Spring Notes

-by Shuinspirer

Spring Core

1. Dependency Injection in Spring

Dependency Injection is same as IOC (Inversion of control)

Dependency Injection is a design pattern that removes the dependency from the programming code so that it can be easy to manage and test the application. Dependency Injection makes our programming code loosely coupled.

The Dependency Injection is a design pattern that removes the dependency of the programs. In such case we provide the information from the external source such as XML file. It makes our code loosely coupled and easier for testing. In such case we write the code as:

```
1. class Employee{
2. Address address;
3. Employee(Address address){
4. this.address=address;
5. }
6. public void setAddress(Address address){
7. this.address=address;
8. }
9. }
```

In such case, instance of Address class is provided by external souce such as XML file either by constructor or setter method.

Two ways to perform Dependency Injection in Spring framework

Spring framework provides two ways to inject dependency

- By Constructor
- By Setter method

Spring Core is the module that automatically configure the dependencies we don't have to worru about whether class is singleton or not etc.

Spring boot believes that we should focus on convention instead of configuration.

2. Creating a component:

Before adding component:

```
Public static void main()
{
        SpringApplication.run(FirstProjApplication.class, args);
        Alien obj=getbean(Alien.class);
        Obj.code();
}
```

The above code throws an error.

- In java every we need object in order to call a method so Getbean needs a method.
- Getbeans belongs to the ApplicationContext Interface.
- Here run method belongs a class which extends Application Context.

So the code becomes

```
Public static void main()
{
        ApplicationContext  context = SpringApplication.run(FirstProjApplication.class, args);
        Alien obj=context.getbean(Alien.class);
        Obj.code();
}
```

It again throws an error because we have to tell the spring to add this alien class as an component

So we need to declare the class as an component;

```
@component
Public class alien
{
        Public void code()
        {
                System.out.println("Hello World!");
        }
}
```

3. Autowiring

It simply means that we ask whether we have an object of mentioned class or not.

```
@component
Public class alien
{
	@Autowired
	Laptop lap;
	Public void code()
	{
		lap.compile();
	}
}
@component
Public class Laptop
{
	Public void compile()
	{
		System.out.println(“Compiling”);
	}
}
```

4. Bean Factory

Now we are configuring our own dependencies in maven project.

To use getbean we need to add a dependency so we move to the chrome and search for maven repository and search for spring context then adding the dependency in pom..xml file.

Now they will be downloading.

But firstly we need to instantiate the beanfactory

But we are using maven not spring boot so we need to configure so we use filesystem to identify our object like from where our object belong.

BeanFatory factory= new XmlBeanFactory(new FileSystemRsource(“spring.xml”));

Alien.obj = (Alien)factory.getBean(“alien”);

But since the XmlBeanFactory is deprecated and new new features keep on comping which are not in beanfactory so in order to create new objects, we’ll use the ApplicationContext Interface

```
ApplicationContext context = new ClassPathXmlApplicationContext("spring.xml");
```

This classpathxmlapplicationcontext instantiate the applicationcontext and implements it.

Since this is a class path so we need to add this in java folder.

5. Spring Container

Applicationcontext creates the object locally if we do not even create the object of the class.

Appcon->ApplicationContext

Appcon will simply create the container for you.

Inside in jvm there is a container, inside a container there are beans

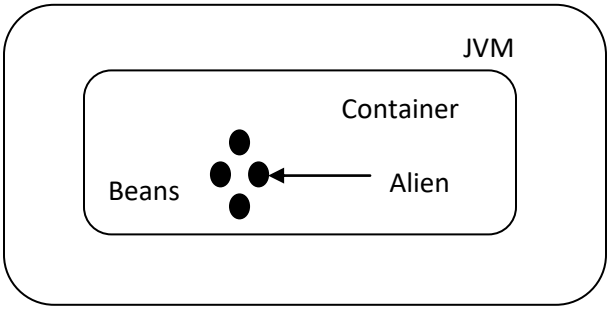
Beans is any class which have variables and getters and setters for the variables

In spring.xml whenever we code following code

```
<bean id="alien" class="com.example.FirstSpringMavenDemo.Alien"></bean>
```

We create a bean just like Alien class.

Every bean tag has property tag which represents the variables.



