Spring Framework

Framework: It allows you to create the complex applications in a simple way, because it takes care of all the common features every application needs so that developers can concentrate on the new functionalities of the project, Framework forces developers to implement the application that is easy to test & maintain, some of the common features the framework provides are:-

1. Exception handling
2. Type conversion
3. Design patterns - related to object creation, initializing the object
4. Internationalization - adapting the applications to specific region or the language
5. Structuring the application - Configurations, Coding, testing and so on

Spring Framework: It is an application framework which can be used to develop any kind of applications like desktop, mobile, web, enterprise, cloud based

It has provided many modules for different types of applications, each modules are independent from other modules so that you can use the ones you need

ex: If you want to connect to the database you can use spring-jdbc module, spring-orm module, spring-data module and so on.

Some of the important modules are:

1. Spring Core: Provides all the basic features or necessities for the application - design patterns, type conversion, exception handling, object creation & initialization, internationalization
2. Spring MVC: For web application & rest based applications
3. Spring Boot: To easily setup the spring related features
4. Spring Cloud: For cloud based applications like microservices
5. Spring Security: Token based Authentication & Authorization (OAuth2.0)
6. Spring AOP: For Cross Cutting Concerns

Spring Core: It is one of the core module in spring that takes care of creating & maintaining the object using Dependency Injection feature, apart from that it provides many design patterns to easily develop the application

Dependency Injection: It is a design pattern where object is supplied to another dependent object ex: Service object supplying to controller, dao object supplying to service, transaction object supplying to service and so on.

Spring IoC: IoC stands for Inversion of Control, it is a container that maintains the objects, since the process of object creation and initializing is reverse it is called as Inversion of Control

Note: Dependency Injection is another way of initializing the object i.e., framework takes care of initializing the object or initializing the dependencies

There are 2 types of dependency injection

1. Setter Injection: Framework supplies the dependencies through setter methods
2. Constructor Injection: Framework supplies the dependencies through Constructor arguments

Benefits of Dependency Injection

* Loosely coupled code
* Flexible to the changes in the implementation

Note: You must use interface based approach when you want to get the full advantage of dependency injection

Usual way of initializing the dependencies i.e., dao of Service that needs instance of EmployeeDao

1. EmployeeDao dao = new EmployeeJdbcImpl();
2. EmployeeDao dao = ObjectFactory.getDaoInstance();

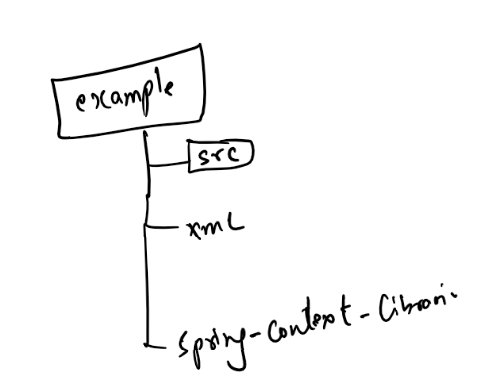
Both the references are going to be initialized by the 2nd statement is loosely coupled compare to the first statement, in the 2nd statement if the implementation object needs to be changed then we only change the factory pattern class no the service layer code

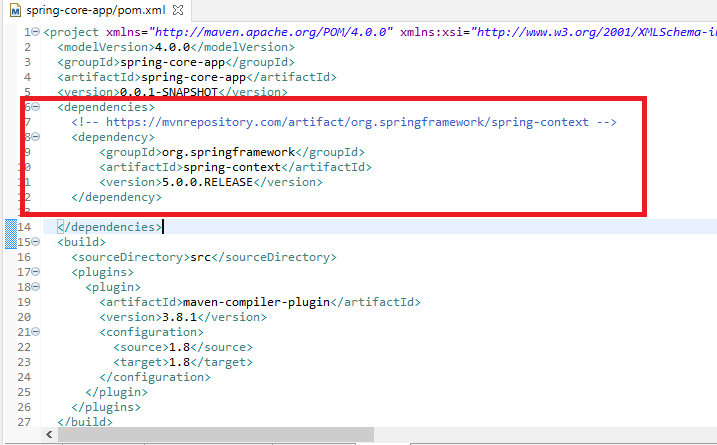
Dependency Injection way of initializing the dependencies

