**Spring All Interview Questions…**

Spring: -

1. What is Spring? Why to use it?

• The Spring Framework (Spring) is an open-source application framework that provides infrastructure support for developing Java applications. One of the most popular Java Enterprise Edition (Java EE) frameworks, Spring helps developers create high performing applications using plain old Java objects (POJOs). • Without a Spring Framework, application code tends to be tightly coupled (interdependent), which is not considered good coding practice. Loose coupling is ideal because loosely coupled components are independent, meaning changes in one will not affect the operation of others.

1. What are the features of Spring framework?

• It is light weight (JARs are smaller in size) & open source. • It is complete end (applicable for all layers in application) & modular framework (applicable for particular layer in application). • We can achieve loose coupling. (Provides features of IoC and DI) • Non-invasive framework (doesn’t force to extend or implement any base class or interface). • We can develop easy to test kind of applications. • Spring is a lightweight framework. It can be thought of as a framework of frameworks because it provides support to various frameworks such as Struts, Hibernate, Tapestry, EJB, JSF, etc.

1. Does Spring contain in-built server?

• Spring don’t have in-built server. We have to add it explicitly.

1. What is tight coupling and loose coupling?

• Tight Coupling ---> Change in one class forces to change in other classes • Loose Coupling ---> Change in one class doesn’t forces to change in other classes

5. Enlist modules in Spring framework? • Core Container ---> This module provides IoC and DI feature. It also contains BeanFactory, ApplicationContext. • Data Access/Integration ---> Provides support to interact with DB and to integrate with ORM (Object Relationship Mapping) like Hibernate. • Web ---> Provides basic support for web development. It has WebApplicationContext built on ApplicationContext of a core container which provides complete MVC (Model View Controller) implementation to develop presentation layer.

6. What is IoC? • The IoC container creates an object of the specified class and also injects all the dependency objects through a constructor, a property or a method at run time and disposes it at the appropriate time. • It means object creation is not a task of programmer it will be handled by IoC container. Called as Inversion of Control because represents the inversion responsibility of object creation, initialization and destruction (object life cycle) from application to Spring Container. • Normal Object creation----> using new keyword In Spring, Core Container provides the required dependency object. • It converts object from tight coupling to loose coupling which is achieved by dependency injection.

7. What is purpose of Spring container? • It takes care of object creation, initialization and managing object dependencies.

8. What is Dependency? Types of Dependencies. • Data member or properties are dependencies. • Types--> Primitive, Object/secondary, Collection

9. What is Dependency Injection? Types of Dependency Injection. • Assigning values to those data members(dependencies) is Dependency Injection. • Can be achieved in two ways--->1. Setter Based 2.Constructor Based

10.Difference between setter based and constructor-based DI? Setter based Constructor-based 1.Allows partial injection 1. Doesn’t allow partial injection 2.Can override constructor-based DI 2.Cannot override setter-based DI 3.Mutable 3.Immutable

11.What is container? Different types of container. • Container is the one which is responsible for object life cycle in Spring. • Types----> 1. Core container------>BeanFactory (interface) 2.JEE container ------>ApplicationContext (Child interface of BeanFactory)

12.Difference between BeanFactory and ApplicationContext container? BeanFactory ApplicationContext 1.Implemented by XMLBeanFactory 1.Implemented by WebApplicationContext 2.Doesn’t support annotation concept 2.Supports annotation concept 3.Doesn’t support internationalization 3.Supports internationalization 4.Lazy loading of beans 4.Eager/aggressive loading of beans 5.Supports desktop-based application 5.Supports both desktop-based and webbased applications Note ---> Each bean has a unique id.

13.What is Lazy and Eager loading? • Lazy Loading --->instantiates bean only when called. • Eager Loading --->instantiates bean at the time of container start.

14.What is Scope in Spring? What are its types? • When the Spring Framework creates a bean, it associates a scope with the bean. A scope defines the runtime context within which the bean instance is available. • By default, Spring bean scope is Singleton. • Types---> 1.Singleton ---> - Only one instance will be created for a single bean definition per Spring IoC container and the same object will be shared for each request made for that bean. - BeanFactory: Lazy ApplicationContext: Eager 2.Prototype ---> - A new instance will be created for a single bean definition every time a request is made for that bean. - BeanFactory: Lazy ApplicationContext: Lazy 3.Request (Valid for Spring based applications, used for http request) 4.Session (Valid for Spring based applications, used for http session) 5.Application 6.Websocket

15.How to make ApplicationContext as Lazy loading? • By declaring ApplicationContext bean scope as Prototype.

