**Spring Core**

🡪Spring Is Told To Be FrameWork Of The FrameWorks, So We Can Use Many Of The Framework In The Spring

🡪Spring is a Dependency Injection framework to make java application loosely coupled

🡪Spring Provides us IOC(Inversion Of Control Container) by help of which we do the dependency injection

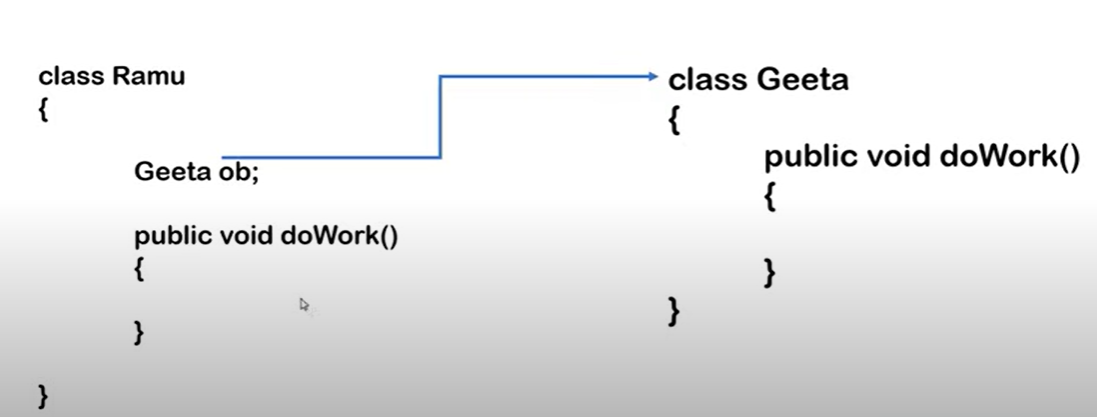
🡪Spring framework makes the easy development of javaEE application

🡪Spring was developed by Rod Johnson in 2003

🡺Dependency Injection

🡪Dependency injection is a design pattern in which an object or function receives other objects or function that it depends on. A form of inversion of control, dependency injection aims to seprate the concerns of construction of objects(i.e. Neel n=new Neel()) and using them, leading to loosely coupled programs.

🡪Example Of Dependency Injection:



🡪In this example the class named ramu uses the class named geeta and for doing this we have to make the object of the geeta in the class ramu and usually we do this by the help of the new keyword, but this is called as the tightly coupled, but the help of the spring or dependency injection’s inversion of control (IOC) we give the control of the object creation from our hand to the spring and spring would make the object of all dependencies on the runtime and would inject it in the required class and this process is called as the inversion of control(IOC)

🡪But by the help of the spring we make the application loosely coupled, So we will not use the method of using new keyword instead we will use the spring’s dependency injection, by the help of which it would inject the required class in the our working class as done in the above example of ramu and geeta

🡪Hence the work will be done by the help of dependency injection, we have to only tell the spring about oue requirement by help of meta-data or annotation or xml

🡪This was only one feature of the spring, And other than it, spring provides various modules, services, api by the help of which we can use many of the other services

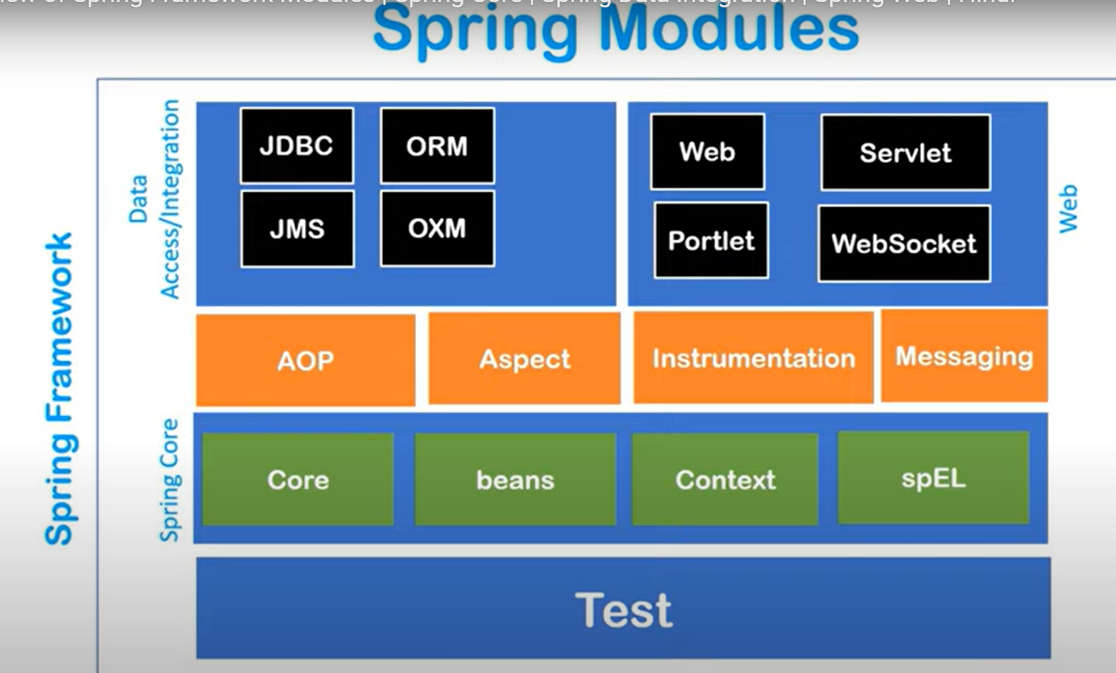
Eg. Spring JDBC which helps us to connect with our database, Spring ORM for using the hibernate as hibernate is an ORM framework this are used for the data access layer.

🡪Genrally in any of the project there are three layers, UI Layer, Business/Service layer and data access layer

🡪For the UI Layer we use the implementation of the spring which is the spring mvc(model view controler)

🡪And in the business or service layer we can use the Spring Security, Spring transaction management

🡪There are various modules of the spring framework some of which are as follow:



🡺Spring IOC Container

🡪Spring IOC container is an predefined program or is a component of spring framework which we get with spring framework, like with JVM we get different program like garbage collector, etc

🡪Which is used to create object, And if object are created than, to make it hold in the memory , And it can also inject one object in the other object i.e. it can also do work of dependency injection

🡪So that it maintains the life cycle of any object, Spring container maintains all the stages from the creation of an object to the destruction of object, itself only we have to told the two things to the spring container the first is the beans i.e. we have to tell the spring container that it had to manage which of the beans or java’s POJO classes and the second one is the configuration file i.e. XML configuration file

🡪So that in short we have to give it two information the beans information(i.e. pojos) and the xml information and in the xml information we will define that which of the bean is dependent on which of the thing and spring container will use this all information and will make the object of the all the given beans and as per our requirement that we told in the xml, It will do injection in the object as per instrunction in the xml

🡪Genrally in xml we define that which of the bean is dependent on which of the bean and bean contains which of the property

🡪And due to this our application can directly use all these objects

🡪In spring we have the application context which basically represents our spring IOC container, So that it is an interface that represent IOC container.

🡪Application context extends the bean Factory, so application context contains the property of the bean factory as well as additionally its additional property

🡪So that by the help of which we can get the values of the object that are stored in the container by the help of the application context

🡪But the problem is that we cannot make the object of the application context as application context is an interface and we cannot make the object of interface, And due to which we have to create object of any other subclass

🡪We have many of the subclasses of the application context, some of them are as follow:

1. ClasspathXMLApplicationContext
2. AnnotationConfigApplicationContext
3. FileSystemXMLApplicationContext

🡪This all there are used for the different purposes, first one searches the xml configuration from the class path of java, i.e. If we want to put the xml in our class path and init, if we want to define the configuration, Then simply we will use classpathXMLApplicationContext

🡪And AnnotationConfigApplicationContext basically searches the beans on which we have used the annotations

🡪FileSystemXMLApplicationContext searches those config file from the file system, So that if we want to use the configuration from any file than we will use it

🡪As we will work more with the xml only which are puted on the classpath, So we will work more with the classpathXMLApplicationContext

🡺IOC Container can make dependency injections in two ways:

1. Setter Injection
2. Constructor Injection

🡪Setter injection is also called as the property injection, In the setter method there are setter methods of each thing to set its value

🡪In the constructor method, while making the objects IOC container uses the constructor method instead of the setter method, The constructor is an method in the class in which the name of the class and the method have same name

Eg. class Address{

String city, state, country;

Address(city, state, country){

}

}

🡺Configuration File

🡪As we used the configuration file of hibernate in the hibernate to set the configuration settings of hibernate, Here in spring there would be also an xml file which is told as configuration file, In this xml file we would tell all the things to our container

🡪Technically configuration file is a file where we declare beans and its dependency, The starting of the file is done with the beans tag and init we would write the details in the bean tag, Example of it is as follow:

<beans>

<bean></bean>

</beans>

🡺Datatypes of dependencies that can be handeled by IOC container

1. Primitive Data Type

byte, short, char, int , float, double, long, Boolean

1. Collection Type

List, Set, Map and Properties

1. Reference Type

Other class Objects

🡺Basic requirement of softwares for spring practicals:

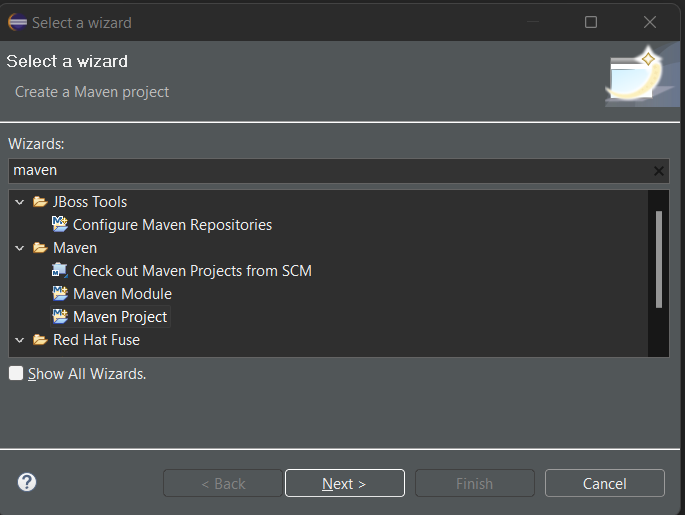
1. Eclipse / NetBeans / IntelliJ =>IDE
2. TomcatServer =>Server
3. Mysql database =>Database
4. Sqlyog/workbench/ phpMyAdmin =>mysql gui

