to get the list of employees, by passing first name?(default method/customize method)**

**what is build.gradlifference between gradle and maven?**

**Microservices**

**Note : https://www.guru99.com/microservices-interview-questions.html**

**1.what is microservice architecture.**

* **Microservices is an architectural style that structures an application as a collection of services that are highly maintainable and testable, loosely couple, independently deployable and modeled around a business domain.**
* **So large applications using this architectural pattern can be broken into small multiple micro services.**
* **Which together act as a one large application. But behind the scene it’s a microservices. These microservices communicating each other through Api’s**

**2.what is advantage and disadvantage of Microservice**

**Advantages:**

* **Each service can only focus on one single business capability.**
* **It is possible to change or upgrade each service individually rather than upgrading in the entire application.**
* **Less dependency, and easy to test. Faster release cycle.**

**Dis Advantages:**

* **Microservices has all the complexities of the distributed system.**
* **There is a higher chance of failure during communication between different services.**
* **Difficult to manage a large number of services.**
* **Complex testing because of a distributed environment.**
* **When more services interact with each other, the possibility of failure also increases.**

**3.how two microservice are communicating with each other**

**4. Fault tolerance**

* **Fault tolerance is the individual service that does not bring down the overall system.**
* **Without fault tolerance, a single failure in the system may cause a total breakdown.**
* **Consider a scenario in which six microservices are communicating with each other. The microservice-5 becomes down at some point, and all the other microservices are directly or indirectly depend on it, so all other services also go down.**
* **The solution to this problem is to use a fallback in case of failure of a microservice. This is called fault tolerance.**
* **Fault tolerance can be achieved with the help of a circuit breaker.**
* **The circuit breaker is a pattern that wraps the request to external service and detects when they are faulty.**
* **If a failure is detected, the circuit breaker opens. All the subsequent requests immediately return an error instead of making requests to the unhealthy service.**

**5. what is Circuit Breaker**

* **Circuit breaker is used to identify which of the services is not running .**
* **If a failure is detected, the circuit breaker opens. All the subsequent requests immediately return an error instead of making requests to the unhealthy service.**
* **It rejects calls until it becomes healthy again**
* **Till now it will run the fall back method available**

**6. what is Euraka Server or Discovery Server**

* **Eureka Server is an application that holds the information about all client-service applications.**
* **Every Micro service will register into the Eureka server and Eureka server knows all the client applications running on each port and IP address.**
* **Eureka Server is also known as Discovery Server**
* **Whenever a service wants to talk with another service it should first talk with eureka naming server.**
* **The naming server provides the instances of particular service that are currently running.**
* **The process of providing instances to other services is called Service Discovery.**

**Steps:**

**1. Create Eureka Server**

* **Create spring boot application**
* **Add eureka server starter in build tool dependency**
* **Add EnableEurekaServer annotation in main class**

**2. Create Eureka Client**

* **Eureka discovery client started in build tool dependency**
* **Add EnableEurekaClient annotation in main class**
* **Add eureka service properties, url in application.properties file**
* **When call another microservice we don’t need to mention the ip address , we can simply specify the application name which is registered in Eureka**

**7. what is API Gateway**

* **The API Gateway is a server. It is a single entry point into a system. API Gateway encapsulates the internal system architecture**
* **All the requests made by the client go through the API Gateway. After that, the API Gateway routes requests to the appropriate microservice.**
* **It also has an inbuilt load balancer to load the balance of all incoming request from the client.**
* **Default port: 8765**

**There is a total of two different methods by which the API gateway handles the request made by the client:**

* **It routes the request of the client to a suitable service.**
* **It spreads out the request of clients to multiple services.**

**Advantage:**

* **The most important advantage of API Gateway is that it encapsulates the internal structure of the application.**
* **Rather than invoking the specific service, the client directly talks to the API Gateway.**

**Dis Advantage:**

* **It requires routing rules.**
* **There is a possibility of a single point of failure.**
* **Risk of complexity due to all the API rules are in one place.**

