# Spring Framework

## Course flow

1. Spring Core
   1. Dependency Injection
2. Spring Data Integration ,
   1. Spring JDBC , Spring ORM
3. Spring Web
   1. Spring MVC
   2. Spring REST API
   3. Spring BOOT
4. Others
5. Spring AOP

## Prerequisite :

* Core java
* Strong oops
* JDBC
* Servlet and JSP
* Web terminology (HTML ,CSS )
* DB terminology (MYSQL / Oracle /Posgres)

## Chapter 1 : What is Spring ?

1. It is framework of framework
2. It is Dependency Injection Based Framework to make java app loosely coupled and removes

Tight coupling.

1. It provides IOC containers to inject Dependency.
2. It makes Java EE application development easier.
3. RODD Johnson in 2003 development this most popular framework in j2EE app development.

## What is Dependency Injection ? : It is a design pattern.

|  |
| --- |
| Class UserX  {  UserY obj;  Public void doWork()  {  }}In this scenario:  Class UserX want to do some work , and takes UserY help to do it .  **It means UserX is totally dependent on UserY.**  Here UserY is Also another class performing some work along with doing UserX ‘s work.  In old practices we embed dependent class in this case UserY manually in UserX, by new Keyword , Which is tight coupling.  If in future there are changes in UserY class ,  There will be manual work in class UserX also to be done , recompile also.  SO with DI from Spring , we can create Dependent class object and inject it into it.  This complete technique is know as Inversion of control .  i.e object creation control will be completely taken care by Spring where spring will dynamically inject all object dependency .  All the dependency metadata info will be given with the help of XML file or annotation. |

## Spring in J2EE Application real Scenario:

ProductController

Inject

ProductDAO dao = new ProductDAO()

UI Layer

SpringMVC

Security ,

Transaction management

ProductService

Business / Service Layer

cvcv

Inject

ProductDAO dao = new ProductDAO()

DB

ProductDAO

Data Access Layer

SpringJDBC , Spring ORM

With the help of XML / Annotation we can do all this injections

## Chapter 2 : Spring Modules

### Module : Spring core (these modules cover DI , IOC)

* Core
* Beans
* Context
* Expression language

**Core and Beans :**

They provide fundamental features like DI, IOC.

**Context :**

It inherits features from Beans.

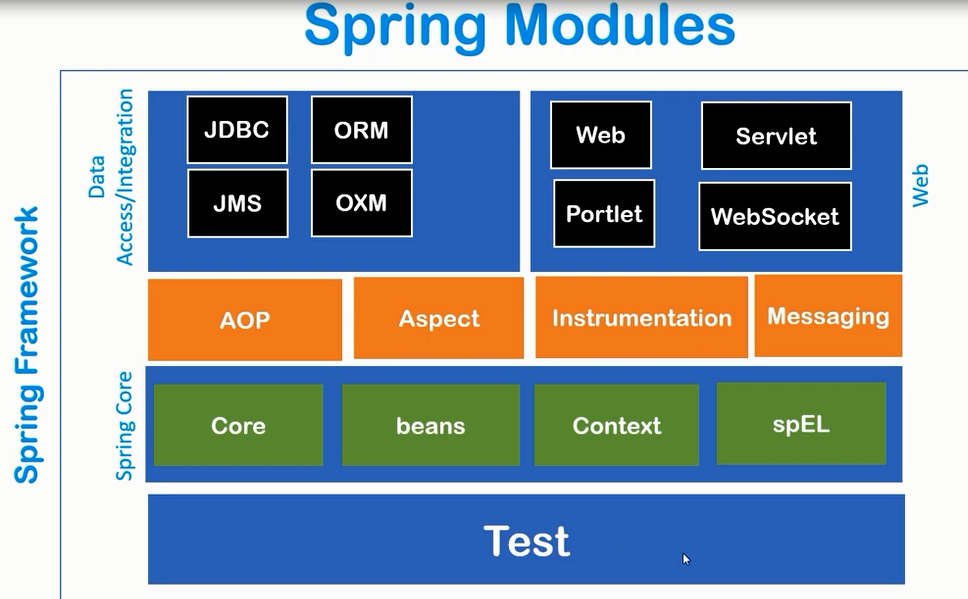
It added internationalization along with all Beans module facility.