🡺Basic Steps For Creating Project With Spring

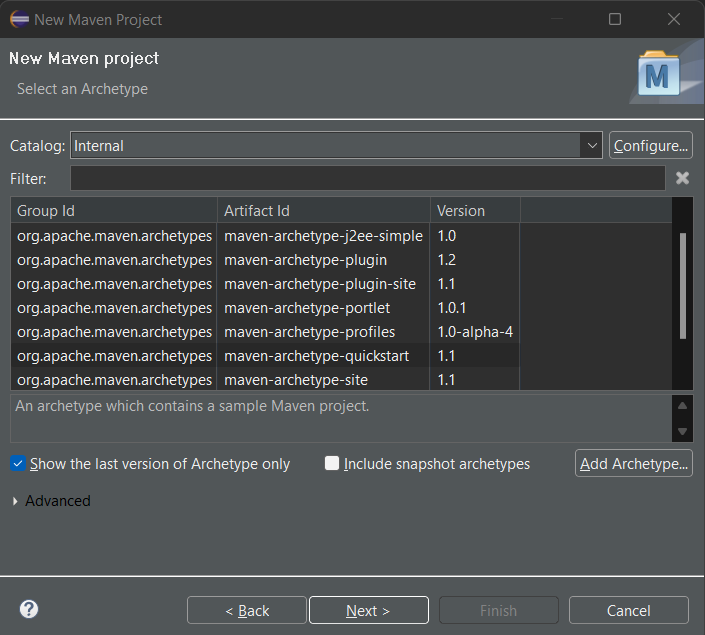
1. Create A Maven Project
2. Adding Dependencies =>Spring Core, Spring context
3. Create beans – POJO (Having getter, setter, constructor, toString\*(For Printing of data))
4. Creating Configuation file =>config.xml(It can have any other name)
5. Setter Injection
6. Main Class: Which can pull the objects and use

🡺Performing Steps:

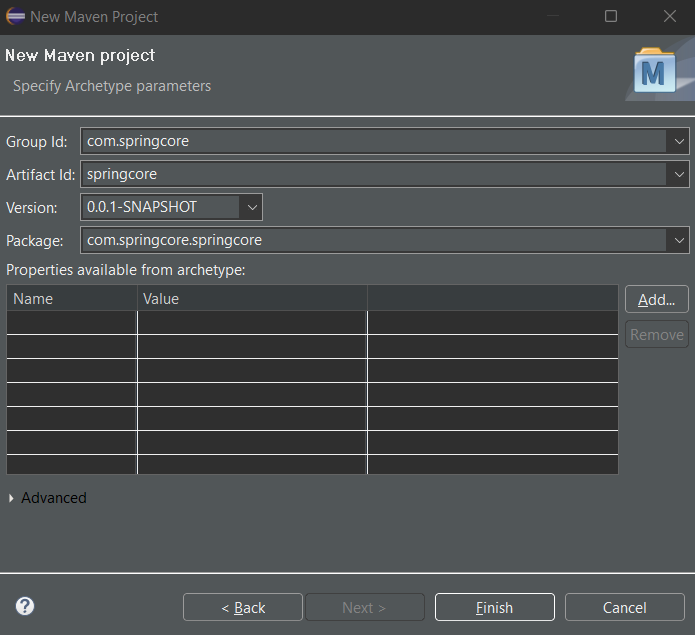
1. First make the maven project in the eclipse



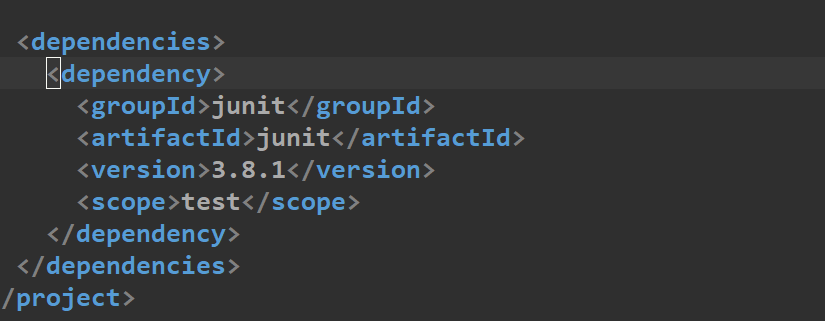
1. Create a simple quick start project from the internal options



1. And than the group id is for uniquely identify the project and the artifact id is the name of the our project and you can also select the package name as per your choice



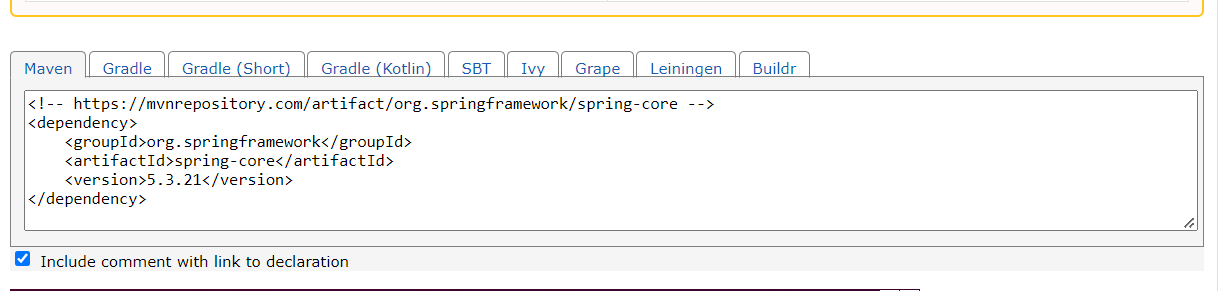
1. Than it would take some time for processing for downloading some of the dependencies
2. And our main operating app, which would render first, like that of the index.html of the html
3. This App.java is present in the package that we made, Here it is com.springcore.springcore
4. And than we can also see an pom.xml file in which we would write the required dependencies that is to be downloaded by the maven itself, If maven would be not there than we would have to manually download it and than put it in the project, but by the help of maven all this process would be done automatically
5. So we will go into the pom.xml for adding dependencies, but for adding dependencies we would have to get the link of the dependencies from the mvnrepository.com
6. Initally the file would look like as that of shown below, There would be dependencies tags and in between of them there would be tags of the dependency, And we have to add the dependency tag that we want to add from the website



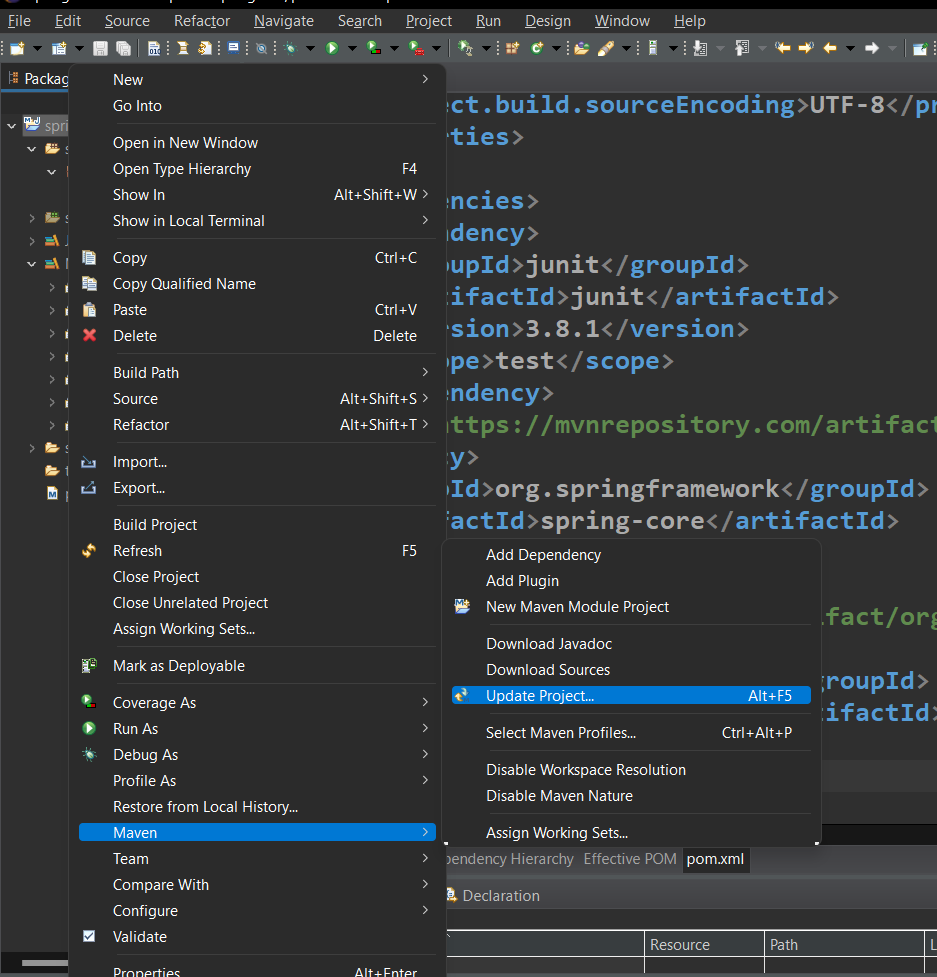
1. For current purpose we have to get two dependencies from the maven the first is spring core and the other is spring context, for that we will go to mvnrepository.com and would copy the text of the lines of the most stable dependencies, but make sure that the dependencies of springs would be of the same versions not of different versions

