CourseCube®
(Formerly Java Learning Center)

Spring 5.2
Study Guide

Author
Srinivas Dande

Java Learning Center

Spring Framework

- The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications
- Spring Framework is a Java platform that provides comprehensive infrastructure support for developing Java applications.
- Spring handles the infrastructure so you can focus on your application.
- Using Spring Framework,We can develop Large-Scale , Distributed High-Performance Applications

Spring Framework Modules

Module 1: Spring IOC

Module 2: Spring AOP

Module 3: Spring DAO Support.

Module 4: Spring with JDBC

Module 5: Spring with Hibernate

Module 6: Spring Transaction Management

Module 7: Spring Web MVC

Module 8: Spring Rest WebServices.

Module 9: Spring Security

Module 10: Spring with Java Mail

Spring Other Projects

1) Spring Boot
2) Spring Data.
3) Spring Cloud
4) Spring Messaging
5) Spring AMQP
6) Spring Kafka
7) Spring Android.
8) Spring Batch
9) Spring Social.
10) Spring Web Flow.
11) Spring LDAP.
12) Spring Hadoop.
13) Spring Scala.
14) Spring Mobile.
15) Spring Roo.

Spring IOC (Inversion Of Control)

- When you are developing any component or bean, it may contain some dependencies.

class A{
void m1(){ ... }
}
 	
class B{
void m2(){ ... }
}
	class Hello{
	
	A aobj=null;
	B bobj=null;
	
	void show(){
	aobj.m1();
	bobj.m2();
	}
	}

- Hello Bean needs A object and B object.
- Hello Bean needs A resource and B resource.
- Hello Bean needs A Bean and B Bean

- Hello Bean is depending on two beans called A bean and B bean.
- Hello Bean dependencies called A and B has to be Initialized.
- Hello Bean dependencies called A and B has to be Injected
- Spring container Injects the Hello Bean dependencies called A and B.

Dependency Injection

- Dependency Injection is the process of Injecting bean dependencies automatically.
- Spring Dependency Injection uses 3 ways to inject the dependencies
  1) Setter Injection
  2) Constructor Injection
  3) Field Injection (using Annotations)

Ex:

```
class Student{
        String sid;                     //1 Setter Injection

        String sname;           //2 Constructor Injection

        @Autowired
        Address add;                    //3 Field Injection.( using Annotations)

        public void setSid(String sid){
         this.sid=sid;                  //1 Setter Injection
        }
        public  Student(String sname){
         this.sname=sname;              //2 Constructor Injection
        }
}
```

Download the Spring Jars

1) Open this url
   https://repo.spring.io/release/org/springframework/spring/
2) Click on 5.2.5.RELEASE
3) Click ON spring-5.2.5.RELEASE-dist.zip ... It dowloads the ZIP
4) Unzip the spring-5.2.5.RELEASE-dist.zip then you will see the folder called spring-framework-5.2.5.RELEASE
5) You can see 63 Jars under libs
6) These 63 jars as belongs 3 groups
   a. 21- binary jars (We need this only)
   b. 21-source jars
   c. 21 - java doc jars

7) Copy 21-Binary Jars into required Folder to use in your Eclipse Projects.

First Spring Example Using Java Configuration

1) Open the Eclipse with Some Workspace.
2) Create the Java Project with the Name : Lab1
3) Add the JDK1.8 to Project.
4) Add 21 Spring-5 Jars to Project build path.
5) Create the Package called com.coursecube.spring
6) Write the following 3 Beans
   a. A.java
   b. B.java
   c. Hello.java
7) Write the Client Program without Spring
   a. Lab1A.java
8) Run Lab1A
9) Configuring the Beans in Configuration Class
   JLCAppConfig.java
10) Write the Client Program
    a. Lab1B.java
11) Run the Lab1B

