Spring frameworks-

Spring is light weight frameworks. It was developed by Red Johnson in 2003. Spring frameworks make easy development of J2EE applications.

Benefits of spring frameworks-

1. Predefined template-

   It provides templates for hibernate, JPA, JDBC, etc. there is no need to write too much code.

   Example- in JDBC template, you don't need to write code for exception handling, creating connection, creating statement, closing connection. Only the thing is that you need to write the code for executing the query only.

2. Loosing coupling-

   Spring applications are loosely coupled because of dependency injection.

3. Easy to test-

   Spring does not require server to run the program. It only need JDK and Jar file.

4. Light weight-

   Spring is light weight component due to its POJO implementation. It does not force any programmer to inherit the class or implement any interface.

5. Development is very fast.

Difference between heavy weight and light weight components-

EJB are depends on application server. So it is called as heavy weight components.

Spring are not depends on application server. They just use your JDK and jar file to run application.

Difference between tightly dependency and loosely dependency.

Tightly coupled dependency-

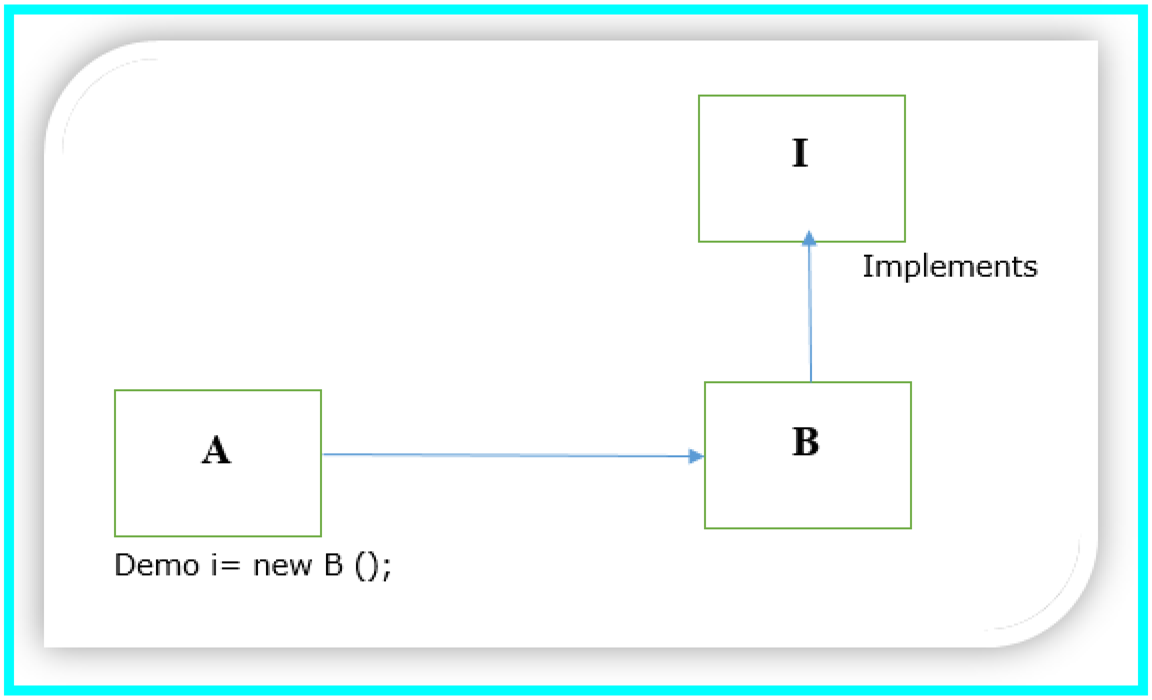
Suppose I have two classes A & B, I have created the object of B class into A class as B b=new B () called as tightly coupled dependency.

Example-

```
package com.test

public class A {

        B b = new B ();
}


public class B{


}
```

Loosely coupled dependency-

Suppose I have two classes A & B which will implements interface test.



```
package com.test

public class A extends B {

        Demo i= new B ();
}

package com.test

public class B implements Demo {

}

package com.test

public interface Demo {

}
```

Here class A is compatible with any class. In future, if you don't want B class then you can create the one more class named as class C and pass the object as below. Demo i= new C (); // i is the interface reference.

