Table of Contents

Java Code:

```
public SpringFramework() {
    BeanFactory factory = new XmlBeanFactory(new FileSystemResou
rce("Spring.xml"));
    Student student = (Student) factory.getBean("triangle");
    student.draw();
}
```

XML Code:

```
<bean id="triangle" class="spring.framework.Student"></bean>
```

If you use bean factory then the xml file need to have in the root directory of the project.

Java Code:

```
public SpringFramework() {
    ApplicationContext context = new ClassPathXmlApplicationCont
ext("Spring.xml");
    Student student = (Student) context.getBean("triangle");
    student.draw();
    System.out.println(student.toString());
}
```

XML Code:

```
<bean id="triangle" class="spring.framework.Student">
        <property name="id" value="1"/>
        <property name="name" value="Mijan"/>
</bean>
```

If you use AppliacationContext then the xml file need to have in the class path or in the default package. Property tag use set method for initialize.

Java Code:

```
public SpringFramework() {
    ApplicationContext context = new ClassPathXmlApplicationCont
ext("Spring.xml");
    Student student = (Student) context.getBean("triangle");
    student.draw();
    System.out.println(student.toString());
}
```

XML Code:

```
<bean id="triangle" class="spring.framework.Student">
        <constructor-arg index="0" value="1"/>
        <constructor-arg index="1" value="Mijan"/>
        <constructor-arg type="int" value="1"/>
        <constructor-arg type="java.lang.String" value="1"/>
</bean>
```

We can use index or type any property we want.

Java Code:

```
public SpringFramework() {
    ApplicationContext context = new ClassPathXmlApplicationCont
ext("Spring.xml");
    Triangle triangle = (Triangle) context.getBean("triangle");
    System.out.println(triangle.toString());
}
```

Java Code:

```
public class Triangle {
    private Point A;
    private Point B;
    private Point C;
}
```

XML Code:

```
<bean id="triangle" class="spring.framework.Triangle">
    <property name="A" ref="point1" />
    <property name="B" ref="point2" />
    <property name="C" ref="point3" />
</bean>

<bean id="point1" class="spring.framework.Point">
    <property name="x" value="1" />
    <property name="y" value="2" />
</bean>

<bean id="point2" class="spring.framework.Point">
    <property name="x" value="3" />
    <property name="y" value="4" />
</bean>

<bean id="point3" class="spring.framework.Point">
    <property name="x" value="5" />
    <property name="y" value="6" />
</bean>
```

Java Code:

```
public SpringFramework() {
    ApplicationContext context = new ClassPathXmlApplicationCont
ext("Spring.xml");
    Triangle triangle = (Triangle) context.getBean("triangle");
    System.out.println(triangle.toString());
}
```

Java Code:

```
public class Triangle {
    private Point A;
    private Point B;
    private Point C;
}
```

XML Code:

```
<bean id="triangle" class="spring.framework.Triangle" name="tria
ngle2">
    <property name="A" >
        <bean class="spring.framework.Point">
            <property name="x" value="1" />
            <property name="y" value="2" />
        </bean>
    </property>
    <property name="B"  >
            <bean  class="spring.framework.Point">
                <property name="x" value="3" />
                <property name="y" value="4" />
            </bean>
    </property>
    <property name="C" >
        <bean class="spring.framework.Point">
            <property name="x" value="5" />
            <property name="y" value="6" />
        </bean>
    </property>
</bean>

<alias name="triangle" alias="triangle1" />
```

Alias gives you freedom to change the name of the bean into another name and you can get the same object using that name.

Inner bean is put a bean into the property tag

Java Code:

```
public SpringFramework() {
    ApplicationContext context = new ClassPathXmlApplicationCont
ext("Spring.xml");
    Triangle triangle = (Triangle) context.getBean("triangle");
    System.out.println(triangle.toString());
}
```

Java Code:

```
public class Triangle {
    private List<Point> points = new ArrayList<>();
}
```

XML Code:

```
<bean id="triangle" class="spring.framework.Triangle">
    <property name="points">
        <list>
            <ref bean="point1"/>
            <ref bean="point2"/>
            <ref bean="point3"/>
        </list>
    </property>
</bean>

<bean id="point1" class="spring.framework.Point">
    <property name="x" value="1" />
    <property name="y" value="2" />
</bean>

<bean id="point2" class="spring.framework.Point">
    <property name="x" value="3" />
    <property name="y" value="4" />
</bean>

<bean id="point3" class="spring.framework.Point">
    <property name="x" value="5" />
    <property name="y" value="6" />
</bean>
```

