31 to 1 march Video:-

Tuesday, May 24, 2022 12:58 PM

Constructor injection:- Whenever we enjecting dependency throw constructor then it is called constructor injection.
Setter Method Base injection:- Whenever we are enjecting dependency throw setter method then it is called setter method based injection.

When one object depend upon another object then if we want to add one more property into item then we have to be also change into the order object we have to set that property data into order object so that is called hardcoded java program but this thing is not good that if we change one object data then we have to be change manually into dependent object so spring core not allow us to do so in spring core it will be happen automatically.

If we do not want to hard code data then we should use beans.xml file. Because in future if we want to change price of item we have to change into only one file that is called beans.xml file.

For more information go into the spring core file in shared location on git ..

IOC Container:- our order class have dependency on item class so initialize the order throw constructor or setter method so this is the wrong this is the hardcoded. So better way we should adapted we configure this object into a xml file and xml file specify <property> dependency. When we apply this dependency it will automatically inject into the initializer automatically injected so whenever you injected the dependency throw generic programs this is called as IOC Container.

IOC is says we should write a xml file there we mention our beans and give it to xml file I will find out what interdependency in this xml file we have and write such generic code using reflection (Class.forName()) we should injecting the item into the order automatically And give us injected code hence I can say this program is called IOC Container.

Note:- spring framewrok provides us a IOC Container. And our jobs only write xml file.

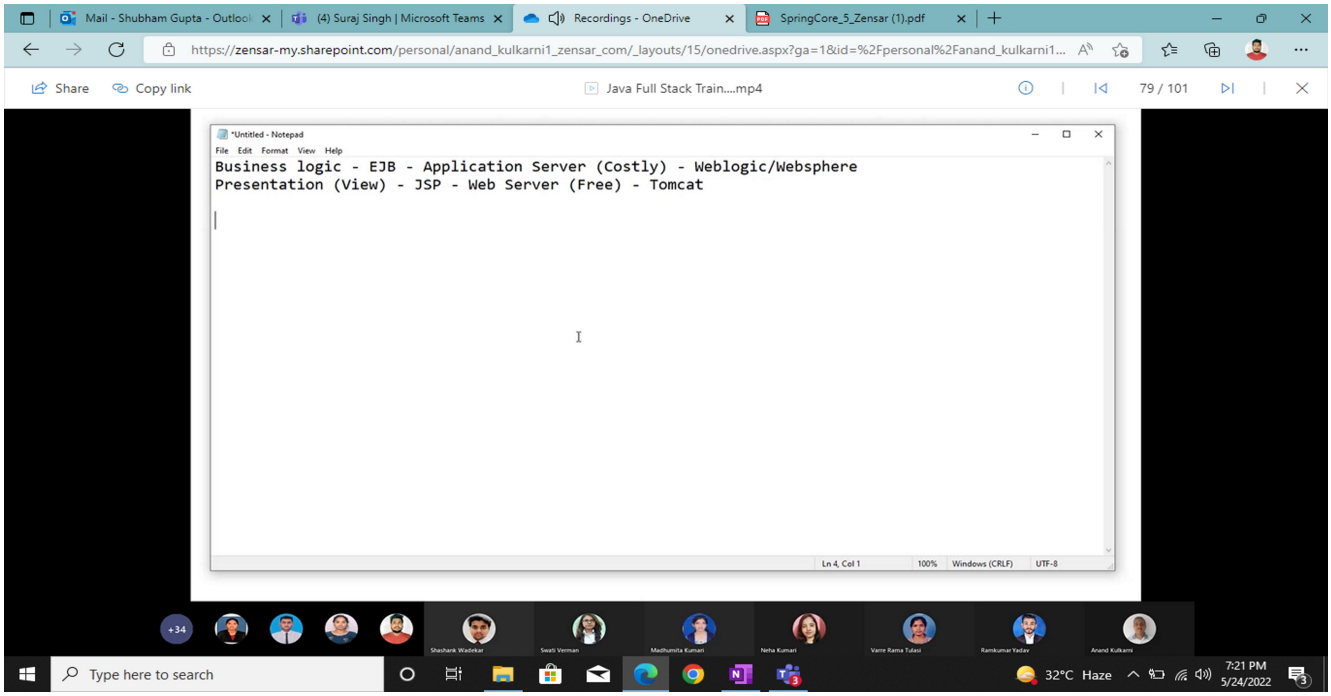
History :- Spring framework is an open source Java platform and it was initially written by Rod Johnson and was first released under the Apache 2.0 license in June 2003.

Business logic -> EJB -> Application Server (costly) -> weblogic/webspare.
Presentation (View) -> JSP ->Web Server -> Tomcat.

In earlier day when spring framework had not come then when we had to written complex query or business logic using EJB(Enterprise Java Bean) and then we had to used Application Server and Application server cost is very high so this was the major drawback small company to use Application server but after spring framework have come into picture then we can write complex business logic into the JSP(Java Server Pages) and we can also use.
Web Server like Tomcat.

Application Server:- it is having web server feature as well as lots of built in services like collection pulling services, Jindia services.. and it very costly..

web server:- whenever any normal server we will take which will handle http request coming from the browser and send the http response to the client this is called simple web server. And web server is commercially free.



Benefits of Using Spring Framework :----

1) Spring enables developers to develop enterprise-class applications using POJOs. No need of EJB container.
2) It truly makes use of some of the existing technologies like several ORM frameworks, logging frameworks, JEE, Quartz and JDK timers, other view technologies.
3) Spring's web framework is a well-designed web MVC framework.
4) Spring provides a convenient API to translate technology-specific exceptions (thrown by JDBC, Hibernate, or JDO, for example) into consistent, unchecked exceptions.
5) Lightweight IoC containers tend to be lightweight, especially when compared to EJB containers, for example. This is beneficial for developing and deploying applications on computers with limited memory and CPU resources.
6) Spring provides a consistent transaction management interface that can scale down to a local transaction (using a single database, for example) and scale up to global transactions (using JTA, for example).

Let see how to create spring project let see the steps click on

File -> New -> maven project -> catlog(intarnal) -> choose artifact id -> and give the project name and group id …….

Add below dependency into POM file

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>5.3.9</version>
        </dependency>
```

After created project right click on project change jdk int 1.8

And create a "resource" folder into src -> main folder and inside the resource folder we have to create xml file

Why we are putting xml file inside the main folder because spring core take it automatically from src-> main folder
Hame batane ki jarurat ni padti h ki xml file kha h…

Go to the main app.java file and mention the IOC container ClassPathXmlApplicationContext("beans.xml");
This is the Ioc container it is return reference

