POST – CONSTRUCT AND PRE – DESTROY

PostConstruct:

In Java Spring Framework, the **@PostConstruct** annotation is used to indicate a method that should be called after a bean has been instantiated and its dependencies have been injected.

The method annotated with **@PostConstruct** will be called immediately after the bean has been created, but before it is returned to the caller of the bean factory method. This method is typically used to perform any initialization logic that needs to be done on the bean before it can be used.

It's important to note that the method annotated with **@PostConstruct** must be public and have no arguments.

PreDestroy:

In Java Spring Framework, the @PreDestroy annotation is used to indicate a method that should be called just before a bean is destroyed by the container.

The method annotated with @PreDestroy will be called immediately before the bean is destroyed. This method is typically used to perform any cleanup logic that needs to be done on the bean before it is destroyed.

It's important to note that the method annotated with **@PreDestroy** must be public and have no arguments.

The **@PreDestroy** annotation is useful for performing any cleanup or resource release logic that needs to be done before the bean is destroyed by the container.

Example: @PostConstruct

```
package com.naveen.learnspringframework.PostConstructPreDestroy;

import
org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.stereotype.Component;

import jakarta.annotation.PostConstruct;

@Component
class Server{
    private Database database;

    public Server(Database database) {
        super();
```

```
        this.database = database;
        System.out.println("All dependencies are wired and ready");
    }

    @PostConstruct
    public void initialize() {
        database.connect();
    }
}

@Component
class Database{
    public void connect() {
        System.out.println("Requesting for database connection");
        System.out.println("CONNECTED");
    }
}
@Configuration
@ComponentScan
public class PostConstructPreDestroyApp {

    public static void main(String[] args) {

        try(var context =
                new AnnotationConfigApplicationContext
                    (PostConstructPreDestroyApp.class)){
        }
    }
}
```

OUTPUT:

```
05:53:19.926 [main] DEBUG org.springframework.beans.factory.suppc
05:53:19.931 [main] DEBUG org.springframework.beans.factory.suppc
05:53:19.947 [main] DEBUG org.springframework.beans.factory.suppc
05:53:19.960 [main] DEBUG org.springframework.beans.factory.suppc
05:53:19.961 [main] DEBUG org.springframework.beans.factory.suppc
05:53:19.999 [main] DEBUG org.springframework.beans.factory.suppc
All dependencies are wired and ready
Requesting for database connection
CONNECTED
05:53:20.100 [main] DEBUG org.springframework.context.annotation.
```

Example: @PreDestroy

```
package com.naveen.learnspringframework.PostConstructPreDestroy;

import java.util.Arrays;

import
org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.stereotype.Component;

import jakarta.annotation.PostConstruct;
import jakarta.annotation.PreDestroy;

@Component
class Server{
    private Database database;

    public Server(Database database) {
        super();
        this.database = database;
        System.out.println("All dependencies are wired and ready");
    }

    @PostConstruct
    public void initialize() {
        database.connect();
    }

    @PreDestroy
    public void cleanUp() {
        database.close();
    }
}

@Component
class Database{
    public void connect() {
        System.out.println("Requesting for database connection");
        System.out.println("CONNECTED");
    }

    public void close() {
        System.out.println("Requesting for close connection.");
        System.out.println("CONNECTION - ENDED");
    }
}
@Configuration
@ComponentScan
public class PostConstructPreDestroyApp {

    public static void main(String[] args) {
```

```
        try(var context =
                new AnnotationConfigApplicationContext
                    (PostConstructPreDestroyApp.class)){
            Arrays.stream(context.getBeanDefinitionNames())
                .forEach(System.out::println);
        }
    }
}
```

OUTPUT:

```
05:59:09.423 [main] DEBUG org.springframework.beans.factory.suppo
All dependencies are wired and ready
Requesting for database connection
CONNECTED
org.springframework.context.annotation.internalConfigurationAnnot
org.springframework.context.annotation.internalAutowiredAnnotatic
org.springframework.context.annotation.internalCommonAnnotationPr
org.springframework.context.event.internalEventListenerProcessor
org.springframework.context.event.internalEventListenerFactory
postConstructPreDestroyApp
database
server
05:59:09.525 [main] DEBUG org.springframework.context.annotation.
Requesting for close connection.
CONNECTION - ENDED
```