



Why Spring Boot over Spring?

Easy to use-remove boiler plate code

Production ready application

Rapid development -Autoconfiguration enables developers to quickly develop apps

Embedded server- provide tomcat server by default

Working Of Spring Boot?

Spring boot Starts by Scanning Starter dependencies for Pom.xml

Then download and auto configure module as you included in pom.xml

2. Suppose we are building Spring boot web application then at the time of project creation we select starter web automatically dependency will add in pom like (Spring-boot -Strater-web)

When we We start the project Spring boot Automatically configure required things to run web Application

How Spring boot Starts?

Spring boot Starts from main() method of our main class

```
package com.lcwd.root;  
  
> import ...  
  
@SpringBootApplication  
public class SpringBootWorkApplication {  
  
    public static void main(String[] args) {  
        SpringApplication.run(SpringBootWorkApplication.class, args);  
    }  
}
```

The

Run() method is Start Sprig Boot Application. This method Starts the application by creating an Application context(contains beans) and initialize it

Once application context is initialized the run () method Start the application embedded web server

top Spring Boot Annotations?

1. @SpringBootApplication- its combinations of three
@configuration, @enableAutoConfiguration, @componentScan
2. @component //it's used to mark class as Spring Bean that will manage by Spring container

With help of Component scan. scan the component from bean class

```
1 package beans.container;  
2  
3 import org.springframework.stereotype.Component;  
4  
5  
6 @Component  
7 public class Student {  
8     public Student() {  
9         System.out.println("This is student creation");  
10    }  
11 }
```

here is bean class object call

```
@SpringBootApplication  
@ComponentScan("beans.container")  
public class SpringBootWorkApplication {  
1  
    public static void main(String[] args)  
    {  
    }
```

```
public class SpringBootWorkApplication {  
    public static void main(String[] args)  
    {  
        ConfigurableApplicationContext context = SpringApplication.run(SpringBo  
        Student bean = context.getBean(Student.class);  
        System.out.println(bean);  
    }
```

If you remove @component, application context didn't get any bean and Thow error like

```
13012 [main] com.tienda.springbootworkapplication  
org.springframework.beans.factory.NoSuchBeanDefinitionException Cre
```

3.@Autowired – this annotation is used to add automatically inject dependencies into a Spring Managed bean (copy different class object into our class)

4. @Service : its used to annotate class that contains buiseness logic

Class is represented as service component in application

5. @RestController- marks class as REST controller. its specialized version of

Always use with each other

@controller || @ResponseBody // control the request and in response whatever you write in class share as response through response body class will be automatically converted to JSON or XML

6. Request Mapping: Used to map HTTP requests to specific methods in a controller.

Specifies the path and HTTP method for the endpoint (e.g., GET, POST)

```
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.GetMapping;

@RestController // This annotation makes the class a RESTful controller.
@RequestMapping("/api") // This sets a base path for all methods in this class.
public class MyController {

    @GetMapping("/greet") // Maps the endpoint "/api/greet" to this method for HTTP GET requests
    public String greet() {
        return "Hello, welcome to our API!";
    }
}
```

7. @Repository: Marks Class As Dao mostly used on class if logic is database related

What are springBoot Starters

Spring Boot Starter is collection of pre-configure dependencies that make it easy to develop particular application like below These include dependencies ,version control, configuration

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-devtools</artifactId>
  <scope>runtime</scope>
  <optional>true</optional>
</dependency>
<dependency>
  <groupId>com.mysql</groupId>
  <artifactId>mysql-connector-j</artifactId>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
```

What are the key dependencies of Spring Boot?

- 1.Spring-Boot-Starter-Parent
 2. Spring -Boot-Maven-plugin
 - 3.Spring-Boot-Starter-test
 4. Spring-Boot-Starter-security
 5. Spring-Boot-Starter-actuator
 6. Spring-Boot-Starter-web
 - 7.....
-

What is Spring-Boot-Starter-Parent

Its starter project that provides the default configuration for Spring-Boot application

- The dependency management
- provide default compiler
- Provide default configuration for maven plugins such as maven-surefire-pluggins,maven-jar-pluggin,and maven-failsafe-plugin
- Resource Filtering
- everything is managed by parent like version and all