🡪For this example we have taken 5.3.21 version’s dependencies

🡪Make sure you take the maven’s link, not of the other one’s as shown below:



1. Take the links of both spring core and spring context and paste them in the pom.xml file between the dependencies tags
2. And than by saving them dependencies will automatically be downloaded in the folder named maven dependencies, if the dependencies are not downloaded or is not showing that it is downloading in the footer of eclipse than go the main project folder and than press right click on it and than go to maven And than click on the update project as shown below in image



1. Now the third step is to create a beans or java pojo and for it we have to make a class, For this tutorial we will take the class name as the Student, And we have to make it in the pacakage where there is App.js
2. And init we will make three private field studentId, studentName, studentAddress and than we will generate getter and setter and than we will also generate constructor(using fields., using super class) and toString by right clicking and than going to source and than selection of the required methods, Example of it is as shown below:



1. Now fourth step is to create a spring configuration file, in this project we would keep the name of that file as the config.xml but we can keep any name as per our requirement, but make sure that it is an .xml file
2. So we will make this new file in **src/main/java** folder and it can be make it by right-clicking on the folder and than go to new than others than search for the xml file and than make one

🡺If we make the spring configuration file at different location than that of the src/main/java than we have to make the address of its in the Controller file or App.js in the different way

🡪Example if we make that exampleConfig.xml file in the com.springcore.collection than its address would be like: **com/springcore/collection/exampleConfig.xml**

1. Than we have to write the name space of the spring in it after the default statement, we can get the name space from the documentation, it is also defined as below and some changes are done init as per necessity(So copy bellow one), It is recommended to read the spring documentation.

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:p="http://www.springframework.org/schema/p"

xsi:schemaLocation="

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.1.xsd

http://www.springframework.org/schema/context

<http://www.springframework.org/schema/context/spring-context-3.1.xsd>">

</beans>

\*\*Changed After completing some of programs, So in some of example in the file there would be some old format

1. Now as we want to declare the beans in the spring configuration file, and our bean is the Student class that we made, so we will make a new bean in the spring configuration file by the help of the bean tag as shown below:

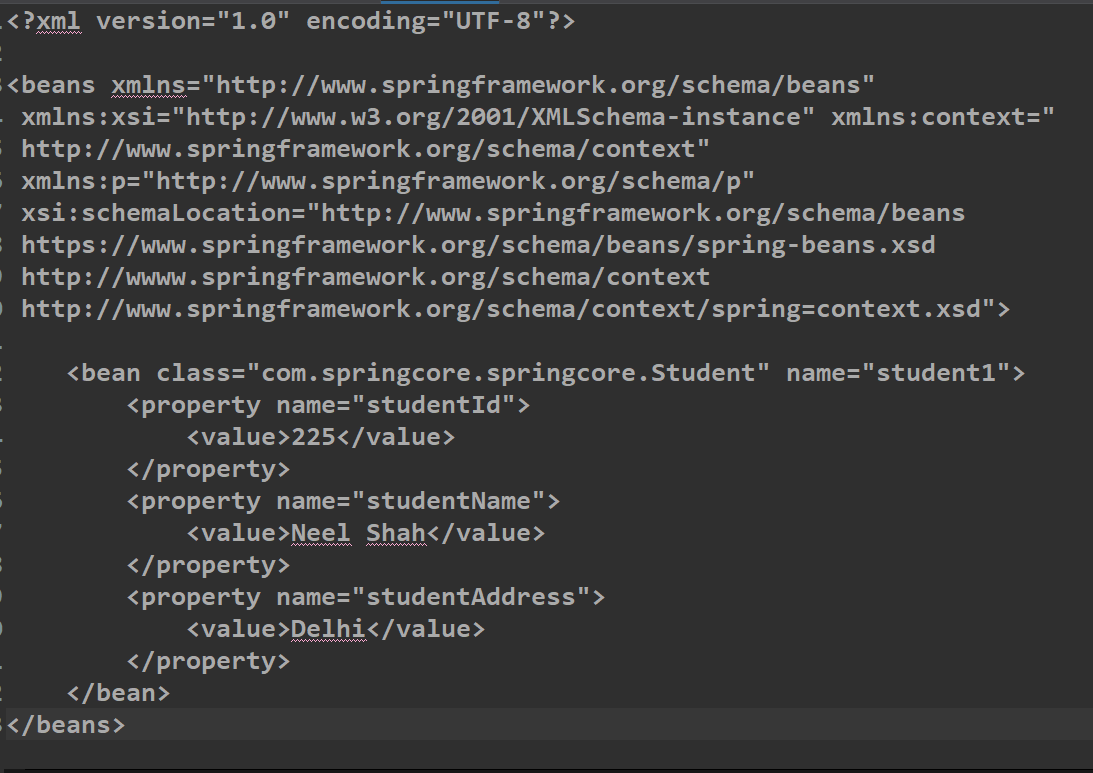
🡪In the beans tag we have to add the bean tag in which we have to write the name and the class, where the class would be the full name of the class which we want to write

🡪And the name should be given anything there is no norms for it

🡪And now we are going to use the setter injection, So we have to write here that what value do we want to set using the setter Injection i.e. what would we want value do we want to keep in the variables present in the bean

🡪So that in the this case we have the bean by the name of the Student and it has the three variables present init they are studentId, studentName and studentAddress

🡪And as we are using the **setter injection**, we will use the **property tag** and init we will use the value tag to define it’s value, this method of the property tag is used for all the variables whose value we want to define in the bean by the setter injection, Example of it is as shown below:



🡪By doing this we have told the spring container metadata, And by the help of this meta data it will automatically take the value from it

1. Now as we have done the fourth and fifth steps which are creating configuration file and setter injection the next and sixth step is to make the container in main class and pull the object to use, **So now we will write code in App.java**
2. And for using the container we first have to initialize the container and the container can be initialized by the by three ways, But as we have the xml class path, We will use that method only for now
3. And in the xmlclasspath method we can give one or more than one xml locations, For more than one we will use the array method and for only one we will use the simple method
   1. For only one class path

**ApplicationContext context=new ClassPathXmlApplicationContext(“config.xml”)**

🡪Currently we are using this one

* 1. For more than one class path

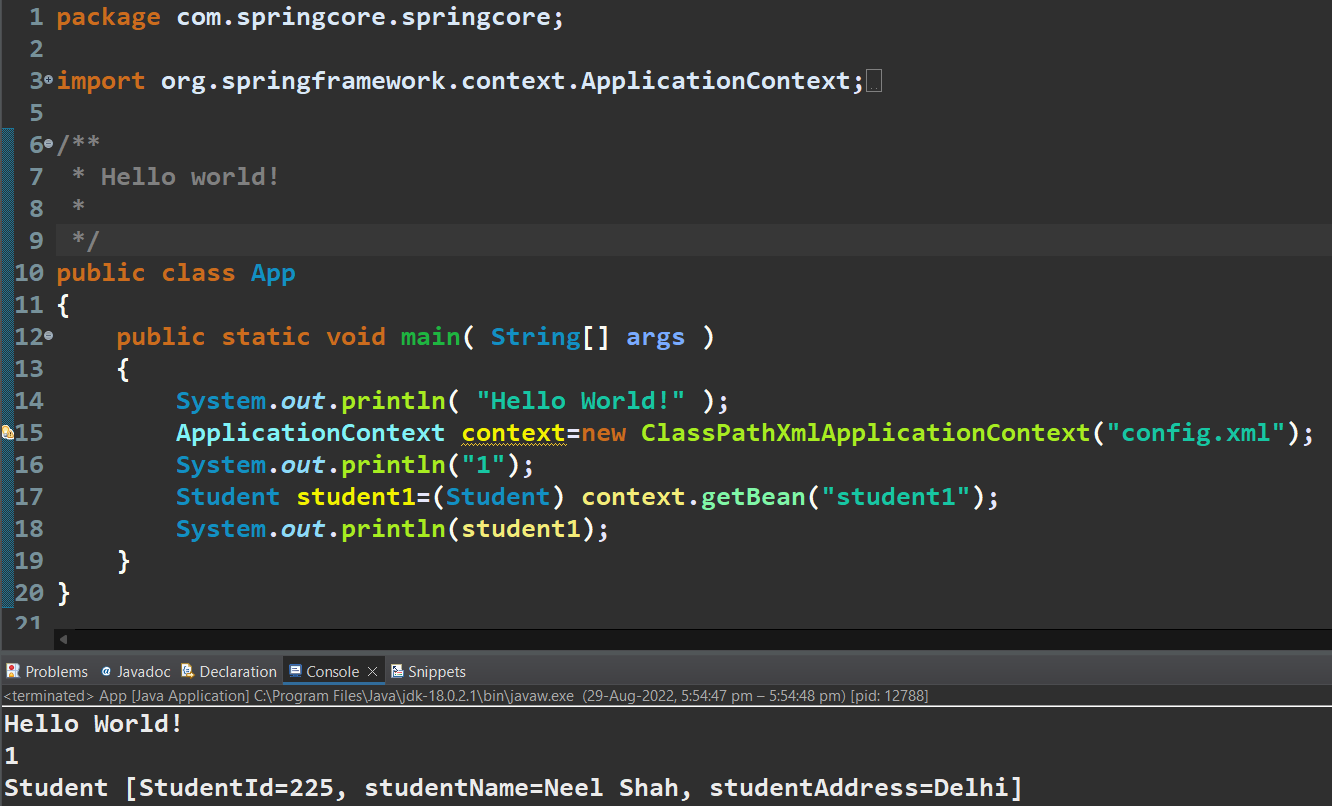
**ApplicationContext context=new ClassPathXmlApplicationContext(new String[] {“services.xml”, “daos.xml”})**