It provides EJB , J2EE feature

**Expression Language:**

It is similar to JSTL.

**AOP , ASPECTS , Instrumentations , Messaging**



### Chapter 3 : Spring IOC Container

What is IOC container ?

* It is predefined program
* It is framework component
* Instance creation , memory management , DI injection

First step is to tell IOC container :

* Beans classes
* Configuration

Types :

ApplicationContext :

* this is IOC Container .
* It is derive from BeanFactory .
* It is an interface.
* ClasspathXMLApplicationContext (used when xml configuration are used ), FIleSystemXMLApplicationContext(passing any other file),AnnotationConfigApplicationContext (used when annotation are used )

### Practicals

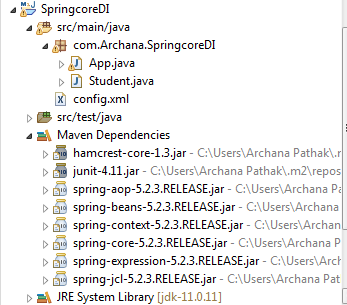
### Example 1 : Setting up Spring core Maven Project

### Example 2 : Simple Dependency Injection (Setter Injection ) concept using ClassPathXMLApplicationContext

## Chapter 4 : Setter Injection (Full practicals)

### Example 1 : Setting up Spring core Maven Project

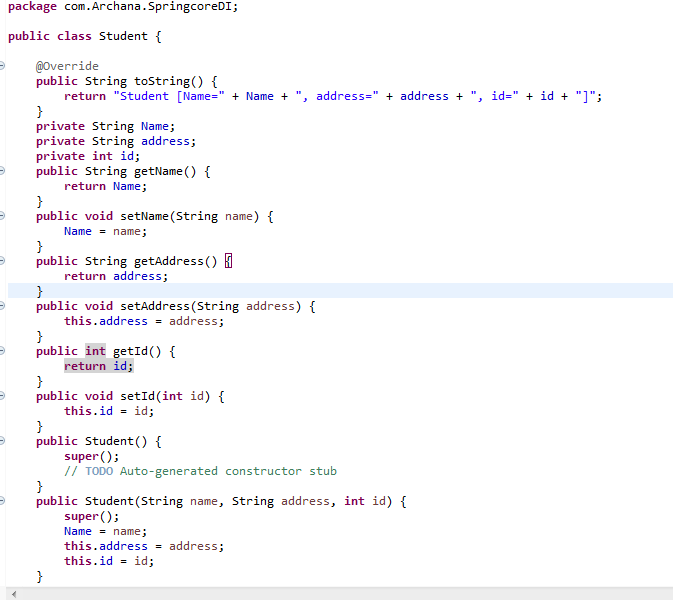