6. Singleton vs prototype

Alien obj1 = (Alien) factory.getBean(“alien”);

Obj1.age=15;

Sustem.out.println(obj1.age);

Alien obj2 = (Alien) factory.getBean(“alien”);

Sustem.out.println(obj2.age);

Output 15

15

Instead of 15 0

So this is because the object is created only once. This is called as singleton bean(The object is created only once). This is because the scope is by default set as singleton

```
<bean id="alien" class="com.example.FirstSpringMavenDemo.Alien" scope=”singleton”></bean>
```

If we need different objects for different calls we can set the scope as prototype

```
<bean id="alien" class="com.example.FirstSpringMavenDemo.Alien" scope=”prototype” ></bean>
```

Now the output is 15

0

In singleton, even if we don’t ask for object, it Is created but in protype if you need object then you need to call the object.

7. Setter Injection

Now if we want to set the default age not from class but from the spring.xml file then we can do by simply using the property tags.

Object creation process

1. Object is created
2. Constructor is called
3. Setter is called(the value was set to 10).

Don’t change any setter and getter function names like setage1 or setterage is not allowed.

8. Reference Attribute

If the variable is reference type not primitive type then instead of value ref is used.

```
<bean id="alien" class="com.example.FirstSpringMavenDemo.Alien">
    <property name="age" value="10"></property>
    <property name="laptop" ref="laptop"></property>
</bean>
<bean id="laptop" class="com.example.FirstSpringMavenDemo.Laptop">
</bean>
```

9.Constructor Injection

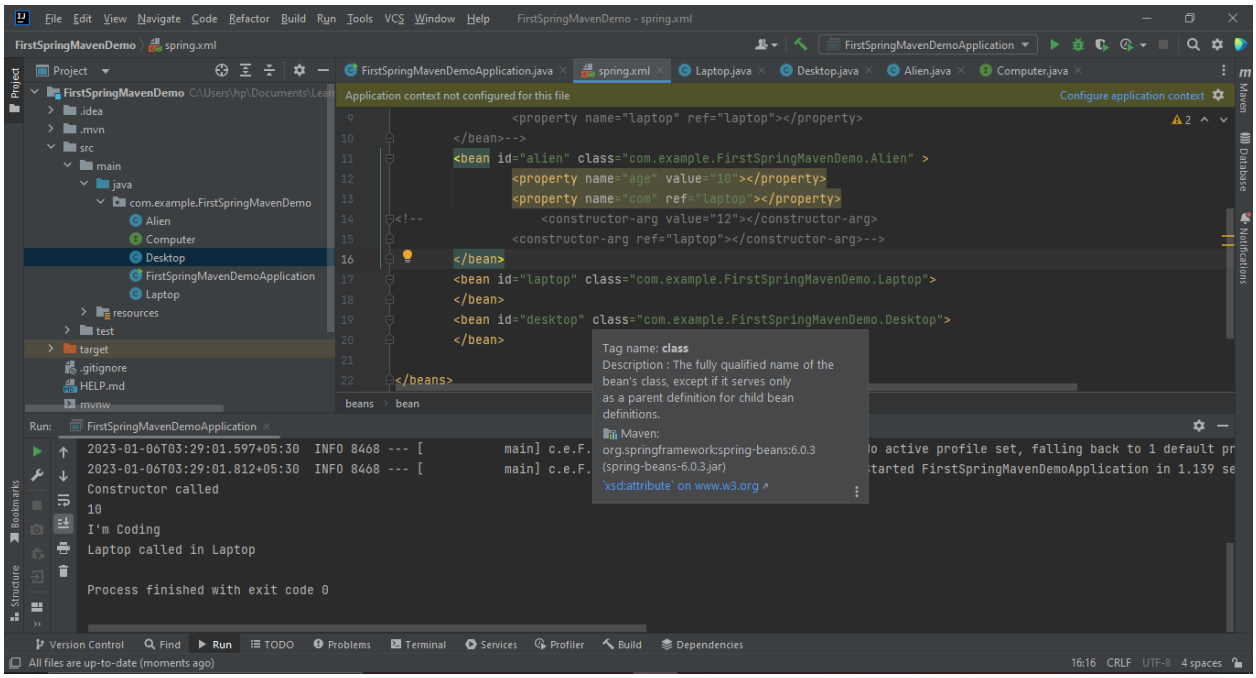
If we want to use the parametrised constructor instead of default without parsing the parameters from the spring.xml file then we can do this by using the tag <constructor-arg>