16.What is an AutoWiring? How to enable AutoWiring? • Autowiring feature of spring framework enables you to inject the secondary dependencies implicitly (automatically). It internally uses setter or constructor injection. • Autowiring can't be used to inject primitive and string values. It works with reference only. • Can be achieved by --->

17.By default, AutoWire supports which autowire type? • By default, it uses byType.

18.How to enable autowire in .xml file? • By default, it is disabled. • Add in bean configuration xml file

19.What is front-controller or Dispatcher servlet? Its uses? • It manages entire process. Operation starts from Dispatcher-servlet so, no need of main method. • It finds appropriate class as per request 20.What is Java based? • In this we don’t use .xml, instead of that we use java class. byName - Dependency is injected according to name of the bean. - In this scenario Has-A variable name in POJO and bean name must be same. byName byType - This option enables the autowire based on bean type. - In this scenario Has-A variable name in POJO and bean name can be different but there must be only one bean of a type. byType constructor - Injects dependency by calling constructor. Internally uses byType. Bean Type Bean Name

21.What is scope combination?

22.what is lookup method? When to use?

23.What is Stereotype Annotation? Explain them? • @Controller: Identifies class as a controller class & marks it as a bean. • @Service: Identifies class as a service class & tells it has business logic. • @Repository: Identifies class as a DAO layer class & tells it has a database connection.

24.Can we interchange any of above stereotype annotations? • Yes, we can shuffle @Repository & @Service. • It is just to understand other programmers to identify business logic class and DAO layer class. • But we can’t replace @Controller class as it marks as bean.

25.What are Spring validations? • The Spring MVC Validation is used to restrict the input provided by the user. It can validate both server-side as well as client-side applications.

26.What is Spring MVC? • Helps to develop web-based applications. Helps to separate different aspects of programming. e.g., input logic, business logic, UI logic Model= binds the called data of an application View= generates the html output Controller=It processes the user request, and built model passes it to view.

27.Execution flow of MVC? DispatcherServlet is a class that receives the incoming request and maps it to the right resource such as controllers, models, and views. o Model - A model contains the data of the application. A data can be a single object or a collection of objects. o Controller - A controller contains the business logic of an application. Here, the @Controller annotation is used to mark the class as the controller. o Front Controller - In Spring Web MVC, the DispatcherServlet class works as the front controller. It is responsible to manage the flow of the Spring MVC application. o View - A view represents the provided information in a particular format. Generally, JSP+JSTL is used to create a view page. Although spring also supports other view technologies such as Apache Velocity, Thymeleaf and FreeMarker. Spring provides view resolvers, which enable you to render (to show) models in a browser without tying you to a specific view technology. As displayed in the figure, all the incoming request is intercepted by the DispatcherServlet that works as the front controller. o The DispatcherServlet gets an entry of handler mapping from the XML file and forwards the request to the controller. o The controller returns an object of ModelAndView. o The DispatcherServlet checks the entry of view resolver in the XML file and invokes the specified view component.

28.Difference between URL and URI? • A URI is a sequence of characters that identifies a web resource by location, name, or both available on the internet. Whereas, URL is a sequence of characters that only identifies the location of a resource available on the internet.

URI URL 1. URI is an acronym for Uniform Resource Identifier. URL is an acronym for Uniform Resource Locator. 2. URI contains two subsets, URN, which tell the name, and URL, which tells the location. URL is the subset of URI, which tells the only location of the resource. 3. All URIs cannot be URLs, as they can tell either name or location. All URLs are URIs, as every URL can only contain the location. 4. A URI aims to identify a resource and differentiate it from other resources by using the name of the resource or location of the resource. A URL aims to find the location or address of a resource on the web. Endpoint vs URL The term endpoint is focused on the URL that is used to make a request. The term resource is focused on the data set that is returned by a request. Now, the same resource can often be accessed by multiple different endpoints. Also, the same endpoint can return different resources, depending on a query string. Each content present in the database is considered as a resource(objects). A resource can be text file, an image or any dynamic data. URI is used to identify each resource of REST (part of a Webservices) architecture. URI-Format Example ://// <https://en.wikipedia.org/wiki/ReST-api>

**Spring Boot Questions: -**

29.What is SpringBoot? Why do we mean by Stand-alone application? • Spring Boot is the extension of the Spring framework and provides a faster way to build applications. Spring Boot is an open-source Java web framework used for Rapid Application Development (to build stand-alone application with minimum configurations). The Spring Boot framework creates a fully production-ready environment that is completely configurable using its prebuilt code within its codebase. • A standalone application is an application that runs locally on the device and doesn't require anything else to be functional. All the logic is built into the app, so it doesn't need an internet connection nor any other services installed.