```
AbstractApplicationContext iocContainer = new ClassPathXmlApplicationContext("beans.xml");
Item item =(Item) iocContainer.getBean("itemBeans");
System.out.println("Item:- "+item);
```

<Bean id="itemBean" class="com.zensar.Item">
<Property name="name" value="Laptop"></Property>
<Property name="price" value="7000"></Property>
</Bean>

Type of IOC Container:----------

Spring IoC containers:-
1) BeanFactory container This is the simplest container providing basic support for DI and defined by theorg.springframework.beans.factory.BeanFactory interface.

2) ApplicationContext container This container adds more enterprise-specific functionality such as the ability to resolve textual messages from a properties file and the ability to publish application events to interested event listeners. This container is defined by the org.springframework.context.ApplicationContext interface. The ApplicationContext container includes all functionality of the BeanFactory container.

BeanFactory IoC container:--
XmlBeanFactory factory = new XmlBeanFactory (new ClassPathResource("Beans.xml"));
HelloWorld obj = (HelloWorld) factory.getBean("helloWorld");
obj.getMessage();

ApplicationContext IoC container:---

The Application Context is spring's more advanced container. The ApplicationContext includes all functionality of the BeanFactory, it is generally recommended over the BeanFactory. The most commonly used ApplicationContext implementations are. it is a interface and it Has many implementation classes like AbstractApplicationContext and AbstractApplicationContext is not a normal class it is also a Abstract class.
Any classes which is implementing ApplicationContext is called IOC Container.
These three are famous implementation of ApplicationContext.
1) FileSystemXmlApplicationContext (this is use when our xml file present at different location not present inside the project dir).
2) ClassPathXmlApplicationContext &(this is use when our xml file present inside the project dir)
3) WebXmlApplicationContext (this is use when we would create dynamic web application)

FileSystemXmlApplicationContext and WebXmlApplicationContext is not use mostly use.

ApplicationContext Implementations :--

1) FileSystemXmlApplicationContext: This container loads the definitions of the beans from an XML file. ApplicationContext context = new FileSystemXmlApplicationContext (“c:/Beans.xml");

2) ClassPathXmlApplicationContext: This container loads the definitions of the beans from an XML file using CLASSPATH.
ApplicationContext context = new ClassPathXmlApplicationContext("Beans.xml");

3) XmlWebApplicationContext: This container loads the XML file with definitions of all beans from within a web application.
WebApplicationContext ctx = WebApplicationContextUtils.getWebApplicationContext(servletContext);

<Property name="name" value="Laptop"></Property> this is called setter method injection.
<constructor-args name="name" value="Laptop"></constructor-args> this is called constructor based injection.

Lazy-Initialization:- jab hm getBean() method ko call krte h to us time pe agar bean initialize hota h to usse hm lazy-Initialization khte h. if we want to make any bean lazy than we can write like below in bean tag.if we use lzay-init ="true" it's mean that

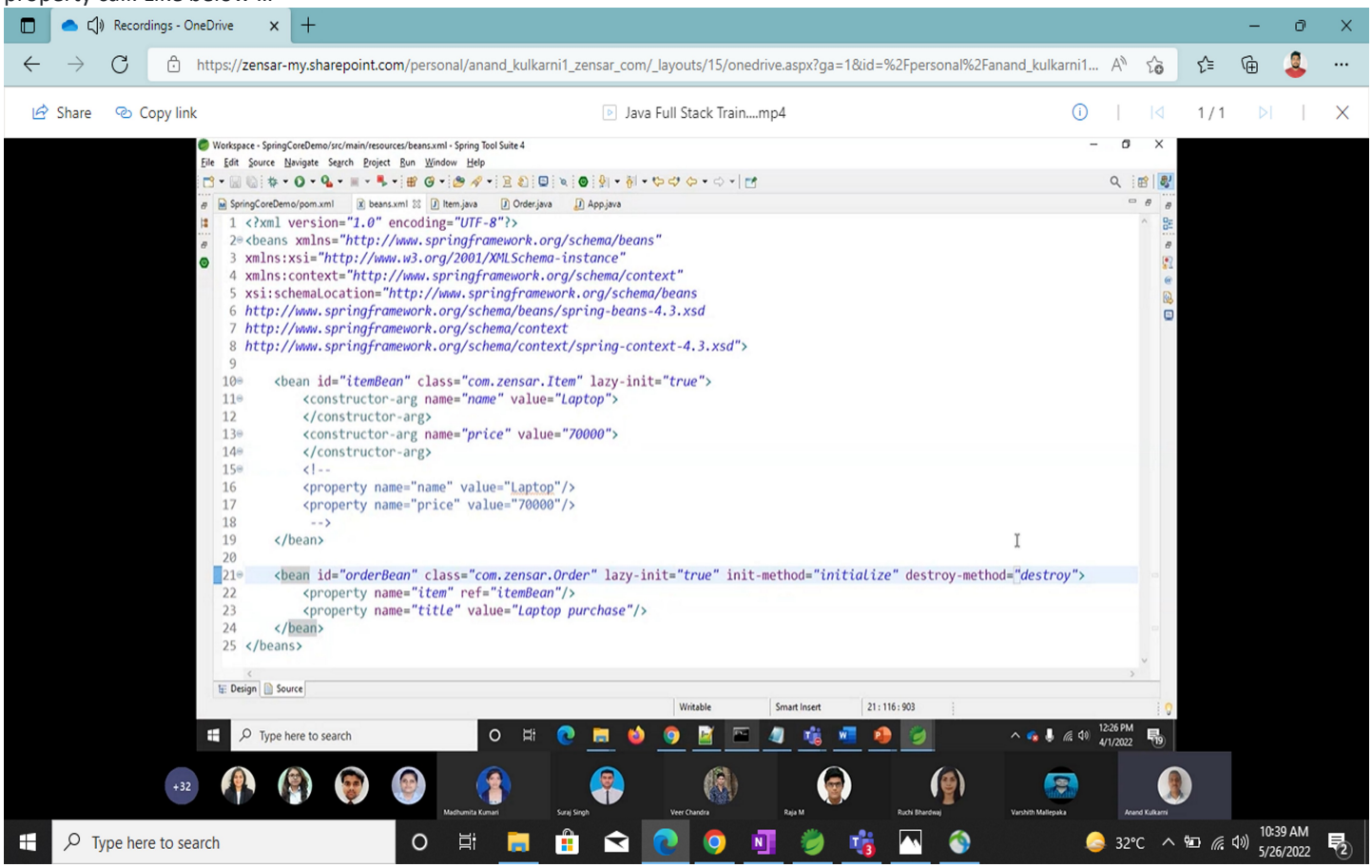
<Bean id="itemBean" class="com.zensar.Item" lazy-init="true">
<Property name="name" value="Laptop"></Property>
<Property name="price" value="7000"></Property>

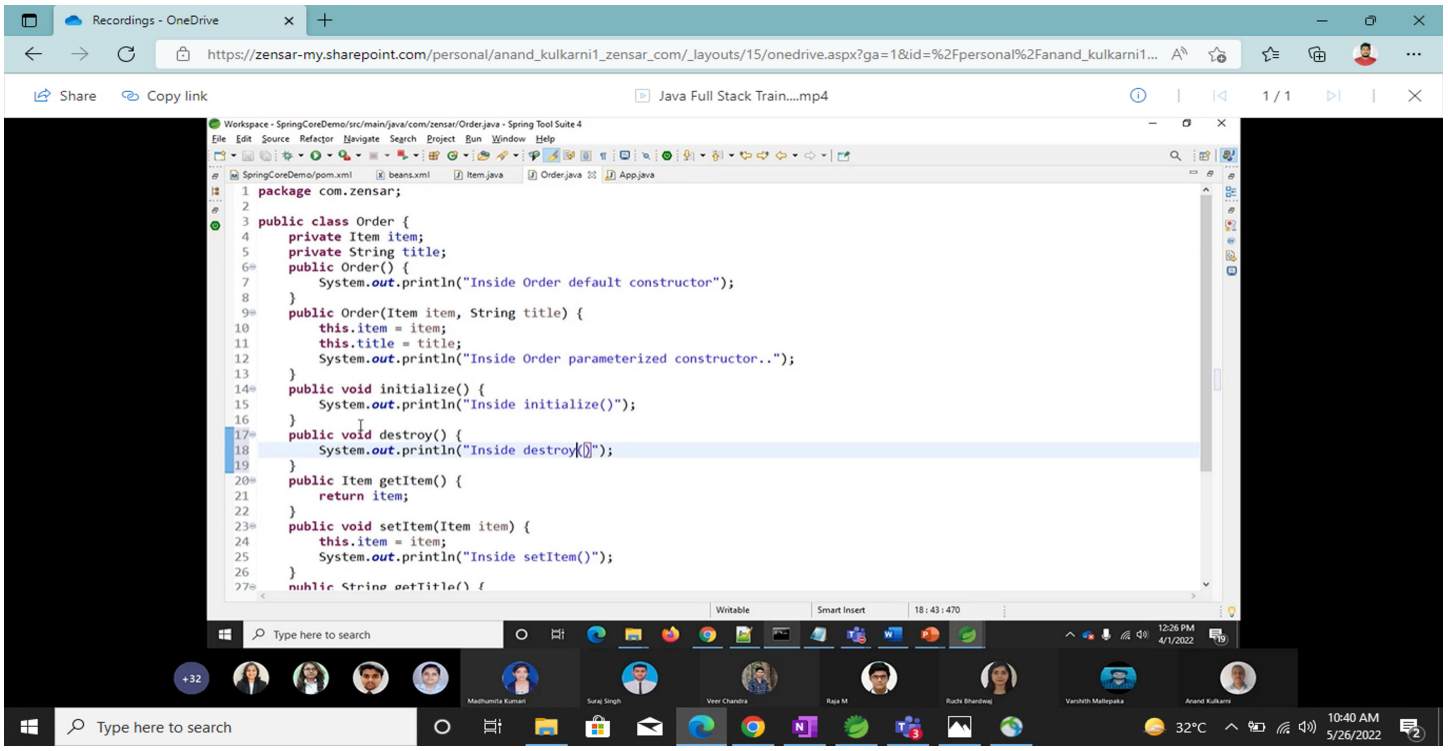
</Bean>

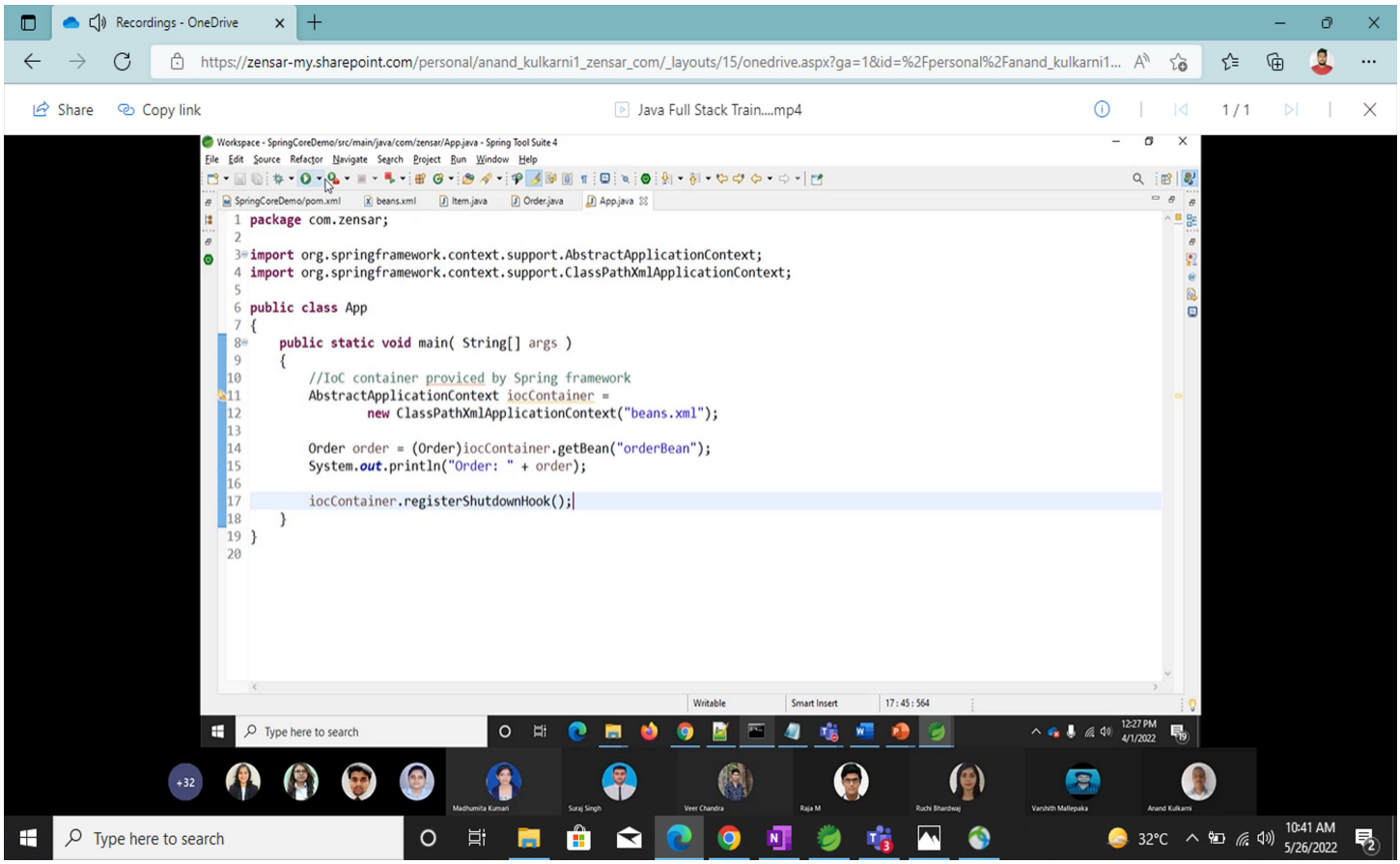
Eager-Initialization:- jaise hi hm beans.xml file classPathXmlApplication me paas krte h to ussi time pe initialization ho jata h to usse Eager-Initialization khte h. and by default eager-initialization hi hota h.