**8. hystrix**

* **Hystrix is a library. Using this we can implement the dashboard there we can identify which are the services running and which are not running.**

**Steps :**

* **Add EnableHystrix annotation in API Gateway main class**
* **Add hystrix dependency in build tool**

**9. what is distributed Log Tracing?**

**Distributed Tracing:**

* **Distributed tracing is following /track single request through across multiple service where as distributed logging is collect the logs from multiple microserivces**
* **This is use to pinpoint bugs, or other issues that impact the application’s performance.**
* **It is used to understand the performance of specific service**
* **We can use Zipkin and spring cloud sleuth or Splunk**

**CONFIGURATION**

**Zipkin Configuration**

1. **download the  and execute it in cmd prompt using java – jar zipkin-server-2.12.9-exec.jar**
2. **Default port for zipkin server is 9411**
3. **Add** spring-cloud-starter-zipkin and **spring cloud sleuth starter in build tool.**
4. **Add** spring.zipkin.baseUrl=http://localhost:9411/ in application.properties
5. **Once we start the server we can see client application zipkin dash board and we can see the trace id span id for each request in console logs which is generated by sleuth**
6. **In zipkin dashboard we can see the request flow for each request and time taken to execute the service and url method everything we can see**

**Splunk**

**10.what is Load balancer**

* **Load balancing is nothing but efficient distribution of network or application traffic across multiple servers.**
* **Each load balancer sits between client devices and backend servers, receiving and then distributing incoming requests to any available server capable of fulfilling them.**
* **We can use netflix ribbon as a load balancer. It provides the client-side balancing algorithm. It uses a Round Robin Load Balancing**

**There are two types of load balancing available:**

* **Server Side Load Balancing: Server side load balancing is a monolithic It applies between the client and the server. It accepts incoming network, application traffic, and distributes the traffic across the multiple backend servers by using various methods. The middle component is responsible for distributing the client requests to the server.**
* **Client-Side Load Balancing: The client holds the list of server’s IPs so that it can deliver the requests. The client selects an IP from the list, randomly, and forwards the request to the server.**

**10. What is Centralized logging mechanism?**

* The Centralised logging approach enables us to consolidate all the logs from distributed application services and monitor them from one interface. It provides easy access to logs for analysis in one single dashboard and we do not have to search on each server to troubleshoot.;
* Eg: kibana , Splunk ,

**10. web client**

**11. rest template**

**12. Feign client**

**13. difference b/w web client and feign client**

**14. Orchestration in microservice /Saga design paatern**

**.**

**2.stream all methods and programs, reverse programs, count the number of car in string, all other interv programs , lambda expression example program**

**3. exectur framework, Async process,Multi treadin, fpa thread scheduler,Future,CompletableFuture, ,**

**4. Desin pattern - singleton, factory , abstract factory pattern, template pattern , facade pattern**

**6.sprin cloud, AOP,**

**7. how to handle the failure of communication (circuit breaker etc)**

**8. all intervw sql query , sample procedure and procedure**

**Interview question**

**AOP**

**Spring Batch processingo**

**Spring security – Oauth , JWT**

**Spring Scheduling**

**Messaging System**

**How you will handle validations in Spring boot**

**How to handle the exception in spring boot**

**How to stream the List<List<String>list; -flatmap syntax**

**constructor injection vs setter injection**

**@autowired vs @injects**

**Write a program for below scenario**

**input=[1,2,3,4,5,6]**

**1) groupSize=4 and iterations=5**

**output:**

**1234**

**5612**

**3456**

**1234**

**5612**

**2) groupSize=5 and iterations = 2**

**output 12345**

**61234**

**Write the program to compare two employees if emp first name equal then comp employee last name , if it also equal then find employee city**

**git commands**

**rebase**

**switch**

**cascadeTyle-ALL**

**@requeriednew**

**Diff b/w @Component and @service**

**Can we replace @service /@repository / @controller with @component, if we replace what will happen**

**Immutable Design Pattern**

**What are the spring modules used in the proj**