Spring Boot

1. What are the steps of rapid application development…….??

1. Business module :

2. Data module

3. Process module

4. Application generation

5. Testing and Turnover.

2. How to change port of embedded tomcat…….??

1. Add server.port = 8080 in application.properties file.

4. Can we replace embedded tomcat with another server...??

1. Yes, we can change tomcat. Tomcat is added because of spring boot start web dependency.

2. Add below dependency to exclusion tomcat and add jetty dependency in pom.xml file.

<dependency>

<groupId>org.springframework.boot</groupId>

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

<exclusions>

<exclusion>

<groupId>org.springframework.boot</groupId>

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

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

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

<scope>provided</scope>

</dependency>

5. How to disable auto configuration class...??

1. Yes, by adding below annotation in main class of spring boot app:

@EnableAutoConfiguration(exclude={class name that to be exclude})

@SpringBootApplication (exclude={JacksonAutoConfiguration.**class**, JmxAutoConfiguration.**class**})

6. What does the @SpringBootApplication annotation do internally...??

1. It is combination of three annotation :

@Configuration :

It Identify the bean for particular spring dependency injection.

@EnableAutoConfiguration

It will be used to auto scan bean, if it is available then add to into set of dependency.

@Component scan.

7. How to use a property defined in application.properties in java class…???

@Value{“${server.port}”}

Private int portNo;

We can also read value form properties files using **@ConfigurationProperties annotation :**

@Component

@ConfigurationProperties("limits-service")

**public** **class** Configuration

{

**private** **int** maximum;

**private** **int** minimum;

**}**

8. What is @RestController annotation…???

1. It is combination of @Controller and @ResponseBody.

2. It convert the response to JSON or XML.

9. Difference between @RestController and @Controller annotation…???

1. @Controller map of the model object to view or template and makes it human readable format.

2. @RestController simply return the object and object data is directly written into Http response as JSON or XML.

10. What is the use of Profiles in spring boot…???

1. When developing application for enterprise, we typically deal with multiple environments such as Dev, QA, Prod. The configuration properties for all environments are different.

2. For example, we might be using an embedded H2 for Dev, but Prod could have the MySQL.

3. To set profile add spring.profiles.active in application.properties file.

11. What is Spring Actuator…???

1. In Spring Boot, Actuator is an additional feature that help you monitor and manage your application when you push it to production. These features include Auditing, health, and metrics gathering and may more features that can be automatically applied to your application.

2. You can enable this feature by adding spring-boot-starter-actuator dependency in pom.xml file.

3. In application.properites file we need to disable security for actuator endpoints.

management.security.enable = false

4. If you want separate port for accessing spring boot app then add management.server.port = 9000 in application.properties files.

5. Actuator Endpoints :

i. Metrics :

To view the application metrics such as memory used, memory free, thread, classes, system uptime etc.

ii. env :

To view list of Environments variables used in application.

iii. beans :

To view the spring beans and its types, scopes and dependency.

iv. health :

To view the application health.

v. info :

To view the information about the spring boot application.

vi. trace :

To view the list of traces of your rest endpoints.

12. Enabling HTTP Trace…???

1. Http trace end point does not enable by default because of memory issue and memory is costly, from sprint boot version 2.2.X.

2. To enable this end points create Configuration class like below :

@Configuration

public class ConfigurationClass{

@Bean

Public HttpTraceRepository httpTraceRepository {

return new InMemoryHttpTraceRepository();

}

}

13. How to change default path of actuator …???

1. To change default path(<http://localhost:8080/actuator/>...) to other path add below property in application.properties file

management.enpoints.web.base-path=/manage

14. How to create Custom end points …???

1. To create custom end points we need to add two annotation

i. @Endpoints

ii. @ReadOperation

2. Example below :

@Component

@EndPoints(id =” customActuator”)

public class CustomActuator{

@ReadOperation

public String currentDbDetails(){

return “Give current db status of appliaction”;

}

}

2. Last hit url for custom endpoints : Http://localhost:8080/actuator/ customActuator.

15. How to deploy spring boot application JAR and WAR files…???

1. To deploy spring boot app jar and war add plugins in pom.xml files like below :

<build>

<plugins>

<plugin>

<groupId>org.springframwork.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

2. Replace :

<packaging>jar</packaging> to

<packaging>war</packaging>

3. Run command :

mvn pacakge

and war file will be created inside target folder of application.