Lab1: Files required

1. Lab1A.java	**2. Lab1B.java**
3. A.java	**4. B.java**
5. Hello.java	**6. JLCAppConfig.java**

1. Lab1A.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab1A {
public static void main(String[] args) {

//Without Spring
//Task1 : Create and Initialize A object
A ao=new A();
ao.setA(101);
ao.setMsg("I am A");

//Task2 : Create and Initialize B object
B bo=new B(102,"I am B");

//Task3 : Create and Initialize Hello object
Hello h=new Hello(bo);
h.setAobj(ao);

h.show();

}
}
```

2. Lab1B.java

```
package com.coursecube.spring;

import
org.springframework.context.ApplicationConte
xt;
import
org.springframework.context.annotation.Annot
ationConfigApplicationContext;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab1B {
public static void main(String[] args) {
//With Spring

ApplicationContext ctx=new
AnnotationConfigApplicationContext(JLCAppCo
nfig.class);
System.out.println("---------Now Spring
Container is Ready-----");

A aobj=(A)ctx.getBean("createA");
System.out.println(aobj);

B bobj=(B)ctx.getBean("BO");
System.out.println(bobj);

Hello h=(Hello)ctx.getBean("myhello");
h.show();

}
}
```

3. A.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class A {
 int a;  //S.I
 String msg; //S.I
 static {
              System.out.println("A - S.B");
 }
public A() {
             System.out.println("A - D.C");
}
public void setA(int a) {
             System.out.println("A - setA()");
             this.a = a;
}
public void setMsg(String msg) {
             System.out.println("A - setMsg()");
             this.msg = msg;
}
 public String  toString() {
             return ""+a +"\t"+msg;
}
}
```

4. B.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class B {
int b;  //C.I
String str; //C.I
static {
              System.out.println("B - S.B");
}
public B(int b, String str) {
             System.out.println("B - 2 arg");
             this.b = b;
             this.str = str;
}
public String  toString() {
             return ""+b+"\t"+str;
}
}
```

5. Hello.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {
A aobj; //S.I
B bobj; //C.I
static {
System.out.println("Hello - S.B");
}
public Hello(B bobj) {
System.out.println("Hello(B) - 1arg");
this.bobj = bobj;
}
public void setAobj(A aobj) {
System.out.println("Hello-setAobj()");
this.aobj = aobj;
}
public void show() {
	System.out.println("Hello-show()");
	System.out.println(aobj);
	System.out.println(bobj);
}
}
```

6. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

	@Bean
	public A createA() {
	//Task1 : Create and Initialize A object
	A ao=new A();
	ao.setA(101);
	ao.setMsg("I am A");
	return ao;
	}

	@Bean("BO")
	public B createB() {
		//Task2 : Create and Initialize B object
		B bo=new B(102,"I am B");
		return bo;
	}

	@Bean("myhello")
	public Hello createHello(A ao,B bo) {
		//Task3 : Create and Initialize Hello object
		Hello h=new Hello(bo);
		h.setAobj(ao);
		return h;
	}
}
```

Bean Scopes

- Bean Instance created by Spring Container can be in one of the following Scopes(Updated from Spring5).
  1) singleton
  2) prototype
  3) request
  4) session
  5) application
  6) websocket

Scope	Description
singleton	◆ When bean scope is singleton then only one instance will be created for that bean and the same instance will be returned when you call getBean() method. ◆ singleton is the default scope in the ApplicationContext container. ◆ When scope is single-ton then default loading type is aggressive loading.
prototype	◆ When bean scope is prototype then every time a new instance will be created for that bean when you call getBean() method. ◆ When scope is prototype then default loading type is lazy loading.
request	◆ Scopes a single bean definition to the lifecycle of a single HTTP request. ◆ Single Bean instance will be created per HTTP Request ◆ Only valid in the context of a web-aware Spring ApplicationContext.
session	◆ Scopes a single bean definition to the lifecycle of an HTTP Session. ◆ Single Bean instance will be created per HTTP Session ◆ Only valid in the context of a web-aware Spring ApplicationContext.
application	◆ Scopes a single bean definition to the lifecycle of a ServletContext. ◆ Single Bean instance will be created per ServletContext. ◆ Only valid in the context of a web-aware Spring ApplicationContext.
websocket	◆ Scopes a single bean definition to the lifecycle of a WebSocket. ◆ Single Bean instance will be created per WebSocket. ◆ Only valid in the context of a web-aware Spring ApplicationContext.

- **Usage:**