30.Features of SpringBoot. 1. Provides embedded tomcat, jetty server. 2. Provides auto-configuration feature. 3. Provides production ready features (Actuators concept) such as matrix, healthcheck, info etc. 4.No requirement for XML configuration. Provides POM.xml starter file. 5.Provides Profiles feature.

31.Why to use SpringBoot over Spring? • Spring Boot not only improves productivity but also provides a lot of conveniences to write your own business logic. 1. Easy to understand and develop spring applications. 2. Spring Boot is nothing but an existing framework with the addition of an embedded HTTP server and annotation configuration which makes it easier to understand and faster the process of development. 3. Increases productivity and reduces development time. 4. Minimum configuration. We don’t need to write any XML configuration, only a few annotations are required to do the configuration. Below are some key points- • Starter POM. • Auto Configuration. • Component Scanning. • Embedded server and Actuators.

32.How to develop SpringBoot application? 1.Maven 2.STS (Spring Tool Suite) 3.Spring Initializer

33.What are main components Spring Boot Framework? • Spring Boot Framework has mainly four major Components 1.Spring Boot Starters. 2.Spring Boot Autoconfiguration. 3.Spring Boot CLI. 4.Spring Boot Actuator. 34.What is Spring Initializer? • Spring Initializer is a web application that helps you to create an internal project structure for you. So, you do not have to manually set up the structure of the project, instead, you can use this feature.

35.Explain the steps to create a Spring Boot project using Spring Initializer? • Spring Initializer is a web tool provided by Spring. With the help of this tool, you can create Spring Boot projects by just providing project details. The following steps need to be followed to create a Spring Boot project using Spring Initializer: 1. Choose the maven project and the required dependencies. Then, fill in the other required details like Group, Artifact, and then click on Generate Project. 2. Once the project is downloaded, extract the project onto your system 3. Next, you have to import this project using the import option on the Spring Tool Suite IDE 4. While importing the project, remember that you have to choose the project type to be Maven and the source project should contain the pom.xml file. Once, all the above steps are followed you will see that the Spring Boot project is created with all the required dependencies.

36.Explain the steps to connect Spring Boot project with JDBC(MySQL)? Step 1: Create a database in MySQL Step 2: Add the JDBC, MySQL and web dependencies in the application. Step 3: Need to configure the database into application.properties.

37.What is SpringBoot CLI and its use? • Spring Boot CLI (Command Line Interface) is a Spring Boot software to run and test Spring Boot applications from command prompt. When we run Spring Boot applications using CLI, then it internally uses Spring Boot Starter and Spring Boot Autoconfiguration components to resolve all dependencies and execute the application.

38.Difference between Spring and SpringBoot. Spring SpringBoot 1.No auto-configuration feature. 1. Provides auto-configuration. 2.No in-built server. 2. In-built tomcat server. 3.Requires xml configuration 3. No need of xml configuration. 4.More boiler plate code. 4. Reduced boiler plate code.

39.Internal Work flow of SpringBoot application. • The entry point is a class which contains @SpringBootAnnotation and has main method. • SpringBoot scan all the components included in the project using annotation @ComponenetScan and generates Beans itself. • Using @EnableAutoConfiguration annotation SpringBoot configures the application automatically.

41.What is SpringBoot starter? Why it is useful? • SpringBoot starter comprises of number of templates which provides a Rapid Application Development (RAD). It contains combination of relative transient dependencies needed to start particular functionality. • Starter is a jar file which predominantly solves the auto-dependency resolution in a SpringBoot application. • Syntax: spring-boot-starter-X, here is X is required dependency. E.g.- spring-boot-starter-web, so here SpringBoot internally downloads and add all web(X) associated dependencies to the application.