Java Code:

```
public SpringFramework() {
    ApplicationContext context = new ClassPathXmlApplicationCont
ext("Spring.xml");
    Triangle triangle = (Triangle) context.getBean("triangle");
    System.out.println(triangle.toString());
}
```

Java Code:

```
public class Triangle {
    private Point pointA;
    private Point pointB;
    private Point pointC;
}
```

XML Code:

```
<bean id="triangle" class="spring.framework.Triangle" autowire="
byName">

</bean>

<bean id="pointA" class="spring.framework.Point">
    <property name="x" value="1" />
    <property name="y" value="2" />
</bean>

<bean id="pointB" class="spring.framework.Point">
    <property name="x" value="3" />
    <property name="y" value="4" />
</bean>

<bean id="pointC" class="spring.framework.Point">
    <property name="x" value="5" />
    <property name="y" value="6" />
</bean>
```

There is different type of autowirng in spring you can check that.

If you are using autowrie byName then you have to have the same name of the other bean you want to autowire.

XML Code:

```
<bean id="triangle" class="spring.framework.Triangle" scope="sin
gleton">

</bean>
```

There is different type of scope in Spring Framework you can check out that.

Java Code:

```
public class Triangle implements ApplicationContextAware, BeanNa
meAware{
    private Point pointA;
    private Point pointB;
    private Point pointC;
    private ApplicationContext context = null;

    public Triangle() {
    }

    public Triangle(Point pointA, Point pointB, Point pointC) {
        this.pointA = pointA;
        this.pointB = pointB;
        this.pointC = pointC;
    }

    public Point getPointA() {
        return pointA;
    }

    public void setPointA(Point pointA) {
        this.pointA = pointA;
    }

    public Point getPointB() {
        return pointB;
    }

    public void setPointB(Point pointB) {
        this.pointB = pointB;
    }

    public Point getPointC() {
        return pointC;
    }

    public void setPointC(Point pointC) {
        this.pointC = pointC;
```

```
    }

    @Override
    public String toString() {
        return "Triangle{" + "pointA=" + pointA + ", pointB=" +
pointB + ", pointC=" + pointC + '}';
    }

    @Override
    public void setApplicationContext(ApplicationContext context)
 throws BeansException {
        this.context = context;
    }

    @Override
    public void setBeanName(String beanName) {
        System.out.println("The bean name is: " +beanName );
    }
}
```

We can do different things using ApplicationContextAware and also BeanNameAware.

XML Code:

```
<bean id="parenttriangle" class="spring.framework.Triangle" abst
ract="true">
    <property name="pointA" ref="pointA"/>
</bean>

<bean id="triangle" class="spring.framework.Triangle" parent="pa
renttriangle" >
    <property name="pointB" ref="pointB"/>
    <property name="pointC" ref="pointC"/>
</bean>

<bean id="pointA" class="spring.framework.Point">
    <property name="x" value="1" />
    <property name="y" value="2" />
</bean>

<bean id="pointB" class="spring.framework.Point">
    <property name="x" value="3" />
    <property name="y" value="4" />
</bean>

<bean id="pointC" class="spring.framework.Point">
    <property name="x" value="5" />
    <property name="y" value="6" />
</bean>
```

We can also inherited list using Bean Inheritance property.

Java Code:

```
public SpringFramework() {
    AbstractApplicationContext context = new ClassPathXmlApplica
tionContext("Spring.xml");
    context.registerShutdownHook();
    Triangle triangle = (Triangle) context.getBean("triangle");
    System.out.println(triangle.toString());
}
```

XML Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

       xsi:schemaLocation="http://www.springframework.org/schema
/beans http://www.springframework.org/schema/beans/spring-beans-
4.0.xsd
" xmlns:aop="http://www.springframework.org/schema/aop" default-
init-method="init" default-destroy-method="destroyM">

    <bean id="parenttriangle" class="spring.framework.Triangle"
abstract="true">
        <property name="pointA" ref="pointA"/>
    </bean>

    <bean id="triangle" class="spring.framework.Triangle" parent
="parenttriangle" init-method="init" destroy-method="destroyM">
        <property name="pointB" ref="pointB"/>
        <property name="pointC" ref="pointC"/>
    </bean>