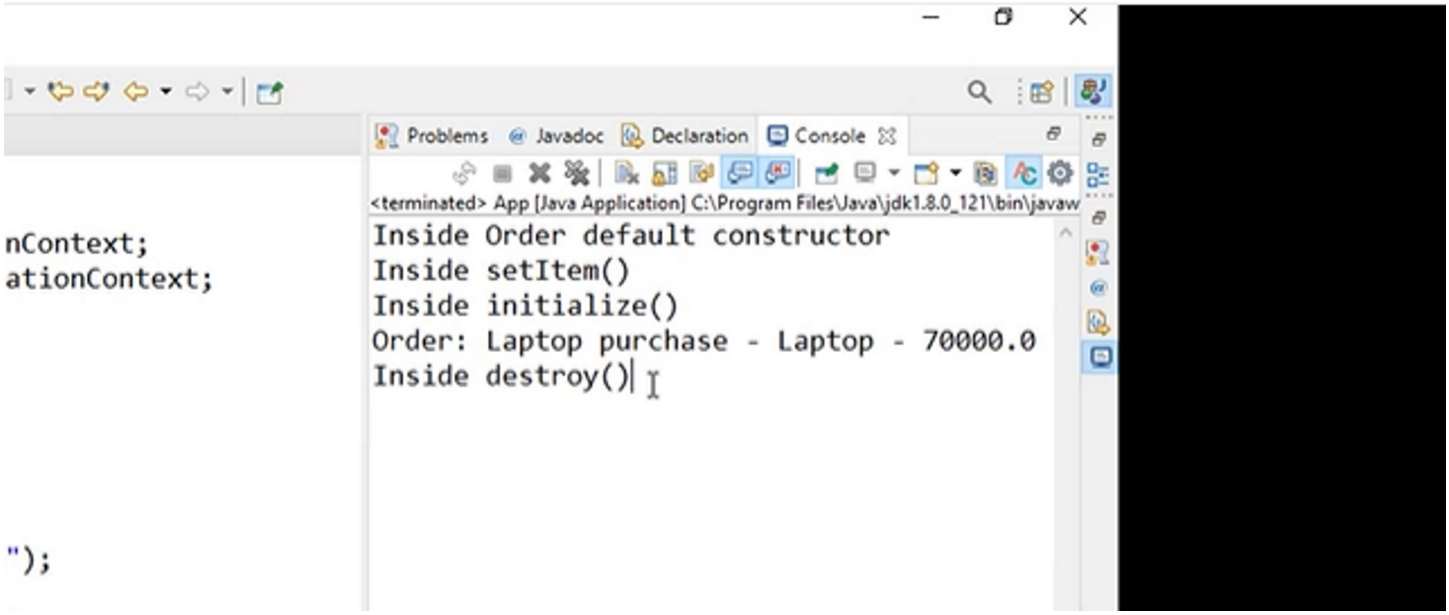
Initialize method:- if we make any bean init-method="initialize" like this it's mean initialize function have in that bean exist and that will be run when that bean all property would be set then it will automatically called like below images.
Any business logic we want to perform only once then we can go for init-method="add" . Like we can use it for getConnection from database.

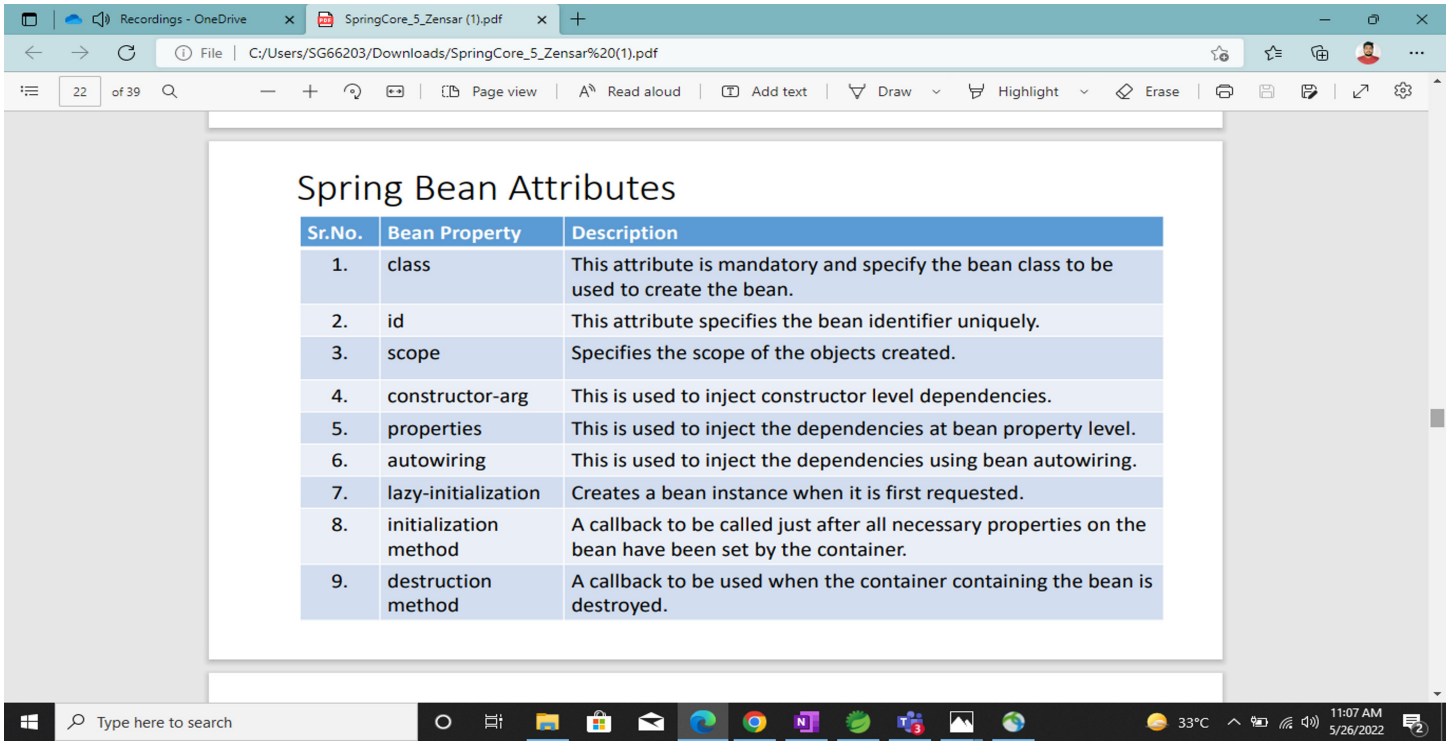
Destroy method :- if we want to close databases connections then we may go for destroy if we make bean destroy-method="destroy" then we should declare that destroy method into our class and again we have to call one more fucntion to call destroy function that is called via iocContainer.registerShutdownHook(); then destroy function will call. It call at last of the all function or property call. Like below …




Recordings - OneDrive
https://zensar-my.sharepoint.com/personal/anand_kulkarni1_zensar_com/_layouts/15/onedrive.aspx?ga=1&id=%2Fpersonal%2Fanand_kulkarni1...
Share
Copy link
Java Full Stack Train....mp4
1 / 1
Workspace - SpringCoreDemo/src/main/java/com/zensar/Order.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
SpringCoreDemo/pom.xml
beans.xml
Item.java
Order.java
App.java
1 package com.zensar;
2
3 public class Order {
4 private Item item;
5 private String title;
6 public Order() {
7 System.out.println("Inside Order default constructor");
8 }
9 public Order(Item item, String title) {
10 this.item = item;
11 this.title = title;
12 System.out.println("Inside Order parameterized constructor..");
13 }
14 public void initialize() {
15 System.out.println("Inside initialize()");
16 }
17 public void destroy() {
18 System.out.println("Inside destroy()");
19 }
20 public Item getItem() {
21 return item;
22 }
23 public void setItem(Item item) {
24 this.item = item;
25 System.out.println("Inside setItem()");
26 }
27 public String getTitle() {
Writable
Smart Insert
18 : 43 : 470
Type here to search
12:26 PM
4/1/2022
+32
Madhumita Kumari
Suraj Singh
Veer Chandra
Raja M
Ruchi Bhardwaj
Varshith Mallepaka
Anand Kulkarni
Type here to search
32°C
10:40 AM
5/26/2022


Recordings - OneDrive
https://zensar-my.sharepoint.com/personal/anand_kulkarni1_zensar_com/_layouts/15/onedrive.aspx?ga=1&id=%2Fpersonal%2Fanand_kulkarni1...
Share
Copy link
Java Full Stack Train....mp4
1 / 1
Workspace - SpringCoreDemo/src/main/java/com/zensar/App.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
SpringCoreDemo/pom.xml
beans.xml
Item.java
Order.java
App.java
1 package com.zensar;
2
3 import org.springframework.context.support.AbstractApplicationContext;
4 import org.springframework.context.support.ClassPathXmlApplicationContext;
5
6 public class App
7 {
8 public static void main(String[] args)
9 {
10 //IoC container proviced by Spring framework
11 AbstractApplicationContext iocContainer =
12 new ClassPathXmlApplicationContext("beans.xml");
13
14 Order order = (Order)iocContainer.getBean("orderBean");
15 System.out.println("Order: " + order);
16
17 iocContainer.registerShutdownHook();
18 }
19 }
20
Writable
Smart Insert
17 : 45 : 564
Type here to search
12:27 PM
4/1/2022
+32
Madhumita Kumari
Suraj Singh
Veer Chandra
Raja M
Ruchi Bhardwaj
Varshith Mallepaka
Anand Kulkarni
Type here to search
32°C
10:41 AM
5/26/2022





Spring bean configuration :-

We can configure beans in two different ways:

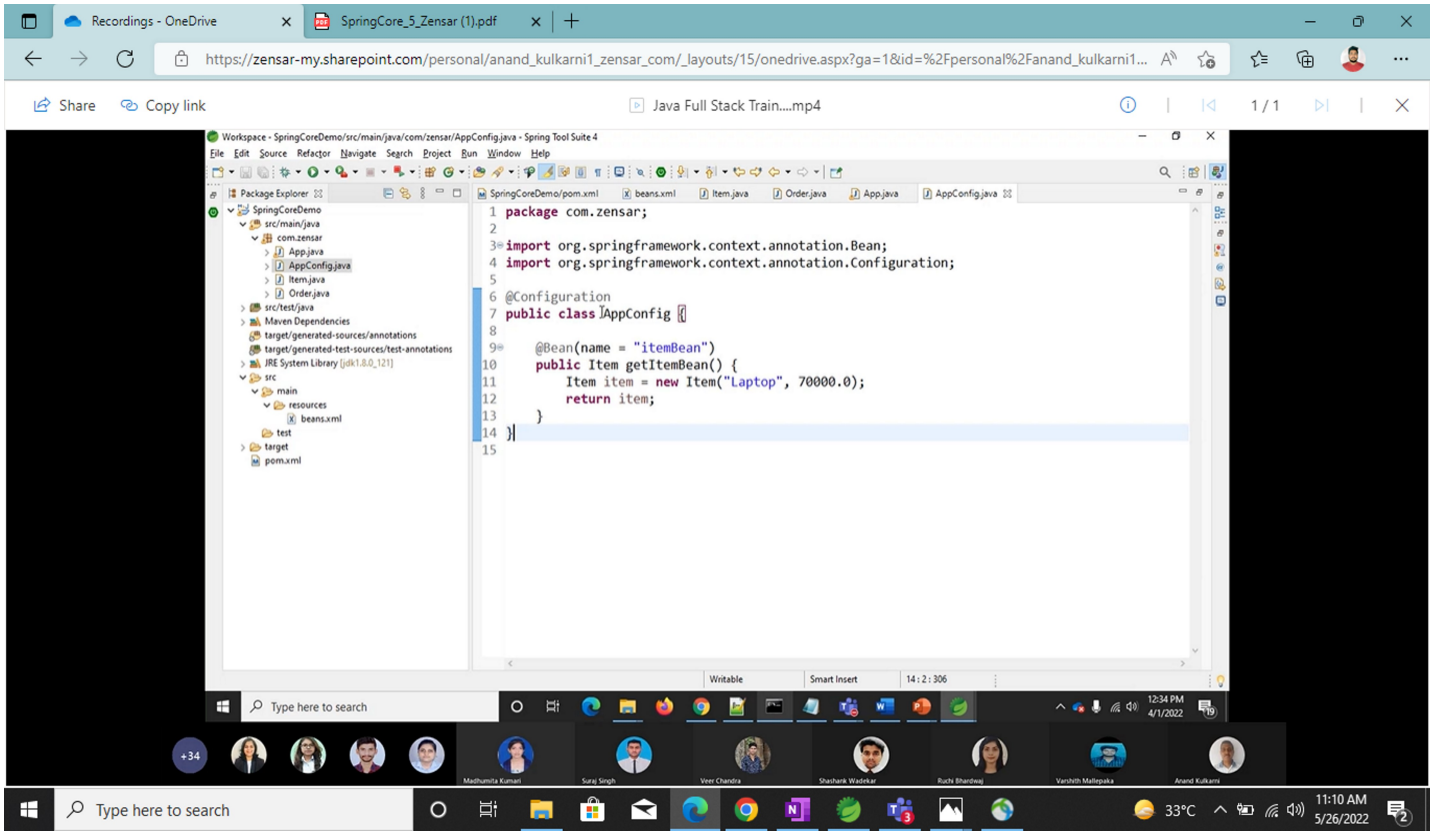
1. XML based configuration file.
2. Java Annotation based configuration.

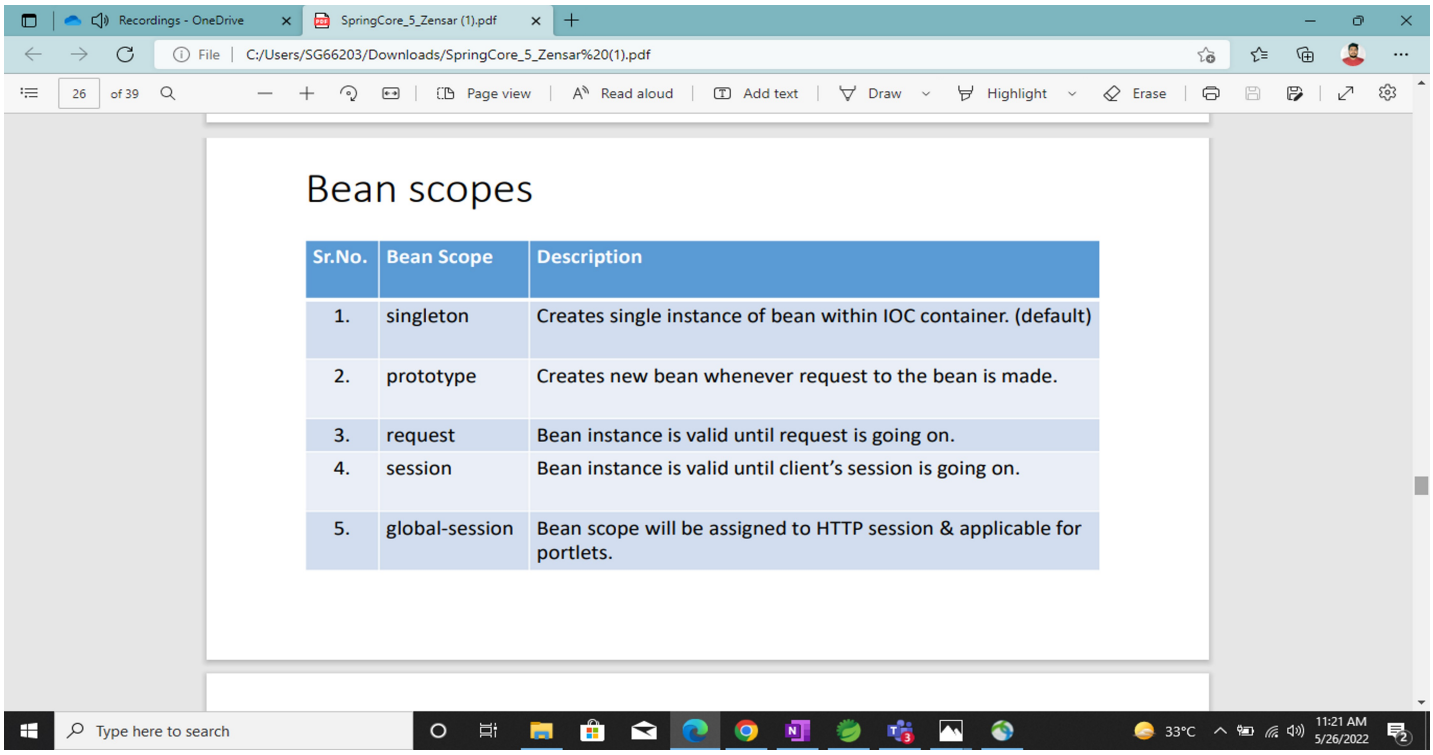
XML based configuration :- we have seen xml based configuration.

