This is where we are going to put our first step in the world of Spring.

In this tutorial, we are going to learn the complete Spring Core.

First let’s discuss what’s is the prerequisite to learn Spring Core.

* Core Java

If you are familiar with all core java concepts you are good to rock.

What is IOC ?

To get an idea about What IOC does? What is the use of IOC? Why IOC?

We have to go through a scenario.

*Suppose we have an interface called* **Sim** *with the following body :*

public interface Sim {

void calling();

void internet();

}

*And it has two implemented classes i.e*. **Airtel** *and* **Vodafone** :

public class Airtel implements Sim {

@Override

public void calling() {

System.out.println("Calling from airtel");

}

@Override

public void internet() {

System.out.println("Internet surfing from airtel");

}

}

public class Vodafone implements Sim {

@Override

public void calling() {

System.out.println("Calling from vodafone");

}

@Override

public void internet() {

System.out.println("Internet surfing from vodafone");

}

}

Now a mobile class will use any of Sim object out of these. So suppose a user mobile is having Airtel Sim in it.

public class Mobile {

public static void main(String[] args) {

Airtel airtel = new Airtel();

airtel.calling();

airtel.internet();

}

}

The above code will work fine and will able to use calling() and internet() through Airtel object.

But the problem will created if the user mind got change and he decided to switch Airtel Sim to Vodafone Sim and this is very common situation.

So what we have to do. We have to change our Source code as shown below highlighted:

public class Mobile {

public static void main(String[] args) {

Vodafone vodafone = new Vodafone();

vodafone.calling();

vodafone.internet();

}

}

But as you can see by using Class reference there is lot many changes we need to apply i.e. 5 changes here.

But by using some software engineering best practices we can improve our code. Instead of creating Class reference just create Interface reference.

public class Mobile {

public static void main(String[] args) {

Sim sim = new Airtel();

sim.calling();

sim.internet();

}

}

Now here, if the user want to switch the Sim from Airtel to Vodafone then we just have to make one change in our source code as showing below code :

public class Mobile {

public static void main(String[] args) {

Sim sim = new Vodafone();

sim.calling();

sim.internet();

}

}

Still we’ll get the same output as before.

But even after applying best practices still we are left with one change. Still if the user change there Sim we are making 1 change in our source code and that is not safe.

So What we want ?

* Don’t touch the Source Code
* Application Should be configurable

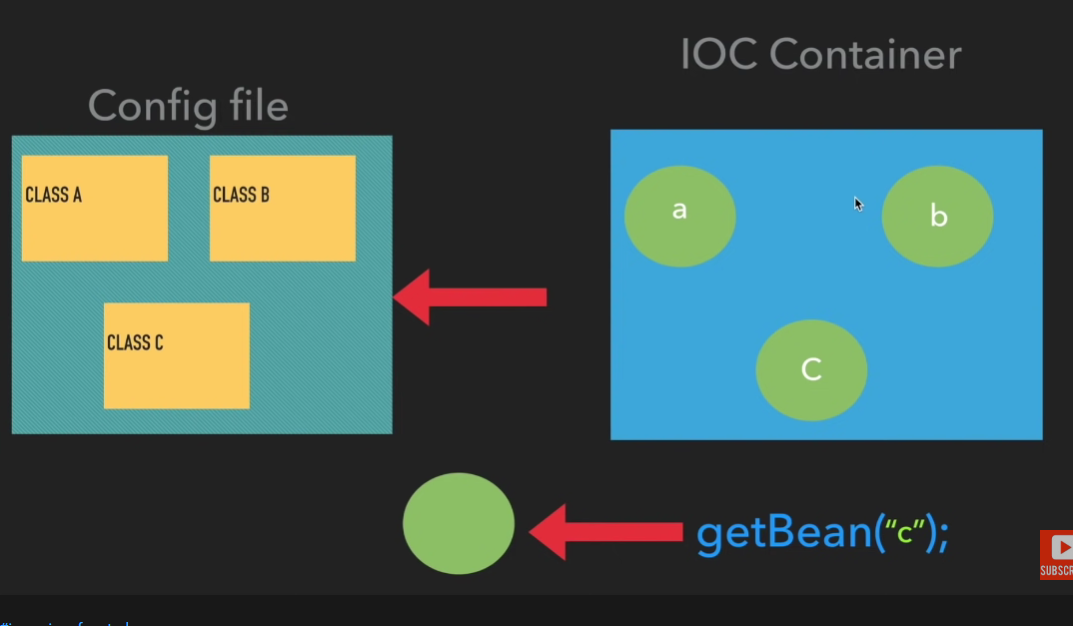
Now who can help you to achieve this kind of functionality in your application.