4. Copy war file from target folder and past in webapp directory of tomcat folder.

16. Advantages of YAML over the Properties file …???

1. More Clarity and better readability.

2. It is Hierarchical structure for file for better understanding.

3. Support map, list, and scalar type.

4. It avoid boiler plate code.

17. @Bean Annotation …???

The**@Bean**annotations are used at the method level and indicate that a method produces a bean that is to be managed by the Spring container. It is an alternative to the XML<bean> tag.

@Bean

Public BeanExample beanExample ()

{

return new BeanExample (),

}

18. @Required Annotation …???

The field which is using this annotation must be configured at the time of configuration with all the required properties, otherwise, it will throw an exception saying BeanInitilizationException. This annotation can be applied on the setter method.

public class Demo

{

private Integer costDemo;

@Required

public void setcostDemo(Integer costDemo)

{

this.costDemo = costDemo;

} }

19. @Get vs Post …???

|  |  |
| --- | --- |
| **GET** | **POST** |
| 1) In case of Get request, only **limited amount of data**can be sent because data is sent in header. | In case of post request, **large amount of data**can be sent because data is sent in body. |
| 2) Get request is **not secured**because data is exposed in URL bar. | Post request is **secured**because data is not exposed in URL bar. |
| 3) Get request **can be bookmarked.** | Post request **cannot be bookmarked.** |
| 4) Get request is **idempotent**. It means second request will be ignored until response of first request is delivered | Post request is **non-idempotent.** |
| 5) Get request is **more efficient**and used more than Post. | Post request is **less efficient**and used less than get. |

**Bookmarked :**

Bookmarking is when the browser remembers the URL of a page. GETs can be bookmarked because the data needed for the request is stored in the URL and hence the bookmark. POSTS cannot because the data is stored in the request body which is not stored as part of the bookmark.

**idempotent** :

An HTTP method is **idempotent** if the intended effect on the server of making a single request is the same as the effect of making several identical requests.

 An idempotent HTTP method is **an HTTP method that can be called many times without different outcomes**.

All [safe](https://developer.mozilla.org/en-US/docs/Glossary/Safe/HTTP) methods are idempotent, as well as GET ,[PUT](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT) and [DELETE](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/DELETE). The [POST](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST) method is not idempotent.

20. @PathVariable and @RequestParam …???

1. **We are using** @PathVariable **and** @RequestParam **annotation to fetch value from URL.**

**2. But in case of** @PathVariable **we need to provide parameter name in get request mapping or any request.**

// <http://localhost:8080/api/employee/1/Priyanka>

**@GetMapping(“/employee/{id}/{name1}”)**

Public Employee getEmployee(@PathVariable int id, @PathVariable(“name1”) String name){

return;

}

**3. But in case of** @RequestParam **we are not providing parameter name in get request mapping or any request, this is used to send query parameter.**

**//** http://localhost:8080/api/employee?id=1&name=priyanka

**@GetMapping(“/employee”)**

Public Employee getEmployee(@RequestParam int id, @RequestParam (“name1”) String name){

return;

}

**4. We are using {} curly brace for path variable and ? question mark for RequestParam Request.**

21. How to use two database in Spring Boot …???

**1. Add the two database properties in application.properties file.**

**Spring.datasource1.url = url**

**For others**

**Spring.datasource2.url = url**

**2. We need to create configuration class**

@Configuration

Public class SpringConfiguration{

@Bean

@Primary

@ConfigurationProperties(prefix="spring.datasource")

public DataSource primaryDataSource() {

return DataSourceBuilder.create().build();

}

@Bean

@ConfigurationProperties(prefix="spring.secondDatasource")

public DataSource secondaryDataSource() {

return DataSourceBuilder.create().build();

}

}