Dependency Injection

In Spring, objects define their dependencies and do not worry about how they will get those dependencies. It is the responsibility of Spring to provide the required dependencies for creating objects.

Types of dependency injection, setter and constructor

Inversion of Control is a principle by which the control of objects or portions of a program is transferred to a container or framework.

IoC enables a framework to take control of the flow of a program and make calls to our custom code.

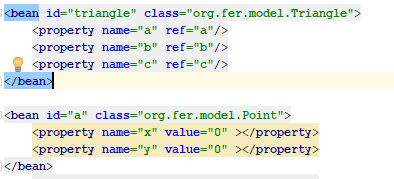
Application Context

Bean Factory (instantiate, keeps track and manage the life cycle of the beans) with additional functionality like event notification and AOP





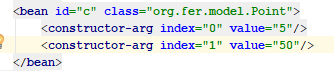
Define a bean in xml file for spring to instantiate it



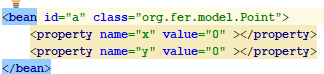
Property tag allows to set a value for the property or use another bean as value with the ref attribute

Constructor injection

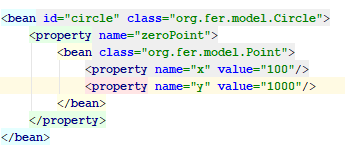
Will make the injection through the constructor instead than the setter. Can use index to specify the position of each arg. Can also specify type in case each arg would be of a different type.



Setter injection, will use the property tag and will set the value through the setter



Inner bean, will be available only inside another bean



Alias

Allows to create an alias for a bean and reference it inside config file or even get it from the spring context. Ex b existing bean, secondPoint its alias



Idref, if you want to restrict a bean to use only id, not aliases or names

Initializing collections on the bean



Autowiring

byName, if the properties of the bean match then they are linked together

byType, only works if there are only one type of the autowired bean

constructor

default

no (default autowiring off)



Spring Scopes

Singleton, will return always the same instance (default). At container initialization these are instantiated

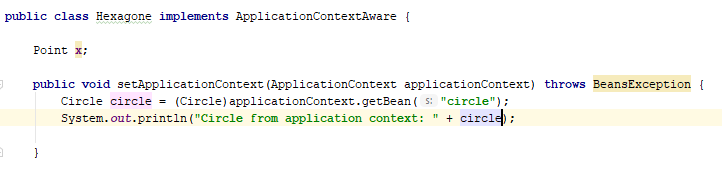
Prototype, will return a new instance of the class on each get bean

Request, session, globalsession also available (web aware context bean scopes)



Application Context Aware Interface

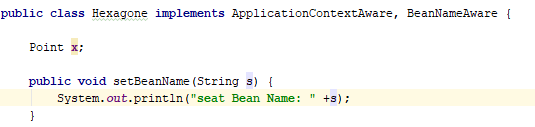
Classes implementing this interface will have access to the application context which will be injected by Spring.



Bean Name Aware Interface

Allows to get the name of the bean in the class.

Ex. Bean declared in spring.xml as hexagone of class Hexagone, in the class Hexagone will have the method setBeanName which will receive the name of the bean (hexagone).

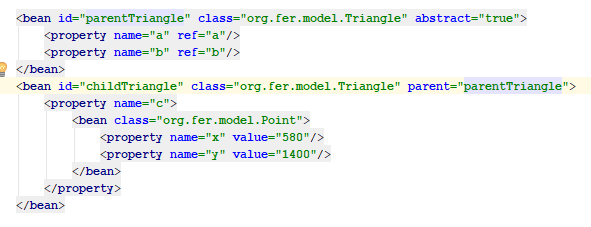


Bean Definition Inheritance

Allows beans to inherit from another bean and share its properties using the attribute parent and the parent’s bean name

Abstract

If set to true, allows a bean to not being instantiated, will just be use as a template for other beans



Inheritance with collections

If a bean inherits from another and they have a collection, then the child would override the values of the parent if the value is specified, or you can use the option merge which will merge the values from parent and child



Initializing Bean Interface

It gives a method afterPropertiesSet which will execute after the bean has bean instantiated

Disposable Bean Interface

Offers a method destroy() which will be executed before the bean gets destroyed.

AbstractApplicationContext

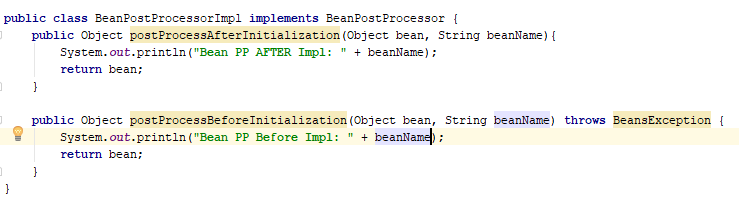
Application context which allows us to use the method registerShutdownHook to identify when the program ends and the beans will be destroyed.

Init-method, destroy-method

Allows the same functionality without implementing the interfaces, declare on the bean the init and destroy method for the bean, If want it for all the beans you can declare it on the beans tag with the attribute default-init-method and default-destroy-method

Bean Post Processor

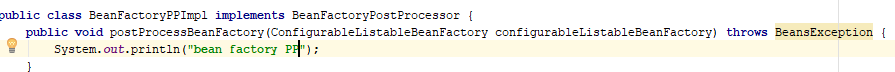
It executes before and or after any bean has been initialized (for all beans) (bean needs to be added to spring xml





Bean Factory Post Processor interface

Has a method which will run after the bean factory itself has been instantiated



@Required

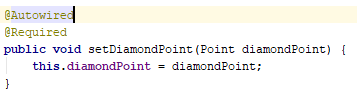
Annotation which force a dependency to be fulfill in order to initialize

(needs to add <context:component-scan base-package="org.fer"/> for it to work)

@Autowired

Spring helps to autowire dependencies, first for type then for name

Add the annotation to the setter

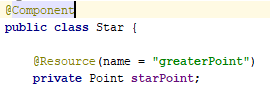


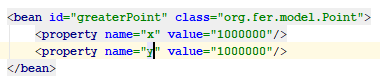
@Qualified

You can add it and a qualifier tag on spring xml for it to be use as bean name

@Resource

Allows to inject a value into a variable



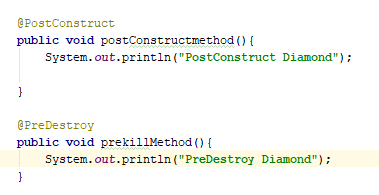


@PostConstruct

You can annotate a method with this and will be executed automatically after it has ben created

@PreDestroy

You can annotate a method with this and it will be executed automatically before the bean is destroyed.



Stereotypes

@Component, allows to define a class as a bean so its not necessary to add it to the spring.xml

@Service

@Controller

@Repository

AOP

Model of programming that helps with cross cutting concerns

Do not make reference to it on the classes and can be applied before or after the method

TARGET METHOD

ASPECT

ASPECT

Aspect

Class with annotations

Advice

Methods inside the aspect which will run

@Before

This will be the moment where the aspect will be applied, before the method with the signature sent is executed this will execute first.

Pointcut

Says where you want the advice to apply

@Before

@After

@AfterReturning

@AfterThrowing

@Around