### Project Structure



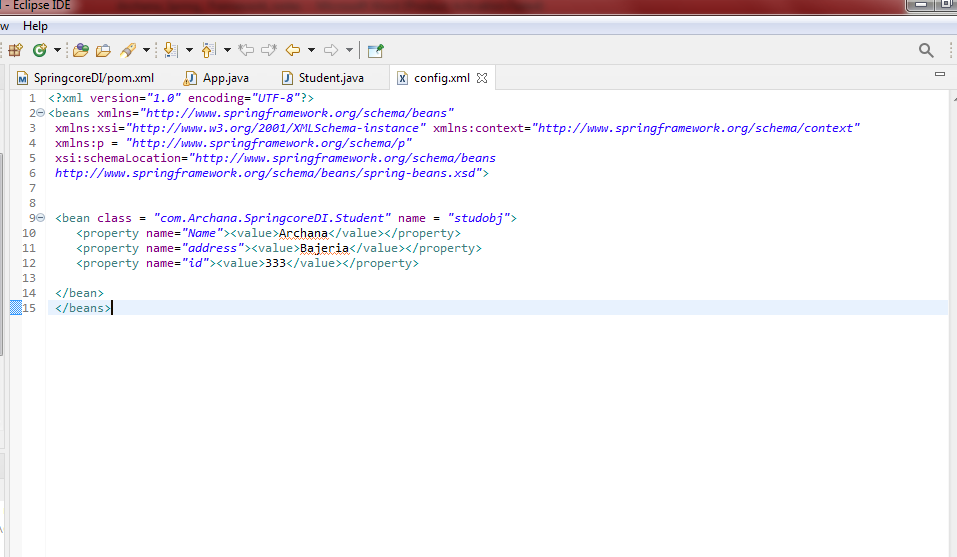
POM with Spring Dependency



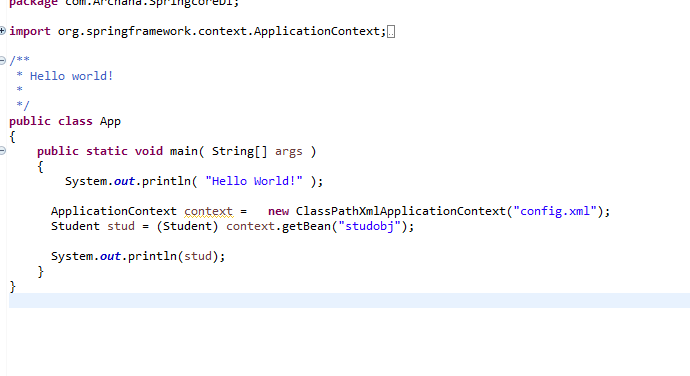
### Example POJO Student .java used



Bean Configuration Sample file : config.xml



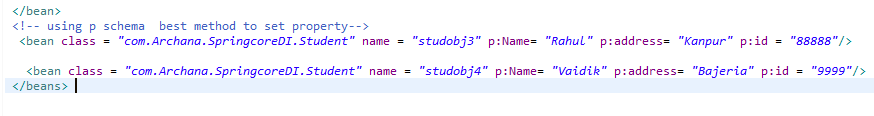
App.java



**Further , Now let us see how to use value as attribute in Bean Property tag.**

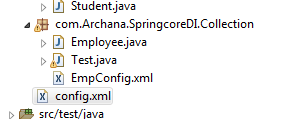


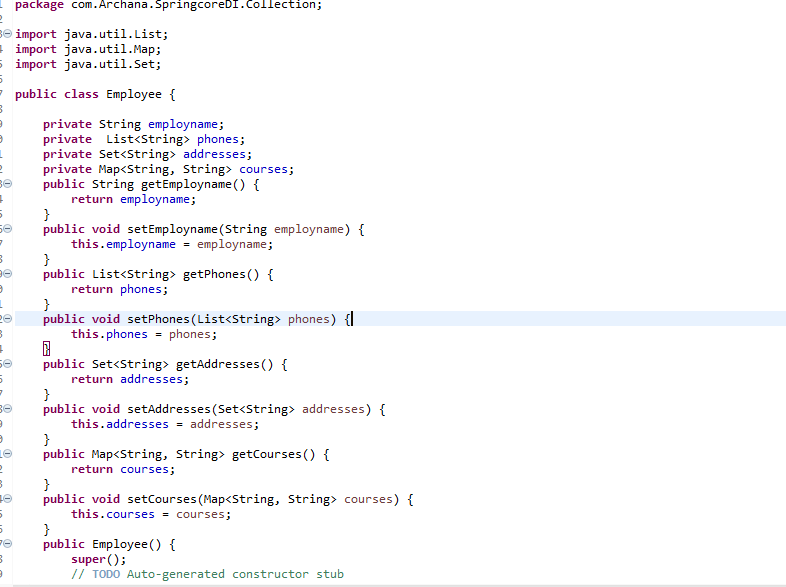
Using P schema