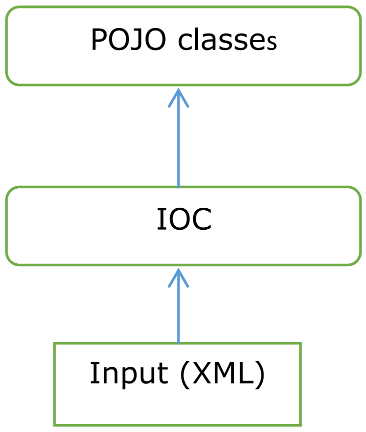
So in this way, we can achieve loosely coupled dependency.

Spring IOC-

IOC container is responsible to instantiate, configure and assemble this objects.

Why container?

If you want to pass / apply any input to pojo classes. You must have container supports.



What does IOC container?

1. Read the XML file.
2. Create the instantiation of xml bean(java classes)
3. It will manage the life cycle of your bean classes.
4. Passing the dynamic parameter to bean(java classes).

There are two types of IOC container as

BeanFactory-

It is called as core container.

By using this, we can develop the standalone application.

XMLBeanFactory is the implementation class for BeanFactory(I).

To use BeanFactory, we need to create the instance of XmlBeanFactory class.

Resource resource= new classPathResource("applicationContext.xml");

BeanFactory factory= new XmlBeanFactory(resource);

The constructor of XmlBeanFactory class receives resource object so we need to pass resource object to create the object of BeanFactory.

ApplicationContext-

It is called as J2EE container.

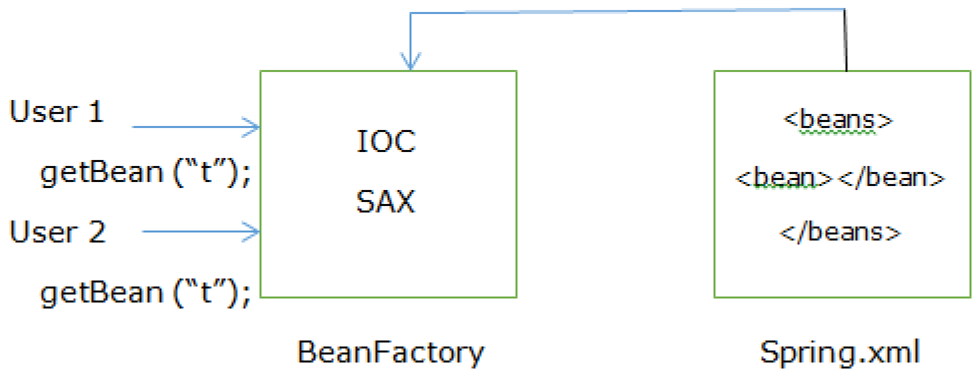
ClassPathXmlApplicationContext is implementation class for ApplicationContext(I).

To use applicationcontext, we need to create the instance of ClassPathXmlApplicationContext as given below-

ApplicationContext context= new ClassPathXmlApplicationContext("applicationcontext.xml");

The constructor of ClassPathXmlApplicationContext class receives string object so we can pass name of xml file to create the object of ApplicationContext.

BeanFactory (Internal working)-



1. At a time of first user request, it will create object of bean class.
2. If second user send request, it will return the same instance, not create the new object for second user.
3. Because default scope of bean is singleton.

Note-

1. SAX Parser will check your document at loading time.
2. BeanFactory not create object at loading time.

ApplicationContext (Internal working)

1. At time of class loading. It will create object of bean class.
2. Suppose you have 10 declaration of bean then it will create the object of, if and only if scope is singleton.
3. If scope is prototype, at loading time, it will not create the instance.

Note-

1. In case of BeanFactory and ApplicationContext, if scope is prototype then it will create the object per user request.
2. If constructor is private or public then spring will create the instances in both cases(By using reflection only).