1. EmployeeService(EmployeeDao dao) { this.dao = dao; }
2. setDao(EmployeeDao dao) { this.dao = dao; }

Both the code are called by Framework by supplying the instance of dao, here the framework creates the instance of EmployeeDao & EmployeeService & supplies the EmployeeDao instance to the EmployeeService

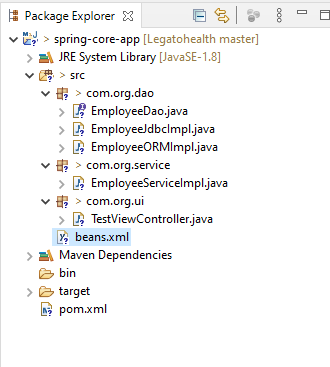
Both the statements are loosely coupled & flexible to the changes as you never create object of either EmployeeService or EmployeeDao because framework takes care of object creation.



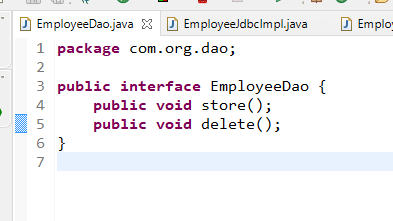


Setter Injection: Object initialization happens via setter method

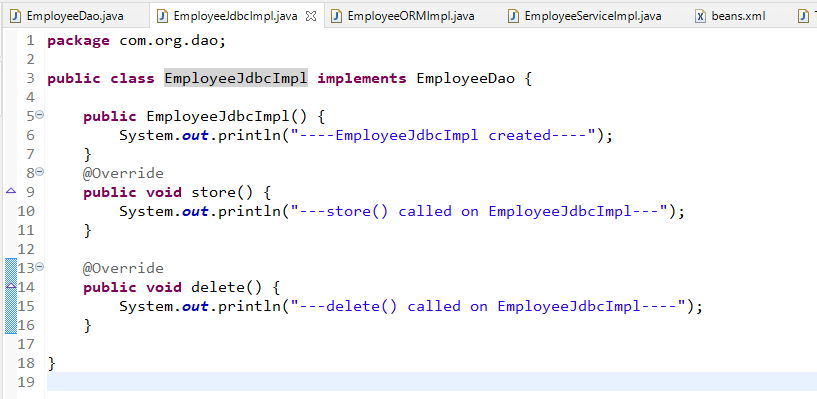
Project Structure:



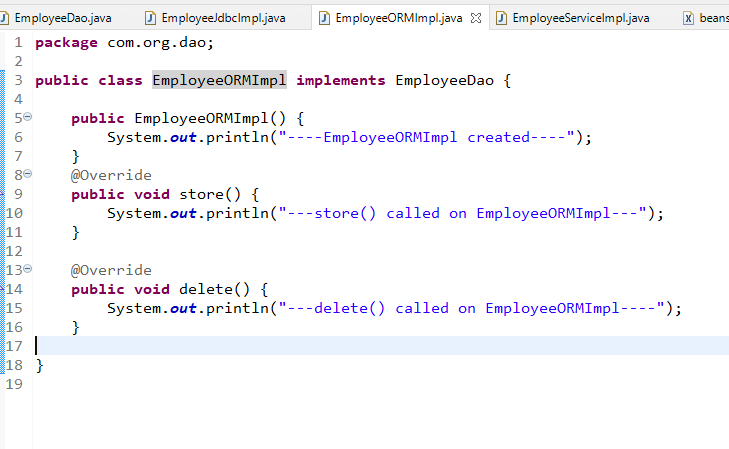
EmployeeDao.java



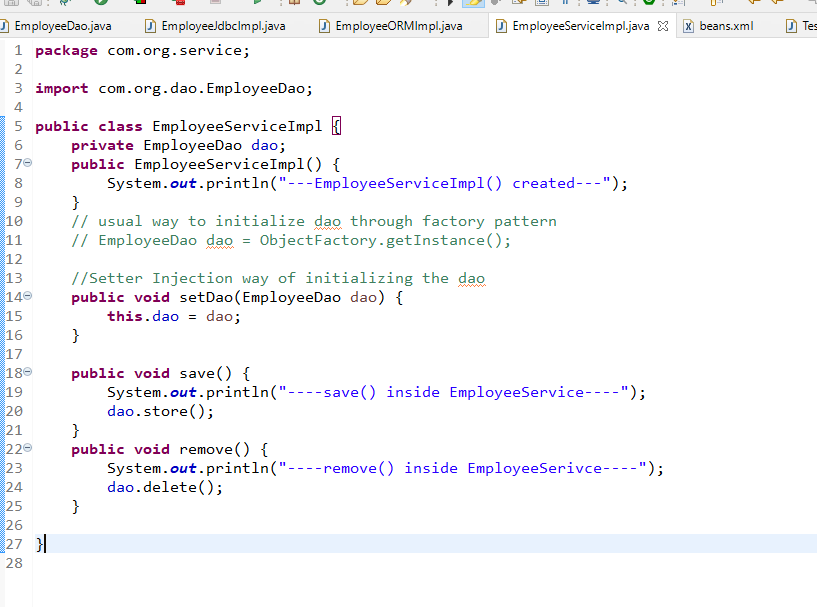
EmployeeJdbcImpl.java



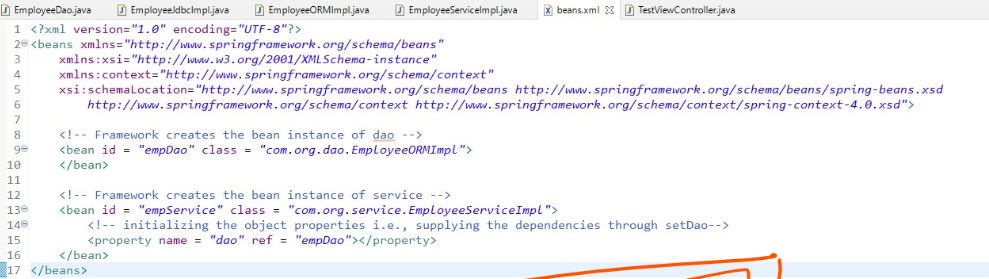
EmployeeORMImpl.java

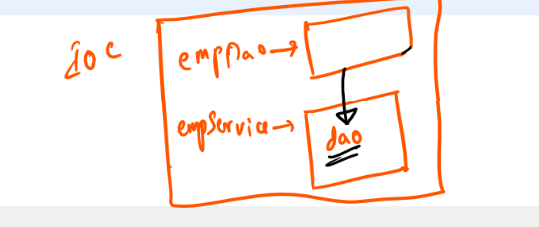


EmployeeServiceImpl.java

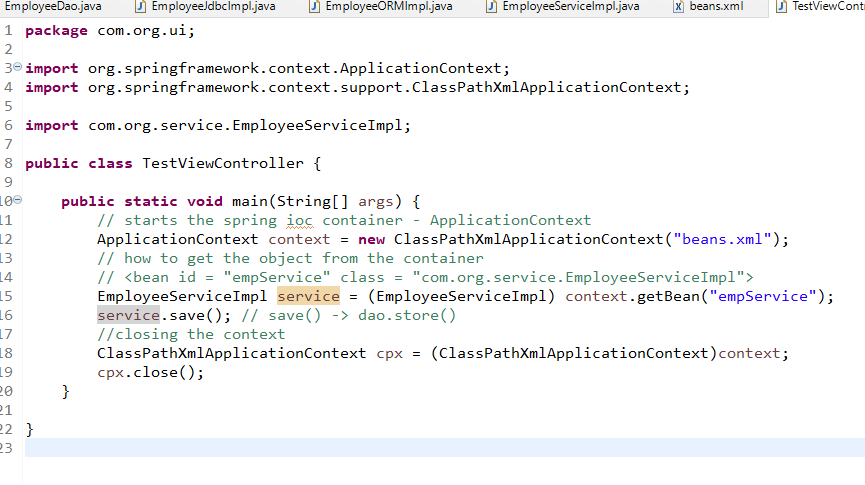


beans.xml

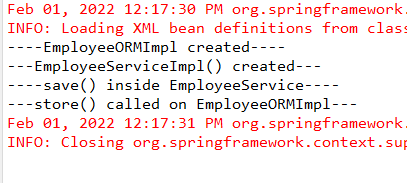




TestViewController.java



Output:

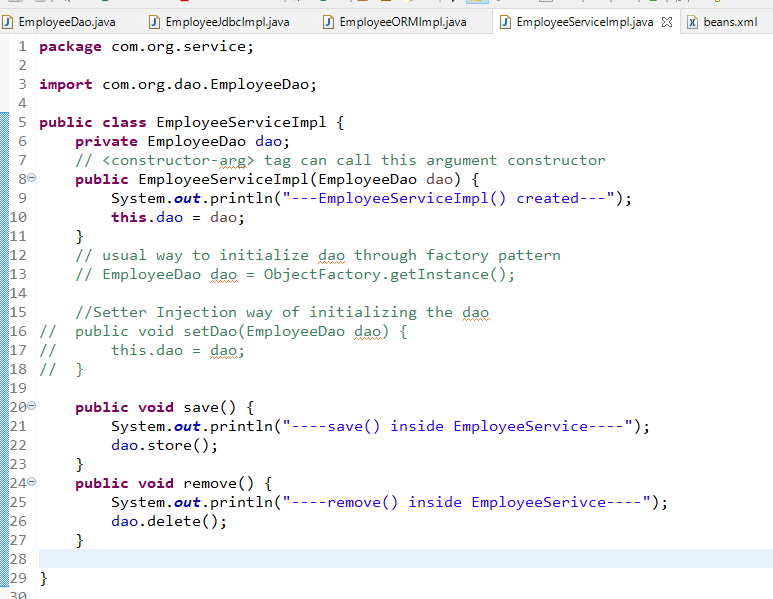


The above code is supplying the EmployeeDao implementation instance using setter injection i.e., it is calling setDao() method & passing the object that implements EmployeeDao, but another way of supplying the dependency is via constructor argument.

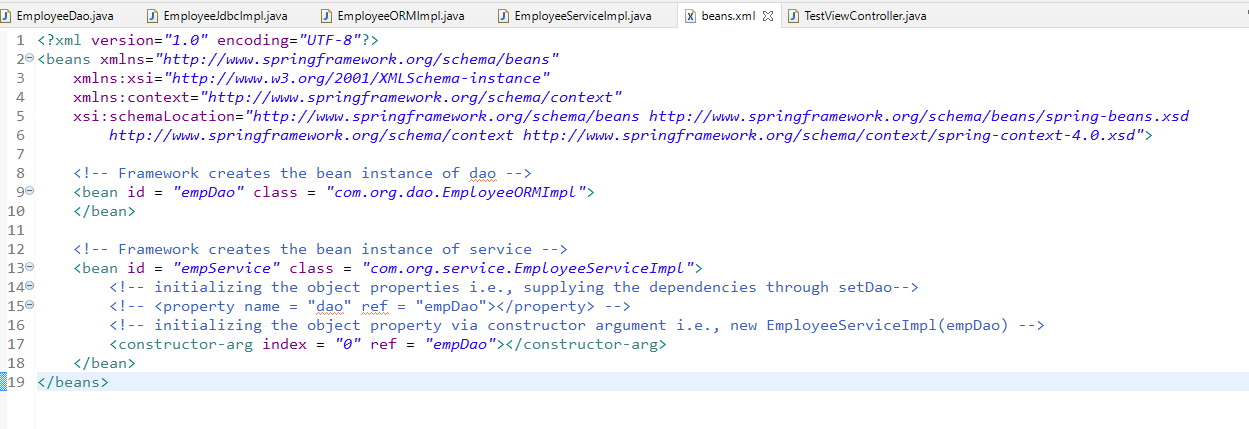
Constructor Injection: Object initialization happens via constructor argument

Sometimes you may have a class where the properties must be initialized only through constructors and after the object is created nobody needs to modify its property in that case you there wouldn’t be setter methods for that property, in such cases you can initialize only by constructors