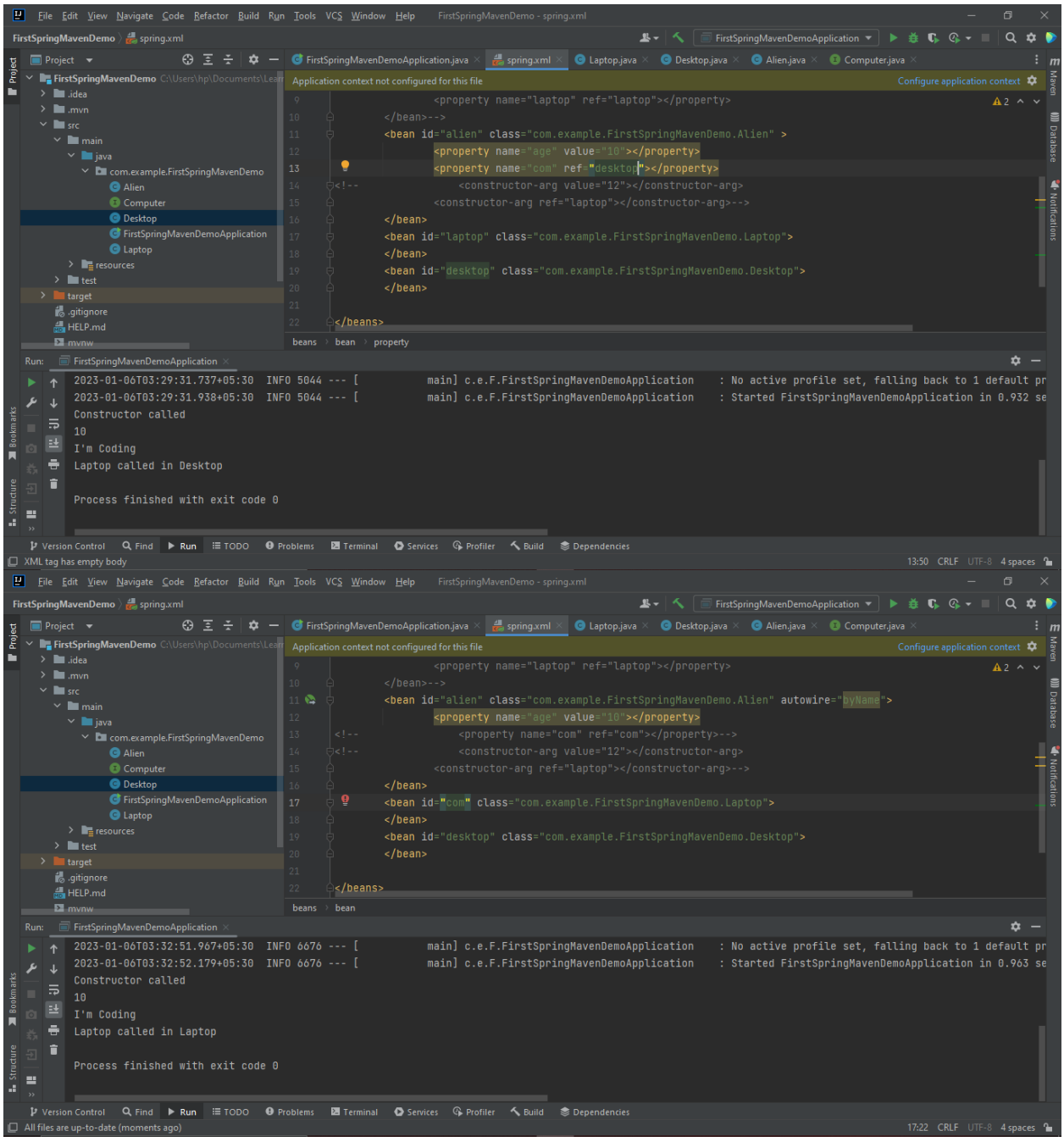
```
<bean id="alien" class="com.example.FirstSpringMavenDemo.Alien">
    <!-- <property name="age" value="10"></property> -->
    <constructor-arg value="12"></constructor-arg>
    <!--<constructor-arg ref="laptop"></constructor-arg> -->
    <property name="laptop" ref="laptop"></property>
</bean>
```