```
@Scope(value="singleton")
@Scope(value=" prototype ")
@Scope("singleton")
@Scope("prototype")
```

Bean Scope Example with Java Configuration

Lab2: Files required

1. **Lab2.java**	2. **Hello.java**
3. **Hai.java**	4. **JLCAppConfig.java**

1. Lab2.java

```
package com.coursecube.spring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab2 {
public static void main(String[] args) {

ApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig.class);
System.out.println("---------Now Spring Container is Ready-----");

Hello hello1=(Hello)ctx.getBean("hello");
Hello hello2=(Hello)ctx.getBean("hello");
System.out.println(hello1==hello2);

Hai hai1=(Hai)ctx.getBean("hai");
Hai hai2=(Hai)ctx.getBean("hai");
System.out.println(hai1==hai2);

Hello hello=(Hello)ctx.getBean("hello");
hello.showHello();

Hai hai=(Hai)ctx.getBean("hai");
hai.showHai();
}
}
```

2. Hello.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {

static {
```

```
System.out.println("Hello - S.B");
}
public Hello() {
System.out.println("Hello - D.C");
}
public void showHello() {
        System.out.println("Hello-showHello()");
}
}
```

3. Hai.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hai {

static {
System.out.println("Hai - S.B");
}
public Hai() {
System.out.println("Hai - D.C");
}
public void showHai() {
        System.out.println("Hai-showHai()");
}
}
```

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Scope;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

        @Bean("hello")
        @Scope("singleton")
        public Hello createHello() {
                System.out.println("-------createHello() ------ called");
                return new Hello();
        }
        @Bean("hai")
        @Scope("prototype")
        public Hai createHai() {
                System.out.println("-------createHai() ------ called");
                return new Hai();
        }

}
```

Bean Loading Types

- Bean configured in the Spring Configuration Class can be loaded in two ways.
    1) Aggressive loading or Eager loading
    2) Lazy loading.

- **Usage:**

```
@Lazy(value="true")
@Lazy(value="true")
@Lazy("true")
@Lazy("false")
```

1) Aggressive loading or Eager loading

- In the case of aggressive loading, all the Beans will be loaded, instantiated and initialized by the container at the container start-up.

Ex:

```
@Bean
@Lazy(false)
public Hello hello() {
        return new Hello();
}
```

2) Lazy loading.

- In the case of lazy loading, all the Beans will be loaded, instantiated and initialized when you or container try to use them by calling getBean() method.

Ex:

```
@Bean
@Lazy(true)
public Hello hello() {
        return new Hello();
}
```

Bean Loading Types Example with Java Configuration:

Lab3: Files required

1. Lab3.java	**New One**
2. Hello.java	**Same as Lab2**
3. Hai.java	**Same as Lab2**
4. JLCAppConfig.java	**New One**

1. Lab3.java

```
package com.coursecube.spring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab3 {
public static void main(String[] args) {

ApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig.class);
System.out.println("---------Now Spring Container is Ready-----");

Hello hello=(Hello)ctx.getBean("hello");
hello.showHello();

Hai hai=(Hai)ctx.getBean("hai");
hai.showHai();

}
}
```

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Lazy;
import org.springframework.context.annotation.Scope;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
```

```
@Configuration
public class JLCAppConfig {

        @Bean("hello")
        @Scope("singleton")
        @Lazy(true)
        public Hello createHello() {
                System.out.println("-------createHello() ------ called");
                return new Hello();
        }