```
parent>  
<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-starter-parent</artifactId>  
<version>3.3.5</version>  
<relativePath/> <!-- lookup parent from repository -->  
/parent>
```

Can we use only Spring Boot Dependency feature and configure maven pluggin manually

Ans. Yes

We can do manually dependency management section but first we should remove parent

And then add versions, pom, import

What is Spring Boot CLI what are its benefits

Its cli to create run and manage SpringBoot Application

Commands

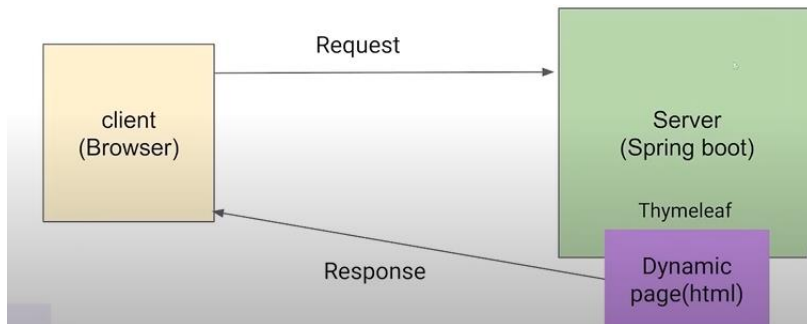
- ➔ Spring
- ➔ Spring help init
- ➔ Spring version
- ➔ Spring-Init - - dependencies =web,data-jpa MyProject_Name /// using command line to create project

But we should set path first

- ➔ Spring path= here you paste downloaded path of Cli
-

What is thymeleaf?

Java based server side templating Engine used in java web Appliaction to render dynamic web pages



What is IOC or Inversion Of Controller

→ Inverting the control of creating object using new keyword to container or framework

```
Class Battery{  
    }  
Class Phone{  
    Battery BT=new Battery();  
}
```

No need to create manually object like upper we give to Inversion container framework to create object and manage it

inversion of control

IOC Container[its Framework]

Note- DI And IOC is heart of Spring

Explain the Spring Bean-Lifecycle

→ **Bean**—its simple plane java object

Lifecycle-

Spring bean life cycle is maintained by IOC Container

- 1.Container gets Started
- 2.container creates and object
- 3.it will go in pom.xml to create dependencies
- 4.dependancies is injected
- 5.destroyed when container closed



```
9 public class Battery {  
10     public Battery() {  
11         System.out.println("battery is created ");  
12     }  
13  
14     @PostConstruct  
15     public void init() {  
16         System.out.println("initializing battery");  
17     }  
18  
19     @PreDestroy  
20     public void destroy() {  
21         System.out.println("distroy");  
22     }  
23  
24 }
```

? **@PostConstruct** runs after the object is created and initialized.

? **@PreDestroy** runs before the object or bean is destroyed.

Or if you want to using java without annotation

Using java --

To just implement

Initializing_Bean

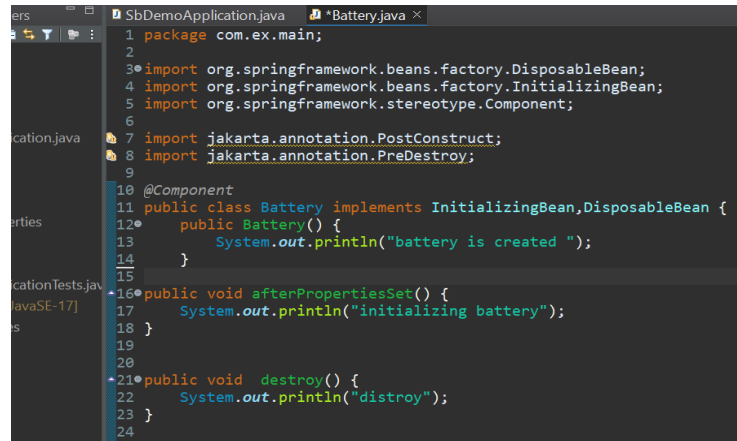
After properties set

Disposable_Bean

Destroy

Note-

using XML Also we can create



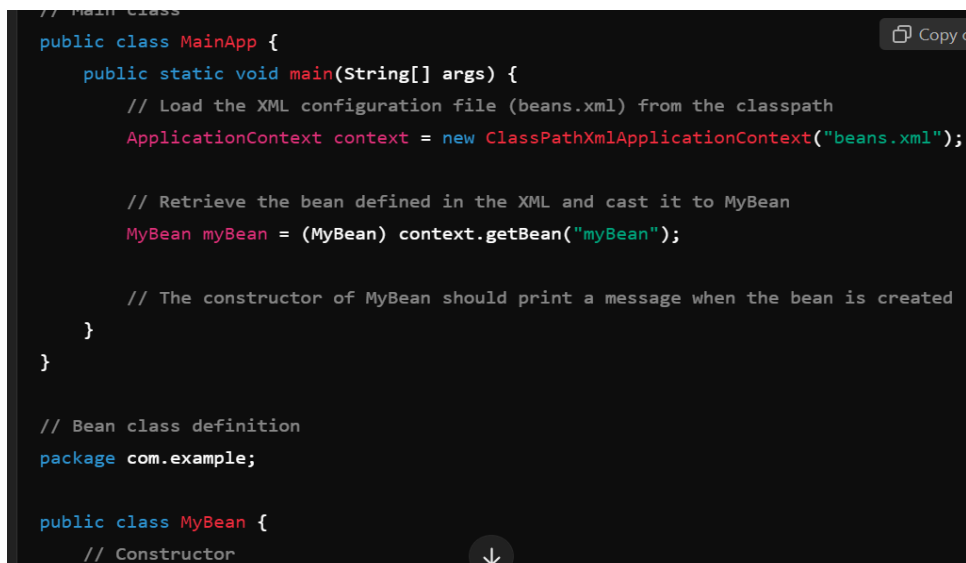
```
1 package com.ex.main;
2
3 import org.springframework.beans.factory.DisposableBean;
4 import org.springframework.beans.factory.InitializingBean;
5 import org.springframework.stereotype.Component;
6
7 import jakarta.annotation.PostConstruct;
8 import jakarta.annotation.PreDestroy;
9
10 @Component
11 public class Battery implements InitializingBean, DisposableBean {
12     public Battery() {
13         System.out.println("battery is created ");
14     }
15
16     public void afterPropertiesSet() {
17         System.out.println("initializing battery");
18     }
19
20     public void destroy() {
21         System.out.println("distroy");
22     }
23 }
24
```

What is Bean Factory ,Have you used XMLBeanFactory?

Ans→

1. This is root interface for Accessing a Spring bean Container
2. ClassPathXmlApplicationContext loads the bean definitions from an XML file located in the class path

XMLBeanFactory was primarily used in older versions of Spring and is not recommended for use today.



