



CS544 EA
Spring
Startup

Startup Order

- By default Spring beans are **eager singletons** created in the order that they are found
 - Eager can be made lazy
 - Singleton can be made 'prototype'
 - If a bean is needed (for DI) it is created earlier

Call Order

- When a bean is created:
 - Constructor is called (possibly with injection)
 - Setter injection happens
 - An init (PostConstruct) method is called if indicated
- When the context is closed:
 - Destroy (PreDestroy) methods are called

Singleton

- By default only one object is made per bean

```
package cs544.spring25.startup.singleton;

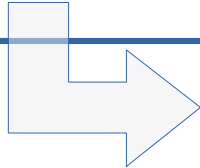
import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {
    public static void main(String[] args) {
        ConfigurableApplicationContext context;
        //context = new ClassPathXmlApplicationContext("cs544/spring25/startup/singleton/springconfig.xml");
        context = new AnnotationConfigApplicationContext(Config.class);

        CustomerService customerService1 = context.getBean("customerService", CustomerService.class);
        CustomerService customerService2 = context.getBean("customerService", CustomerService.class);
        System.out.println(customerService1);
        System.out.println(customerService2);

        context.close();
    }
}
```

The exact same object



```
cs544.spring25.startup.singleton.CustomerService@65b3f4a4
cs544.spring25.startup.singleton.CustomerService@65b3f4a4
```

Prototype

- Instead of having only one object per bean
 - You can tell Spring to make a new object every time the bean is needed (for injection or getBean)

With Annotations:

```
package cs544.spring29.startup.proto;  
  
...  
  
@Service  
@Scope("prototype")  
public class CustomerService {  
    ... // same content as eager  
}
```

With Java Config:

```
package cs544.spring29.startup.proto;  
  
...  
  
@Configuration  
public class Config {  
    @Bean  
    @Scope("prototype")  
    public CustomerService customerService() {  
        return new CustomerService();  
    }  
}
```

With XML:

```
<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans" ...  
    <bean id="customerService" scope="prototype"  
        class="cs544.spring29.startup.eager.CustomerService"  
        init-method="start" destroy-method="stop"/>  
</beans>
```

Demonstration of Prototype

```
package cs544.spring29.startup.proto;

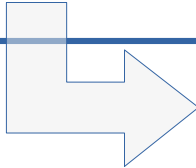
import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {
    public static void main(String[] args) {
        ConfigurableApplicationContext context;
        //context = new ClassPathXmlApplicationContext("cs544/spring25/startup/singleton/springconfig.xml");
        context = new AnnotationConfigApplicationContext(Config.class);

        CustomerService customerService1 = context.getBean("customerService", CustomerService.class);
        CustomerService customerService2 = context.getBean("customerService", CustomerService.class);
        System.out.println(customerService1);
        System.out.println(customerService2);

        context.close();
    }
}
```

Two different objects



```
cs544.spring29.startup.proto.CustomerService@708f5957
cs544.spring29.startup.proto.CustomerService@68999068
```