        @Bean("hai")
        @Scope("prototype")
        @Lazy(true)
        public Hai createHai() {
                System.out.println("-------createHai() ------ called");
                return new Hai();
        }
}
```

Using Existing Singletons

Lab4: Files required

5. **Lab4.java**	6. **Hello.java**
7. **JLCAppConfig.java**	

1. Lab4.java

```
package com.coursecube.spring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab4 {
public static void main(String[] args) {

ApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig.class);
System.out.println("---------Now Spring Container is Ready-----");

Hello hello=(Hello)ctx.getBean("myhello");
hello.show();
Runtime rt =(Runtime)ctx.getBean("myruntime");
System.out.println(rt.availableProcessors());
}   }
```

2. Hello.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {
static Hello hello;
static {
        hello=new Hello();
}
private Hello() {
}
public static Hello getHello() {
        return hello;
}
public void show() {
        System.out.println("Hello-show()");
}  }
```

3. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {
	@Bean("myhello")
	public Hello createHello() {
		System.out.println("-------createHello() ------ called");
		Hello hello=Hello.getHello();
		return hello;
	}
	@Bean("myruntime")
	public Runtime createRT() {
		System.out.println("-------createRT() ------ called");
		Runtime rt=Runtime.getRuntime();
		return rt;
	}
}
```

Writing Multiple Configuration classes

- JLCAppConfig - Spring Configuration class which contains all the Bean Definitions.
- You can write Multiple Configuration classes in One Spring Application.
- There are 3 ways to Handle Multiple Configuration classes
  1) Using the constructor of AnnotationConfigApplicationContext
  2) Using @Import annotaion
  3) Using @ImportResource annotation

1) Using the constructor of AnnotationConfigApplicationContext (Refer Lab5)

- Specify the Multiple Configuration classes as Parameters for Constructor of AnnotationConfigApplicationContext

```
ApplicationContext ctx= new AnnotationConfigApplicationContext (
JLCAppConfig1.class, JLCAppConfig2.class, JLCAppConfig3.class);
```

2) Using @Import annotation (Refer Lab6)

- Specify the One Configuration class as Parameters for Constructor of AnnotationConfigApplicationContext

```
ApplicationContext ctx=new AnnotationConfigApplicationContext (JLCAppConfig1.class);
```

- Import Remaining Configuration classes using @Import Annotation

```
@Import({JLCAppConfig1.class,JLCAppConfig3.class})
public class JLCAppConfig2{

}...
```

3) Using @ImportResource annotation (Refer Lab7)

- This is Required only when some of your beans are configured in XML file.
- Consider the Bean Configurations as follows.

```
jlc1.xml   - Configure A bean here
jlc2.xml   - Configure B bean here
JLCAppConfig -- Hello,Hai
```

Note : place jlc1.xml and jlc2.xml in src folder.

- Specify the One Configuration class as Parameters for Constructor of AnnotationConfigApplicationContext

```
ApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig.class);
```

- Import Remaining Configuration XML files using @ ImportResource Annotation

```
@ImportResource({"jlc1.xml","jlc2.xml"})
public class JLCAppConfig {

}
```

Lab5: Files required

1. Lab5.java	**2. A.java**
3. B.java	**4. Hai.java**
5. Hello.java	**6. JLCAppConfig1.java**
7. JLCAppConfig2.java	**8. JLCAppConfig3.java**

1. Lab5.java

```
package com.coursecube.spring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
```

```
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab5 {
public static void main(String[] args) {
ApplicationContext ctx=new
AnnotationConfigApplicationContext(JLCAppConfig1.class,JLCAppConfig2.class,JLCAppConfig3.class
);
System.out.println("---------Now Spring Container is Ready-----");