    <bean id="pointA" class="spring.framework.Point">
        <property name="x" value="1" />
        <property name="y" value="2" />
    </bean>

    <bean id="pointB" class="spring.framework.Point">
        <property name="x" value="3" />
        <property name="y" value="4" />
    </bean>

    <bean id="pointC" class="spring.framework.Point">
        <property name="x" value="5" />
        <property name="y" value="6" />
    </bean>

</beans>
```

Java Code:

```
public class Triangle implements Shape, InitializingBean, Dispos
ableBean{
    Point pointA;
    Point pointB;
    Point pointC;
}
```

Java Code:

```
@Override
public void afterPropertiesSet() throws Exception {
    System.out.println("Before initilizination");
}

@Override
public void destroy() throws Exception {
    System.out.println("After initilizination");
}
```

Java Code:

```
public Spring() {
  ApplicationContext context = new ClassPathXmlApplicationContex
t("spring.xml");
  Triangle triangle = (Triangle) context.getBean("triangle");
  System.out.println(triangle.toString());
}
```

Java Code:

```
public class BeanpostProcessor implements BeanPostProcessor{

    @Override
    public Object postProcessBeforeInitialization(Object bean, S
tring beanName) throws BeansException {
        System.out.println(beanName);
        return bean;
    }

    @Override
    public Object postProcessAfterInitialization(Object bean, St
ring beanName) throws BeansException {
        System.out.println(beanName);
        return bean;
    }

}
```

XML Code:

```
<bean id="triangle" class="spring.Triangle" autowire="byName">

</bean>

<bean id="pointA" class="spring.Point">
    <property name="x" value="1"/>
    <property name="y" value="2"/>
</bean>

<bean id="pointB" class="spring.Point">
    <property name="x" value="3"/>
    <property name="y" value="4"/>
</bean>

<bean id="pointC" class="spring.Point">
    <property name="x" value="5"/>
    <property name="y" value="6"/>
</bean>

<bean class="spring.BeanpostProcessor"/>
```

You have to only give the class name of the BeanPostProcessor.

Java Code:

```
public Spring() {
    ApplicationContext context = new ClassPathXmlApplicationCont
ext("spring.xml");
    Triangle triangle = (Triangle) context.getBean("triangle");
    System.out.println(triangle.toString());
}
```

Java Code:

```
public class BeanpostProcessor implements BeanFactoryPostProcess
or{

    @Override
    public void postProcessBeanFactory(ConfigurableListableBeanF
actory clbf) throws BeansException {
        System.out.println("BeanFactoryPostProcessor is called")
;
    }
}
```

Properties File:

```
pointA.X=1
pointA.Y=2
```

XML Code:

```
<bean id="triangle" class="spring.Triangle" autowire="byName">

</bean>

<bean id="pointA" class="spring.Point">
    <property name="x" value="${pointA.X}"/>
    <property name="y" value="${pointA.Y}"/>
</bean>

<bean id="pointB" class="spring.Point">
    <property name="x" value="3"/>
    <property name="y" value="4"/>
</bean>

<bean id="pointC" class="spring.Point">
    <property name="x" value="5"/>
    <property name="y" value="6"/>
</bean>

<bean class="spring.BeanpostProcessor"/>
<bean class="org.springframework.beans.factory.config.PropertyPl
aceholderConfigurer">
    <property name="locations" value="spring.properties" />
</bean>
```

Java Code:

```
public Spring() {
    ApplicationContext context = new ClassPathXmlApplicationCont
ext("spring.xml");
    Shape shape = (Circle) context.getBean("circle");
    System.out.println(shape.toString());
}
```

Java Code:

```
public class Circle implements Shape{
    Point center;
}
```

XML Code:

```
<bean id="triangle" class="spring.Triangle" autowire="byName">

</bean>

<bean id="circle" class="spring.Circle">
    <property name="center" ref="pointA"/>
</bean>

<bean id="pointA" class="spring.Point">
    <property name="x" value="${pointA.X}"/>
    <property name="y" value="${pointA.Y}"/>
</bean>

<bean id="pointB" class="spring.Point">
    <property name="x" value="3"/>
    <property name="y" value="4"/>
</bean>