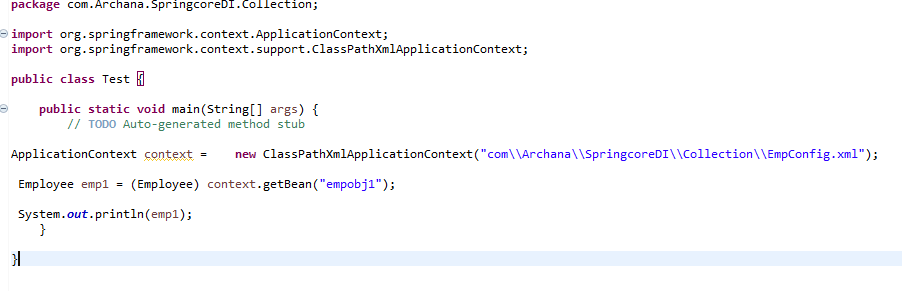


### How to Inject Java collections types

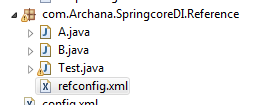


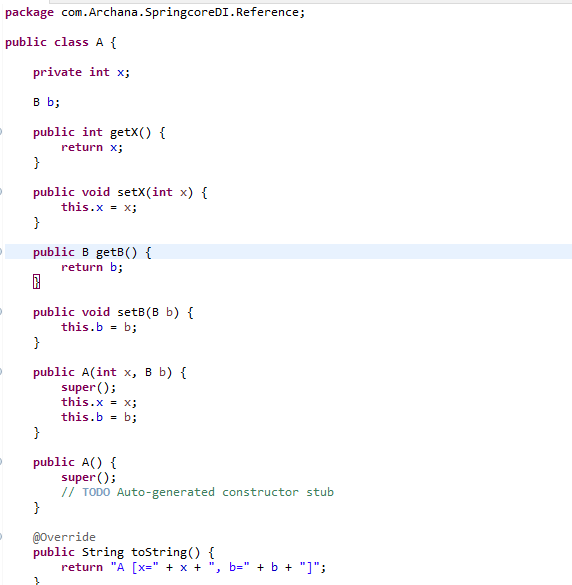


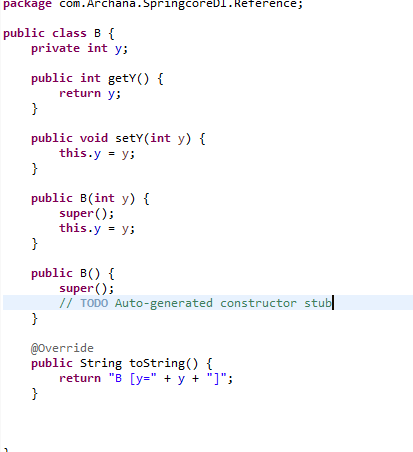


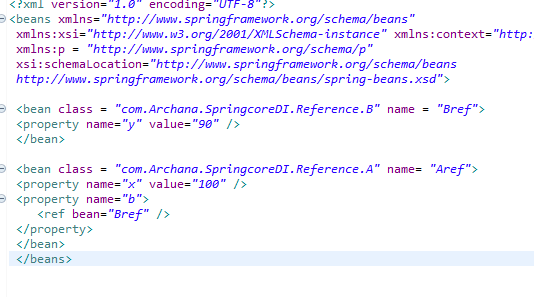


### How to inject Reference type :





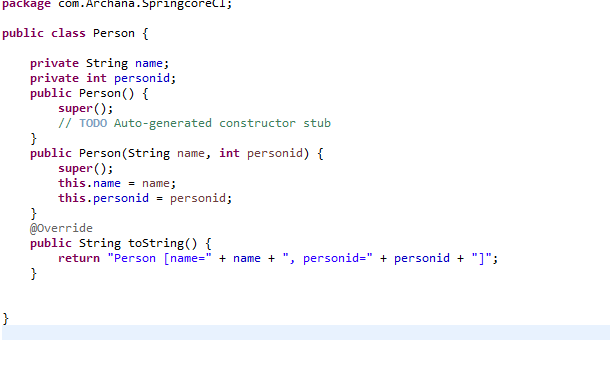




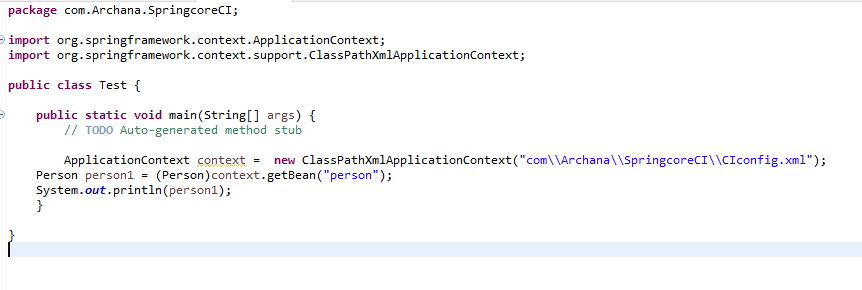


* Setter Injection
* Constructor Injection

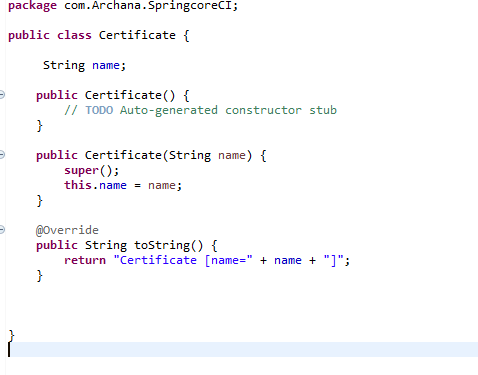
## Chapter 5 : Constructor Injection

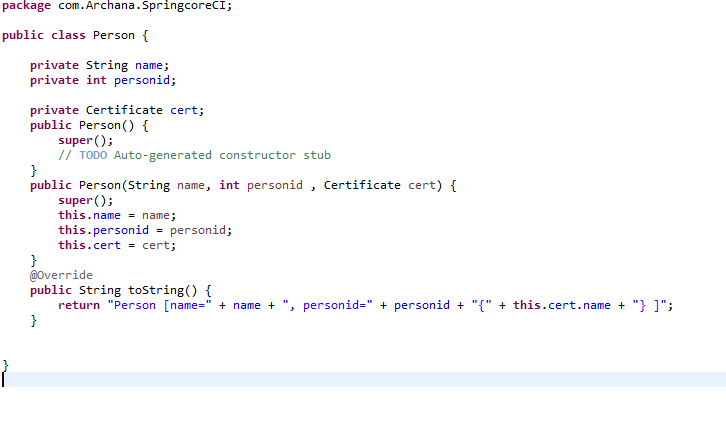