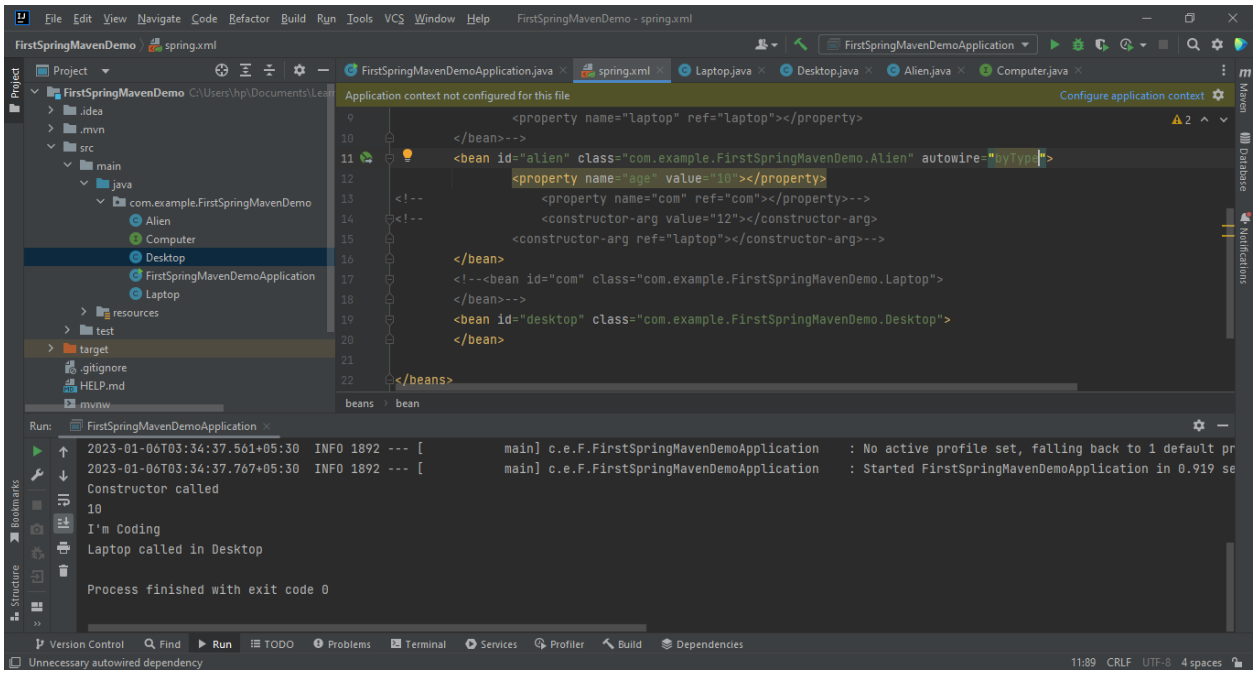
Constructor is used when the property(variable) is compulsory to assign or wired.