<bean id="pointC" class="spring.Point">
    <property name="x" value="5"/>
    <property name="y" value="6"/>
</bean>
```

Java Code:

```
public Spring() {
    ApplicationContext context = new ClassPathXmlApplicationCont
ext("spring.xml");
    Shape shape = (Circle) context.getBean("circle");
    System.out.println(shape.toString());
}
```

Java Code:

```
@Required
public void setCenter(Point center) {
    this.center = center;
}
```

XML Code:

```
<bean class="org.springframwwork.beans.factory.annotation.Requir
edAnnotationBeanPostProcessor" />
```

Java:

```
public Spring() {
    ApplicationContext context = new ClassPathXmlApplicationCont
ext("spring.xml");
    Shape shape = (Circle) context.getBean("circle");
    System.out.println(shape.toString());
}
```

XML Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema
/beans http://www.springframework.org/schema/beans/spring-beans-
4.0.xsd
       http://www.springframework.org/schema/context http://www.
springframework.org/schema/context/spring-context.xsd
" xmlns:context="http://www.springframework.org/schema/context">
    <bean id="triangle" class="spring.Triangle" autowire="byName
">

    </bean>

    <bean id="circle" class="spring.Circle">

    </bean>

    <bean id="pointA" class="spring.Point">
        <qualifier value="circle"/>
        <property name="x" value="1"/>
        <property name="y" value="2"/>
    </bean>

    <bean id="pointB" class="spring.Point">
        <property name="x" value="3"/>
        <property name="y" value="4"/>
    </bean>
```

```
    <bean id="pointC" class="spring.Point">
        <property name="x" value="5"/>
        <property name="y" value="6"/>
    </bean>
    <bean class="org.springframework.beans.factory.config.Proper
tyPlaceholderConfigurer">
        <property name="locations" value="spring.properties" />
    </bean>
    <context:annotation-config />
</beans>
```

Java Code:

```
@Autowired
@Qualifier("circle")
public void setCenter(Point center) {
    this.center = center;
}
```

Java Code:

```
@Resource(name = "pointB")
public void setCenter(Point center) {
    this.center = center;
}

@PostConstruct
public void initialization(){
    System.out.println("Before");
}

@PreDestroy
public void destroy(){
    System.out.println("after");
}
```

Java Code:

```
@Component
@Service
@Repository
@Controller
public class Circle implements Shape{
    Point center;
}

@Resource(name = "pointB")
public void setCenter(Point center) {
    this.center = center;
}
```

XML Code:

```
<context:component-scan base-package="spring" />
```

Java:

```
public Spring() {
    ApplicationContext context = new ClassPathXmlApplicationCont
ext("spring.xml");
    Shape shape = (Circle) context.getBean("circle");
    System.out.println(shape.toString());
    System.out.println(context.getMessage("greeting",null, "Defa
ult Message", null));
}
```

XML Code:

```
<bean id="messagesource" class="org.springframework.context.supp
ort.ResourceBundleMessageSource">
    <property name="basenames">
        <list>
            <value>spring</value>
        </list>
    </property>
</bean>
```

Java Code:

```
public void draw() {
    System.out.println(this.messagesource.getMessage("greeting",
null, "optional",null));
    System.out.println(this.messagesource.getMessage("greeting",
new Object[]{center.getX(), center.getY()}, "optional",null));
    System.out.println("Circle is drawn");

}
```

Property File:

```
greeting=Hello!
draing.point={0},{1}
```

Java Code:

```
public class Circle {
    Point center;
    MessageSource messagesource;

    public MessageSource getMessagesource() {
        return messagesource;
    }
    @Autowired
    public void setMessagesource(MessageSource messagesource) {
        this.messagesource = messagesource;
}
```

Java Code:

```
public class Circle implements Shape, ApplicationEventPublisherA
ware{
    Point center;
    ApplicationEventPublisher  publisher;
}

@Override
public void setApplicationEventPublisher(ApplicationEventPublish
er publisher) {
    this.publisher = publisher;
}
```

Java Code:

```
@Component
public class Application implements ApplicationListener{

    @Override
    public void onApplicationEvent(ApplicationEvent e) {
        System.out.println(e.toString());
    }

}
```

Java Code:

```
public class DrawEvent extends ApplicationEvent{

    public DrawEvent(Object source) {
        super(source);
    }

    public String toString(){
        return "draw event occured";
    }
}
```