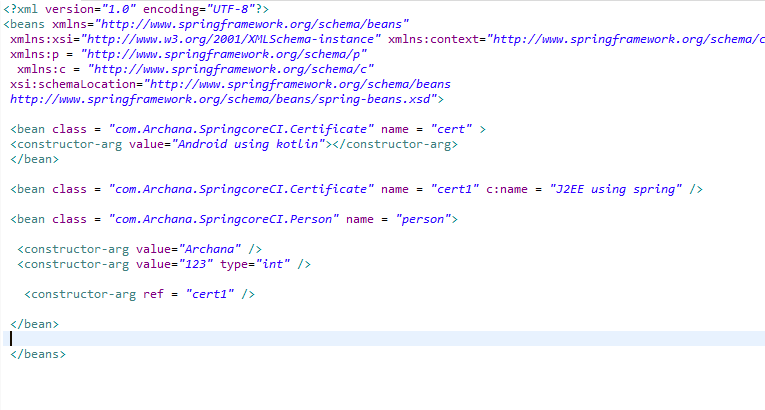


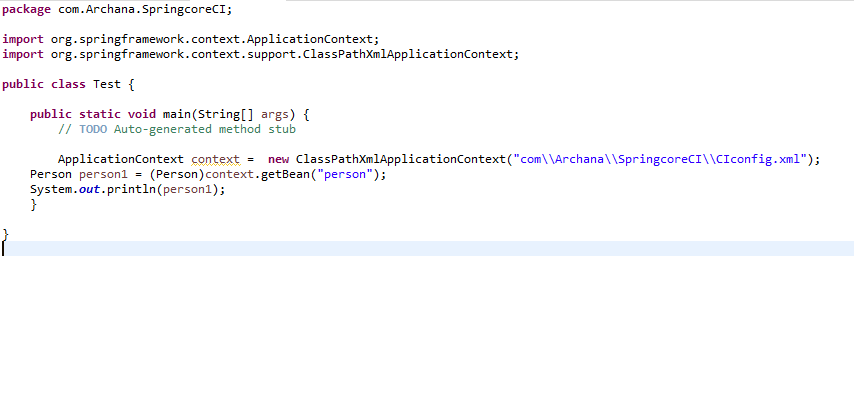


## Injection Reference type using constructor











# Chapter 6 Spring Bean Lifecycle Method

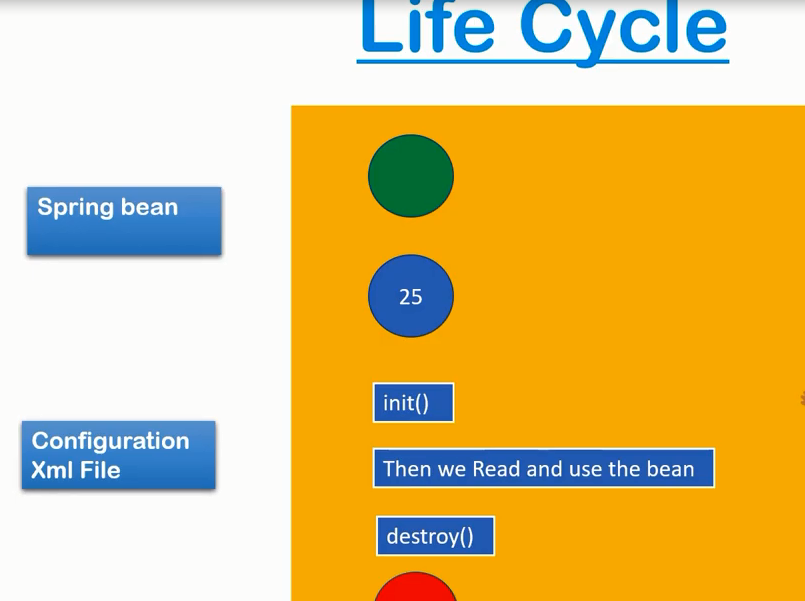
Spring Provide 2 important lifecycle method to every Bean.

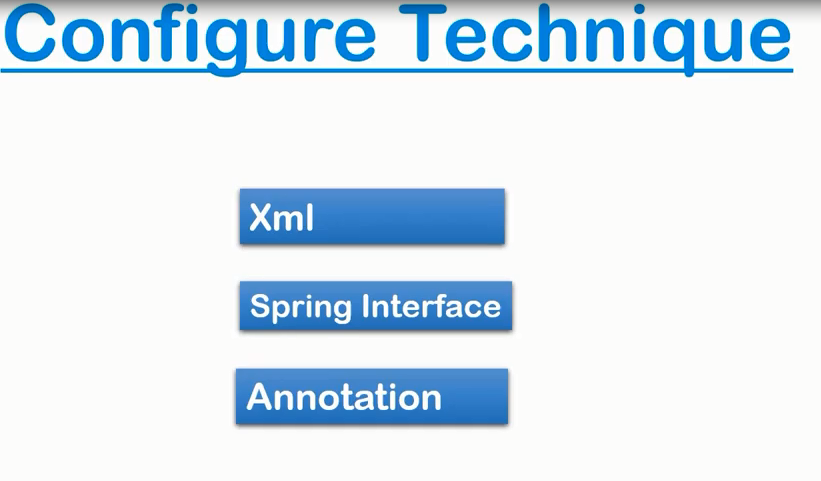
1. Public void Init() - initialization

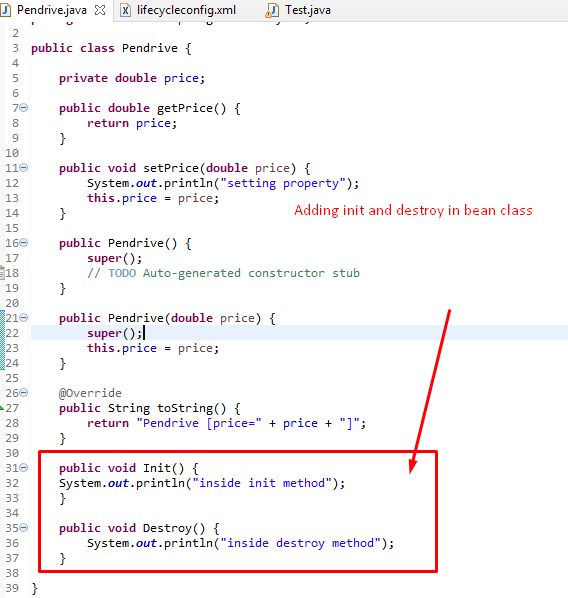
Initialization , code loading , config , connecting db , Webservice etc.