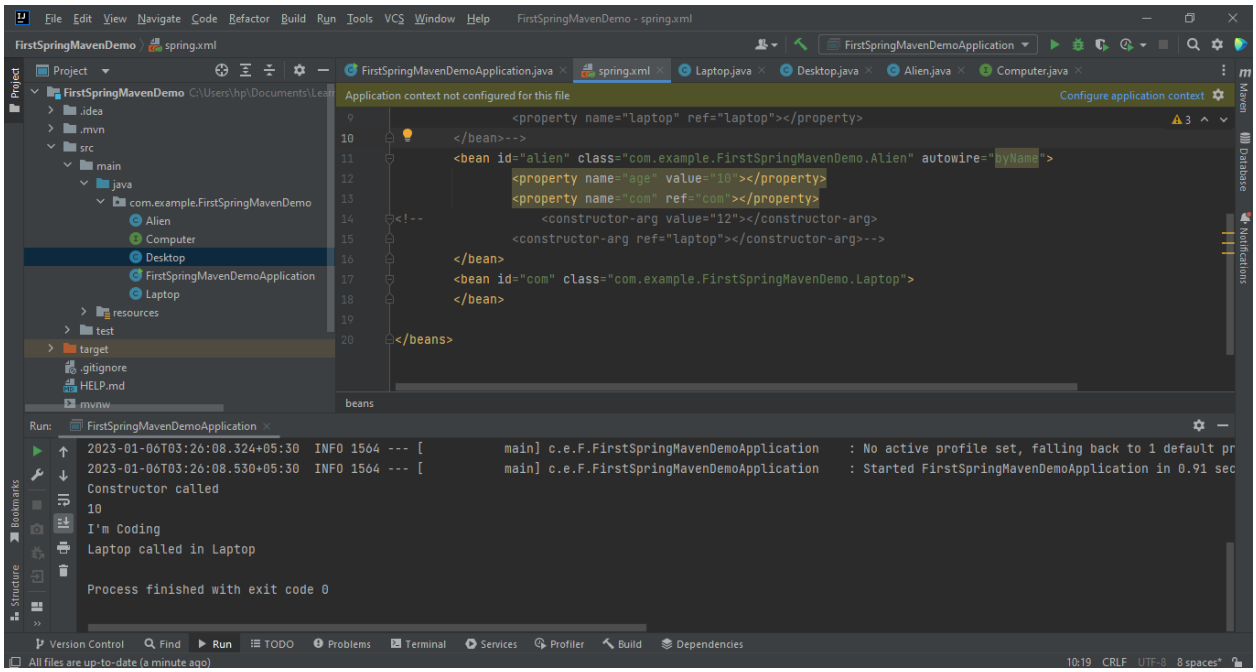
Setter is used when when the peoperty is optional.