42.Purpose of SpringApplication.run()? • SpringApplication.run() bootstraps a spring application as a stand-alone application from the main method. It creates an appropriate ApplicationContext instance and load beans. It also runs embedded Tomcat server in Spring web application.

43.Does tag add any dependencies? • No, it doesn’t add any dependency. It helps to maintain the application. • ----> One reason to use a parent is that you have a central place to store information about versions of artifacts, compilersettings etc. • ---->A dependency is libraries you need to get your code to compile. This can be your own code, or libraries such as Apache Commons. A parent contains information, but nothing to actually build, that is shared between a number of your projects.

44.What is devtools? • DevTools stands for Developer Tool. The aim of the module is to improve the development time while working with the Spring Boot application. Spring Boot DevTools pick up the changes and restart the application automatically (no need to re-run).

45.What is SpringBoot dependency management? • Spring Boot dependency management is used to manage dependencies and configuration automatically without specifying the version for any of those dependencies.

46.What is HotSwapping? • Hot swapping means you can modify the source code and see the changes immediately, simply by reloading the page in the web browser, without rebuilding your project and have to restart the web server.

47.Which is default port in SpringBoot? • Default port -----> 8080

48.Can we change port number of embedded Tomcat server? How? • Custom port ----->server.port=9999 • If server.port=0 then it will automatically assign any available port.

49.Can we run SpringBoot Application on custom port? (YES)

50.Can we replace embedded Tomcat server? • Yes. By using starter dependency in POM.xml. Exclude tomcat and add required dependency.

51.Can we disable default web server in SpringBoot? • Yes. By using application.properties to configure the web application type. spring.main.web-application-type=none

52.What is an actuator? Its uses, advantages. • An actuator is a feature of SpringBoot that helps to monitor and manage the application when pushed to production environment. These actuators include auditing, health, CPU usage, HTTP hits, and metric gathering, and many more that are automatically applied to our application. • It’s a tool which has http endpoints.

53.How to activate actuator? What do we mean by http endpoints? • To enable the spring actuator feature, we need to add the dependency of “spring-boot-starter-actuator” in pom.xml • Spring Boot Actuator Endpoints lets us monitor and interact with our application.

54.How to run actuator endpoints and server on different ports? • For Actuator ----->management.server.port=9999 Management.endpoints.health.show-details=true On chrome hit----> http://localhost: 9999/actuator/ health • For Server-----> server.port=8800

55.How can we access actuator properties using custom URL? • Default endpoint -----> http://localhost: 9999/actuator/ add “management.endpoints.web.base-path=/customEndpointName” to application.properties • Hit Custom endpoint-----> http://localhost: 9999/customEndpointName/

57.What is profiles concept? Its uses, advantages. • While developing the application we deal with multiple environments such as dev, maintenance, Prod, and each environment requires a different configuration. • Even if DBMS is the same across all the environments, the URLs will be different. To make this easy and clean, Spring has the provision of Profiles to keep the separate configuration of each environment and those segregated configurations are available to that particular environment only. (Properties of dev available to dev only)

58.How to disable AutoConfiguration feature for a particular class? • Using exclude attribute @EnableAutoConfiguration(exclude = {XYZ.class}) @SpringBootApplication(exclude = {XYZ.class})

59.Difference between @RestController and @Controller. • @Controller map the model object to view or template and make it human readable but @RestController simply returns the object and object data is directly written in HTTP response as JSON or XML.

61.Difference between @RequestMapping and @GetMapping. • @RequestMapping can be used with GET, POST, PUT, and many other request methods using the method attribute on the annotation. Whereas @GetMapping is only an extension of @RequestMapping which helps you to improve on clarity on request. @Component @Endpoint(id="customActuator") public class CustomEndPoint { @ReadOperation public String test() { return "test"; } } http://localhost: 9999/actuator/customActuator Output: test @ReadOperation=Get method @Writeoperation=Post method

62.How to use property defined in an application.properties file into java class? • Use @Value (springframework.beans) to access value associated with application.properties E.g., @Value(“${server.port }”) public String serverPort;

63.How to get list of all Beans? • Spring Boot actuator “/beans” is used to get the list of all the spring beans in your application. e.g., http://localhost: 8200/actuator/ beans 64.About .yaml file and .properties file.