```
<Bean id="itemBean" class="com.zensar.Item" lazy-init="true">
<Property name="name" value="Laptop"></Property>
<Property name="price" value="7000"></Property>
</Bean>
```

Java Annotation based configuration:- if we use java Annotation based configuration then we no need to use beans.xml file.

```
@Configuration
public class AppConfig {

@Bean(name="messagePrinter")
public MessagePrinterBean getMessagePrinterBean()
{
return new MessagePrinterBean();
}
}
```





Requast:-- When the cient makes request to your bean whichever components is trying to access our order bean all of them is going to share The same bean instance.

Session and global-session rarely use and also request onlyb two scope is important singleton and prototype.

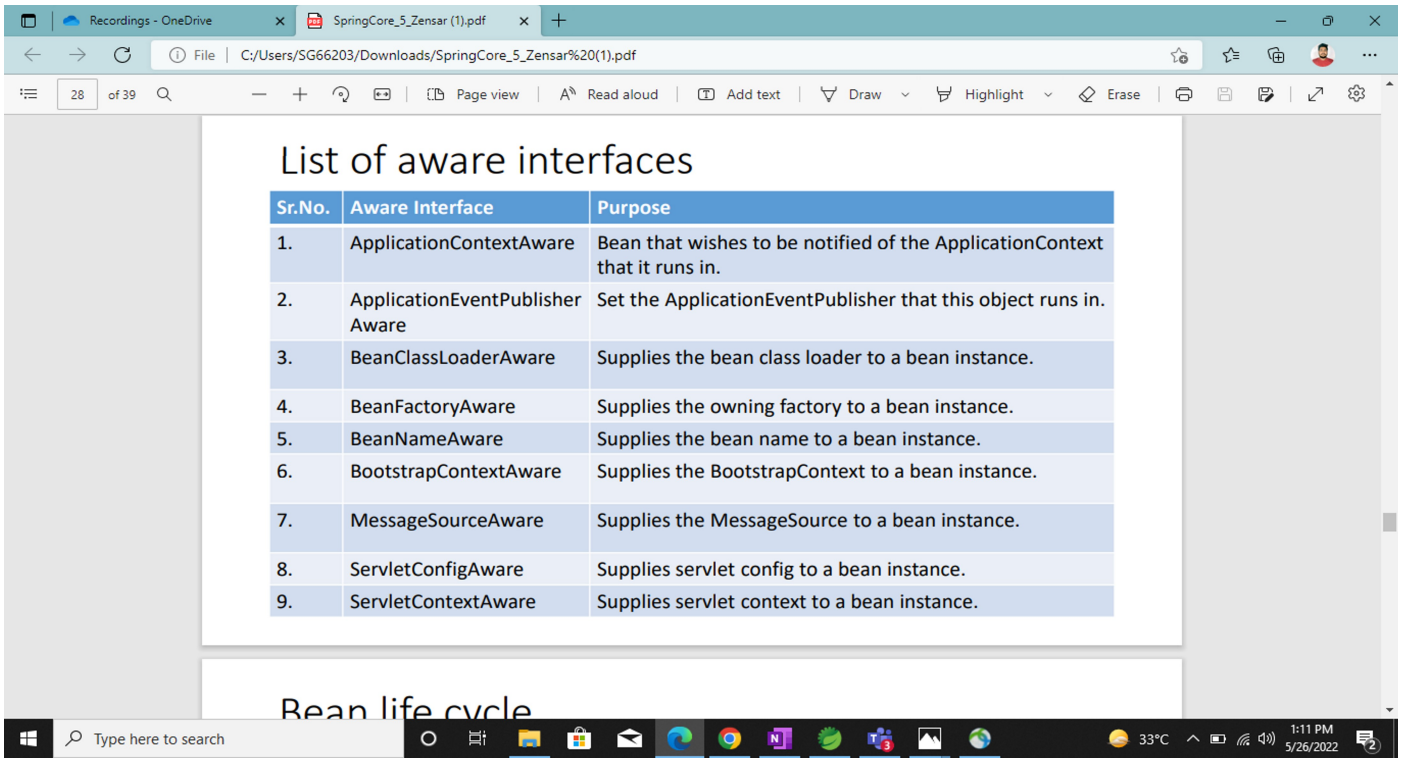
Aware interfaces:-

• Spring offers a range of Aware interfaces that allow beans to indicate to the container that they require a certain infrastructure dependency.

• Each interface will require you to implement a method to inject the dependency in bean.

• For example:

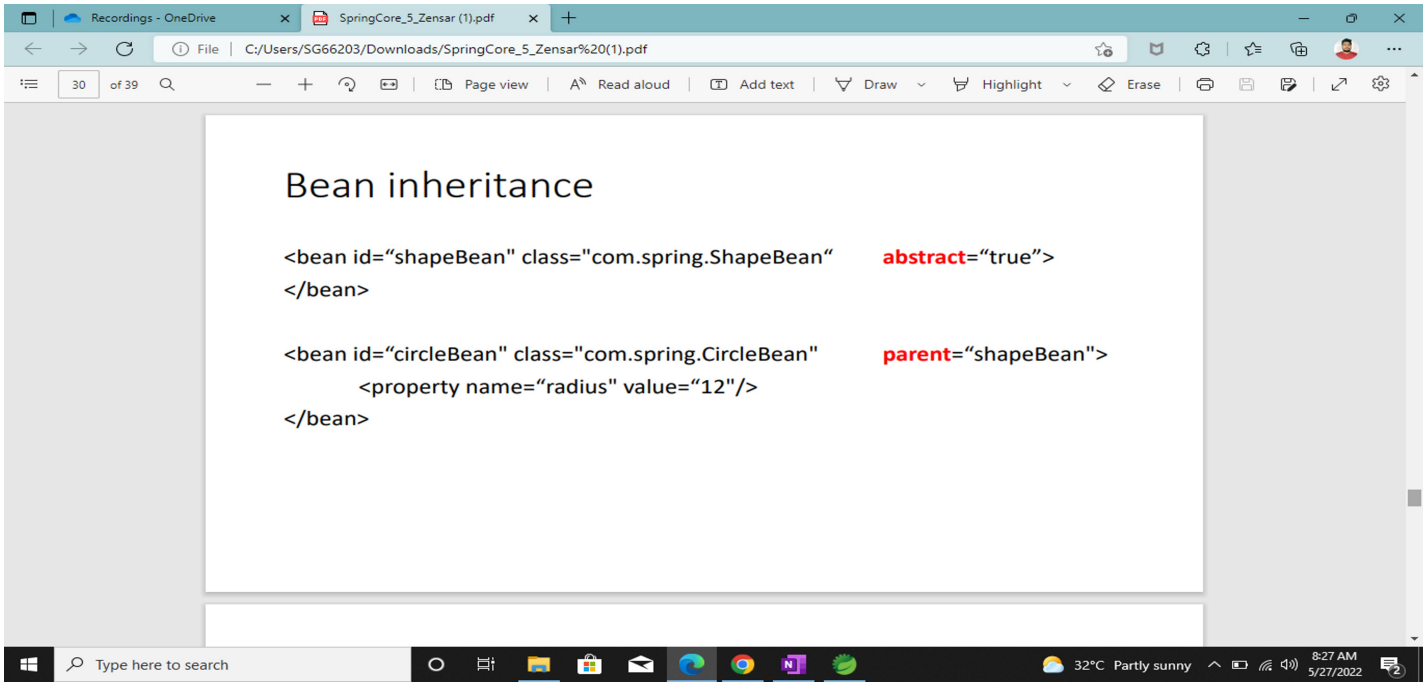
```
public class MessagePrinterBean implements BeanNameAware {
 public void setBeanName(String beanName) {
System.out.println("Bean Name: " + beanName);
 }
 }
```



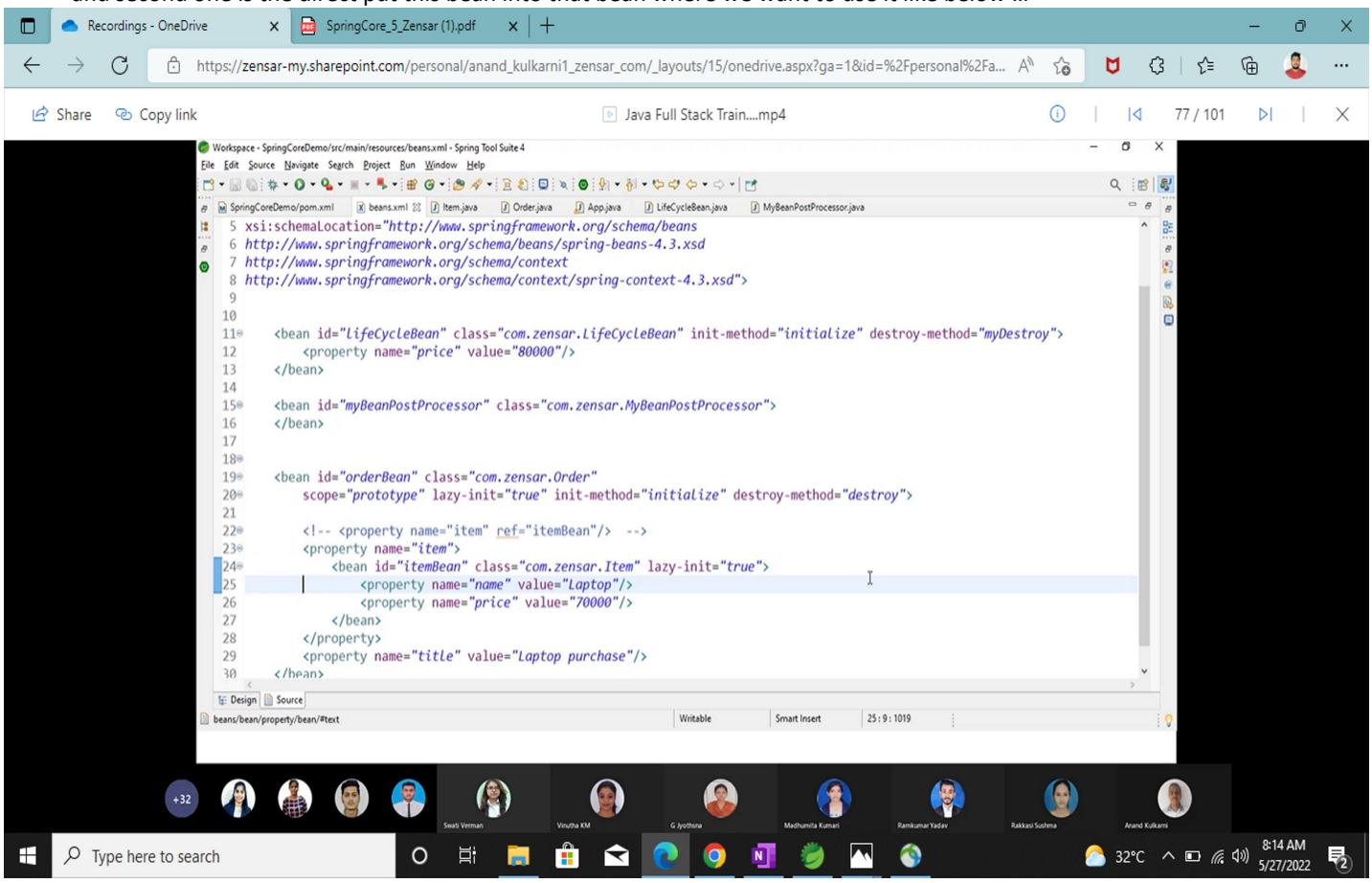
Bean life cycle

1. Reads bean configuration from xml file.
2. Creates instance of bean class.
3. Calls setter methods on bean to set the specified properties.
4. Calls aware interface methods on bean if it implements any.
5. If BeanPostProcessor is associated then calls postProcessBeforeInitialization() method on implementing class.
6. If bean implements InitializingBean interface then calls afterPropertiesSet() method on bean.
7. If bean configuration has "init-method" or @PostConstruct specified then it calls the defined init method on bean.
8. If BeanPostProcessor is associated then calls postProcessAfterInitialization() method on implementing class.
9. If bean implements DisposableBean interface then it calls destroy() method on bean.
10. If bean configuration has "destroy-method" or @PreDestroy specified then it calls the defined destroy method on bean.

