Spring boot : Bean and its Lifecycle

What is bean?

In a Simple term, Bean is a Java Object, which is managed by Spring container (also known as IOC Container).

IOC container -> contains all the beans which get created and also manage them.

How to create a Bean?

1. Using ***@Component*** Annotation:

- ***@Component*** annotation follows "convention over configuration" approach.
- Means Spring boot will try to auto configure based on conventions reducing the need for explicit configuration.
- @Controller, @Service etc. all are internally tells Spring to create bean and manage it.

So here, Spring boot will internally call **new User()** to create an instance of this class.

```java
@Component
public class User {

    String username;
    String email;

    public String getUsername() { return username; }

    public void setUsername(String username) { this.username = username; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }
}
```

But what will happen to this now:

```java
@Component
public class User {
    String username;
    String email;

    public User(String username, String email){
        this.username =username;
        this.email = email;
    }

    public String getUsername() { return username; }

    public void setUsername(String username) { this.username = username; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }
}
```

```
***************************
APPLICATION FAILED TO START
***************************
```

Why? Because Spring boot does not know what to pass in these Constructor parameters.

So what to do?

@Bean comes into the picture, where we provide the configuration details and tells Spring boot to use it while creating a Bean.

```java
public class User {
    String username;
    String email;

    public User(String username, String email){
        this.username =username;
        this.email = email;
    }

    public String getUsername() { return username; }

    public void setUsername(String username) { this.username = username; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }
}
```

```java
@Configuration
public class AppConfig {

    @Bean
    public User createUserBean(){
        return new User( username: "defaultusername",  email: "defaultemail");
    }
}
```

If we add 2 times in Configuration file, Spring will create 2 beans of it.

```java
@Configuration
public class AppConfig {

    @Bean
    public User createUserBean(){
        return new User( username: "defaultusername",  email: "defaultemail");
    }

    @Bean
    public User createAnotherUserBean(){
        return new User( username: "anotherUsername",  email: "anotheremail");
    }
}
```

Now, we know what is bean and how its getting created. But few Questions comes to our minds:

-> How Spring boot find these Beans?
-> At what time, these beans get created?

How Spring boot find these Beans?

1. Using @ComponentScan annotation, it will scan the specified package and sub-package for classes annotated with @Component, @Service etc.

```java
@SpringBootApplication
@ComponentScan(basePackages = "com.conceptandcoding.learningspringboot")
public class SpringbootApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringbootApplication.class, args);
    }
}
```

2. Through Explicit defining of bean via @Bean annotation in @Configuration class.

```java
@Configuration
public class AppConfig {

    @Bean
    public User createUserBean(){
        return new User( username: "defaultusername",  email: "defaultemail");
    }
}
```

At what time, these beans get created

Eagerness

-> Some Beans get created, When we start up an application.

-> For ex: Beans with Singleton Scope are Eagerly initialized.

Lazy

-> Some Beans get created Lazily, means when they actually needed.

-> For ex: Beans with Scope like Prototype etc. are Lazily initialized.

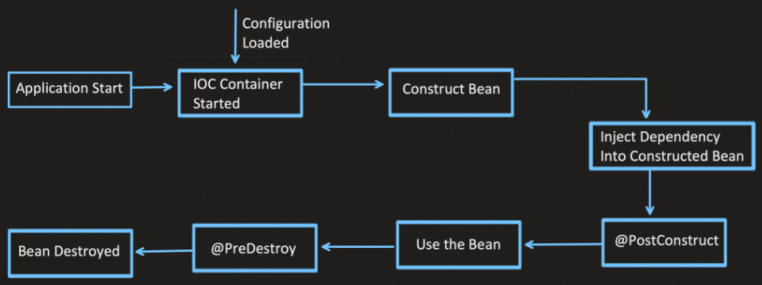
Or

Beans with @Lazy annotation.

```java
@Lazy
@Component
public class Order {

    public Order(){
        System.out.println("initializing Order");
    }
}
```

Lifecycle of Bean:



Step1:

-> During Application Startup, Spring boot invokes IOC Container

(ApplicationContext provides the implementation of IOC container)

-> IOC Container, make use of Configuration and @ComponentScan to look out for the Classes for which beans need to be created.

