TESTYANTRA

SOFTWARE SOLUTIONS (INDIA) PVT. LTD.

Spring Boot

EXPERIENTIAL
learning factory

- It is an open source Java-based framework
- It is used to create a micro Service
- We can develop stand-alone application as well as enterprise application

Advantages

- Easy to understand and develop spring applications
- Increases productivity
- Reduces the development time
- Avoids complex XML configuration in Spring
- Easy way to develop Spring application

Auto-Configuration

- Spring Boot automatically configures application based on the dependencies
- **@EnableAutoConfiguration** annotation is used to activate auto configuration
- Spring Boot automatically scans all the components included in the project by using **@ComponentScan** annotation
- If **@SpringBootApplication** annotation is used then nether **@EnableAutoConfiguration** nor **@ComponentScan** and **@SpringBootConfiguration** have to use in the application
- **@SpringBootApplication** will do both the task

Auto-Configuration continued

- Spring boot provide in memory database for auto configuration
- To use is we have to add the h2 dependency and enable h2 by adding spring.h2.console.enabled=true to the application.properties
- Now all the data will store in the in memory database

Spring Boot Starters

- Dependency handling is a difficult task
- Developer have to check all the dependency supports each other
- Spring Boot resolves this issue by spring boot starter pack
- All the Spring Boot starters follows the pattern **spring-boot-starter-***
- Few starter dependencies are
  - Spring Boot Starter Actuator
  - Spring Boot Starter Test
  - Spring Boot Starter web
  - Spring Boot Starter Thyme Leaf

Spring Boot Code Structure

- **com**
  - testyantra
    - project
      - *Application.java*
      - pkg1
        - *JavaFiles*
      - pkg2
        - *JavaFiles*
- **This should be the package structure**
- **Application.java should have the main method**
- **Rest of the package will have the code for the application**

Spring Boot - Runners

- **Runners helps you run any code immediately after the spring boot application started**
- **Types of Runners**
  - ApplicationRunner
  - CommandLineRunner
- **We have to implement any one of the Interfaces and override run method**

Spring Boot RESTful web services

- Add boot-starter web dependency
- Create a class and add @RestController for Rest Controller class
- Create mapping methods and add @RequestMapping for URI mapping or @PostMapping, @GetMapping, @PutMapping, @DeleteMapping for respective HTTP methods
- Add method = RequestMethod.{HTTP Method} attribute to the RequestMapping annotation
- ResponseEntity object used to give back the response

Spring Boot - Interceptor

- **Interceptor is used to execute some code before**

Sending request to the controller
Sending response to the client

- **To create interceptor add @Component annotation to the class and implement HandlerInterceptor interface**
- **There are 3 methods**
  - preHandle() – This method is used to execute code before sending the request to the controller
  - postHandle() – This method is used to execute code before sending the response to the client
  - afterCompletion() – This method is used to execute code after completing the request and response

Spring Boot Overriding Auto Configuration

- We can override auto-configuration
- To do that we just have to add configuration properties for our DB into the application.properties file
- We can override auto-configuration using java classes
- Create a @Configuration class with @Bean method which returns DataSource
- We will have more control on Java based configuration than properties configuration
- We can have conditions to create bean objects using *@ConditionalOnClass, @ConditionalOnMissingClass, @ConditionalOnBean, @ConditionalOnMissingBean etc annotations*

Properties vs Java Configuration


```java
@Bean
public DataSource dataSource(){
    DriverManagerDataSource dataSource = new DriverManagerDataSource();
    dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");
    dataSource.setUrl("jdbc:mysql://localhost:3306/spring_jpa");
    dataSource.setUsername( "tutorialuser" );
    dataSource.setPassword( "tutorialmy5ql" );
    return dataSource;
}
```

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.username=mysqluser
spring.datasource.password=mysqlpass
spring.datasource.url=
  jdbc:mysql://localhost:3306/myDb?createDatabaseIfNotExist=true
```

- In spring boot JPA there is an interface called CrudRepository which can provide all the crud operation using hibernate
- There is another interface called JpaRepository which internally extends CrudRepository which helps us to write JPQL using @Query annotation

Spring Boot - Actuator

- **Spring Boot Actuator provides secured endpoints for monitoring and managing your Spring Boot application**
- **By default all the endpoints in actuator is secured**
- **To enable the actuator there only one step**
  Add dependency to the pom.xml
- **To enable all endpoints add management.endpoints.web.exposure.include=* in the application.properties**

Spring Boot - Exception Handling

- @ControllerAdvice annotation is used to handle exceptions in Spring Boot
- At the class level @ControllerAdvice annotation is used
- At the method level @ExceptionHandler annotation is used
- We can have multiple exception handler with different exception to accept

TESTYANTRA

SOFTWARE SOLUTIONS (INDIA) PVT. LTD.

Thank You !!!

No.01, 3rd Cross Basappa Layout, Gavipuram Extension,
Kempegowda Nagar, Bengaluru, Karnataka 560019

praveen.d@testyantra.com

www.testyantra.com

EXPERIENTIAL
learning factory