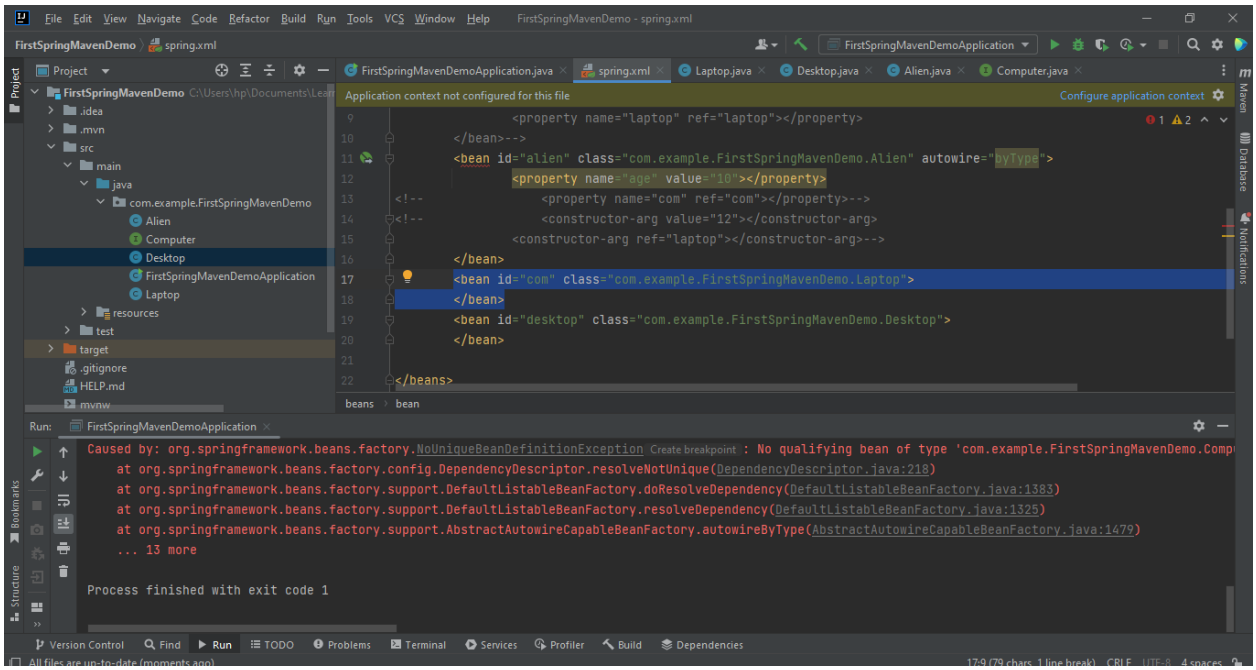
10.Autowire




FirstSpringMavenDemo - spring.xml
FirstSpringMavenDemo
spring.xml
FirstSpringMavenDemoApplication
Application context not configured for this file
Configure application context
<property name="laptop" ref="laptop"></property>
</bean>-->
<bean id="alien" class="com.example.FirstSpringMavenDemo.Alien" >
<property name="age" value="10"></property>
<property name="com" ref="desktop"></property>
<!-- <constructor-arg value="12"></constructor-arg>
<constructor-arg ref="laptop"></constructor-arg>-->
</bean>
<bean id="laptop" class="com.example.FirstSpringMavenDemo.Laptop">
</bean>
<bean id="desktop" class="com.example.FirstSpringMavenDemo.Desktop">
</bean>
</beans>
beans › bean › property
Run: FirstSpringMavenDemoApplication
2023-01-06T03:29:31.737+05:30 INFO 5044 --- [main] c.e.F.FirstSpringMavenDemoApplication : No active profile set, falling back to 1 default pr
2023-01-06T03:29:31.938+05:30 INFO 5044 --- [main] c.e.F.FirstSpringMavenDemoApplication : Started FirstSpringMavenDemoApplication in 0.932 se
Constructor called
10
I'm Coding
Laptop called in Desktop
Process finished with exit code 0
XML tag has empty body
13:50 CRLF UTF-8 4 spaces
FirstSpringMavenDemo - spring.xml
FirstSpringMavenDemo
spring.xml
FirstSpringMavenDemoApplication
Application context not configured for this file
Configure application context
<property name="laptop" ref="laptop"></property>
</bean>-->
<bean id="alien" class="com.example.FirstSpringMavenDemo.Alien" autowire="byName">
<property name="age" value="10"></property>
<!-- <property name="com" ref="com"></property>-->
<!-- <constructor-arg value="12"></constructor-arg>
<constructor-arg ref="laptop"></constructor-arg>-->
</bean>
<bean id="com" class="com.example.FirstSpringMavenDemo.Laptop">
</bean>
<bean id="desktop" class="com.example.FirstSpringMavenDemo.Desktop">
</bean>
</beans>
beans › bean
Run: FirstSpringMavenDemoApplication
2023-01-06T03:32:51.967+05:30 INFO 6676 --- [main] c.e.F.FirstSpringMavenDemoApplication : No active profile set, falling back to 1 default pr
2023-01-06T03:32:52.179+05:30 INFO 6676 --- [main] c.e.F.FirstSpringMavenDemoApplication : Started FirstSpringMavenDemoApplication in 0.963 se
Constructor called
10
I'm Coding
Laptop called in Laptop
Process finished with exit code 0
All files are up-to-date (moments ago)
17:22 CRLF UTF-8 4 spaces