65.What is Spring Data JPA? (Java Persistence API) • Provides functionality for connection between application and relational database. It has 3 repository interfaces in that • CrudRepository (It is parent interface, used for CRUD operations) • PagingAndSortingRepository (It extends CrudRepository, used for Paging & Sorting purpose) • JPARepository (It extends PagingAndSortingRepository, used for all above and its additional services)

66.How to create custom query? • Using @Query annotation. E.g., @Query("select c from City c where c.name like %?1") List findByNameEndsWith(String chars);

67.How to Configure Hibernate in SpringBoot? application.properties spring.datasource.driver-class-name=com.mysql.jdbc.Driver spring.datasource.url=jdbc:mysql://localhost:3306/sbcurd spring.datasource.username=root spring.datasource.password=root spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL55Dialect spring.jpa.show-sql=true spring.jpa.hibernate.ddl-auto=update

69.Difference between @RequestBody and @ ResponseBody 70.What is the difference between @component and @bean? • @Component is a class level annotation whereas @Bean is a method level annotation and name of the method serves as the bean name. • @Component need not to be used with the @Configuration annotation where as @Bean annotation has to be used within the class which is annotated with @Configuration

72.Disadvantages of SpringBoot? • When we provide starters into POM.xml it downloads lots of dependencies from central repository to .m2 folder, at some point many of those dependencies may not be useful. So increases the size of an application. • Suitable for micro-services but not for large or monolithic web services.

73.What is AutoConfiguration? How it helps? Why SpringBoot is called Opinionated? • AutoConfiguration – automatically configures Spring application based on the dependencies added in POM.xml. • Spring Boot is opinionated because it follows the opinionated(ideal) default configuration that reduces developer efforts to configure the application. • Opinionated – a way of approaching a business process is inherently better and provides a software crafted around that approach.

**Webservices: -**

1. What is Webservices? • It is a software system that is specially designed to propagate communication between the client and server applications on WWW (World Wide Web). • It is the method, medium or way of communication among two or more devices over a network. It allows multiple applications built on different programming languages to communicate with each other without any trouble. It uses the internet for direct application-to-application interaction, and also allows you to expose business logic using API. • It is a software system for the interoperable machine to machine communication. • It is a collection of standards or protocols for exchanging information between two devices or application.

2. Advantages of Webservices? • Interoperability: Web services are accessible over network and runs on HTTP/SOAP protocol and uses XML/JSON to transport data, hence it can be developed in any programming language and on any OS. Web service can be written in java programming and client can be PHP and vice versa. • Reusability: One web service can be used by many client applications at the same time. • Loose Coupling: Web services client code is totally independent with server code, so we have achieved loose coupling in our application. • Easy to deploy and integrate. • Multiple service versions can be running at same time. • Supports the transparent exchange of data to facilitate business integration. • Supports Remote Procedure Call (RPC), a powerful technique for constructing distributed, client-server-based applications.

3. What are the types of webservices? SOAP (Simple Object Access Protocol) Web Services: It is also referred to as transport-independent messaging protocol whose main purpose is to transfer a message, and is based on XML protocol. RESTful (Representational State Transfer) Web Services: It is developed to fulfil the shortcomings of SOAP and to make the web services more effective. SOAP ReST

1. It supports XML data format only. It supports XML, HTML, Plain text, JSON, etc. data formats

2. It is protocol based It is architecture based.

3. It has its own security. Need to add security.

4. It is less preferred. It is more preferred.

5. Need JAX-WS API to implement SOAP. Need JAX-RS API to implement REST. -JSP---> Java Servlet Pages -json---> JavaScript Object Notation (data gets stored in key value pair)

4. How to create Restful webservices? • Add REST dependency in pom.xml. (spring-boot-starter-data-rest) • Add @RestController annotation for controller class. • Map the methods written in controller class. Each content present in the database is considered as a resource(objects) in ReST. A resource can be text file, an image or any dynamic data. URI is used to identify each resource of REST architecture. URI-Format Example ://// <https://en.wikipedia.org/wiki/ReST-api>

5. What is REST? Why to use it? • REST is the acronym for Representational State Transfer. REST is an architectural style for developing applications that can be accessed over the network. • REST is a stateless client-server architecture where web services are resources and can be identified by their URIs (Uniform Resource Identifiers). It uses http protocol to fetch those resources. • We choose Restful web service because - Simple and easy to implement and test. - It is programming language and platform (OS) independent - Support different formats such as JSON, XML, HTML, etc - Faster and provide better performance. - It provides caching mechanism to minimize server calls for the same type of repeated requests. The protocol used by REST web services to interact with server is HTTP. Some of the HTTP methods----> GET: Used to get and read a resource (retrieve data). POST: Used to create a new resource (insert data). PUT: Used to update existing resources (update data). DELETE: Used to delete the resource. PATCH: Used to apply partial modifications to a resource. • Medium of communication between client and server is called “messaging”.