Change the EmployeeServiceImpl.java

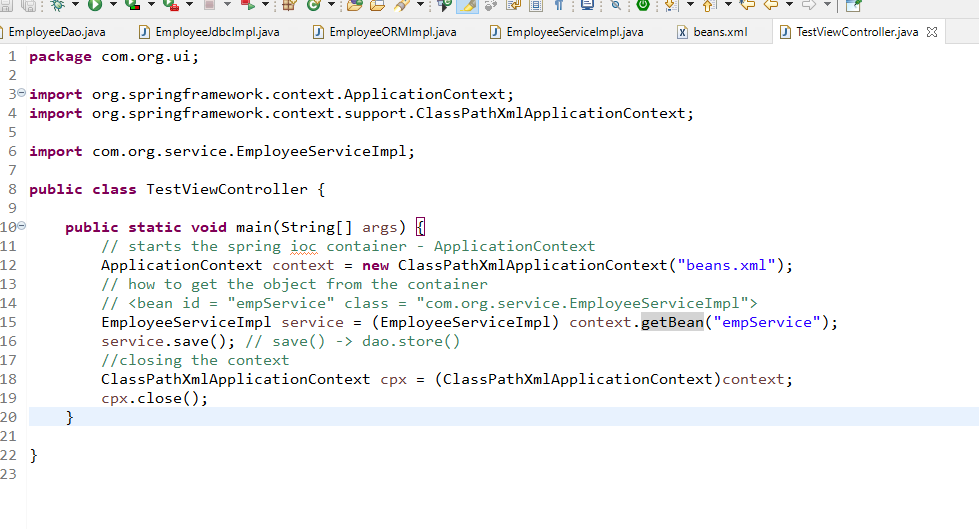


beans.xml



The above configuration supplies the dao to the service using argument constructor,

TestViewController.java

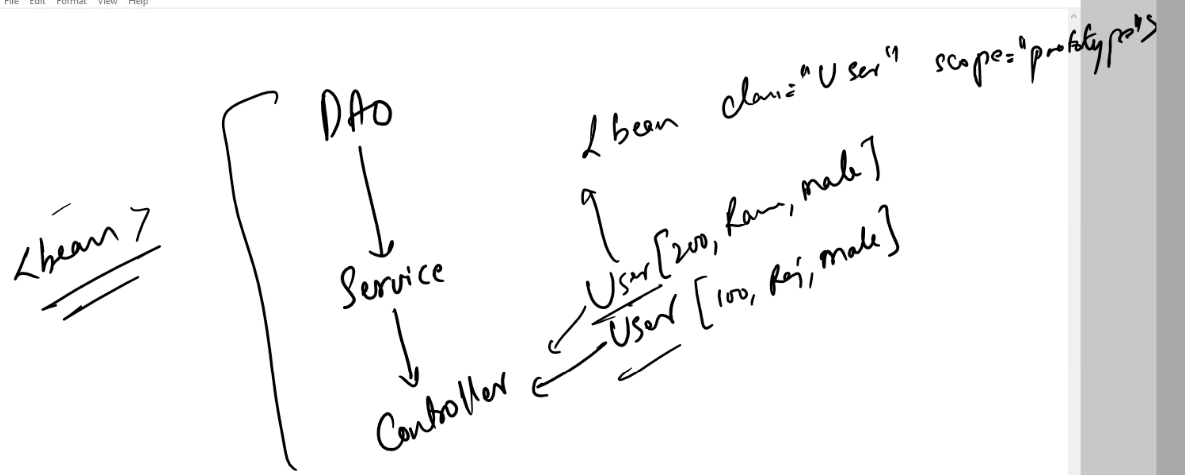


Scopes of bean

1. singleton: By default every bean is singleton per <bean> configuration
2. prototype: This allows multiple instance of the bean to be created per <bean> configuration.

<bean id = ‘xyz’ class = ‘com.org.Xyz’> : Xyz is single ton

<bean id = ‘abc’ class = ‘com.org.Abc’ scope = ‘prototype’>: Abc can have multiple instances



Note: If you omit scope attribute in the <bean> tag then they are considered singleton.