1. And than we can use the bean by the help of the getBean method of the context and init we have to give the name of the bean that we have defined in the property tag of the xml file

🡪And in return it will give the object of the student as we know the bean is the object of the student, and we have to also do the type casting of the Student

(\*Here the student is the bean class that we have used)

🡪Example of the above steps is as follow:



🡪And than by running the App.java as java application gives the results as shown above

🡪So when we want to change the value of the student’s variable than we do not have to make change in the java file, instead we have to make change in the .xml file and give value as the meta-data, hence we do not have to compile java file again, Hence it is called as loosely coupled, And this is what usage of the dependency injection is…..in which we do not make object it is automatically made by spring

🡺Types of Value We Can Give In Property Tag In Spring Configuration Xml

🡪We can give the value to the property tag by the three ways :

1. Normal Way(Not-Prefered)

<bean class=”com.springcore.springcore.Student” name=”student1”>

<property name=”studentId”>

<value>2233</value>

</property>

<property name=”studentName”>

<value>neel</value>

</property>

</bean>

1. Values As Attribute In Property Tag(Not-Prefered)

<bean class=”com.springcore.springcore.Student” name=”student1”>

<property name=”studentId” value=”223344”/>

<property name=”studentName” value=”neel”/>

</bean>

1. Using P Schema(Mostly Prefered And Easy)

<bean class=”com.springcore.springcore.Student” name=”student1” p:studentId=”2233” p:studentName=”neel”/>

🡪In this p schema make sure you have written the following statement in the starting of the .xml file for the spring configuration

xmlns:p=http://www.springframework.org/schema/p

🡪And in the p schema type we do not have to write the property tag

🡪All of these three method gives the same result, So we can use any of them

🡺As we know that there are three types of datatypes handled by the IOC Container and the method for putting their values are different for .xml file

1. Primitive Type

As we have discussed in the upper part there are three types to store the primitive types normal way, value as attribute in property tag and using p schema

🡪Refer the above section for them

1. Collection Type
2. List

<bean>

<property name=””>

<list>

<value>10</value>

<value>1023</value>

<value>1052</value>

</list>

</property>

</bean>

**🡪If we want to define null value than we will use the null tag i.e. <null/>**

**🡪If there is only one value in the list than we do not need to use the list tag, We can directly use the value tag without usage of list tag**

1. Set

<bean>

<property name=””>

<set>

<value>10</value>

<value>1023</value>

<value>1052</value>

</set>

</property>

</bean>

1. Map

<bean>

<property name=””>

<map>

<entry key=”java” value=”2month”/>

<entry key=”java” value=”1month”/>

</map>

</property>

</bean>

1. Properties

<bean>

<property name=””>

<props>

<prop key=”name”>neel</prop>

<prop key=”surname”>shah</prop>

</props>

</property>

</bean>

1. Reference Datatype

🡪It is one of the most used datatypes in the industry and it is very important

🡪Before we begin let us see how the refrence datatype is been made to use

**A 🡪 B**

🡪Consider the Two classes A and B, In which the class A has the object of the class B, So that the class A is depend upon the B so that the B is the dependency and class A is dependent object

🡺Genral Type(or pseudo-code) For Setting the value of the refence datatype:

<bean class=”A” name=”sample”>

<property name=”obj”>

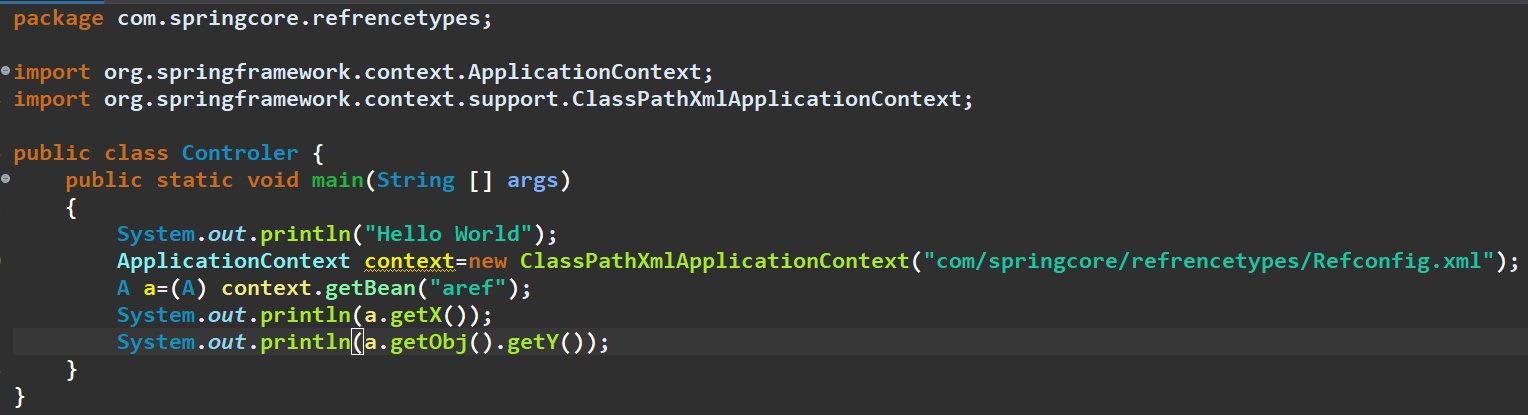
<ref bean=”B”>

</property>

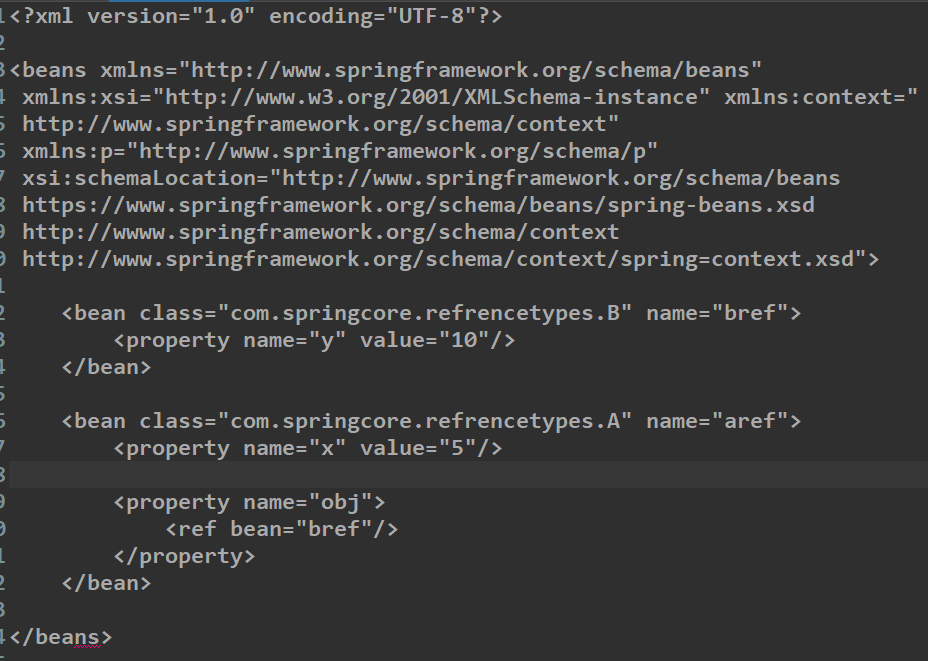
</bean>

🡪Let us consider an example of it, For making the concept clyster clear

🡪controller File



🡪 Reqconfig.xml



🡺The Following Statement Given bellow can be written in three ways:

<bean class=”com.springcore.refrencetypes.A” name=”aref”>

<property name=”x” value=”5”/>

<property name=”obj”>

<ref bean=”bref”/>

</property>

</bean>



OR

<bean class=”com.springcore.refrencetypes.A” name=”aref”>

<property name=”x” value=”5”/>

<property name=”obj” ref=”bref”/>

</bean>

OR

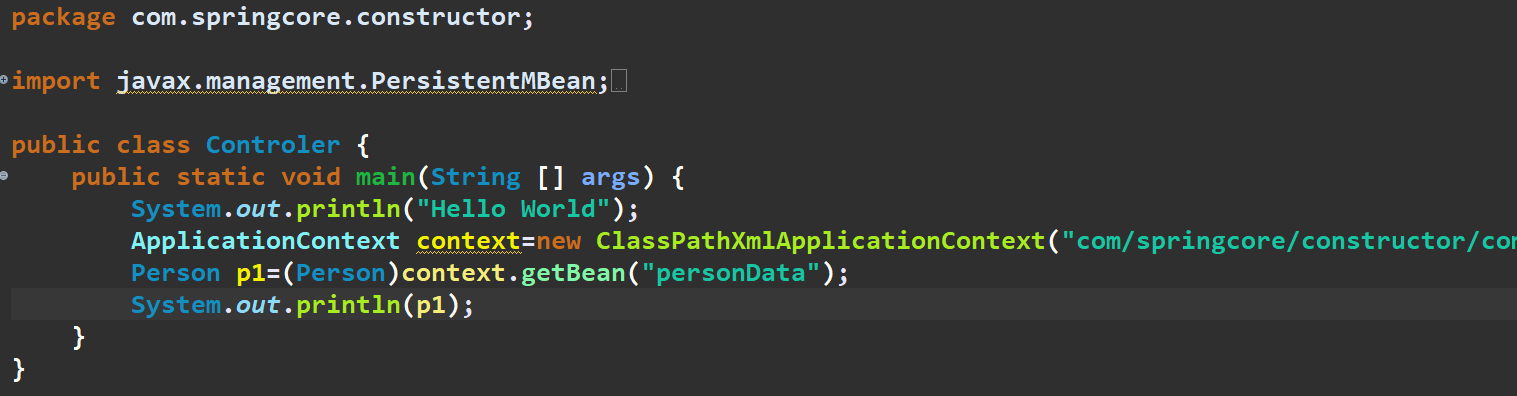
<bean class=”com.springcore.refrencetypes.A” name=”aref” p:x=”33” p:ob-ref=”bref”/>