6. What is statelessness w.r.t. ReST? Advantages. • Statelessness is basically a condition or restriction where RESTful web services are not allowed to keep a client state on the server as per the REST architecture. Clients are responsible to pass their context to the server. To process the client’s request, the server then further stores this context. E.g., if you get data using Get method, but if you want to fetch same data again then we need to use Get method again to get the response and every time it will be considered as a new request. • Advantages: -No need to maintain previous interactions with clients. -Independent treatment of each method request. -Less complexity and simplified application design. • Example: Simple GET Request using NodeJS

7. What is the use of ARC and Postman tools? • To test the webservice • Postman is an application used for API testing. It is an HTTP client that tests HTTP requests, utilizing a graphical user interface, through which we obtain different types of responses that need to be subsequently validated.

8. Explain http status codes. REST API generally returns the following status codes in HTTP response: • 200 OK • 201 Created • 202 Accepted • 302 Found • 400 Bad Request • 401 Unauthorized • 404 Not Found • 405 Method Not Allowed • 409 Conflict • 500 Internal Server Error

9. Difference between webservices and API webservices API 1. It must use a network. It can be online or offline. 2. All web services are APIs. All APIs are not web services. 3. It is interaction between applications over the network. It is an interface between two different applications. 4. Web service uses styles like REST, SOAP for communication. API can be used for any style of communication. 5. Web Services does not have a light-weight architecture. API has a light-weight architecture.

10.Difference between @PathParam and @PathVariable @PathVariable @PathParam 1. It is from spring framework. It is from JAX-RS. 2. It will work in spring MVC and REST It will work in REST only. 3. It is annotation on a method argument to bind it to the value of a URI template variable. It is a parameter annotation which allows you to map variable URI path fragments into your method call.

11.What is RestTemplate? What is its use? • Rest Template is used to create applications that consume RESTful Web Services. (Provides communication between two templates). Basically, used to communicate, access and operate produce data using consume.(Refer program)

12.Drawbacks of WebService? How to overcome from it? • Lack of state: most web applications require stateful mechanisms. Suppose you purchase a website which has a mechanism to have a shopping cart. It is required to know the number of items in the shopping cart before the actual purchase is made. This burden of maintaining the state lies on the client, which makes the client application heavy and difficult to maintain. • Last of security: REST doesn’t impose security such as SOAP. That is the reason REST is appropriate for public URLs, but it is not good for confidential data passage between client and server. • Monolithic Architecture

14.Components of http? • An HTTP request is divided into three parts: Request line, header and body.

15.Name the web service method that is read-only, and is idempotent? • The web service method that is read-only and safe is the GET method, and the web service methods that are idempotent are PUT and DELETE operations. Post is not idempotent. • Idempotent refers to the operations whose results will always be the same even if these operations are invoked so many times. That is if you hit any server with same request for any number of times then the every time the response should be same (no change in memory for that particular resource). e.g., With each Post operation there is change in the database memory allocation so it is not an idempotent but if we take Put operation there will not be any change in the databases memory even if you use it any number of times so, this is idempotent. Delete is also idempotent as if you delete a particular resource, you cannot delete the same resource again so, there will not be change using same method again.

16.What is Consumed or Produced Restful web service? Explain? • E.g., makemytrip(consume) accesses all airlines(produce) data to show availability of tickets for various location-locations. So here consume access produce data to show availability of services to the user and provides interfaces to book the tickets and after booking the ticket it updates both consume and produce data for that booking resources. That’s why user get mail from both produce(particular airline) as well as consume(makemytrip). Remember here consume isn’t scheduling the flights it is just accessing the produce and after the use of service it is updating the data for both(that is produce must know user details for that booking ) • Above is just an example. But consume can access produce as per need Note: Produce and Consume can have different databases

18.What is broken API? If we hit an API and if that API is not mapped in the controller, then such APIs are called broken API. (Status code ---> 404 not found) 19.How to convert java object to json and vice versa? • Object to json----> 1. Add gson dependency in POM.xml 2. new Gson().toJson(object) • Json to object----> 1. Add gson dependency in POM.xml 2. Gson g= new Gson(); Student s = g.fromJson(jsonString, Student.class);

USE OF…..