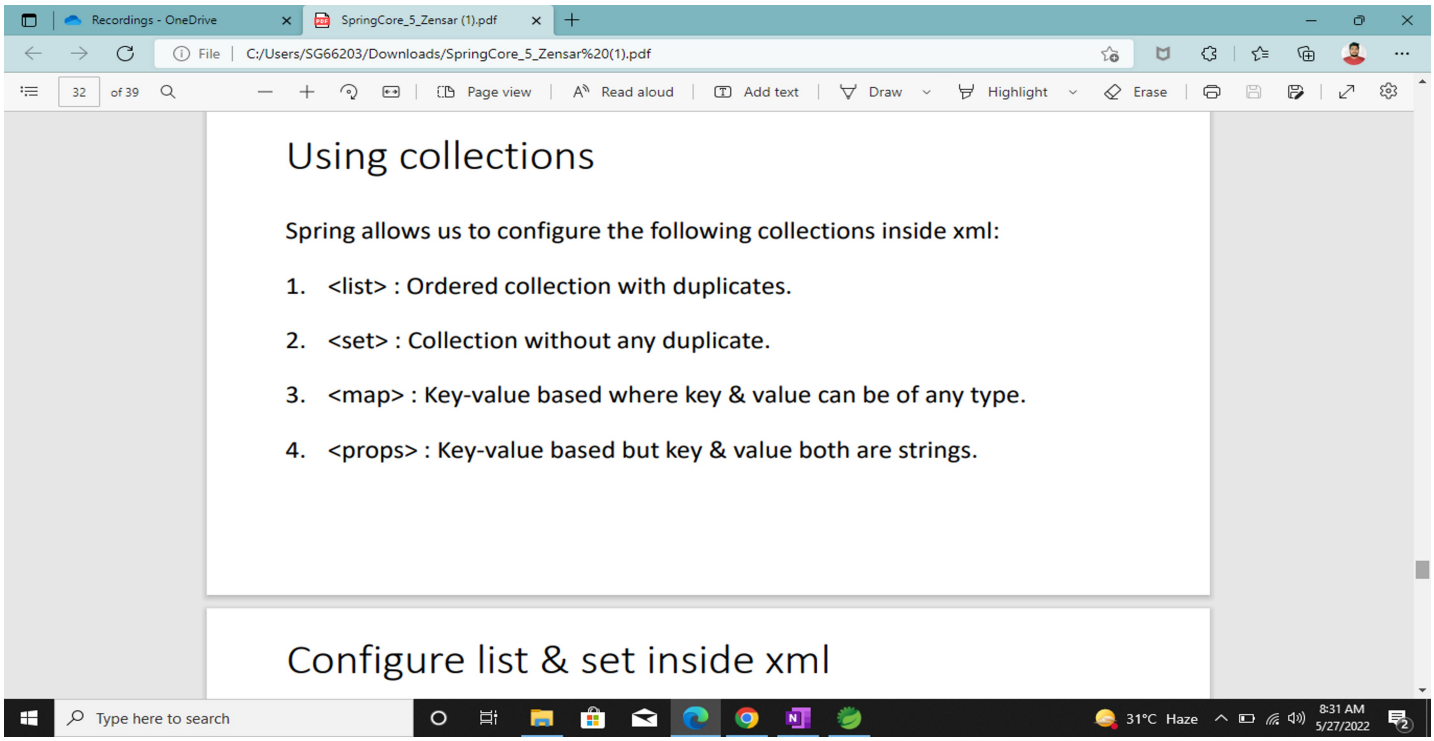
Bean inheritance :- if we want inheritance then we can do also…

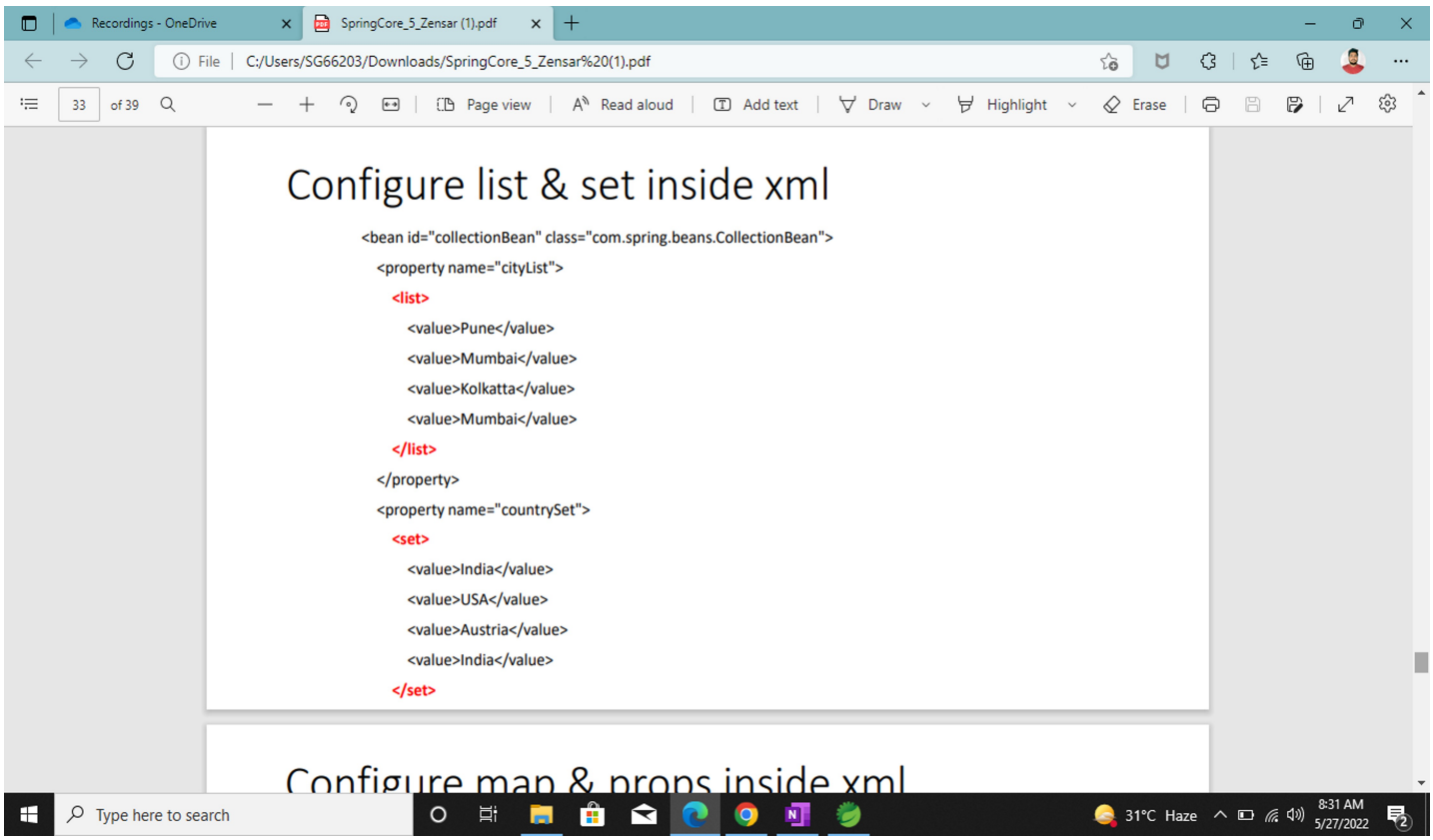


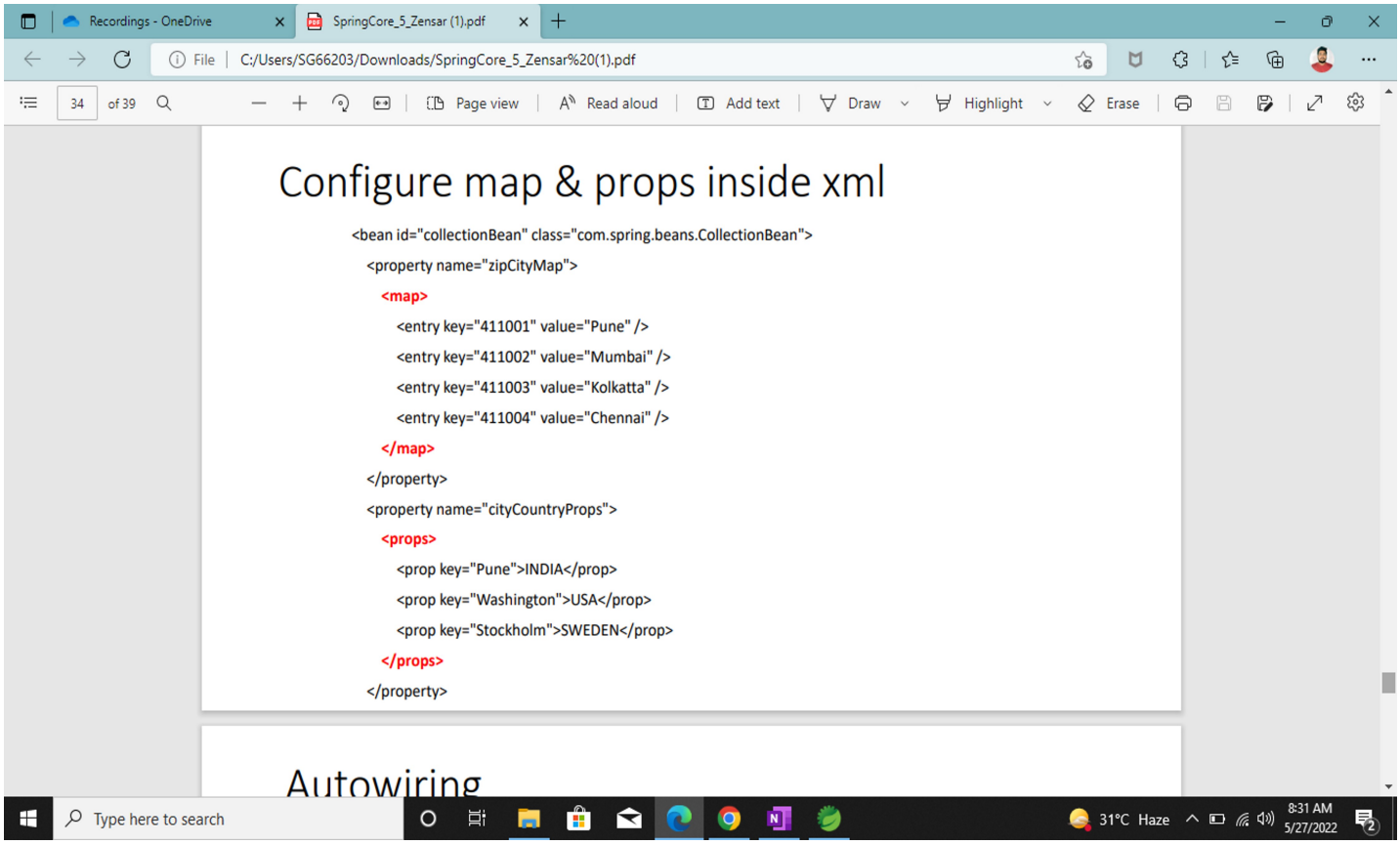
Note:- if we not want to use another class object bean into other class bean then we can use one way is to using ref variable and second one is the direct put this bean into that bean where we want to use it like below …

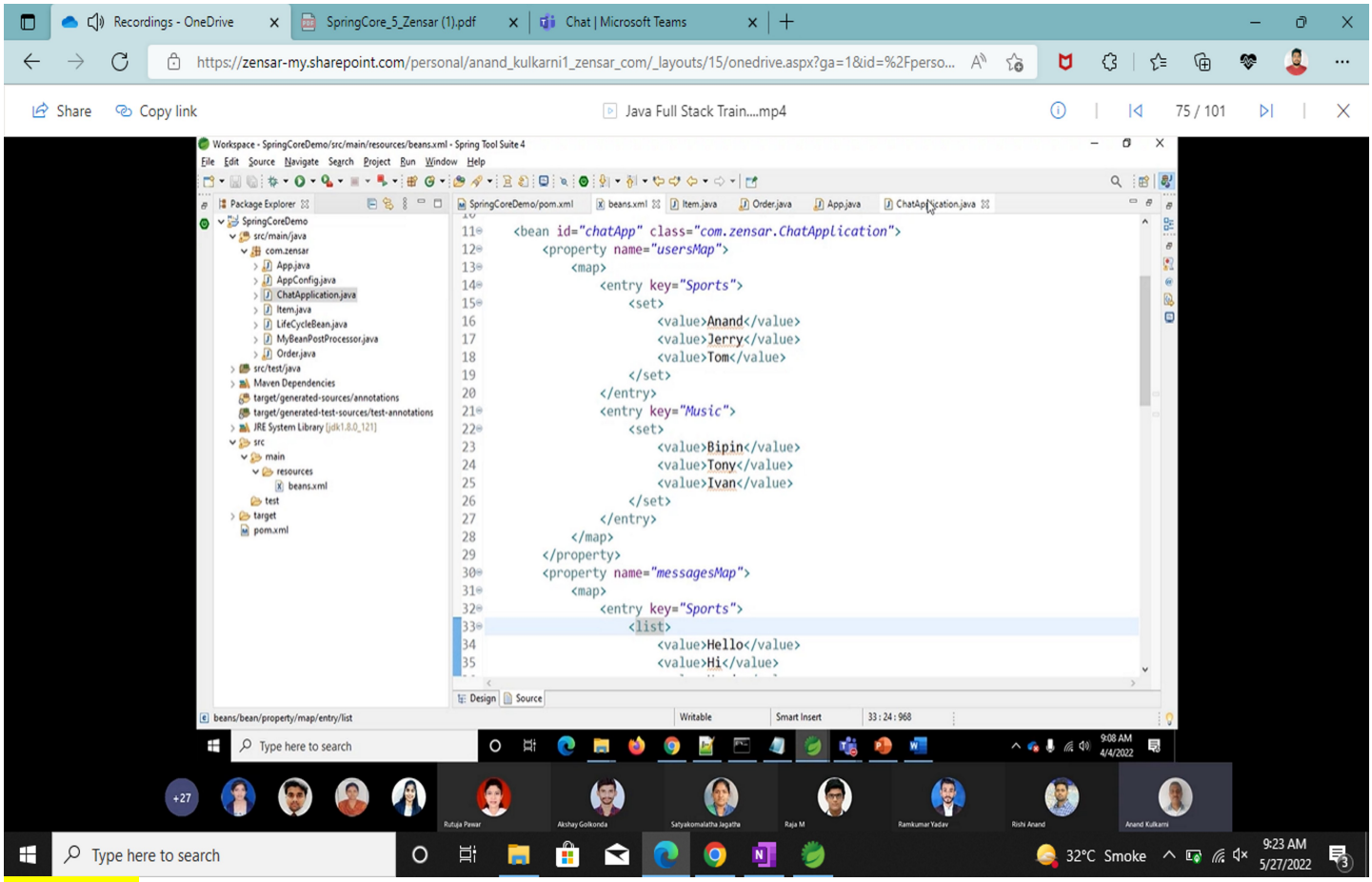


In above image we can see we have used item bean into order bean directly……


Recordings - OneDrive
SpringCore_5_Zensar (1).pdf
File | C:/Users/SG66203/Downloads/SpringCore_5_Zensar%20(1).pdf
32 of 39
Page view
Read aloud
Add text
Draw
Highlight
Erase
Using collections
Spring allows us to configure the following collections inside xml:
1. <list> : Ordered collection with duplicates.
2. <set> : Collection without any duplicate.
3. <map> : Key-value based where key & value can be of any type.
4. <props> : Key-value based but key & value both are strings.
Configure list & set inside xml
Type here to search
31°C Haze
8:31 AM
5/27/2022


Recordings - OneDrive
SpringCore_5_Zensar (1).pdf
File | C:/Users/SG66203/Downloads/SpringCore_5_Zensar%20(1).pdf
33 of 39
Page view
Read aloud
Add text
Draw
Highlight
Erase
Configure list & set inside xml
<bean id="collectionBean" class="com.spring.beans.CollectionBean">
<property name="cityList">
<list>
<value>Pune</value>
<value>Mumbai</value>
<value>Kolkatta</value>
<value>Mumbai</value>
</list>
</property>
<property name="countrySet">
<set>
<value>India</value>
<value>USA</value>
<value>Austria</value>
<value>India</value>
</set>
Configure map & props inside xml
Type here to search
31°C Haze
8:31 AM
5/27/2022





Autowiring :-

Spring framework provides facility to inject a bean into another bean implicitly which is known as 'Autowiring'.

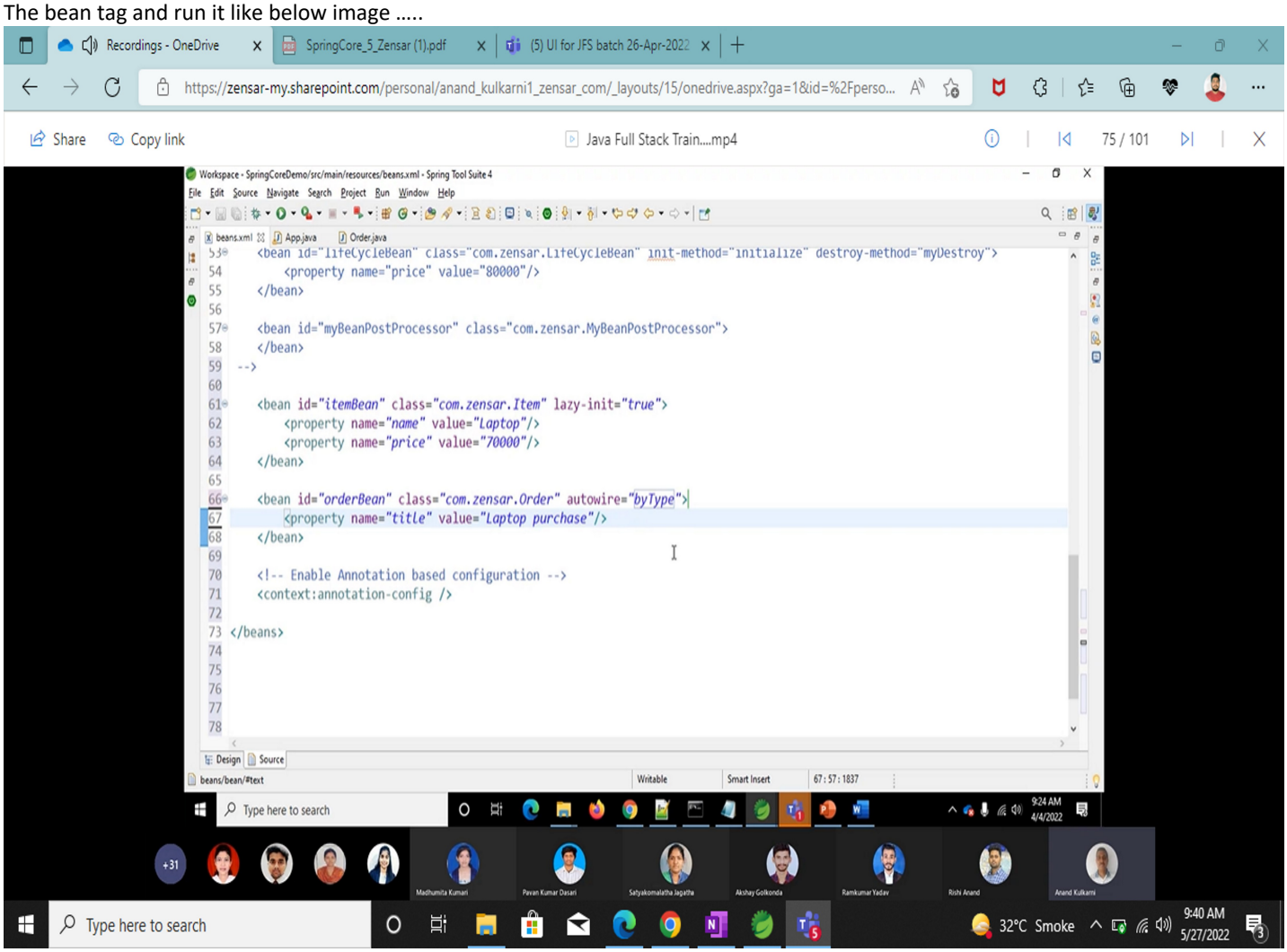
Thus, Autowiring allows Spring to resolve and inject collaborating beans into our bean using Ioc Container. Suppose, UserController bean

has dependency on UserService bean then autowiring is done as below:

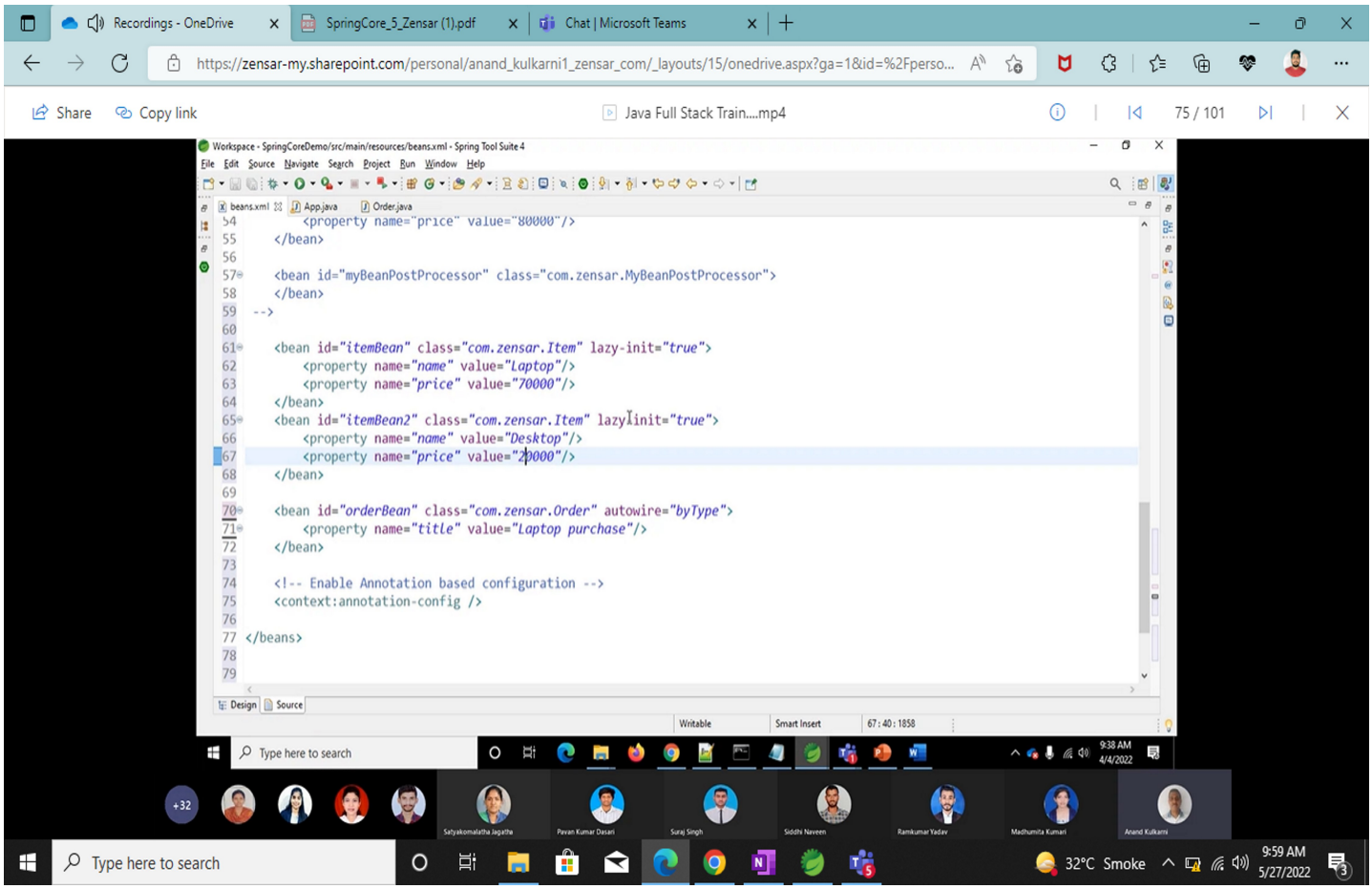
```
 public class UserController {