🡺Constructor Injection

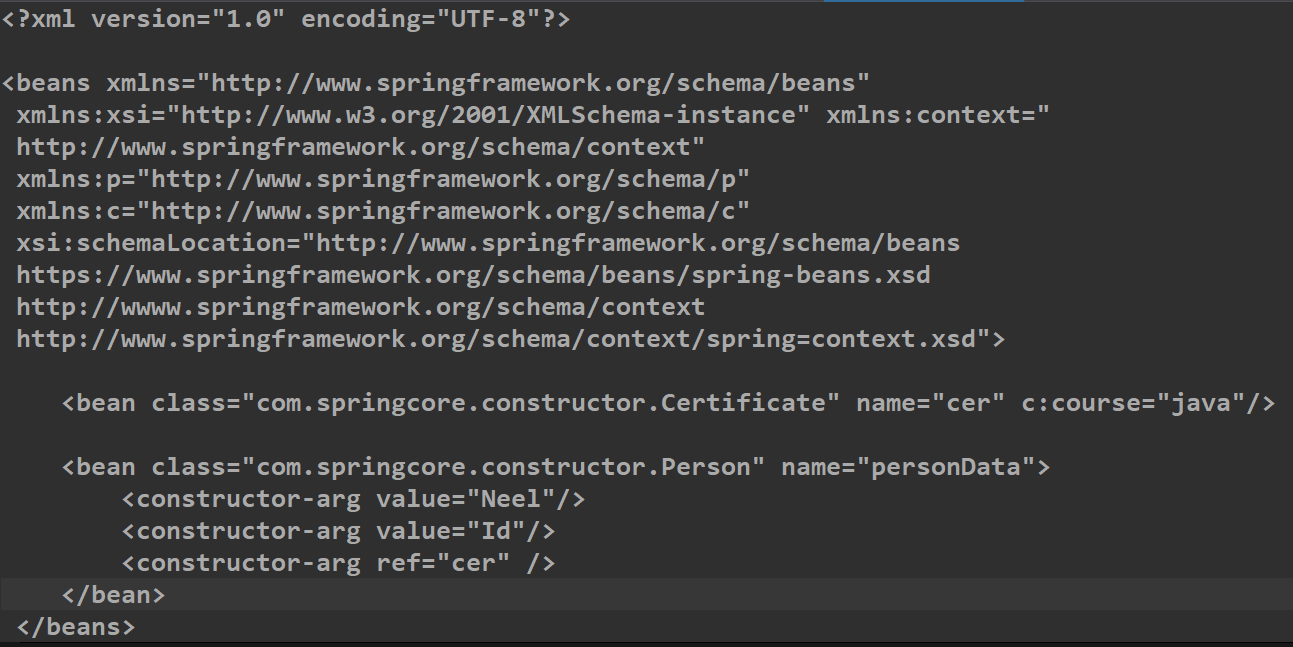
1. In constructor injection instead of the property tag we use the constructor-arg tag and in it we give the values, but in the constructor-args, we do not define the name
2. And the spring is smart enough to recognize that the given value is of the which datatype, but this fails when there are more than one types in the constructor which is called as the ambiguity problem which we will talk some time after
3. Now for example we will take the class named person and the certificate to use both the simple constructor method and the refrence method also, Example is as shown below:

🡪Person.java and Certificate.java

🡪Controler.java



🡪Config.xml

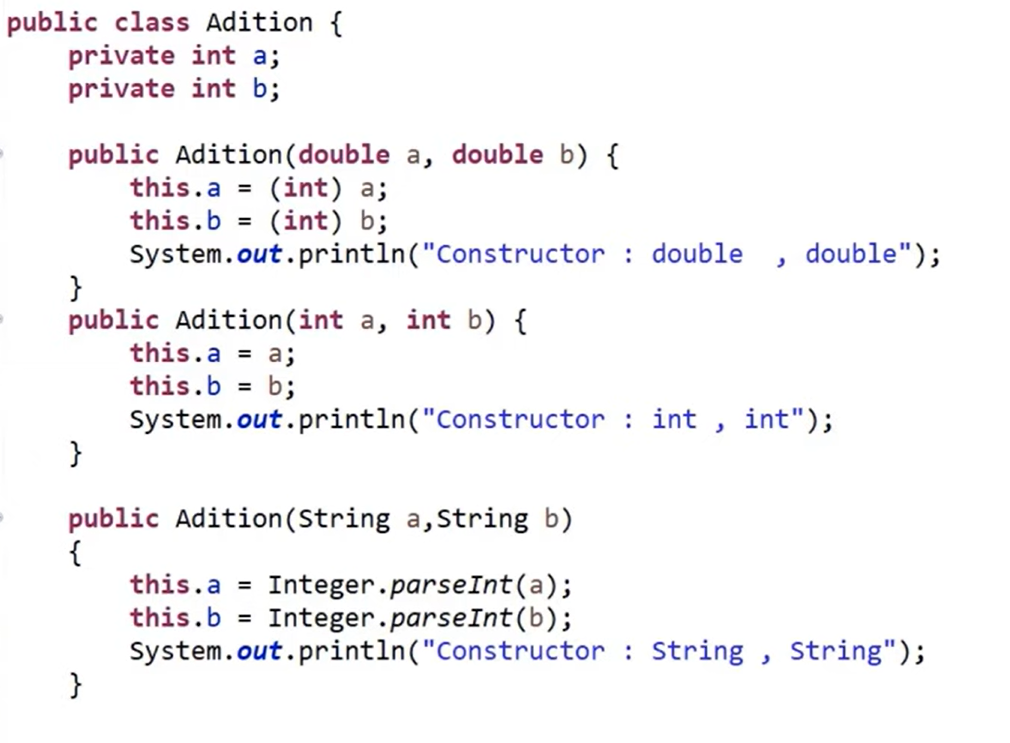


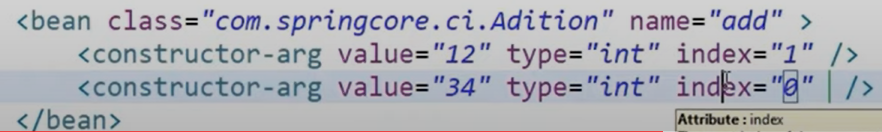
🡪As shown we have used the c:courses instead of the constructor-arg and if we want to use it than we have to add the following statement highlighted by red colour in the code

🡺Ambiguity Problem

🡪By default spring assumes any given value as the String and if spring do not get the constructor as the String than it will give the preference to the constructor which is on the top as compared to the other one

🡪This problem arrises when there are more than one constructor or in any constructor there are more than one same type of the variables, Example:



🡪But we can handle this calling of the constructor by using giving the data like **value** and the **index** in the constructor-arg tag, Example:

🡪This problem happens only with the constructor, So this is called as the constructor injection with ambigutiy problem and its solution is as told in the upper line

🡺Life Cycle Methods

1. Spring Provides two important methods to every bean

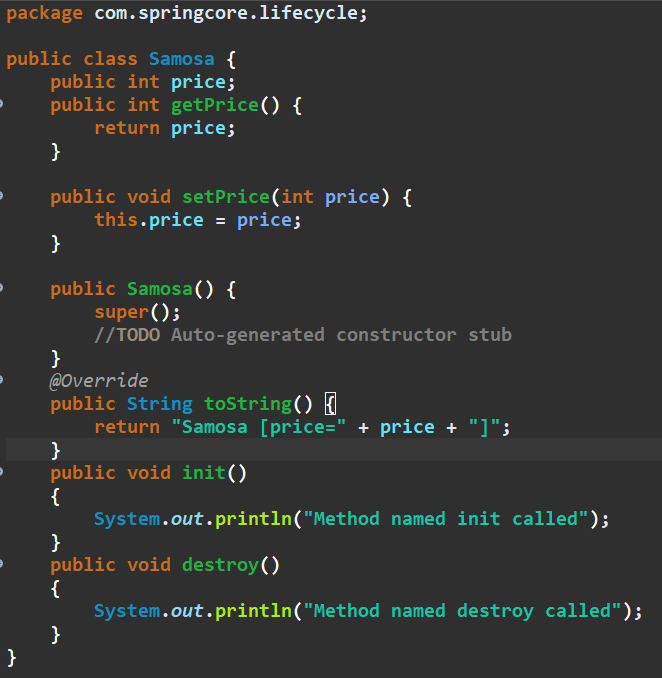
public void init()

public void destroy()