So the answer is **Spring Framework**.

**How Spring Framework will help us ?**

Spring framework says,

I have something called **IOC Container (Inversion Of Control Container )**. You have to tell me through **config file** the class name and id for them which object you want to be create. Once you have mentioned and when you run the application then I will read the **config file** and I will create objects for all those classes and put inside my **IOC container.** Now these object can be available if you ask me them through their id.



I know it can be little bit confusing to understand now. But you will get the charm once you have achieved your goal.

There is two types of IOC container spring have :

* **BeanFactory** (legacy interface)
* **ApplicationContext** (latest interface)

The only difference between these two is **ApplicationContext** is latest version of BeanFactory with advance features.

In this particular tutorial we are going to use **ApplicationContext**.

As we have mentioned **ApplicationContext** is an interface so it is not possible to create it’s object. We have it’s implemented class called **ClassPathXmlApplicationContext .**

class ClassPathXmlApplicationContext implements ApplicationContext { }

So let’s take a quick practical how will perform all these?

***STEP 1 :*** *As a first step just add a new xml file into your project called* **config.xml** *with the given body.\*

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.xsd">

<bean class="com.paulspring.ioc1.Airtel" id="sim" />

</beans>

* The above xml file configuration is very easy.
* <beans> tag can consist of multiple <bean> tag.
* <beans> tag have some attributes like xmlns, xmlns:xsi, xsi:schemaLocation. So this attributes just we import Classes and use their methods. Similarly here we are importing some of tag which we going to use inside our <beans> tag.

Now the important line is :

<bean class="com.paulspring.ioc1.Airtel" id="sim" />

When spring read this line, it will create an object of Airtel class like below :

Airtel airtel = new Airtel();

* **class attribute :** it specify the bean class name which object going to create.
* **id attribute :** it specify the reference to that object.

**STEP 2 :** *Rewrite the Mobile class again.*

public class Mobile {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("com/paulspring/ioc1/config.xml");

Sim sim = context.getBean("sim", Sim.class);

sim.calling();

sim.internet();

}

}

**Explanation :** You have mentioned in your config.xml that create Airtel class object and referenced it with the name “sim”.

Now when you run main() method it’s first line will read the config.xml and create all the object mentioned in it and put them inside IOC container.

Second line will search for an object which has associated id “sim” and return that object. So due to this we will get Airtel object and with the help of this object we can call Airtel members.

Now suppose user want to switch their Sim from Airtel to Vodafone. We don’t need to even touch our source code. We just have to mentioned inside config.xml that we don’t need Airtel object anymore. Give me Vodafone object now. That’s it.

<bean class="com.paulspring.ioc1.Vodafone" id="sim" />

**Questions**

**1.) How can you prove that spring creating our class object ?**

**Ans :** Just add an constructor inside any of your class Airtel or Vodafone and put an System.out.println() statement inside it. If spring would creating the object the constructor will must call.

**2.) What if new Sim provider comes inside the market and implements your Sim interface.**

**Ans :** There is nothing to worry even we just have to make a small change in config.xml and our source will not get touch even in this situation.

<bean class="com.paulspring.ioc1.Jio" id="sim" />

**3.) What do you mean by Inversion Of Control ?**

**Ans :** When we need an object we make it ourselves before. But in spring it got reverse, here spring creating object for us. So that’s why this topic name is Inversion Of Control.