



CS544 EA
Spring

Startup: Life Cycle Methods

Life Cycle Methods

- Spring can call a method after creating a bean

```
<bean id="customerService" class="cs544.spring26.startup.lifecycle.CustomerService" init-method="start" />
```

XML config uses:
init-method to specify
a method name

```
package cs544.spring26.startup.lifecycle;
...
@Service
public class CustomerService {
    public CustomerService() {
        System.out.println("constructor");
    }
    @PostConstruct
    public void start() {
        System.out.println("start");
    }
    public void hello() {
        System.out.println("hello");
    }
}
```

When using annotations
place @PostConstruct
on the method

Demonstration of @PostConstruct

- Order of execution:
 - Constructor (including constructor injection)
 - Setter injection
 - Init method (@PostConstruct)

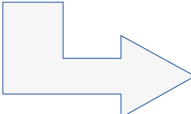
```
package cs544.spring26.startup.lifecycle;

import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {
    public static void main(String[] args) {
        ConfigurableApplicationContext context;
        context = new ClassPathXmlApplicationContext("cs544/spring26/startup/lifecycle/springconfi
        //context = new AnnotationConfigApplicationContext(Config.class);

        CustomerService cs = context.getBean("customerService", CustomerService.class);
        cs.hello();

        context.close();
    }
}
```



constructor
start
hello

Destroy Method

- Spring can call a method before destroying a bean

```
<bean id="customerService" class="cs544.spring26.startup.lifecycle.CustomerService" destroy-method="stop" />
```

```
package cs544.spring26.startup.lifecycle;
...
@Service
public class CustomerService {
    public CustomerService() {
        System.out.println("constructor");
    }
    public void hello() {
        System.out.println("hello");
    }
    @PreDestroy
    public void stop() {
        System.out.println("stop");
    }
}
```

XML config uses:
destroy-method to specify
a method name

When using annotations
place @PreDestroy
on the method

Demonstration of @PreDestroy

- Destroy methods are called when the Application Context is closed
 - The close() method is only available on ConfigurableApplicationContext

```
package cs544.spring26.startup.lifecycle;

import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {
    public static void main(String[] args) {
        ConfigurableApplicationContext context;
        context = new ClassPathXmlApplicationContext("cs544/spring26/startup/lifecycle/springconfig.xml");
        //context = new AnnotationConfigApplicationContext(Config.class);

        CustomerService cs = context.getBean("customerService", CustomerService.class);
        cs.hello();

        context.close();
    }
}
```

Has to be a
ConfigurableApplicationContext

Destroy methods activates
on context.close()

constructor
hello
stop

Prototype and @PreDestroy

- Spring never calls destroy methods on prototype beans
 - Spring keeps references to all non-prototype beans
 - Uses these references to call destroy methods
- Spring does **not keep references** to prototype objects
 - Because there **could be too many**
 - Therefore cannot call destroy on them