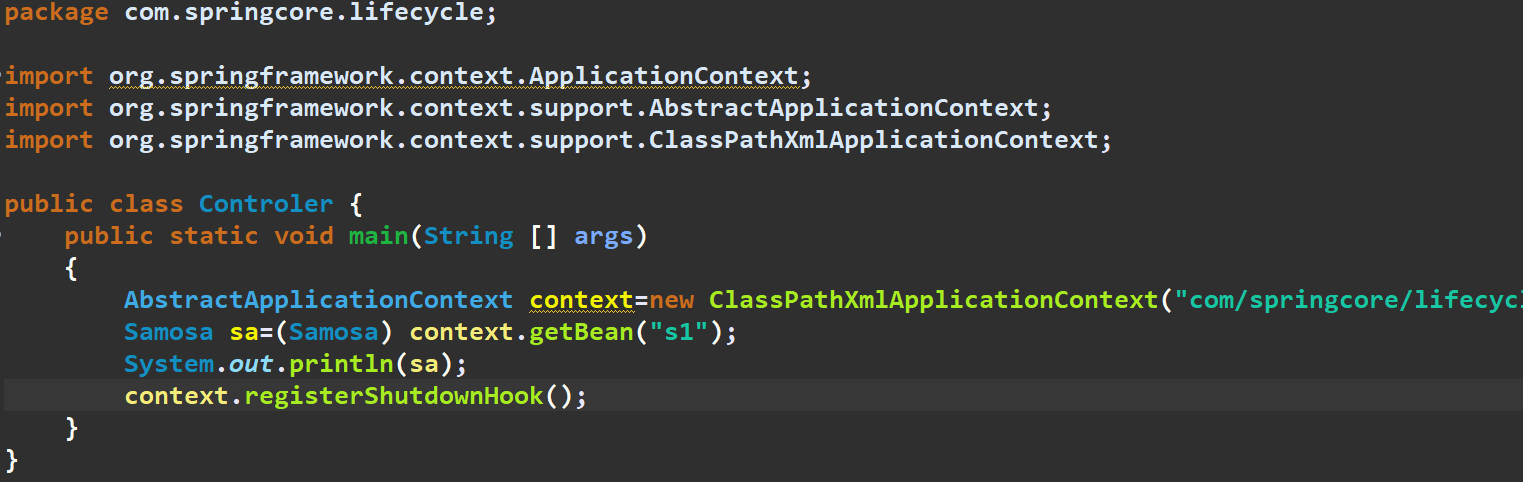
1. **We can change the names of these method, But signature must be same**
2. In the init method we can perform various tasks like initialization code, loading config, Connecting database, webservices, etc
3. And the destroy method is for the clean up of the code, that we had made during the init method
4. As per normal the life cycle includes the following steps as follow:
   1. Creating An Object
   2. Setting values of the properties present in the object
   3. Calling init method if it is wriiten
   4. Then we read and use the bean
   5. And after the usage of the bean at last it will class destroy method for the clean up of code and than it will destroy the bean
5. For configuration of bean there are three methods:
   1. XML method
   2. Spring Interface
   3. Annotation

**\*\*If we use the property method than we have to use the constructor using having super class and if we use the constructor-args method than we have to use the constructor using fields**

🡺Practical of the life cycle methods and configuration of bean by help of xml file

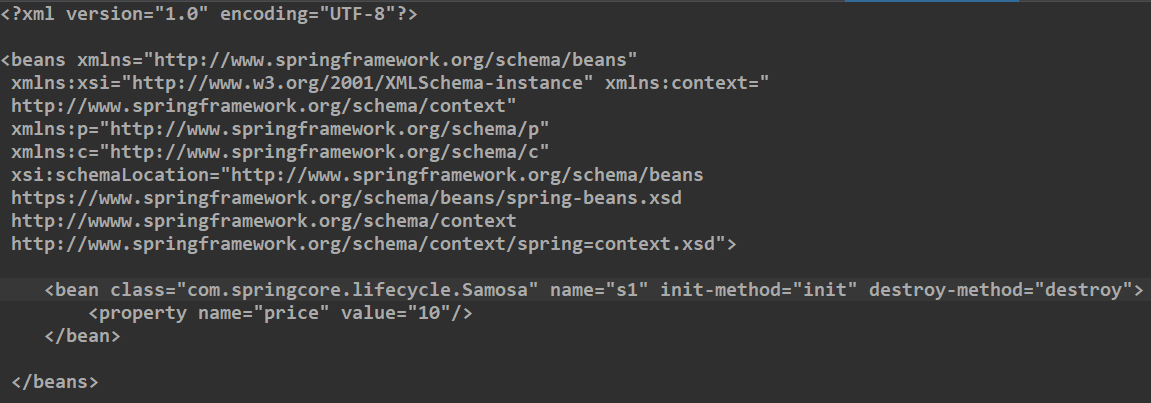


🡪 Consider the above Samosa class which is an bean and init there is one parameter named price and in this bean we have made the two another functions named the init and destroy and this both the methods are been called by the spring during the initiation and the destruction



🡪And this is the Controler.java, which is out main class and init usally we use the ApplicationContext to make an context from the xml file but if we want to use the init and destroy method at the time of the initiation and destruction than we have to use the AbstractApplicationContext instead of the ApplicationContext

🡪And we have to also add the method of the AbstractApplicationContext that is registerShutdownHook() after the usage of that bean to make spring know that there also exists an destroy method of the bean, otherwise it will not use the destroy method



🡪And also in the xml file we have to add the init-method and destroy-method ‘s name in the bean tag of that class, Here we can have the name of the init and destroy method any thing, just we have to define it here, As for this program they are init and destroy

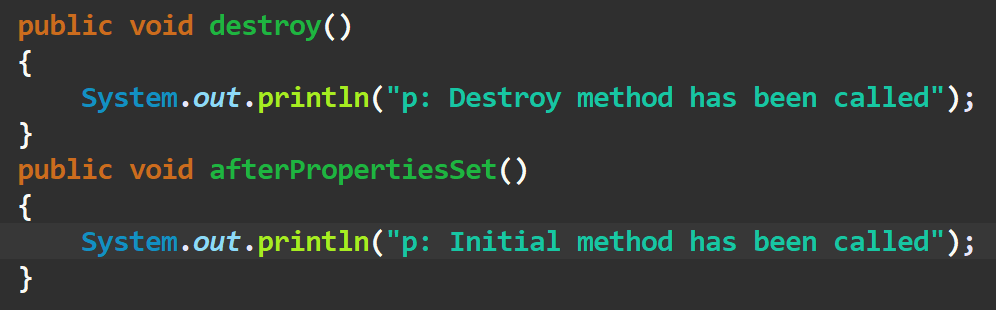
🡺Configuration Of Bean Using Interface And Xml File

1. In this method we have to implement two interfaces in the bean class and use its method instead of writing the init-method and destroy-method in the property tag in the xml file
2. The two interfaces to implement are:
   1. InitializingBean
   2. DisposableBean
3. And we have to also make the two methods by the name
   1. public static destroy()
   2. public static afterpropertiesSet()

🡪The name of the both methods should be same as they are bydefault required name by the interfaces that we implemented

🡪And in this two methods we have to write the code which would run at the time of the initialization and destruction

🡪Also in the xml file we do not need to mention the init-method and destroy-method in the bean tag, they will be called automatically by the bean itself



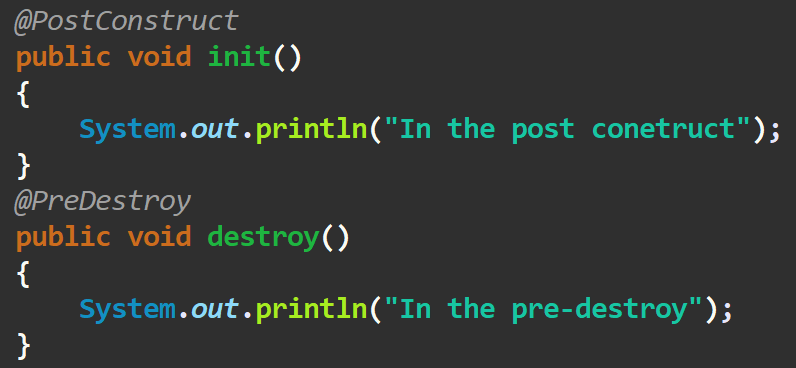
🡺**Configuration Of Bean Using Annotation And Xml File**

🡪Note that both the @PostConstruct and @PreDestroy annotations are part of Java EE. Since [Java EE was deprecated in Java 9](https://www.baeldung.com/java-enterprise-evolution), and removed in Java 11, we have to add an additional dependency to use these annotations:

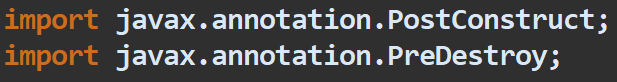
🡪Dependency to be added in the maven’s pom.xml file is as follow:

🡪And than we have to Add the two statements the **@postContruct** above the method which you want to call at the time of the initialization and the **@preDestroy** above the method which you want to call at the time of the destruction, Example

<**dependency**> <**groupId**>javax.annotation</**groupId**> <**artifactId**>javax.annotation-api</**artifactId**> <**version**>1.3.2</**version**> </**dependency**>



🡪And if it do not automatically import the two statement, than you have to import the following two statements:

