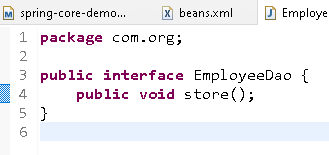
Spring Notes

Spring Framework implements lot of design patterns required to develop the applications like

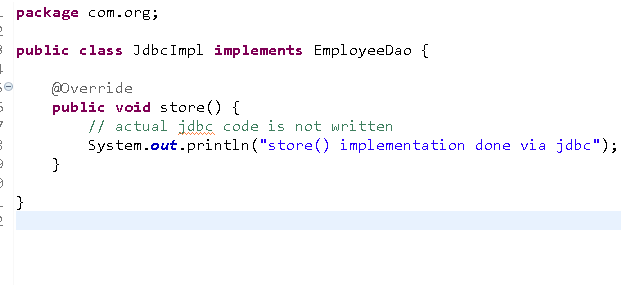
* Dependency Injection
* Singleton pattern
* Front controller pattern
* Proxy pattern
* Factory pattern

Old Approach

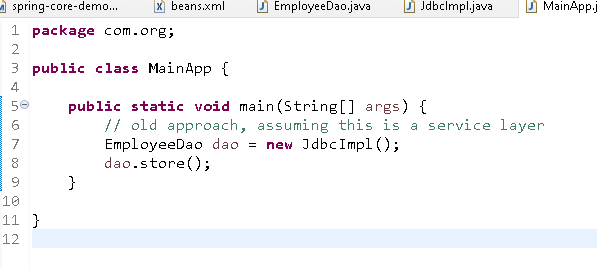
EmployeeDao.java



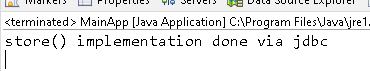
JdbcImpl.java



MainApp.java



Output:



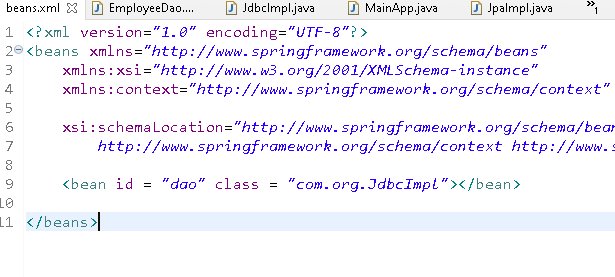
Suppose a new implementation you want to use then you must change the dependency code, which can be completely avoided through spring dependency injection feature

Dependency Injection: process of supplying the dependency to other objects

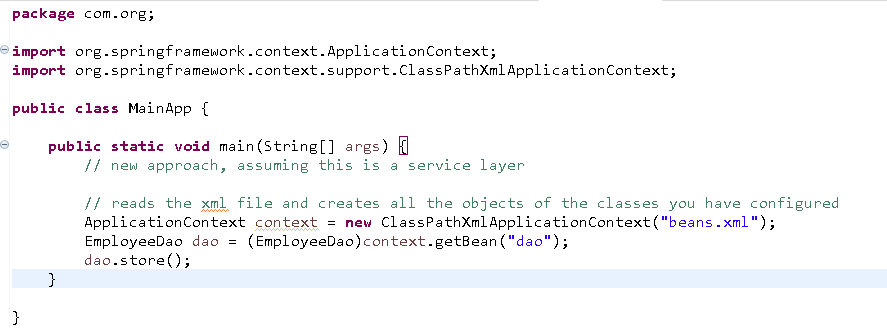
New Approach:

Mention the dependencies to be created in an xml file and use ApplicationContext to load the xml file that enables spring to create the object of all the classes present in the xml

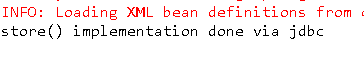
beans.xml



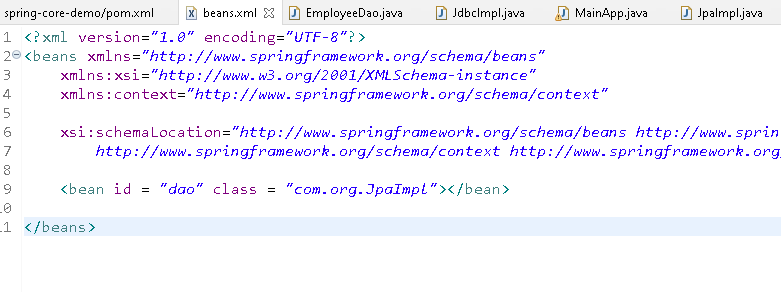
MainApp.java



Output:



Changing the xml to get JpaImpl



Output:



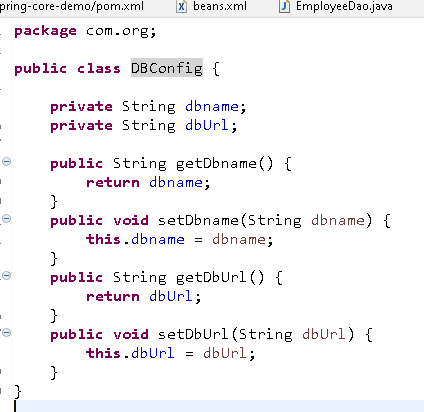
Dependency injection are of two types

1. setter injection
2. constructor injection

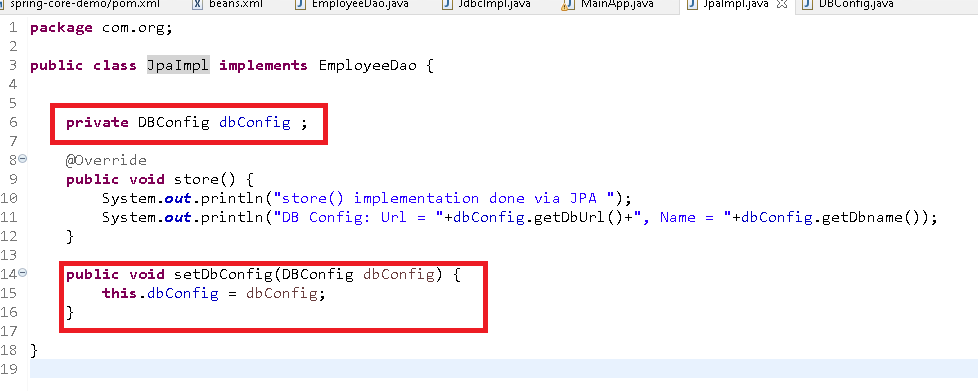
Setter injection means setter method will be called by passing a reference or a value

Constructor injection means parameterized constructor will be called by passing a reference or a value

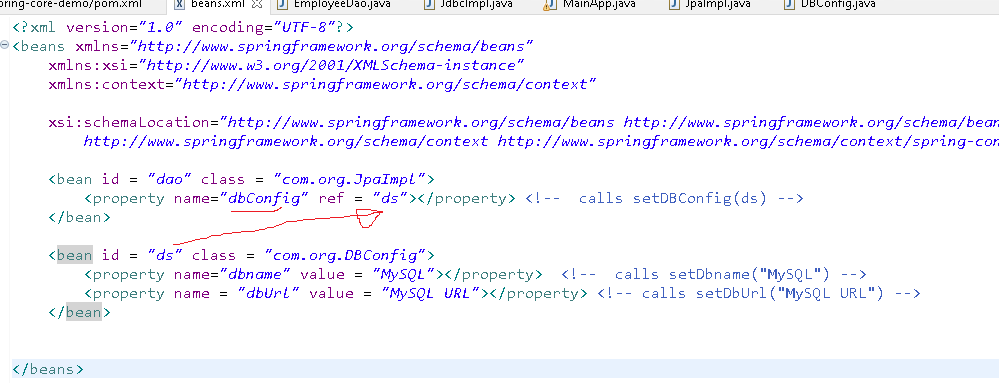
DBConfig.java



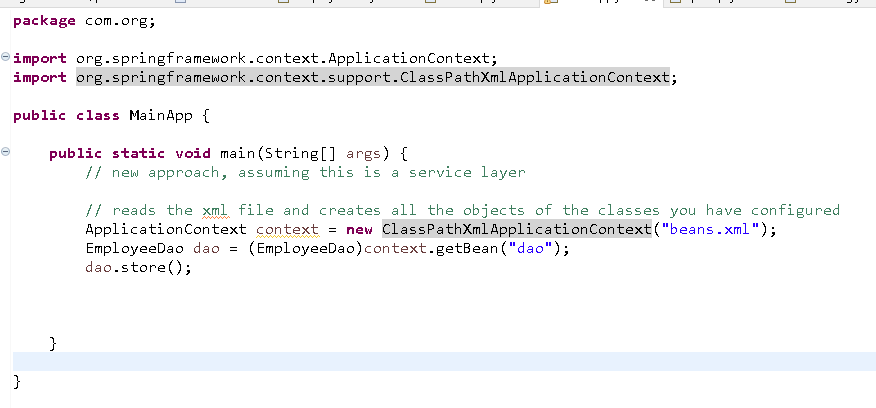
JpaImpl.java



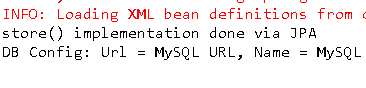
beans.xml



MainApp.java



Output:



Exercise:

Create a Service class named EmployeeService and create a variable of type EmployeeDao and in main method you get the object of EmployeeService, the EmployeeService will have a method storeEmployee() which calls store() method of dao. From main you will call storeEmployee() that prints storeEmployee() method message as well as store() method message of EmployeeDao

