What the fuck is spring anyway?

Just a framework which will help us to build high quality and sustainable code.

Reason for that will be discussed below.

Why the fuck should I choose spring?  
In my pov, there are two main reasons.

1. IOC -> inversion of control
2. DI -> dependency injection

The word “bean” is just a fancy name for object. So beans are nothing but objects.

Spring will create, manage, and instantiate beans for you i.e., developers. But how?

Internally there is a container called IOC which will do all this management/creation of beans.

IOC :

First of all we will try to imagine our lives without IOC but what is the issue that IOC is trying to solve here?

Tight coupling

Tight coupling ante 2-3 classes unnay anuko, oka class lo emaina modification cheste migata classes lo kooda cheyali. Oka daani meeda inkoti ala chala heavy ga depend ayyi untay. Ala unte next time emaina kotta feature or existing source code enhance cheyali ante chaala tough.

IOC lekapote manam beans ela create chestam?  
By using “new” keyword.

General ga code control developer daggara untundi kada, for example objects maname create chestam, maname call cheskuntam.

Kaani IOC lo ala kaadu, as the name says the control gets inverted.

Spring mana developers daggaraki ochesi, nv ila objects create chestunte mottam tight coupling and shitty code avtondi nuvem chestav ante just neeku eh objects kavalo naaku chepu, aa objects neek ichesta antundi. Ide inversion of control.

Lets try to understand the difference between tightly coupled and loosely coupled code.

A screenshot of a computer program

Description automatically generated

Ee case lo

Ipdu person change ayyar anukondi, student class lo kooda changes cheyali.

A screen shot of a computer program

Description automatically generated

Ee case lo

Person change aina emaina student class ki sambandame ledu.

Student class be like : Person class emaite naakenduku bayya, all I want is an object of person class.

IOC will smartly create the beans but how?

Based on dependency injection.

Dependency injection ante enti mari?  
aa creation of objects, managing etc etc aa process mottam DI

If I had to say this in my own words I would say like this -> the process in which the beans are created smartly is DI.

The place where this process happens is IOC.

Dependency injections ela cheyali?

Two ways.

1. Constructor based
2. Setter functions ( POJO classes )

I think you know this. Epudo vinnam anipistundi.

The worst part of spring enti ante ide, XML file based dependency injection.

XML files lo beans ni configure chesi pedte IOC will pick them up for you, but how?

XML file lo unna data ni java pick cheskovali ante ela? We know that IOC will pick.

We’re in the world of Java, so IOC should be surely compatible with java.

IOC lo two main interfaces untay (aa rendu eh unaya or inka unaya anedi idk, I’ve to refer)

1. BeanFactory
2. ApplicationContext (this is used more)

Aa rendu interfaces kabatti, vatiki object create cheyalem

So there are some classes which will implement beanfactory and applicationcontext, by using them we can create objects for this.

XmlBeanFactory is a class which implements beanfactory interface.

ClassPathXmlApplicationContext is a class which implements applicationcontext interface.

\* BeanFactory bf = new XmlBeanFactory(“xml\_path.xml”);

But XmlBeanFactory is deprecated, you can use ClassPathXmlApplicationContext to create obj for beanfactory.

For eg :

BeanFactory bf = new ClassPathXmlApplicationContext("xml\_data.xml");

ApplicationContext ac = new ClassPathXmlApplicationContext("xml\_data.xml");

(But I don’t know why and how. ClassPathXmlApplicationContext should be implementing BeanFactory interface but I don’t see that in the source code of org.springframework.context. idk )

Lets try to understand a few examples :

1. Constructor based DI :

Please refer this: <https://github.com/notthefakekowshik/spring-prac/tree/master/constructor-DI>

1. Setter based DI :

Please refer this: <https://github.com/notthefakekowshik/spring-prac/tree/master/setter-DI>

Until now, we have injected dependencies and configured beans using XML files. The time has come to dump XML and move on with Annotations.

1. Autowiring

This can be done by using XML files too but we have an annotation called autowired (@Autowired), I am not discussing XML related autowiring here.

You can apply this annotation on instances, constructors.

We use this when there is has-a relationship.

For Eg :

Class student{

@Autowired //person is injected into student.

Person p;

}

1. Configuration, qualifier, bean :

Configuration is a class based annotation which says that this class will have some beans and this will be managed by IOC.

In this, you will create an object for “AnnotationConfigApplicationContext” instead of ClassPathXmlApplicationContext because we are using annotations but not XML.

Eg :

ApplicationContext ac = new AnnotationConfigApplicationContext(BeansConfig.class);

BeansConfig is a class which will return the beans, please refer the source code attached below. You will get an idea.

Refer this, you will understand easily: <https://github.com/notthefakekowshik/spring-prac/tree/master/configuration-annotation>

1. Component, Primary :

Talking is cheap, see the code :

https://github.com/notthefakekowshik/spring-prac/tree/master/component-annotation

Some of the spring properties you must know :   
A screenshot of a computer

Description automatically generated

Src : https://www.youtube.com/watch?v=dpgwqStAj3k