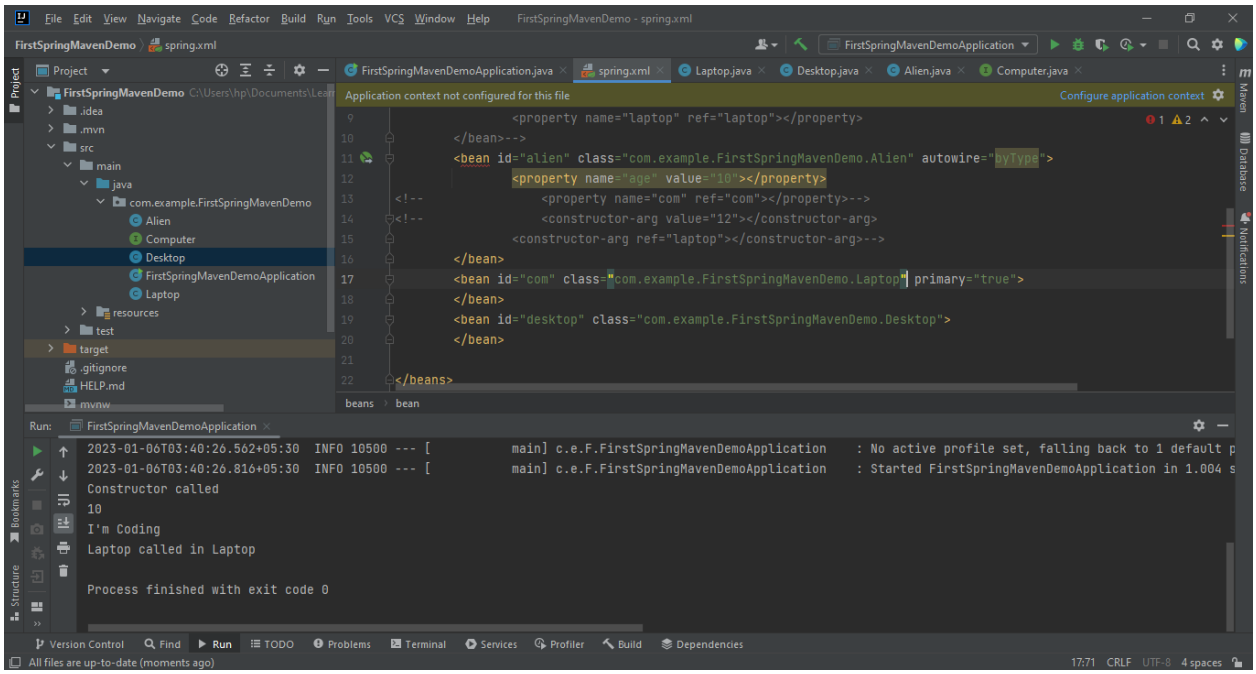

FirstSpringMavenDemo - spring.xml
FirstSpringMavenDemoApplication.java
spring.xml
Laptop.java
Desktop.java
Alien.java
Computer.java
Application context not configured for this file
Configure application context
<property name="laptop" ref="laptop"></property>
</bean>-->
<bean id="alien" class="com.example.FirstSpringMavenDemo.Alien" autowire="byType">
<property name="age" value="10"></property>
<!-- <property name="com" ref="com"></property>-->
<!-- <constructor-arg value="12"></constructor-arg>
<constructor-arg ref="laptop"></constructor-arg>-->
</bean>
<!--<bean id="com" class="com.example.FirstSpringMavenDemo.Laptop">
</bean>-->
<bean id="desktop" class="com.example.FirstSpringMavenDemo.Desktop">
</bean>
</beans>
beans › bean
Run: FirstSpringMavenDemoApplication
2023-01-06T03:34:37.561+05:30 INFO 1892 --- [main] c.e.F.FirstSpringMavenDemoApplication : No active profile set, falling back to 1 default pr
2023-01-06T03:34:37.767+05:30 INFO 1892 --- [main] c.e.F.FirstSpringMavenDemoApplication : Started FirstSpringMavenDemoApplication in 0.919 se
Constructor called
10
I'm Coding
Laptop called in Desktop
Process finished with exit code 0
Unnecessary autowired dependency
11:89 CRLF UTF-8 4 spaces


FirstSpringMavenDemo - spring.xml
FirstSpringMavenDemoApplication.java
spring.xml
Laptop.java
Desktop.java
Alien.java
Computer.java
Application context not configured for this file
Configure application context
<property name="laptop" ref="laptop"></property>
</bean>-->
<bean id="alien" class="com.example.FirstSpringMavenDemo.Alien" autowire="byName">
<property name="age" value="10"></property>
<property name="com" ref="com"></property>
<!-- <constructor-arg value="12"></constructor-arg>
<constructor-arg ref="laptop"></constructor-arg>-->
</bean>
<bean id="com" class="com.example.FirstSpringMavenDemo.Laptop">
</bean>
</beans>
beans
Run: FirstSpringMavenDemoApplication
2023-01-06T03:26:08.324+05:30 INFO 1564 --- [main] c.e.F.FirstSpringMavenDemoApplication : No active profile set, falling back to 1 default pr
2023-01-06T03:26:08.530+05:30 INFO 1564 --- [main] c.e.F.FirstSpringMavenDemoApplication : Started FirstSpringMavenDemoApplication in 0.91 sec
Constructor called
10
I'm Coding
Laptop called in Laptop
Process finished with exit code 0
All files are up-to-date (a minute ago)
10:19 CRLF UTF-8 8 spaces*