@Autowired
UserService userService;

}
```

Types of Autowiring :-
Autowiring is divided into following types:
1. autowire byType
2. autowire byName
3. autowire by constructor
4. @Autowired annotation
5. @Qualifier annotation

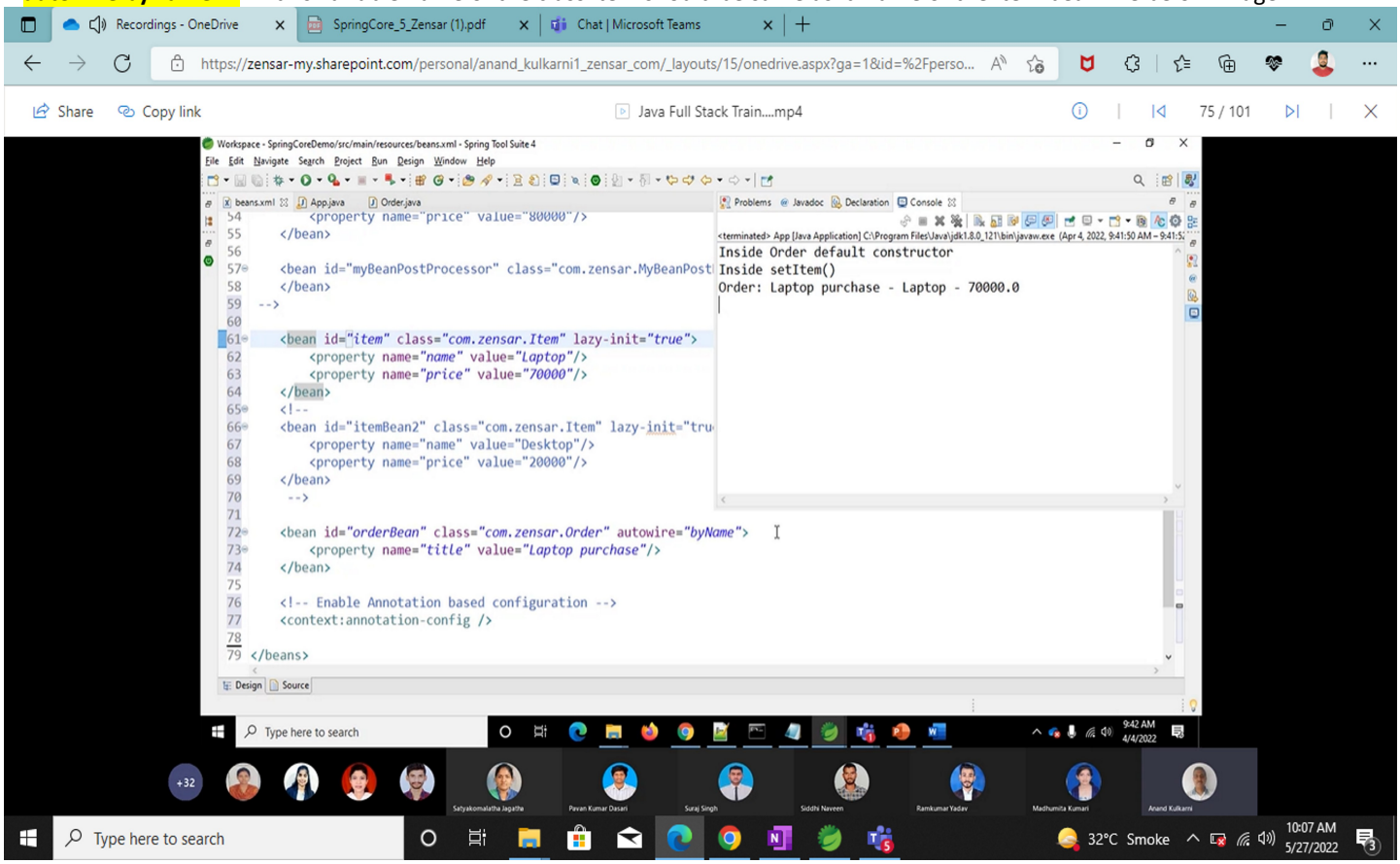
1. **autowire byType :-** if we want to inject item object into order object then we can use autowire="byType" into bean tag and this will search that type into the xml file and if ioc container get the same type of object name then it will put it into the The bean tag and run it like below image …..



But in byType has a problem when we are using two or more item bean then it will not distinct that which one it has to insert into order bean because both bean is same type like below image have two bean it name is itemBean and itemBean2 it will give error ….

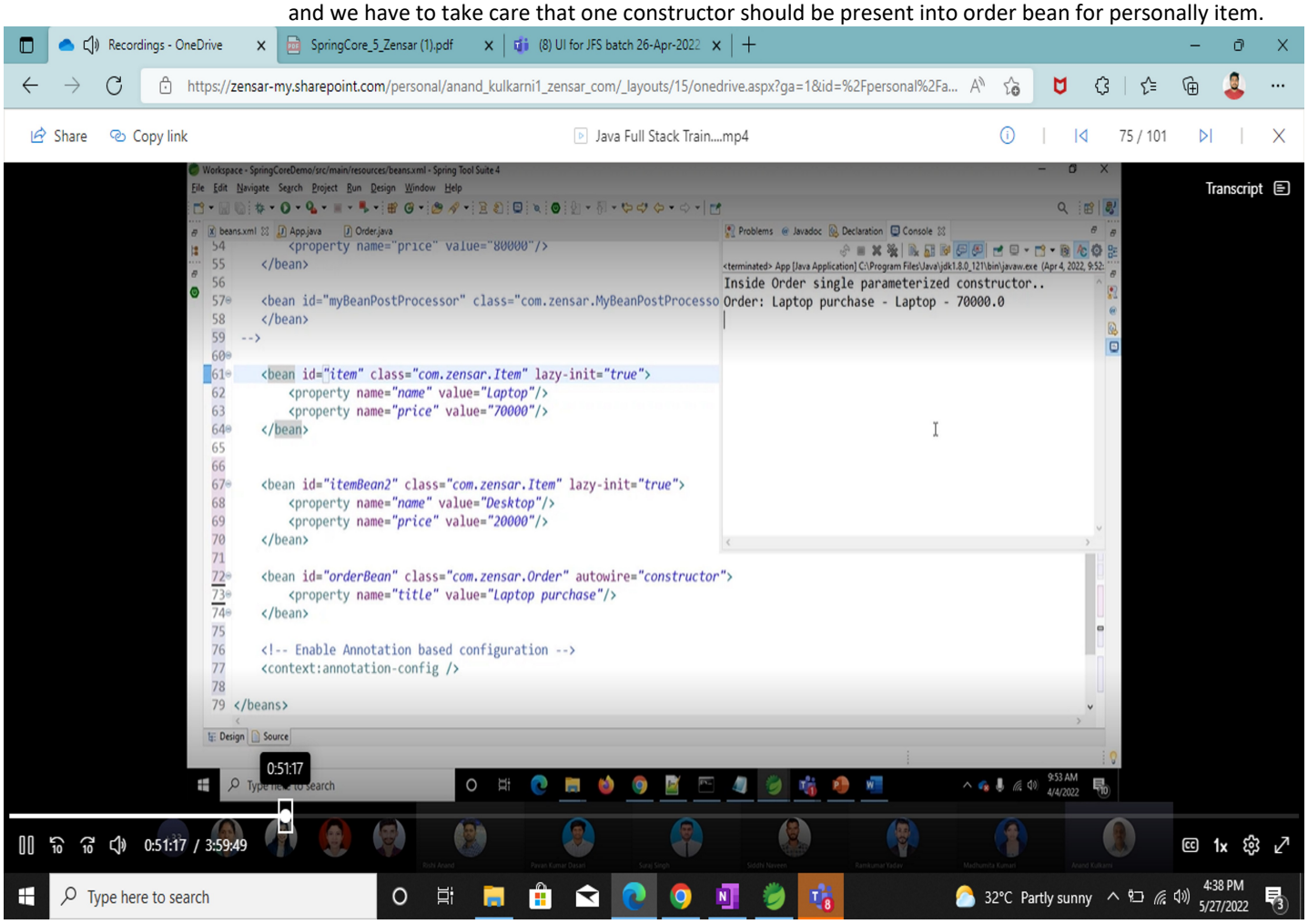


2.**autowire byName :-** in this variable name of the class item should be same as id name of the item bean like below image ....

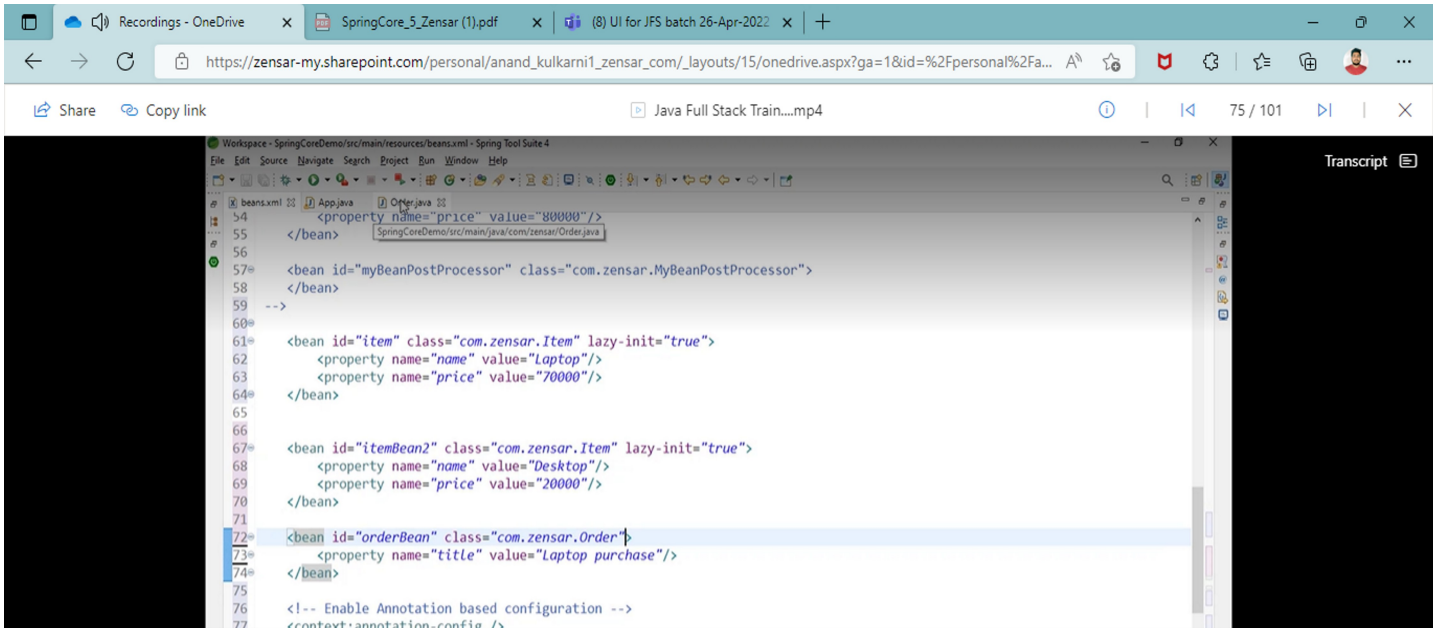


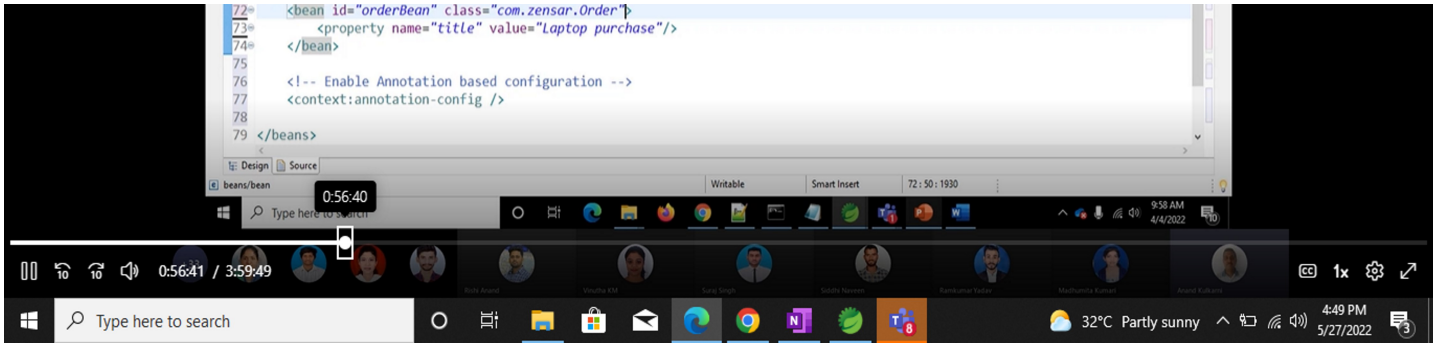
In the above image we can see that id name is same of the class variable name as a item then we can use autowire="byName".

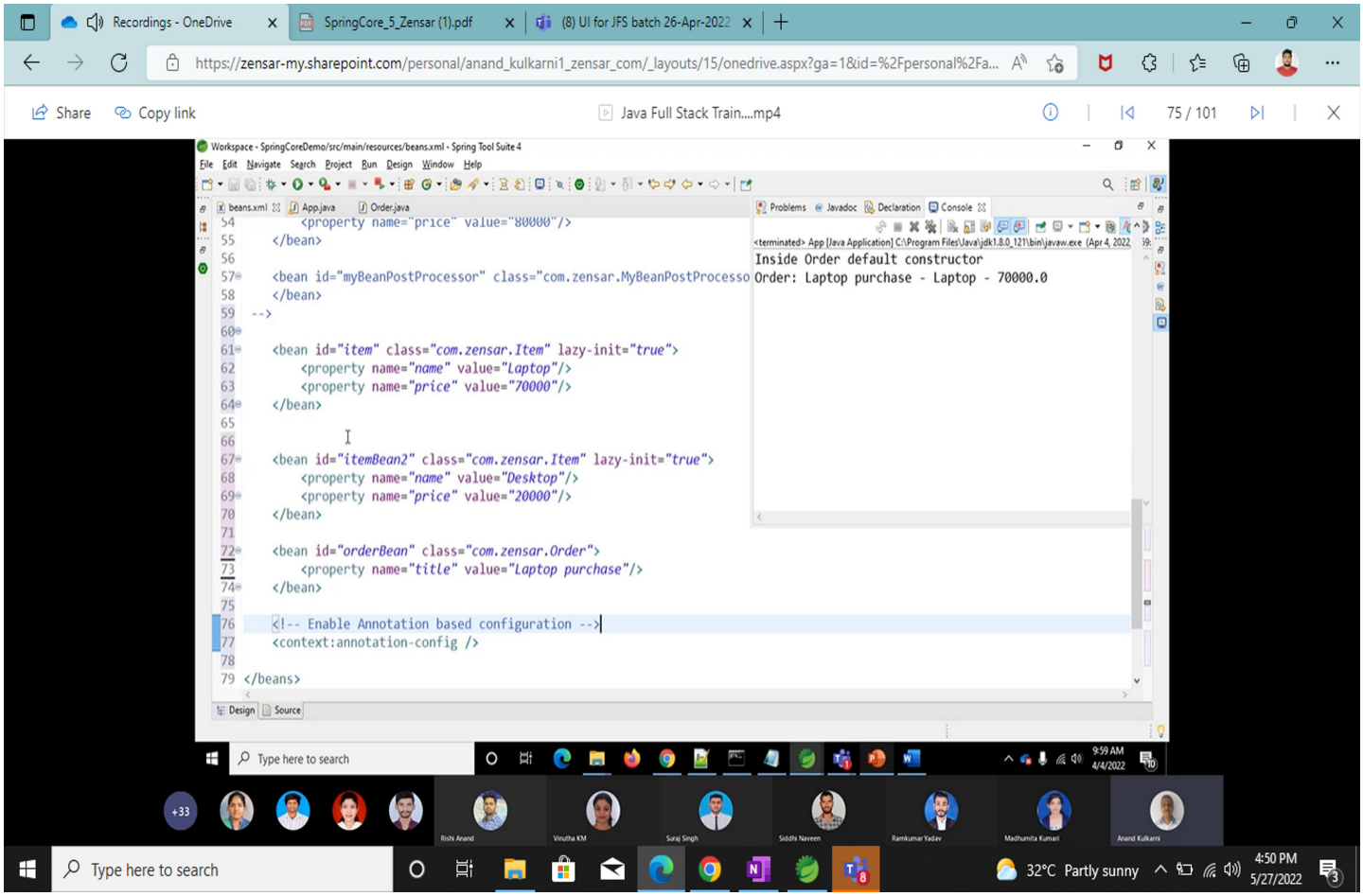
3. **autowire by constructor :-** by constructor is combination of both byType or byName in this use simply autowaire="constructor" and we have to take care that one constructor should be present into order bean for personally item.

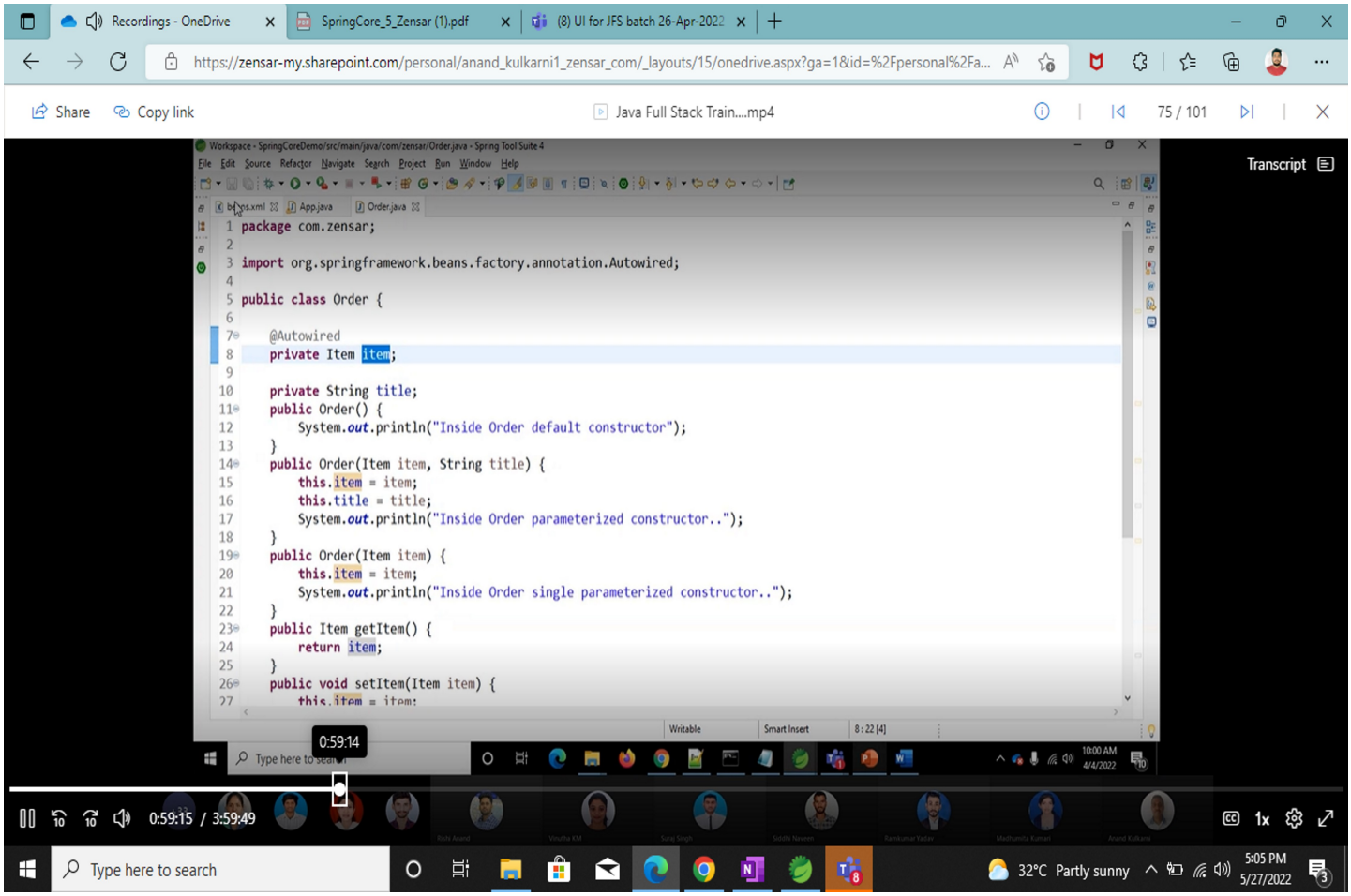


1. **@Autowired annotation:-** in this we no need to define autowire annotation into order bean we have to use this annotation into order class above to the declaration of item variable and it working is that first ioc container goes byType check that any item bean is present in xml file or not if it take to or more bean or same class then ioc container again go for byName and this annotation is applicable for each order bean by constructor if we use autowire="constructor" then it is only applicable for that bean there we have use autowire="constructor".

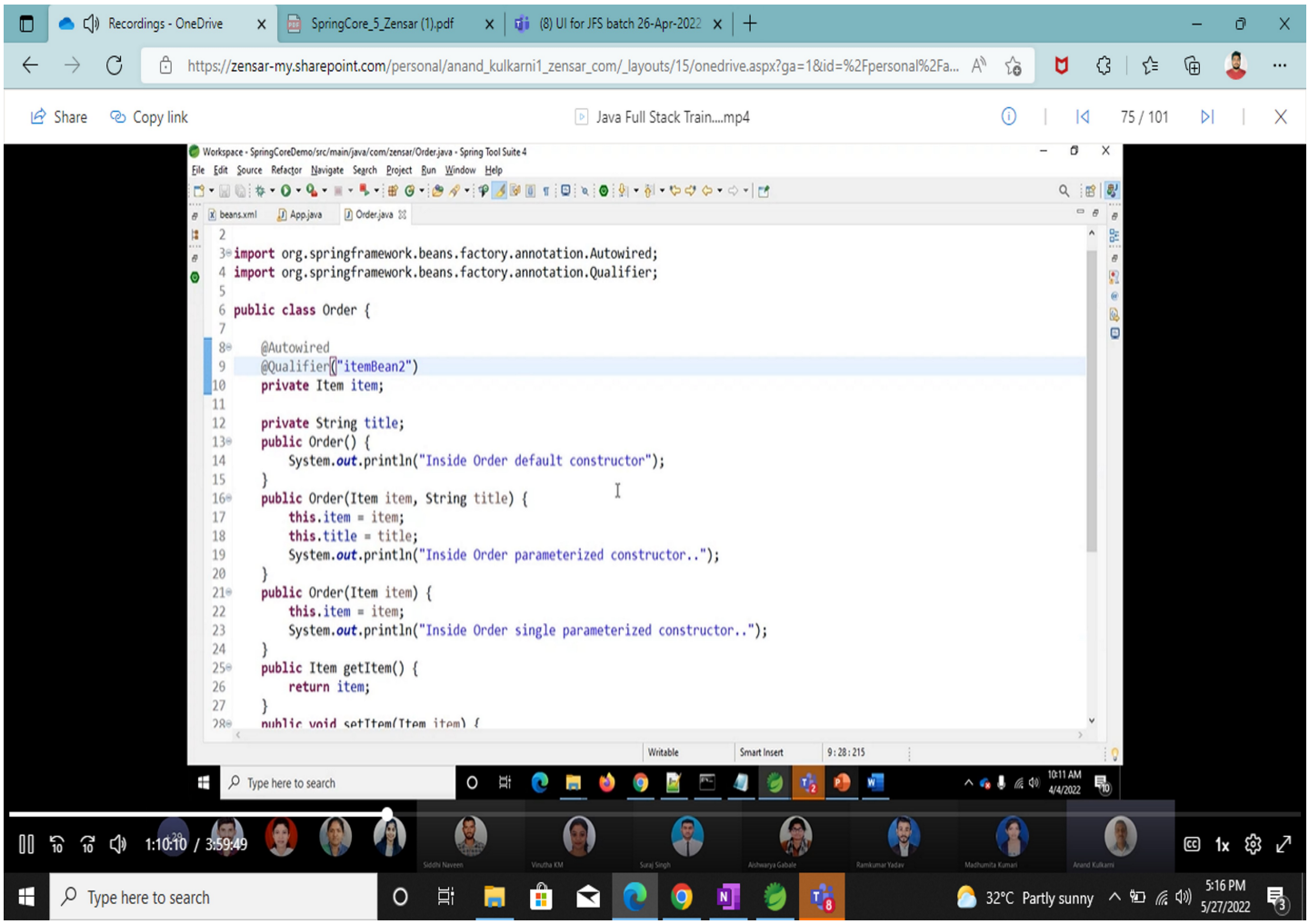



72 <bean id="orderBean" class="com.zensar.Order">
73 <property name="title" value="Laptop purchase"/>
74 </bean>
75
76 <!-- Enable Annotation based configuration -->