```
// Main class
public class MainApp {
    public static void main(String[] args) {
        // Load the XML configuration file (beans.xml) from the classpath
        ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");

        // Retrieve the bean defined in the XML and cast it to MyBean
        MyBean myBean = (MyBean) context.getBean("myBean");

        // The constructor of MyBean should print a message when the bean is created
    }
}

// Bean class definition
package com.example;

public class MyBean {
    // Constructor
}
```

What the difference between Bean-factory AND applicationContext in Spring?

Aspect	BeanFactory	ApplicationContext
Basic Functionality	Basic container for bean instantiation.	Advanced container with extra features.
Event Handling	Not supported.	Supports event handling.
Annotation Support	Limited or no support.	Full support for annotations.
Use Case	Suitable for lightweight apps.	Used for complex, enterprise-level apps.

Difference between the setter and constructor in Spring?

In constructor injection is imp to remember the type and order of parameter

Aspect	Constructor Injection	Setter Injection
Order and Type of Parameters	Important to remember the type and order of parameters.	No need to remember the order of parameters.
Dependency Type	Used for mandatory dependencies.	Used for optional dependencies.
Immutability	Promotes immutability as dependencies are injected at creation.	Allows flexibility to change dependencies after object creation.

Note- Constructor injection is typically used for mandatory dependencies, which must be provided at the time of object creation. If a required dependency is not provided, the Spring container will throw an error.

What are different modules in Spring?

Spring has Seven core module

1. The core container module
2. Application Context module
3. AOP module (Aspect Oriented Programming)
4. Jdbc abstraction and dao module
5. Orm Module
6. Test

Difference between @Autowired and @inject

1. @Autowired is used in Spring to automatically inject dependencies.
2. @Inject is a standard Java annotation (JSR-330) for dependency injection, similar to @Autowired, but not specific to Spring.

What is difference between @Bean and @component in Spring?

➔ In the @Bean, we manually define the bean inside a configuration class.

In the @Component, Spring automatically detects the bean by ComponentScan, so we don't need to define it explicitly.

What is Auto wiring in Spring? What are auto wiring modes?

Ans➔

Inject the beans automatically. We don't need to write explicitly

- 1) **no** - this is the default mode, it means autowiring is not enabled.
- 2) **byName** - injects the bean based on the property name. It uses setter method.
- 3) **byType** - injects the bean based on the property type. It uses setter method.
- 4) **constructor** - It injects the bean using constructor

```
<!-- Bean definitions here -->
<bean name="network" class="com.interview.beans.Network"></bean>

<bean name="phone" class="com.interview.beans.Phone" autowire="no">
--   <constructor-arg value="12"></constructor-arg>-->
--   <constructor-arg value="121.23"></constructor-arg>-->
</bean>
```

```
<bean name="phone" class="com.interview.beans.Phone" autowire="byName">
--   <constructor-arg value="12"></constructor-arg>-->
```

What Are different bean Scopes in Spring?