11.Primary Bean

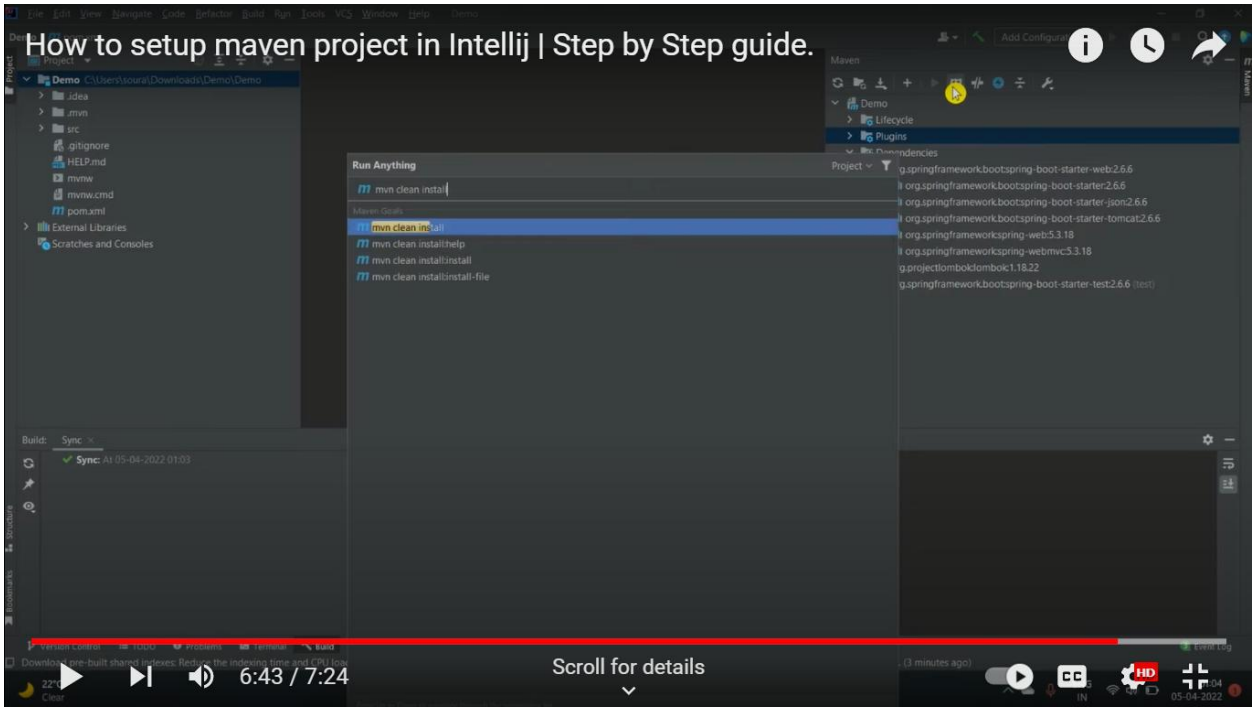
When autowire is byType and there are many beans so need to choose a primary bean



Notes

Maven Installation

- If maven dependencies are not working as per the processing so reload the project. Even then it is not working then click on maven and then type clean install then it should surely work



XmlBeanFactory Substitution Code

```
public class SpringHelloWorldTest {
   public static void main(String[] args) {

      XmlBeanFactory beanFactory = new XmlBeanFactory(new
ClassPathResource("SpringHelloWorld.xml"));

      Spring3HelloWorld myBean =
(Spring3HelloWorld)beanFactory.getBean("Spring3HelloWorldBean");
      myBean.sayHello();
   }
}
```

Alternative of following code is

```
public static void main(String[] args){
   ApplicationContext context=new ClassPathXmlApplicationContext(new
String[]{"SpringHelloWorld.xml"});
   BeanFactory factory=context;
   Spring3HelloWorld myBean=(Spring3HelloWorld)factory.getBean("Spring3HelloWorldBean");
   myBean.sayHello();
}
```

Or

```
BeanDefinitionRegistry beanFactory = new DefaultListableBeanFactory();
XmlBeanDefinitionReader reader = new XmlBeanDefinitionReader(beanFactory);
reader.loadBeanDefinitions(new ClassPathResource("beans.xml"));
Messager msg = (Messager) beanFactory.getBean("Messager");
```

Setter and Constructor Injection

Constructor is used when the property(variable) is compulsory to assign or wired.

Setter is used when when the peoperty is optional.

References

https://stackoverflow.com/questions/5231371/springs-xmlbeanfactory-is-deprecated

JSP Setup in IntelliJ Ultimate

https://stackoverflow.com/questions/44478620/jsp-with-intellij

https://stackoverflow.com/questions/61757663/not-able-to-create-jsp-file-by-template-on-intellij-ideas-springboot-project