77 <context:annotation-config />
78
79 </beans>
Design
Source
beans/bean
Writable
Smart Insert
72 : 50 : 1930
0:56:40
0:56:41 / 3:59:49
1x
Type here to search
32°C Partly sunny
4:49 PM
5/27/2022

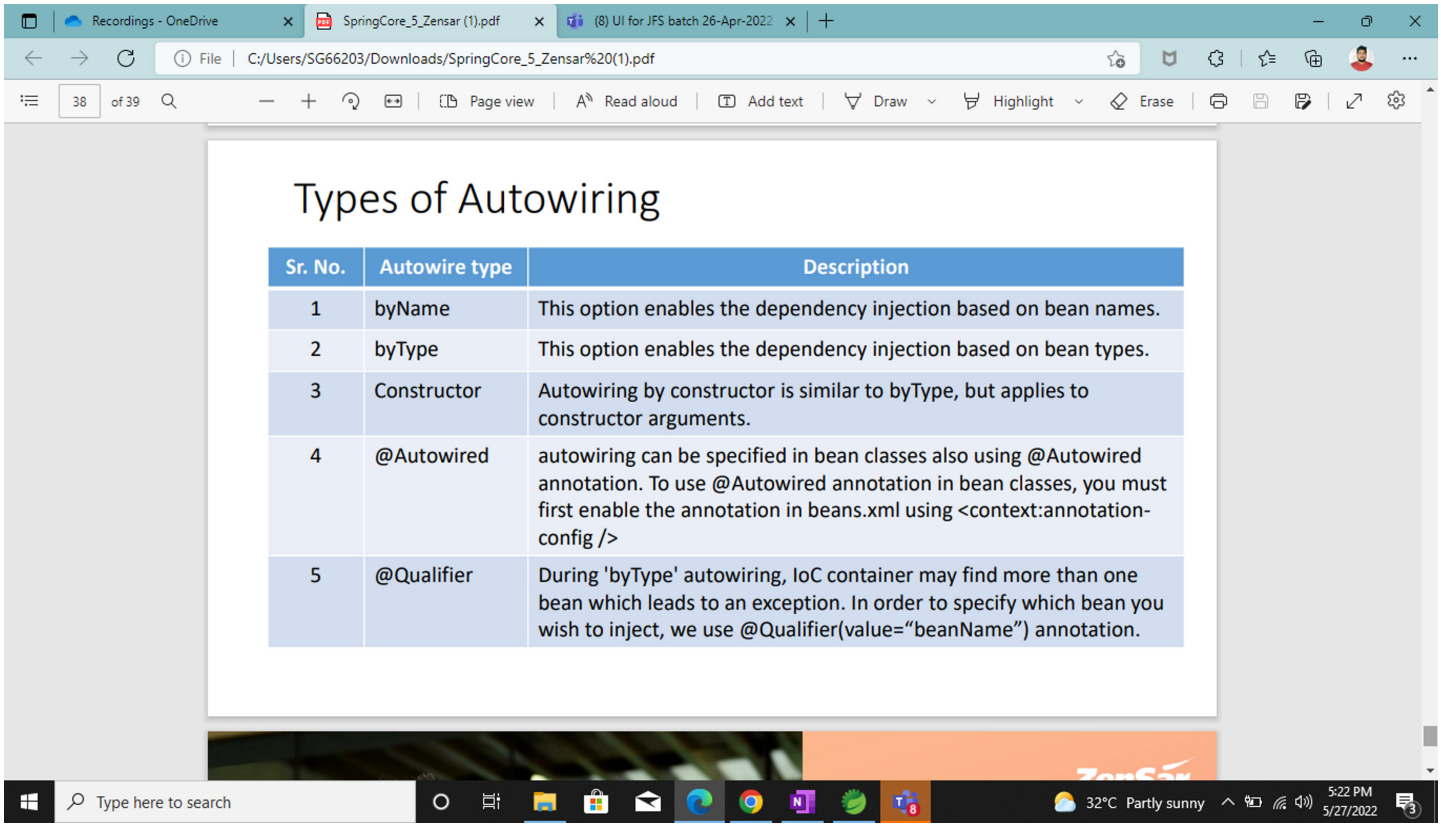

Recordings - OneDrive
SpringCore_5_Zensar (1).pdf
(8) UI for JFS batch 26-Apr-2022
https://zensar-my.sharepoint.com/personal/anand_kulkarni1_zensar_com/_layouts/15/onedrive.aspx?ga=1&id=%2Fpersonal%2Fa...
Share
Copy link
Java Full Stack Train....mp4
75 / 101
Workspace - SpringCoreDemo/src/main/resources/beans.xml - Spring Tool Suite 4
File Edit Navigate Search Project Run Design Window Help
beans.xml
App.java
Order.java
<property name="price" value="80000"/>
</bean>
<bean id="myBeanPostProcessor" class="com.zensar.MyBeanPostProcesso
</bean>
-->
<bean id="item" class="com.zensar.Item" lazy-init="true">
<property name="name" value="Laptop"/>
<property name="price" value="70000"/>
</bean>
<bean id="itemBean2" class="com.zensar.Item" lazy-init="true">
<property name="name" value="Desktop"/>
<property name="price" value="20000"/>
</bean>
<bean id="orderBean" class="com.zensar.Order">
<property name="title" value="Laptop purchase"/>
</bean>
<!-- Enable Annotation based configuration -->
<context:annotation-config />
</beans>
Design
Source
Problems
Javadoc
Declaration
Console
<terminated> App [Java Application] C:\Program Files\Java\jdk1.8.0_121\bin\javaw.exe (Apr 4, 2022
Inside Order default constructor
Order: Laptop purchase - Laptop - 70000.0
Type here to search
9:59 AM
4/4/2022
+33
Rishi Anand
Vinutha KM
Suraj Singh
Siddhi Naveen
Ramkumar Yadav
Madhumita Kumari
Anand Kulkarni
Type here to search
32°C Partly sunny
4:50 PM
5/27/2022



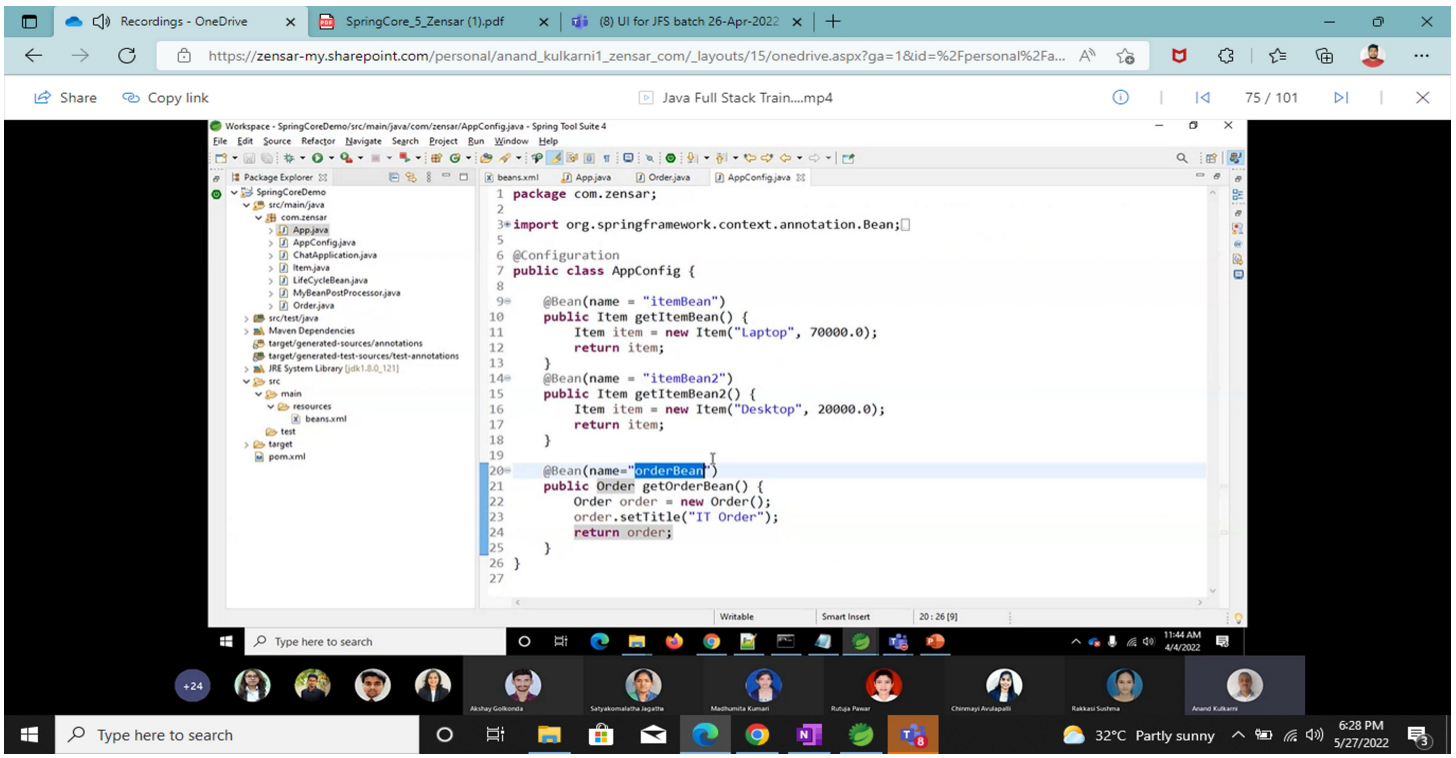
5. **@Qualifier annotation** :- we have seen if we use @autowire annotation into the order bean then if we have two bean of item type then we have got error that not suitable bean is find then we can go for Qualifier annotation like below ..

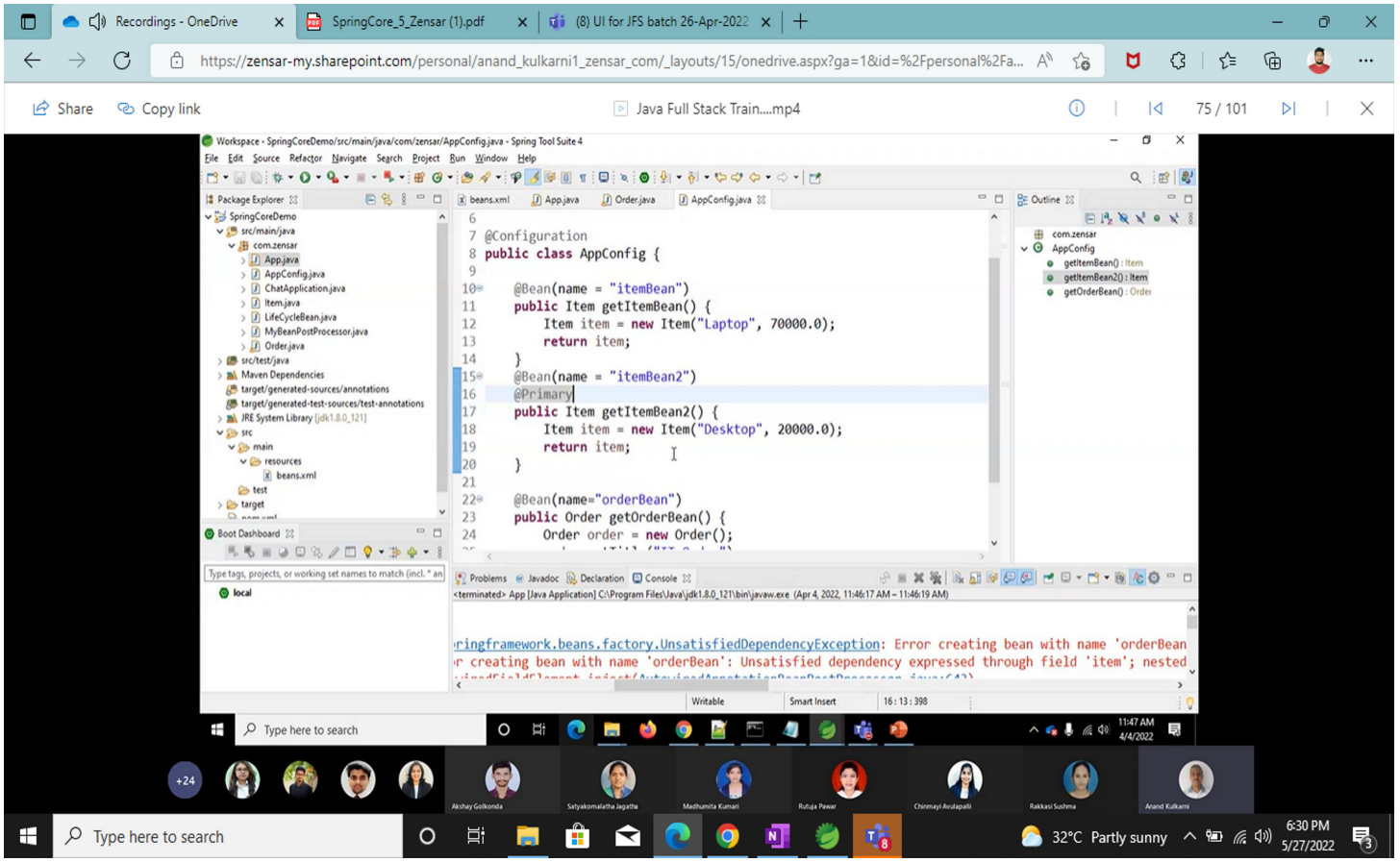


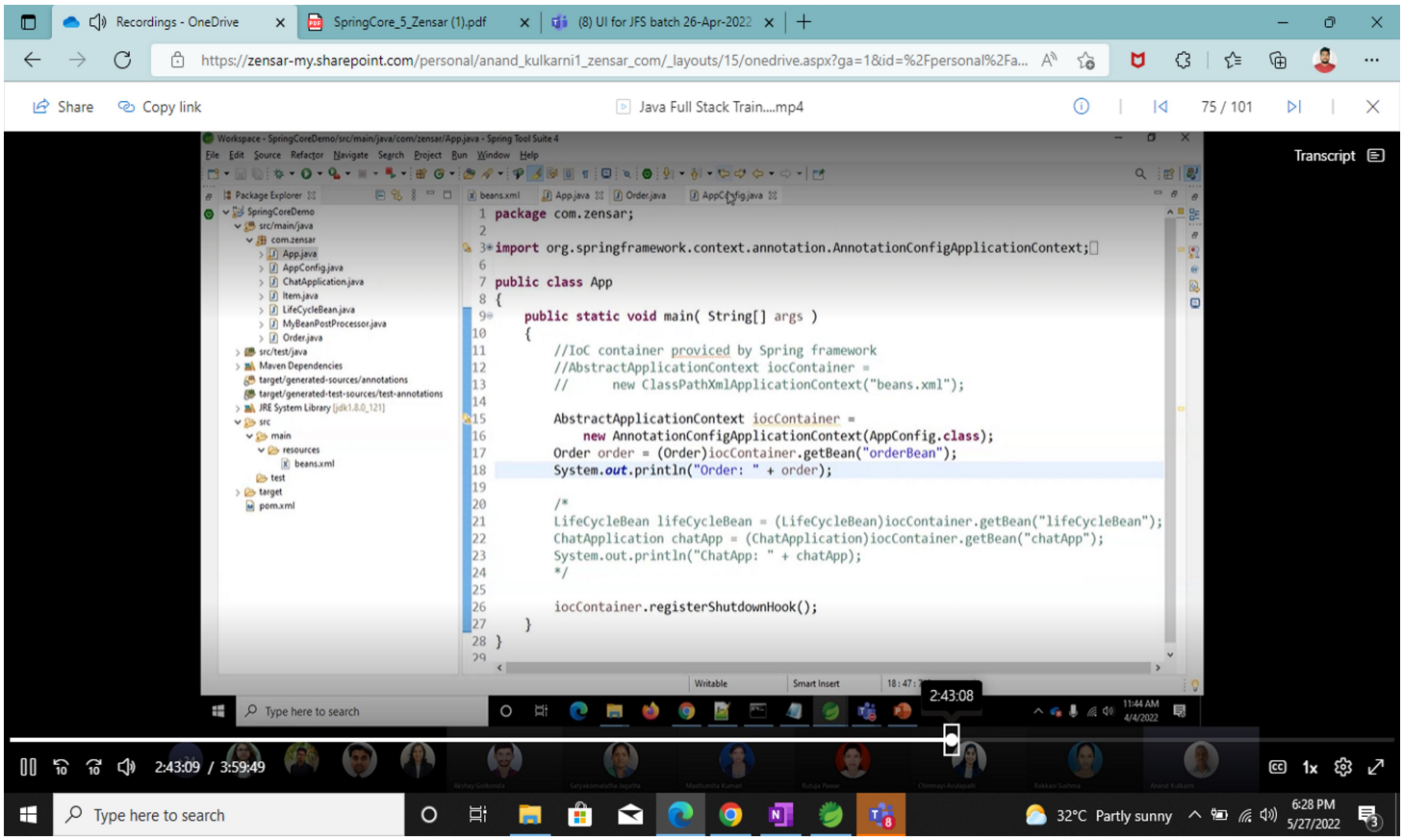
Whenever there is ambiguity which bean should injected using @Qualifier("itenBean1") we can specify ioc container out of multiple beans which exactly we need to inject.



