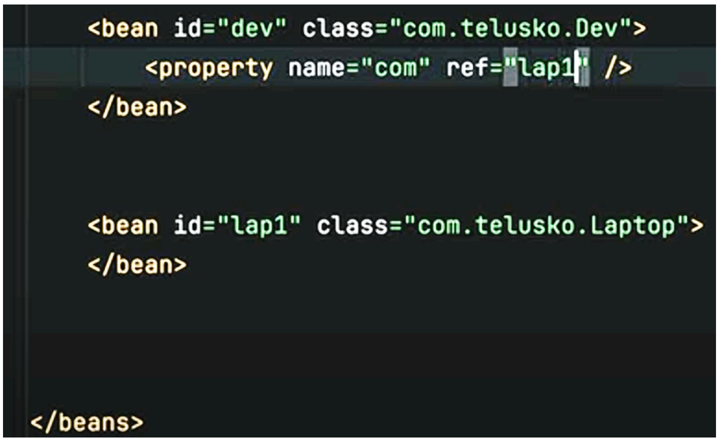
Autowire in Spring

- We've already seen how autowiring works in Spring Boot. Let's look at it again in the Spring Framework.

  Because of this, we made changes to our project and made an interface computer with two classes: laptop and desktop. Both of these classes implement the compile method from the interface computer.

- The classes dev, desktop, and laptop each have a bean tag in our XML file. The bean IDs for these tags are dev, desk1, and lap1.

  Now, since Computer is an interface, we need to make an object reference for its implementing class, which is laptop. To do this, let's add the reference attribute as laptop to the dev bean tag inside the property and see what happens.



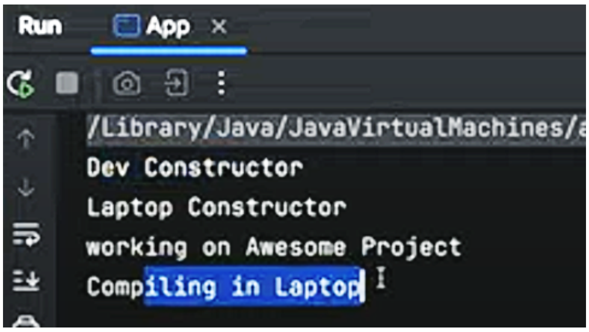
```
    <bean id="dev" class="com.telusko.Dev">
        <property name="com" ref="lap1" />
    </bean>


    <bean id="lap1" class="com.telusko.Laptop">
    </bean>



</beans>
```

- **Output**

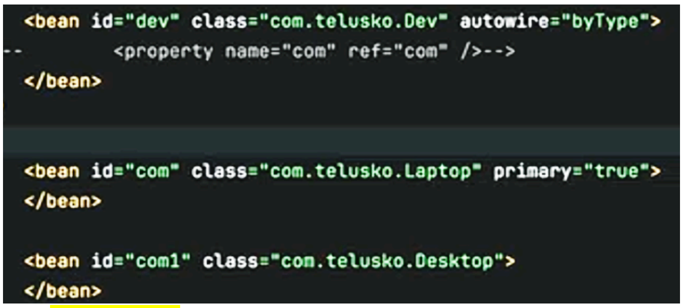


```
/Library/Java/JavaVirtualMachines/a
Dev Constructor
Laptop Constructor
working on Awesome Project
Compiling in Laptop
```

- In the same way, we can add desktop as a reference. Also, now the other way of doing it is to ask Spring to do it automatically, like we did in SpringBoot with the **@Autowired** annotation. In the same way, inside the dev class bean tag, just add the autowire element.

  As we know, autowiring can be done **byType**,**byName** and also by **index**

  Follow the below syntax to make it simple, as classes laptop and desktop are implementing computer so both are of type computer which will be chosen for this ambiguity lets add another element inside bean tag of either of the classes that needs to be selected as **primary,** just as we did in boot style, just in XML format and using Spring.



```
<bean id="dev" class="com.telusko.Dev" autowire="byType">
--        <property name="com" ref="com" />-->
</bean>


<bean id="com" class="com.telusko.Laptop" primary="true">
</bean>

<bean id="com1" class="com.telusko.Desktop">
</bean>
```

- Now we learned how spring runs behind the scenes; further,futher let's move to spring MVC in the next chapter.