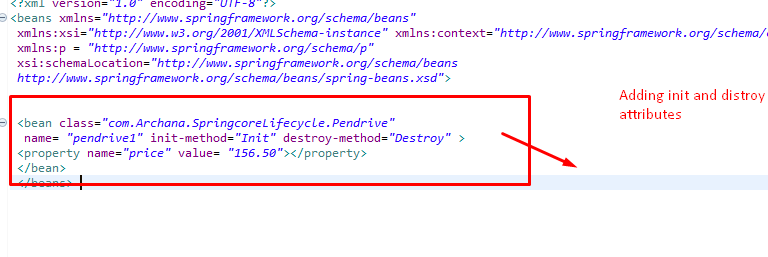
1. Public void destroy() – destroy

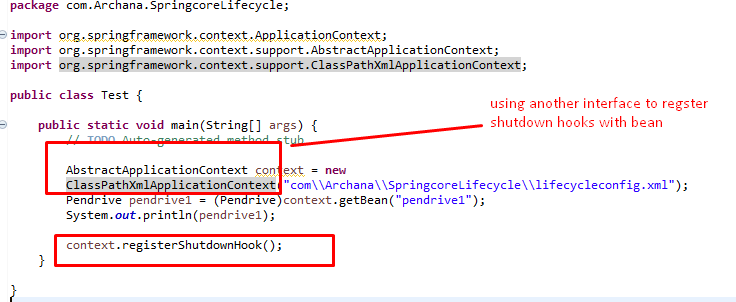
Cleanup code







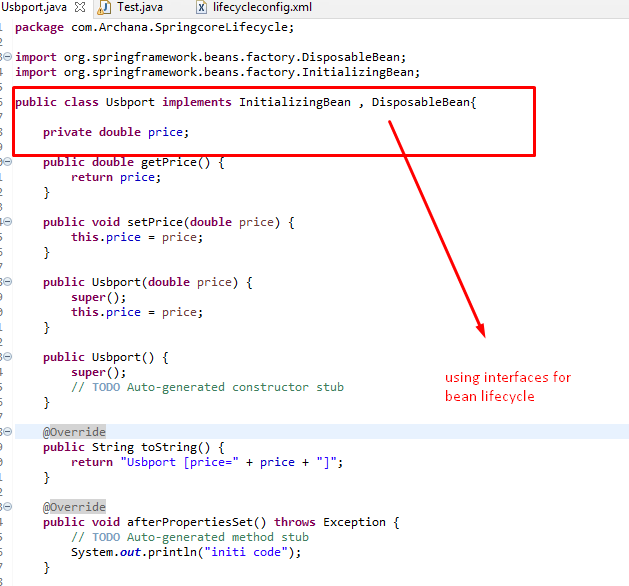


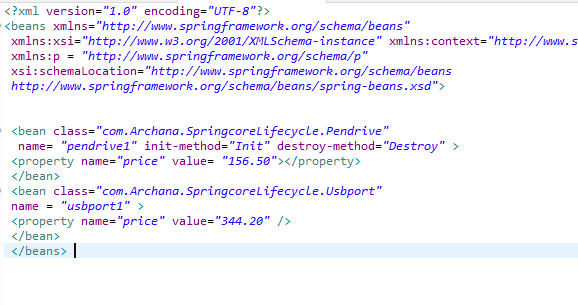


# Chapter 7 Spring Bean Lifecycle using Interfaces

Initializing Bean

Disposable Bean







# Chapter 8 Spring Bean Lifecycle using Annotations

@PostConstruct

@PostDestroy