If we do not want to use xml file then we can do configure that bean in other java file like AppConfig is a file like below but if in this file have two or more item bean then we can use @Primary Annotation with @Autowired annotation then conflict will not occurred.




Recordings - OneDrive
SpringCore_5_Zensar (1).pdf
(8) UI for JFS batch 26-Apr-2022
https://zensar-my.sharepoint.com/personal/anand_kulkarni1_zensar_com/_layouts/15/onedrive.aspx?ga=1&id=%2Fpersonal%2Fa...
Share
Copy link
Java Full Stack Train....mp4
75 / 101
Workspace - SpringCoreDemo/src/main/java/com/zensar/AppConfig.java - Spring Tool Suite 4
@Configuration
public class AppConfig {
@Bean(name = "itemBean")
public Item getItemBean() {
Item item = new Item("Laptop", 70000.0);
return item;
}
@Bean(name = "itemBean2")
@Primary
public Item getItemBean2() {
Item item = new Item("Desktop", 20000.0);
return item;
}
@Bean(name="orderBean")
public Order getOrderBean() {
Order order = new Order();
ringframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name 'orderBean
r creating bean with name 'orderBean': Unsatisfied dependency expressed through field 'item'; nested
Type here to search
32°C Partly sunny
6:30 PM
5/27/2022


Recordings - OneDrive
SpringCore_5_Zensar (1).pdf
(8) UI for JFS batch 26-Apr-2022
https://zensar-my.sharepoint.com/personal/anand_kulkarni1_zensar_com/_layouts/15/onedrive.aspx?ga=1&id=%2Fpersonal%2Fa...
Share
Copy link
Java Full Stack Train....mp4
75 / 101
Transcript
Workspace - SpringCoreDemo/src/main/java/com/zensar/App.java - Spring Tool Suite 4
package com.zensar;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
public class App
{
public static void main(String[] args)
{
//IoC container proviced by Spring framework
//AbstractApplicationContext iocContainer =
//new ClassPathXmlApplicationContext("beans.xml");
AbstractApplicationContext iocContainer =
new AnnotationConfigApplicationContext(AppConfig.class);
Order order = (Order)iocContainer.getBean("orderBean");
System.out.println("Order: " + order);
/*
LifeCycleBean lifeCycleBean = (LifeCycleBean)iocContainer.getBean("lifeCycleBean");
ChatApplication chatApp = (ChatApplication)iocContainer.getBean("chatApp");
System.out.println("ChatApp: " + chatApp);
*/
iocContainer.registerShutdownHook();
}
}
2:43:08
2:43:09 / 3:59:49
1x
Type here to search
32°C Partly sunny
6:28 PM
5/27/2022