Invoking IOC Container

```
main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.19]
main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 419 ms
main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path ''
main] c.c.l.SpringbootApplication              : Started SpringbootApplication in 0.771 seconds (process running for 0.946)
```

Step2: Construct the Beans

```java
@Component
public class User {

    public User(){
        System.out.println("initializing user");
    }
}
```

```
2024-04-09T00:47:43.681+05:30  INFO 60692 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2024-04-09T00:47:43.687+05:30  INFO 60692 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-04-09T00:47:43.687+05:30  INFO 60692 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.19]
2024-04-09T00:47:43.711+05:30  INFO 60692 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2024-04-09T00:47:43.711+05:30  INFO 60692 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 412 ms
initializing user
2024-04-09T00:47:43.859+05:30  INFO 60692 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path ''
2024-04-09T00:47:43.864+05:30  INFO 60692 --- [           main] c.c.l.SpringbootApplication              : Started SpringbootApplication in 0.756 seconds (process running for 0.927)
```

Step3:

-> Inject the Dependency into the Constructed Bean.
-> @Autowired, first look for a bean of the required type.
-> If bean found, Spring will inject it. Different ways of Injection:
- Constructor Injection
- Setter Injection
- Field Injection

******We will see each injection and which one to use in next part ********

-> If bean is not found, Spring will create one and then inject it.

```java
@Component
public class User {

    @Autowired
    Order order;

    public User(){
        System.out.println("initializing user");
    }
}
```

```java
@Lazy
@Component
public class Order {

    public Order(){
        System.out.println("Lazy: initializing Order");
    }
}
```

```
2024-04-09T00:53:11.751+05:30  INFO 60872 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2024-04-09T00:53:11.757+05:30  INFO 60872 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-04-09T00:53:11.758+05:30  INFO 60872 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.19]
2024-04-09T00:53:11.782+05:30  INFO 60872 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2024-04-09T00:53:11.782+05:30  INFO 60872 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 416 ms
initializing user
Lazy: initializing Order
2024-04-09T00:53:11.931+05:30  INFO 60872 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path ''
2024-04-09T00:53:11.936+05:30  INFO 60872 --- [           main] c.c.l.SpringbootApplication              : Started SpringbootApplication in 0.76 seconds (process running for 0.929)
```

Step4: Perform any task before Bean to be used in application.

```java
@Component
public class User {

    @Autowired
    Order order;

    @PostConstruct
    public void initialize(){
        System.out.println("Bean has been constructed and dependencies have been injected");
    }

    public User(){
        System.out.println("initializing user");
    }
}
```

```
2024-04-09T01:01:14.147+05:30  INFO 61161 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2024-04-09T01:01:14.152+05:30  INFO 61161 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-04-09T01:01:14.153+05:30  INFO 61161 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.19]
2024-04-09T01:01:14.175+05:30  INFO 61161 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2024-04-09T01:01:14.176+05:30  INFO 61161 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 417 ms
initializing user
Lazy: initializing Order
Bean has been constructed and dependencies have been injected
2024-04-09T01:01:14.318+05:30  INFO 61161 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path ''
2024-04-09T01:01:14.322+05:30  INFO 61161 --- [           main] c.c.l.SpringbootApplication              : Started SpringbootApplication in 0.754 seconds (process running for 0.923)
```

Step5: Use the Bean in your application.

Means, we are using the bean (or Object) to invoke some methods to perform some business logic

Step6: Perform any task before Bean is getting destroyed.

```java
@SpringBootApplication
public class SpringbootApplication {

    public static void main(String[] args) {
        ConfigurableApplicationContext context = SpringApplication.run(SpringbootApplication.class, args);
        context.close();
    }
}
```

```java
@Component
public class User {

    @PostConstruct
    public void initialize() { System.out.println("Post Construct initiated"); }

    @PreDestroy
    public void preDestroy() { System.out.println("Bean is about to destory, in PreDestoryMethod"); }

    public User() { System.out.println("initializing user"); }
}
```

```
2024-04-09T18:07:37.600+05:30  INFO 16725 --- [           main] c.c.l.SpringbootApplication              : No active profile set, falling back to 1 default profile: "default"
2024-04-09T18:07:38.016+05:30  INFO 16725 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2024-04-09T18:07:38.024+05:30  INFO 16725 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-04-09T18:07:38.024+05:30  INFO 16725 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.19]
2024-04-09T18:07:38.048+05:30  INFO 16725 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2024-04-09T18:07:38.048+05:30  INFO 16725 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 426 ms
initializing user
Post Construct initiated
2024-04-09T18:07:38.197+05:30  INFO 16725 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path ''
2024-04-09T18:07:38.201+05:30  INFO 16725 --- [           main] c.c.l.SpringbootApplication              : Started SpringbootApplication in 0.764 seconds (process running for 0.931)
2024-04-09T18:07:49.215+05:30  WARN 16725 --- [           main] org.apache.tomcat.util.net.Acceptor      : The acceptor thread [http-nio-8080-Acceptor] did not stop cleanly
Bean is about to destory, in PreDestoryMethod
```