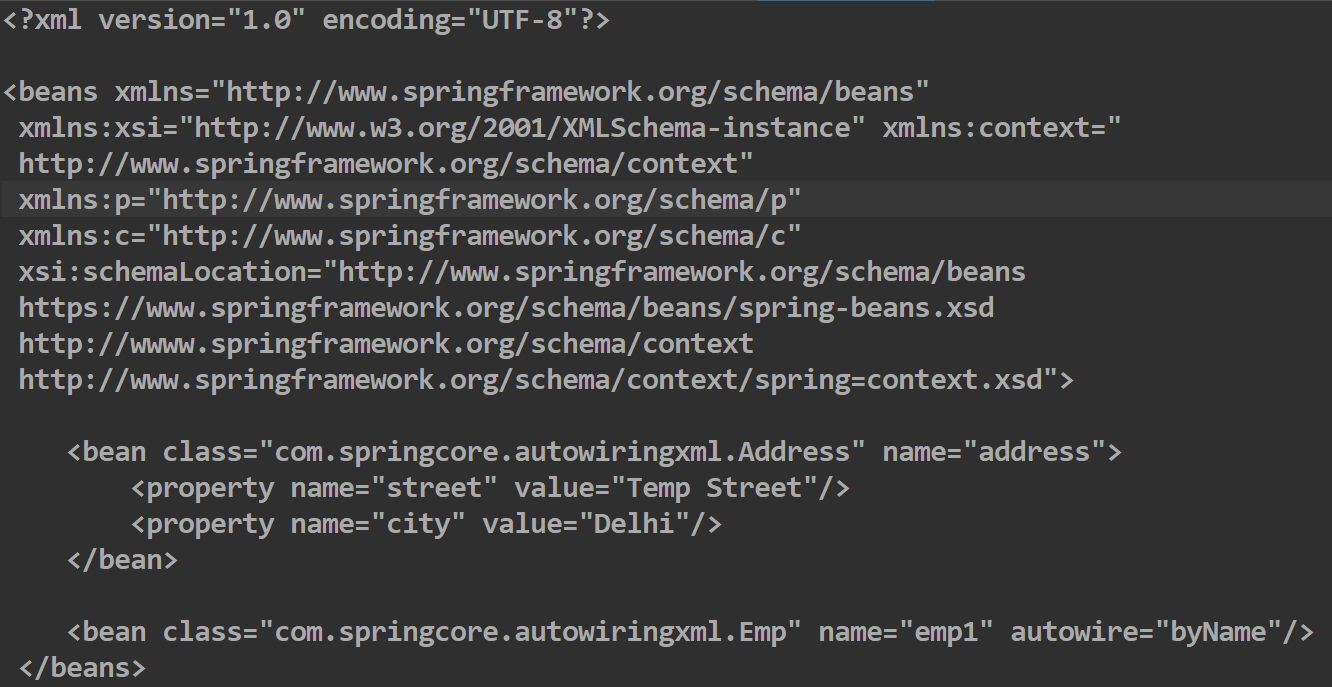


🡺**Autowiring in Spring**

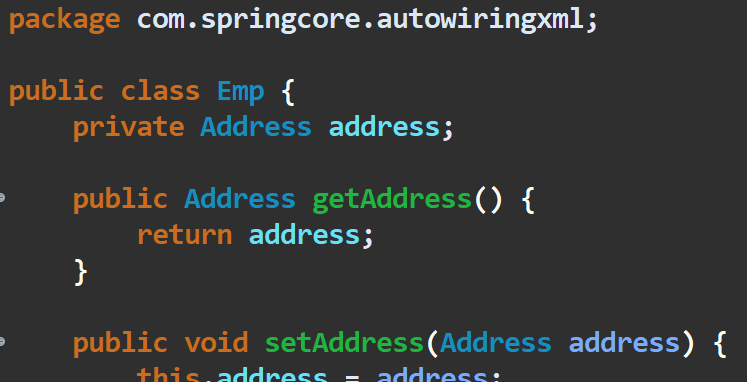
1. Feature of spring framework in which spring container inject the dependencies automatically
2. Autowiring can’t be used to inject primitive and string values. It works with reference only
3. Autowiring can be used in the XML and Annotations
4. In autowiring there are autowirnig modes like no, byName, byType, constructor, autodetect(deprecated from spring 3)
5. And if we are working with the annotations than we have to only use the @autowired and it will automatically wire

**🡺Autowiring in XML**

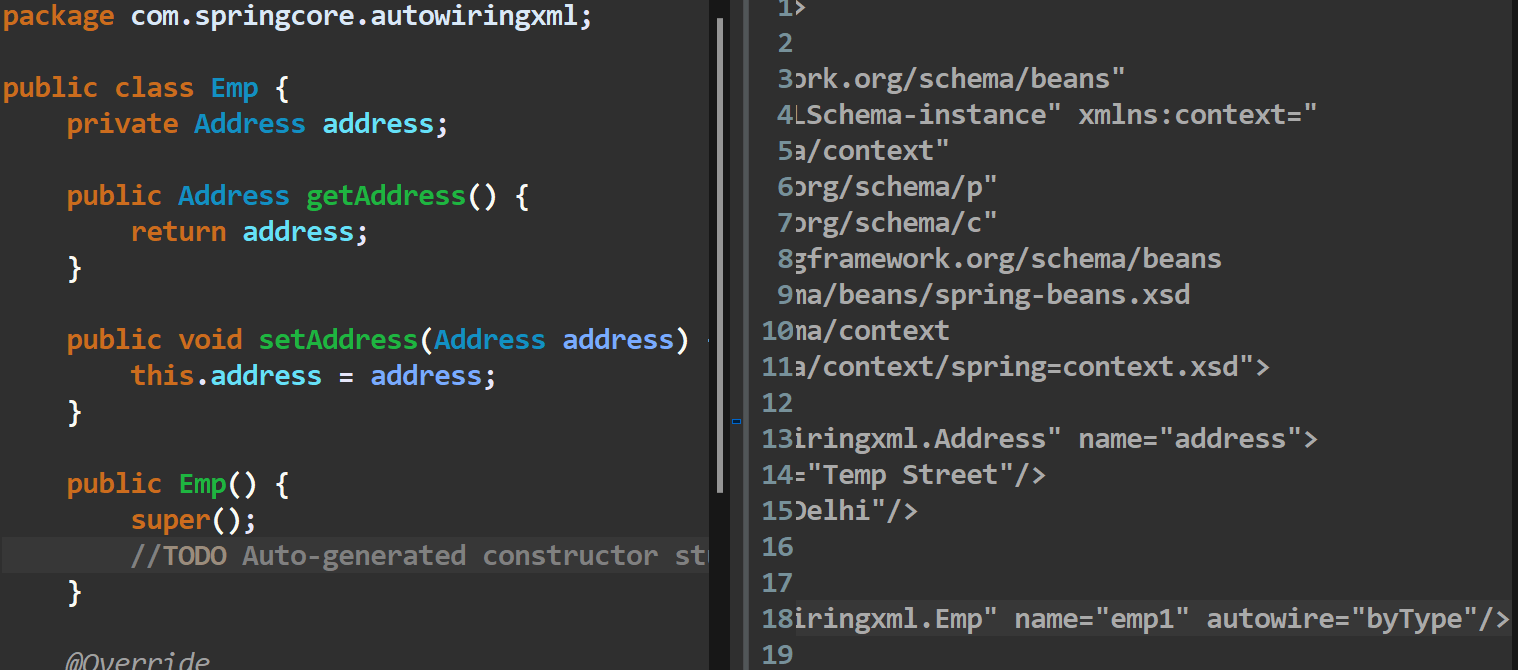
1. Now we have taken two classes named the Emp and Address, And in the Address we have two fields the street and, The Emp is dependent on the Address
2. And we have to made nothing different in the controller Class, everything is as it is, First we will set the values of the street and city in the address, and than we will use it in the Emp class, by help of the autowire in the property tag, Example:



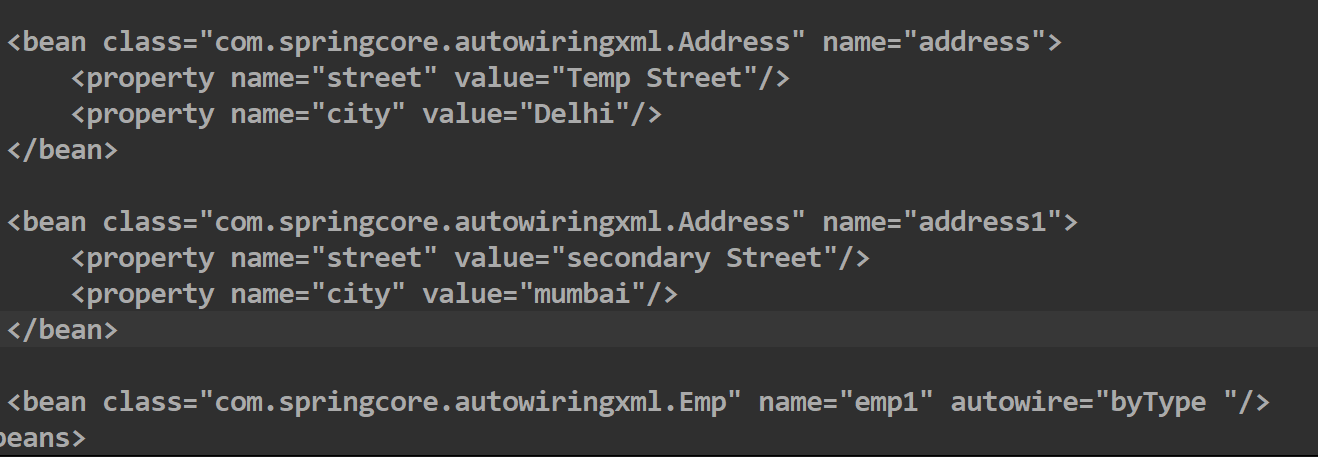
1. But make sure, that the name of the bean that we maked of the dependency class(i.e. Address) is same to that of the object of the object of the address in the dependent class, Otherwise the autowire will not work, Example: (Image of the Emp class)



1. Do see that we have used the byName in the autowire in the xml file
2. We can also use the byType instead of byName in the autowire, by which it will not match the name but it will find that if any bean exists which have type same as the type required in the dependent class, And if it is there than it will use it, Example:



1. Now, If we have the two beans of the same type than the spring container would get confuse and give the exception, That there exists the two bean of the same type, Example:

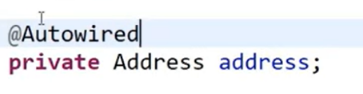


1. Now comes the constructor, by which we can get the bean by help of the constructor, in which we just have to write the **autowire=”constructor”**

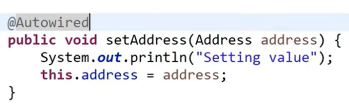
**🡺Autowiring Using Annotations**

🡪We can use the **@Autowired** annotations by keeping it above the any one of the following three things:

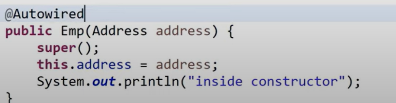
1. Property



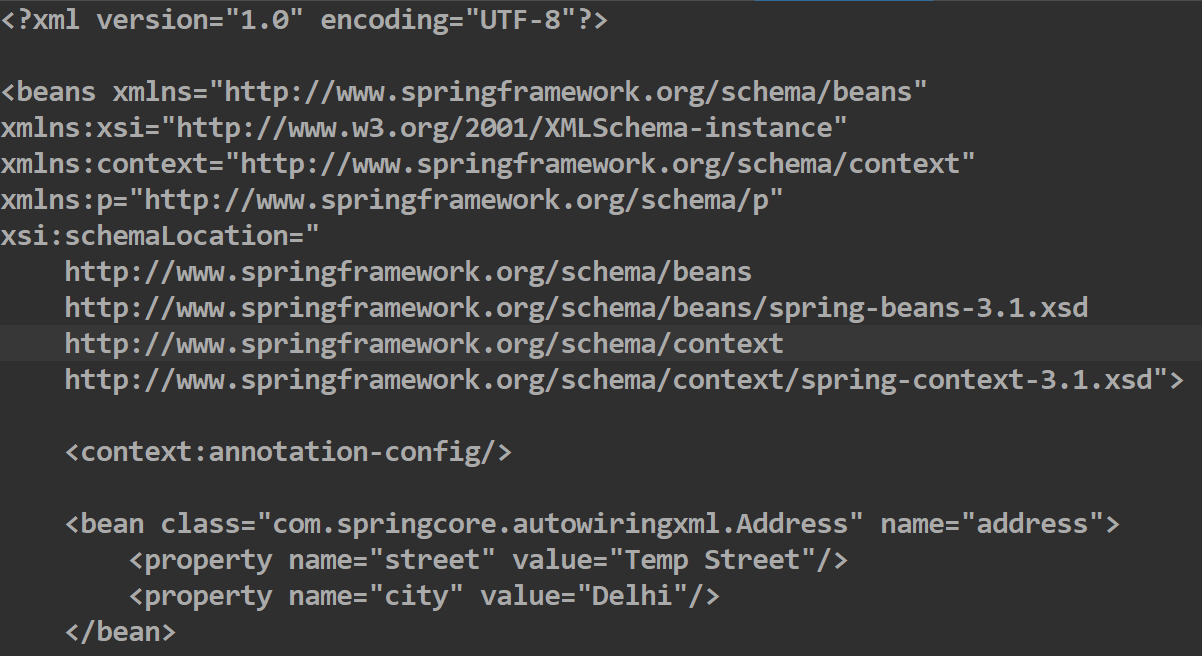
1. Getter Method



1. Constructor



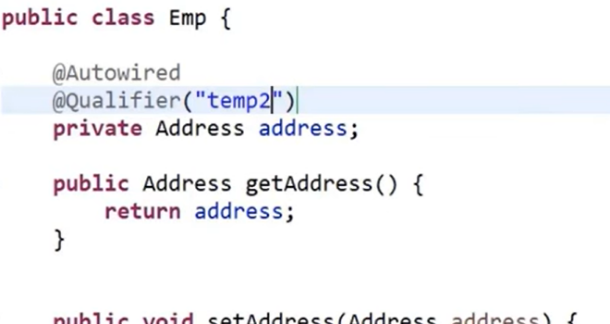
🡪We just have to make one change in the spring configuration xml file, And that change is to add the following tag after the starting beans tag and before the bean tags, As shown in the figure:



🡪It is recommended to keep the name of the variable at the dependent class and the name of the bean at xml file to keep same, It is not required but helps in the readability of code

🡺Usage of the Qualifier Annotation

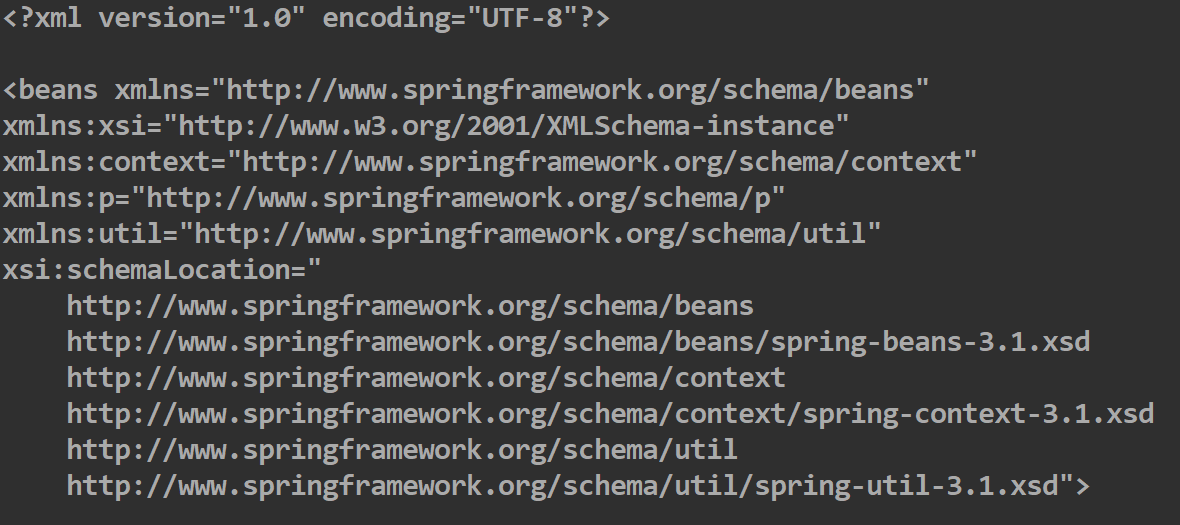
1. Let us assume a case where we have an class named Student and we have the two beans for it bean1 and bean2, And now if we use the **@Autowired** on it than, It will give an error that there exists two beans of given condition bean1 and bean2
2. So to overcome this problem, We have the @Qualifier(“<bean-name-from-xml>”), by which we can define that to choose which of the bean, If there are more than one bean
3. And @Qualifier(“”) annotation is used after the @Autowired annotation, Example:



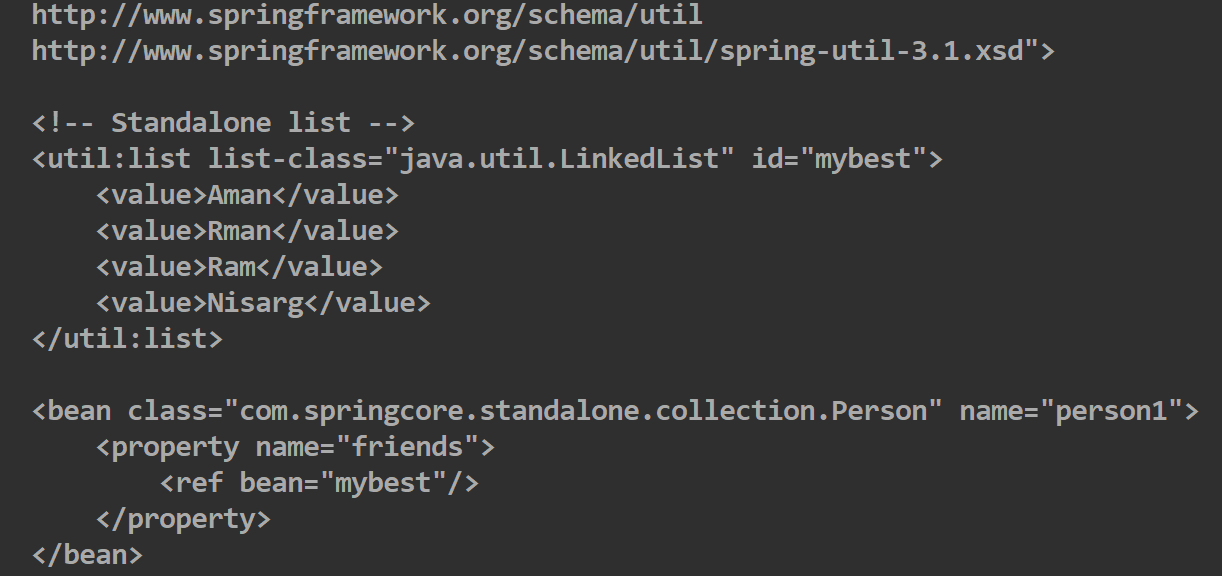
🡺Standalone Collections

🡺For List

🡪We can make the standalone collections in the spring configuration xml file, by first adding the following three things in the name space of the spring,which are as shown in the figure:



🡪Now we can easily make the standalone list and use it, First let see how to make it, We will make it under the beans tag and before starting of bean tags, Example:



🡪For making an list we will make the opening and closing tag of the util:list and in it we will have the list-Class which in which we would define the class of the list like LinkedList, ArrayList, etc and we will give it an unique id to it by the help of which we can use it

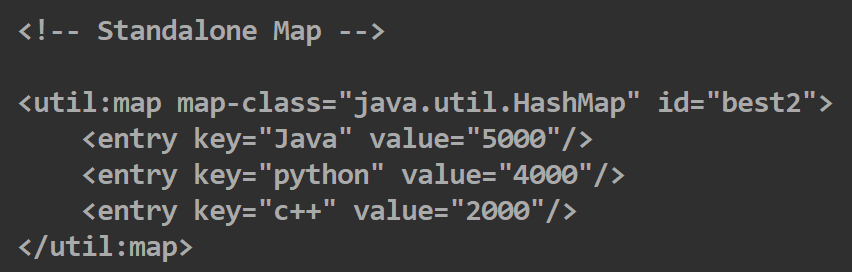
🡪And for using it as the property we will give its refrence inside the property tag by help of ref tag and init we will have bean an in which we will write the id of the list which we want to access, And all other things remains same

🡪We can also write the above property tag also as:

**<property name=”friends” ref=”mybest”/>**

🡺For Map

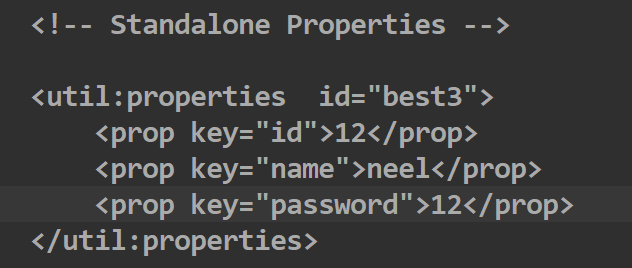
🡪For map only the code with util:list will change to util:map, Example:



🡪We can also get the name of the class by writing following code in the main class:

**System.out.println(context.<obj-of-bean-from-context>.getClass().getName());**

**🡺For Properties**

****