@annotations:

1. @SpringBootApplication Marks the class as a Spring Boot annotated class which contains main (). [@EnableAutoConfiguration + @ComponentScan + @Configuration] @EnableAutoConfiguration: Triggers autoconfiguration when dependency added in POM.xml. @ComponentScan: Scans packages and sub packages for Spring Beans. @ComponentScan (basePackages = {“package name”, “package name”}): Scans packages and sub packages from specified packages for Spring Beans. @Configuration: This annotation declares class as a Configuration class which is a source of beans (returns object).

2. @Bean Used over a method in a configuration class which is used to denote that it is returning a bean and the bean is managed by a Spring Container.

3. @Component @Controller @Repository @Service Used at class level. So, no need to declare @Bean over each method. @Repository public interface Repository extends CrudRepository { } @Controller public class StudentController { } @Service public class ServiceImpl implements Service {

} 4. @Autowired Can be used over property, setters, constructors. @Autowired StudentService ss;

5. @Controller • Part of a component. Defines bean. • Used as Presentation layer. • Provides Spring MVC features. • Used if want return view (ex.- .jsp, .html pages)

6. @Service Used at Service Layer.

7. @Repository Used as DAO (Data Access Object) layer. Database Transactions.

8. @Qualifier (Enter bean id) Used if more than one @Bean returns same type of object. Here object will be created for all those objects but only that object Will be returned whose bean id is entered while specifying Qualifier. Ex- @Bean(“Student1”) → @Qualifier(“Student1”)

9. @Lazy Doesn’t create object if the particular method is not called. Object is crated only on request.

10. @RequestMapping( value = “/URL”, method = RequestMethod.GET) • Used over a method to map specific operation URL. @RequestMapping(value = "/update”, method = RequestMethod.PUT)

11. @ResponseBody This tells controller that the object returned is automatically serialized into Jason and passed back as a http response.

12. @RequestBody • Deserializes and converts Jason data into specified object. • Used as a method parameter in a controller. @RequestMapping(value = "/update”, method = RequestMethod.PUT) public String update (@RequestBody Student s)

13. @RestController = @Controller + @ResponseBody • No need to specify @ResponseBody explicitly on individual method. • It will return data into http response (specifies application as webservice) • Comprises properties of both the annotations. • Defined at a class level and defines class as source of bean. • It is a combination of @Controller and @ResponseBody, used for creating a restful controller. It converts the response to JSON or XML. It ensures that data returned by each method will be written straight into the response body instead of returning a template.

14. @PathVariable • Deserializes and converts Jason data into specified object. • Used as a method parameter in a controller to get data from URI. • Need to change mapping according to required path variable. http://localhost:8090/employee/1/john Syntax: - @RequestMapping(value = "/login/{id}/{name}”, method = RequestMethod.GET) public String login(@PathVariable ("id") int id, @PathVariable ("name") int name)

15. @RequestParam • Used when we want to get data from Form. Brings single variable from client side • @RequestParam is a Spring annotation used to bind a web request parameter to a method parameter in query format. http://localhost:8090/employee?id=1&name=john • No need to change mapping. @RequestMapping("/login”) //read the provided form data public String display(@RequestParam("id") String name, @RequestParam("name") String name)

16.@ModelAttribute @ModelAttribute is an annotation that binds a method parameter or method return value to a named model attribute, and then exposes it to a web view. • Brings complete POJO from client side Syntax: - public String saveRegisteredData(@ModelAttribute UserData ud) Note - @ModelAttribute is used for binding data from request param (in key value pairs), but @RequestBody is used for binding data from whole body of the request like POST, PUT request types which contain another format like json, xml.

17.@EnableJPARepository It is used to scan the packages for configuration and repository class for Spring Data JPA.

18.@EntityScan Used to scan all the classes marked with @Entity annotation

19.@Value Used to get value associated with keys in application.properties files. @Value("${spring.mail.username}") private String fromEmail; 20. @CrossOrigin(\*) 21. @Data () Used at class level to reduce boiler-plate code (repetitive code like settergetter etc.). But to use this annotation Lombok dependency is mandatory