Document Prepared by

Vineet Semwal

vineetsemwal82@gmail.com

Spring

Spring is a container in the sense that it contains and manages the life cycle and configuration of application objects.
The core of Spring's DI container is the BeanFactory(factory of beans). A bean factory is responsible for managing components and their dependencies

Dependency Injection

Done in 2 steps

1) **Component scanning**—Spring automatically discovers beans to be created in the application context

(Use @Component at top of class if it is created by you , in case it is not created by u and you dont't have source , Provide the bean object in the configuration class and use @Bean at the top of method, example later in doc)

2) **Autowiring**—Spring automatically satisfies bean dependencies
(by looking at @Autowired)

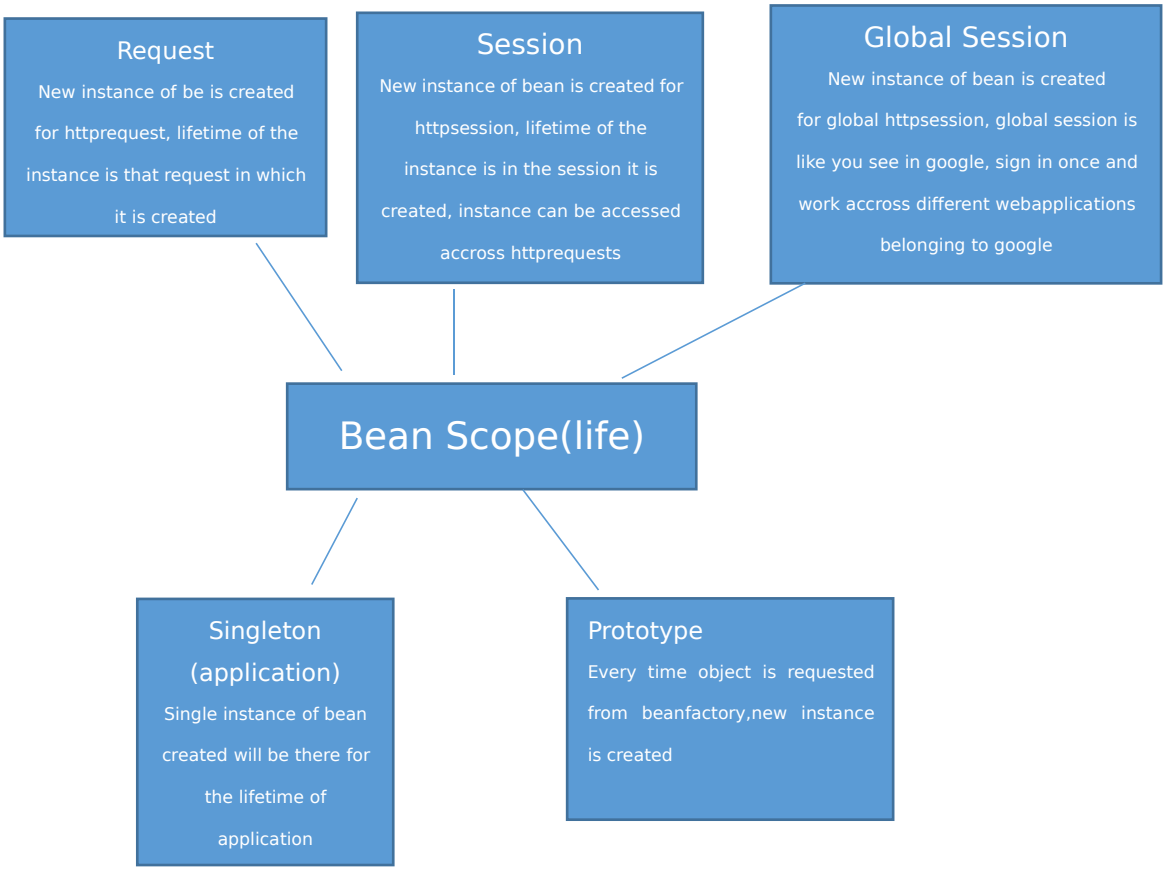
When Bean class is created by you

```
@Component
public class Circle implements Shape {
    private int radius;
    public int getRadius(){
        return radius;
    }
    public void setRadius(int radius){
        this.radius=radius;
    }
}
```

When Bean class is created by someone else and you don’t have source ,
Use @Bean

```
/*
 @Configuration used to mention that it is a configuration class and will have methods that will provide beans to spring container
*/
@Configuration
public class JavaConfig {
     /**
      * @Bean informs spring that this method produces a bean that will be managed by the Spring container
      */
    @Bean
    public   Circle circle(){
          Circle circle=new Circle();
          circle.setRadius(1);
          return circle;
     }
}
```

@ComponentScan : It is used to inform spring the package where spring will scan for finding beans
Foreg. @ComponentScan("com.mycompany") ,used on top of Configuration class
Scan in package "com.mycompany" for the beans



BeanFactory(Interface): container which instantiates and populate the dependencies, these dependencies are mentioned in configuration in xml or annotation

Popular Implementations

XmlBeanFactory(when beans configurations are mentioned in xml)

```
/*
When xml is in filesystem
*/
Resource resource = new FileSystemResource("beans.xml");
XmlBeanFactory factory = new XmlBeanFactory(resource);
```

In case when xml file is there in project classpath

```
Resource resource = new ClassPathResource("beans.xml");
XmlBeanFactory factory = new XmlBeanFactory(resource);
```

ApplicationContext(Interface)

It is also a bean factory with more features like capability to populate bean fields from the properties file etc.

Popular Implementations

FileSystemXmlApplicationContext

```
/**
Xml kept in file system
**/
ApplicationContext context = new FileSystemXmlApplicationContext("fullpathofbeans.xml");
Canvas canvas=context.getBean(Canvas.class);
```

ClasspathApplicationContext

```
/**
When Configurations mentioned in xml
**/
ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
Canvas canvas=context.getBean(Canvas.class);
```

AnnotationConfigApplicationContext

```
AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext();
context.register(JavaConfig.class);// configuration class
Canvas canvas=context.getBean(Canvas.class);
```

Populating from properties file in fields

```
@Component
public class Circle implements Shape {
	@Value("${radius}")
	private int radius;
	public int getRadius(){
		return radius;
	}
	public void setRadius(int radius){
		this.radius=radius;
	}
	@Value("${color}") // value will be populated by spring from properties file automatically
	private String color;
	public String getColor(){
	    return color;
	}
	public void setColor(String color){
		this.color=color;
	}
	@Override
	public double area(){
		return 3.14*radius*radius;
	}
}
@Configuration
@ComponentScan("com.mycompany.app")
// informing spring where properties file is kept
@PropertySource("classpath:shape.properties")
public class JavaConfig {
}
```