Spring Boot

**How does Spring boot Application Start**?

Look for Main class,

Annotations- @Configuration,@EnableAutoConfiguration,@Componentscan

**Important Annotations for Spring boot** :-

@RestController- Combination of @Controller and @Responsebody(Activates to send response automatically as JSON object)

@PathVariable- used to capture vales from URL

@RequestParam- used to capture parameter values from URl present after?

**InterView Questions for JavaSpring Boot**

[**https://www.interviewbit.com/spring-boot-interview-questions/**](https://www.interviewbit.com/spring-boot-interview-questions/)

**Changing port number :**

Server.port=port number

Changing server type: Add exclusion to the tomcat server in pom.xml file and include dependency of spring boot jettty server

**Initiating Actuator**- An actuator is used for monitoring include auditing, health, CPU usage, HTTP hits, and metric gathering using various end points.

**Dependency Injection**

**Setter based-**Passed through Setters | Partial Initialization is allowed

**Constructor Based-** Initialized with Constructor |Partial Injection is not allowed

**Filed Injection-** Injecting a field of an object

**ResponseEntity**- It is a complete HTTP response that contains headers/body and status code

Headers are a king od hashmap added in Key,Value format

**Spring Data JPA**

-Custom Query Overrides repository methods

Annotations

@Entity-For Table

@Table(name = "Customer",schema = "Saleslt")

@Id primary Key column

@Generationtype= Auto-> Choose an Strategy Automatically as per Db

@GenerationType->Identity- Done by DB only.

@GenerationType->Sequence – Query Db and put next sequence

@Repsitory -> To tell interface which serves as JPA repository.

Usage-

public interface CustomerRepository extends JpaRepository<Customer, Integer>

Implement all JPA methods here

Different Types of Mappings :

**One-to-one mappin**g: The one-to-one mapping represents a single-valued association where an instance of one entity is associated with an instance of another entity

**One-to-Many**: The One-To-Many mapping comes into the category of collection-valued association where an entity is associated with a collection of other entities

**Many-to-one mapping:** The Many-To-One mapping represents a single-valued association where a collection of entities can be associated with the entity.

**Many-to-many mapping:** The Many-To-Many mapping represents a collection-valued association where any number of entities can be a ssociated with a collection of other entities.

Joins Using JPA and configuring Fetch Types

1.Configure Join column in source Persistence class

@OneToMany(targetEntity = SalesOrderHeader.class,cascade = CascadeType.ALL,fetch = Fe tchType.EAGER)

@JoinColumn(name="Customerid",referencedColumnName = "Customerid")

The result will be List within a List

Caching-Ehcache

Steps

1.Prepare Echcache xml file to preprare settings for cache setting.

2. @EnableCaching- at main class

3. Optional to implement cacheEvent logger with a new java class and implement CacheEventListener to override CacheEvent.

4@Cacheable(value = "custById",key = "#id") annotation at method level in Service class and value &key configuration in sync with Ehcache XML file

5. In some cases implement the Serializable interface in entity class to make the entitty class writable to memory/disk

@Cacheevict- @CacheEvict(value = "first", key = "#cacheKey")- can be called over a method to clear cache.

Also CacheManager’s clear method can be explicitly used to clear cache.

CacheEventListeners- Set call back methods when a selected cache method is called

Sessions- Can be configured in XML file mentioning the class names that will define its scope or @Scope(‘Session’) on a class

@Required annotation used for setting a filed/method mandatory

To inject values from contructor we use construcot-arg

Difference between Auto wired and Inject Annotation

Inject-Spring Cdi, Autowired-Spring

Default scope-Singleton---- Prototype

Incase of ambiguity we use Qualifier--- we use Named

Singleton Bins are not thread safe

What are SpringInterceptors

used to pre-handle and post-handle the web requests in Spring MVC which are handled by Spring Controllers. This can be achieved by the HandlerInterceptor interface. These handlers are used for manipulating the model attributes that are passed to the controllers or the views

There are 3 call back methods for it

Prehandle()

PostHandle

Oncompletion()

Which method is used for shutting down IoC container in Spring

RegusterShutDownHook()

Ways to implement Actuators

1.Add the spring boot starter actuator dependency

2. Implement Security

spring.security.user.name=Pankaj spring.security.user.password=Pankaj

Ways to implement Spring boot Secirity

1.Add dependency

2Extend The **WebSecurityConfigurerAdapter and override the Configure method**

**3. create a bean with UserDeialsSeriveclass nd add thee details with defaultPasswordEncoder**

**4.you can use inMemoryDetailsManager**

Digest Authentication- Uses password Hash explicitely

Session configuration has two options

* SessionManagementFilter.
* SessionAuthneticationStrategy

OAuth Security Architecture

Owner->Client-> Authentication Server-> Actaul resource

Access token is given to the client by Authentication server

Password Encoder two methods and Comes from defualtencoder factories

Encode and

Mathches

@Secured Method secure methods within a class

Master Exception handling

1.Create a class that extends ResponseEntityExceptionHandler with @RestControllerAdvice Annotation. 2.Create a custom exception class that Extends Exception3.Throw a new conditional Exception of the same class that was created above.4.Controller advice will catch the exception and will get handled by the global exception handler.5.Finally in the global exception handler class create a response Entity that gives out response@ExceptionHandler(value = CustomerNotFoundException.class) return new ResponseEntity<>(apiResponse.toString()**,**HttpStatus.*NOT\_FOUND*)

Actuators

**Initiating Actuator**- An actuator is used for monitoring include auditing, health, CPU usage, HTTP hits, and metric gathering using various end points.

Spring Boot Actuator Endpoints are exposed over JMX and HTTP

Common End Points:-

1. beans: this endpoint returns the list of all the beans configured in our application.
2. env: provides information about the Spring Environment properties.
3. health: Shows application health
4. info: Displays application information, we can configure this in Spring environment properties.
5. mappings: Displays the list of all [@RequestMapping](https://www.digitalocean.com/community/tutorials/spring-requestmapping-requestparam-pathvariable-example) paths.
6. shutdown: allows us to gracefully shutdown the application.
7. threaddump: provides the thread dump of the application.

Adding Dependency

<artifactId>spring-boot-starter-actuator</artifactId>

Add this Information to Application.properties

info.app.name=Spring Actuator Example

info.app.java.version=10

info.app.type=Spring Boot

Making Custom Endpoints

spring-boot-starter-security

To Disable - @SpringBootApplication(exclude = { SecurityAutoConfiguration.class })

Add @Endpoint(id="myendpoint")to the class that you want to add @ReadOperation @Bean public String hi() { return "Hi from custom endpoint"; }Only info and health can be put without Spring Security hence we need to configure Spring Security and the details of below spring.security.user.name=Pankaj spring.security.user.password=Pankaj

Finally add end points to the application properties filemanagement.endpoints.web.exposure.include=health,info,beans,env,myendpoint.

In Application Properties.file

management.endpoints.web.exposure.include=\*

STUDY DTO CLASS