Destroy Method And Scope Types

Contents

Overview 2
Why destroy() Is Called in Singleton Scope 3
Why destroy() Is Not Called in Prototype Scope 4

Overview

This document is intended to provide why destroy method is not called automatically for prototype-scoped beans and called for singleton-scoped beans in java.

Why destroy() Is Called in Singleton Scope

In **singleton scope**, only one instance of the bean is created and managed by the Spring container for the entire application lifecycle.

Key Reasons:

- **Spring manages the full lifecycle** of singleton beans:
  - Bean creation
  - Dependency injection
  - Initialization (`@PostConstruct`, `init-method`)
  - **Destruction (`@PreDestroy`, `destroy-method`)**

- When the **application context is closed** (e.g., calling `context.close()`), Spring:
  - Notifies all singleton beans.
  - Invokes their `destroy()` methods automatically (e.g., via `@PreDestroy` or implementing `DisposableBean`).

Example:

```
@Scope("singleton")
@Component
public class MySingletonBean {
   @PreDestroy
   public void destroy() {
      System.out.println("Singleton bean destroy method called");
   }
}
```

When the context is closed, the above message will be printed because Spring takes care of calling the destroy method.

Why destroy() Is Not Called in Prototype Scope

In **prototype scope**, a new bean instance is created **every time** it is requested from the container.

Key Reasons:

- Spring only manages the **creation and dependency injection** of prototype beans.
- **Destruction is not managed** by Spring.
  - Spring has **no knowledge of when the bean will be discarded** or is no longer used.
  - Therefore, it **does not automatically call `destroy()`**.

You must **manually invoke any cleanup or destruction logic** for prototype-scoped beans.

Example:

```
@Scope("prototype")
@Component
public class MyPrototypeBean {
   @PreDestroy
   public void destroy() {
      System.out.println("Prototype bean destroy method called");
   }
}
```

Even if the context is closed, the above method will **not** be called automatically. You would need to call `destroy()` yourself if cleanup is required.