Bean Scopes

- Bean Instance created by Spring Container can be in one of the following Scopes(Updated from Spring5).
    1) singleton
    2) prototype
    3) request
    4) session
    5) application
    6) websocket

Scope	Description
singleton	◆ When bean scope is singleton then only one instance will be created for that bean and the same instance will be returned when you call getBean() method. ◆ singleton is the default scope in the ApplicationContext container. ◆ When scope is single-ton then default loading type is aggressive loading.
prototype	◆ When bean scope is prototype then every time a new instance will be created for that bean when you call getBean() method. ◆ When scope is prototype then default loading type is lazy loading.
request	◆ Scopes a single bean definition to the lifecycle of a single HTTP request. ◆ Single Bean instance will be created per HTTP Request ◆ Only valid in the context of a web-aware Spring ApplicationContext.
session	◆ Scopes a single bean definition to the lifecycle of an HTTP Session. ◆ Single Bean instance will be created per HTTP Session ◆ Only valid in the context of a web-aware Spring ApplicationContext.
application	◆ Scopes a single bean definition to the lifecycle of a ServletContext. ◆ Single Bean instance will be created per ServletContext. ◆ Only valid in the context of a web-aware Spring ApplicationContext.
websocket	◆ Scopes a single bean definition to the lifecycle of a WebSocket. ◆ Single Bean instance will be created per WebSocket. ◆ Only valid in the context of a web-aware Spring ApplicationContext.

- Usage:

```
@Scope(value="singleton")
@Scope(value=" prototype ")
@Scope("singleton")
@Scope("prototype")
```

Bean Scope Example with Java Configuration

Lab2: Files required

1. Lab2.java	2. Hello.java
3. Hai.java	4. JLCAppConfig.java

1. Lab2.java

```
package com.coursecube.spring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab2 {
public static void main(String[] args) {

ApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig.class);
System.out.println("---------Now Spring Container is Ready-----");

Hello hello1=(Hello)ctx.getBean("hello");
Hello hello2=(Hello)ctx.getBean("hello");
System.out.println(hello1==hello2);

Hai hai1=(Hai)ctx.getBean("hai");
Hai hai2=(Hai)ctx.getBean("hai");
System.out.println(hai1==hai2);

Hello hello=(Hello)ctx.getBean("hello");
hello.showHello();

Hai hai=(Hai)ctx.getBean("hai");
hai.showHai();
}
}
```

2. Hello.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {

static {
```

```
System.out.println("Hello - S.B");
}
public Hello() {
System.out.println("Hello - D.C");
}
public void showHello() {
        System.out.println("Hello-showHello()");
}
}
```

3. Hai.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hai {

static {
System.out.println("Hai - S.B");
}
public Hai() {
System.out.println("Hai - D.C");
}
public void showHai() {
        System.out.println("Hai-showHai()");
}
}
```

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Scope;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

        @Bean("hello")
        @Scope("singleton")
        public Hello createHello() {
                System.out.println("-------createHello() ------ called");
                return new Hello();
        }
        @Bean("hai")
        @Scope("prototype")
        public Hai createHai() {
                System.out.println("-------createHai() ------ called");
                return new Hai();
        }

}
```

Bean Loading Types

- Bean configured in the Spring Configuration Class can be loaded in two ways.
    1) Aggressive loading or Eager loading
    2) Lazy loading.

- Usage:

```
@Lazy(value="true")
@Lazy(value="true")
@Lazy("true")
@Lazy("false")
```

1) Aggressive loading or Eager loading

- In the case of aggressive loading, all the Beans will be loaded, instantiated and initialized by the container at the container start-up.

Ex:

```
@Bean
@Lazy(false)
public Hello hello() {
        return new Hello();
}
```

2) Lazy loading.

- In the case of lazy loading, all the Beans will be loaded, instantiated and initialized when you or container try to use them by calling getBean() method.

Ex:

```
@Bean
@Lazy(true)
public Hello hello() {
        return new Hello();
}
```

Bean Loading Types Example with Java Configuration:

Lab3: Files required

1. Lab3.java	New One
2. Hello.java	Same as Lab2
3. Hai.java	Same as Lab2
4. JLCAppConfig.java	New One

1. Lab3.java

```
package com.coursecube.spring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab3 {
public static void main(String[] args) {

ApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig.class);
System.out.println("---------Now Spring Container is Ready-----");

Hello hello=(Hello)ctx.getBean("hello");
hello.showHello();

Hai hai=(Hai)ctx.getBean("hai");
hai.showHai();

}
}
```

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Lazy;
import org.springframework.context.annotation.Scope;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
```

```
@Configuration
public class JLCAppConfig {

	@Bean("hello")
	@Scope("singleton")
	@Lazy(true)
	public Hello createHello() {
		System.out.println("-------createHello() ------ called");
		return new Hello();
	}

	@Bean("hai")
	@Scope("prototype")
	@Lazy(true)
	public Hai createHai() {
		System.out.println("-------createHai() ------ called");
		return new Hai();
	}
}
```