Hello hello=(Hello)ctx.getBean("myhello");
hello.show();
Hai hai=(Hai) ctx.getBean("myhai");
hai.show();
}
}
```

2. A.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class A {
 int a;  //S.I
 String msg; //S.I
 static {
             System.out.println("A - S.B");
 }
public A() {
            System.out.println("A - D.C");
}
public void setA(int a) {
            System.out.println("A - setA()");
            this.a = a;
}
public void setMsg(String msg) {
            System.out.println("A - setMsg()");
            this.msg = msg;
}
 public String  toString() {
            return ""+a +"\t"+msg;
}
}
```

3. B.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class B {
int b;  //C.I
String str; //C.I
static {
            System.out.println("B - S.B");
}
public B(int b, String str) {
           System.out.println("B - 2 arg");
           this.b = b;
           this.str = str;
}
public String  toString() {
           return ""+b+"\t"+str;
}
}
```

4. Hai.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hai {
           A aobj; //S.I
           B bobj; //S.I
            static {
                       System.out.println("Hai - S.B");
            }
            public Hai() {
                       System.out.println("Hai() - D.C");
            }
           public void setAobj(A aobj) {
           System.out.println("Hai- setAobj() ");
                      this.aobj = aobj;
           }
           public void setBobj(B bobj) {
                      System.out.println("Hai- setBobj() ");
                      this.bobj = bobj;
           }
```

```
        public void show() {
            System.out.println("Hai- show()");
            System.out.println(aobj);
            System.out.println(bobj);
    }
}
```

5. Hello.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {
            A aobj; //C.I
            B bobj; //C.I
             static {
                        System.out.println("Hello - S.B");
             }
             public Hello() {
                        System.out.println("Hello() - D.C");
             }
           public Hello(A aobj,B bobj) {
            System.out.println("Hello(A,B) - 2arg");
                       this.aobj=aobj;
                       this.bobj = bobj;
            }
           public void show() {
            System.out.println("Hello-show()");
            System.out.println(aobj);
            System.out.println(bobj);
           }
}
```

6. JLCAppConfig1.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig1 {
        @Bean("ao")
        public A createA() {
        A ao=new A();
        ao.setA(101);
        ao.setMsg("I am A");
        return ao;
        }
        @Bean("bo")
        public B createB() {
                B bo=new B(102,"I am B");
                return bo;
        }
}
```

7. JLCAppConfig2.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig2 {

        @Bean("myhello")
        public Hello createHello(A ao,B bo) {
                Hello h=new Hello(ao,bo);
                return h;
        }
}
```

8. JLCAppConfig3.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig3 {
        @Bean("myhai")
        public Hai createHai(A ao,B bo) {
                Hai hai=new Hai();
                hai.setAobj(ao);
                hai.setBobj(bo);
                return hai;
        }
}
```

Lab6: Files required

1. Lab6.java	**New One**
2. A.java	**Same as Lab5**
3. B.java	**Same as Lab5**
4. Hai.java	**Same as Lab5**
5. Hello.java	**Same as Lab5**
6. JLCAppConfig1.java	**New One**
7. JLCAppConfig2.java	**Same as Lab5**
8. JLCAppConfig3.java	**Same as Lab5**

1. Lab6.java

```
package com.coursecube.spring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab6 {
public static void main(String[] args) {

ApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig1.class);
System.out.println("---------Now Spring Container is Ready-----");

Hello hello=(Hello)ctx.getBean("myhello");
hello.show();

Hai hai=(Hai) ctx.getBean("myhai");
hai.show();
}
}
```

6. JLCAppConfig1.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
@Import({JLCAppConfig2.class,JLCAppConfig3.class})
public class JLCAppConfig1 {

        @Bean("ao")
        public A createA() {
        A ao=new A();
        ao.setA(101);
        ao.setMsg("I am A");
        return ao;
        }
        @Bean("bo")
        public B createB() {
                B bo=new B(102,"I am B");
                return bo;
        }
}
```

Lab7: Files required

1. Lab7.java	**New One**
2. A.java	**Same as Lab5**
3. B.java	**Same as Lab5**
4. Hai.java	**Same as Lab5**
5. Hello.java	**Same as Lab5**
6. JLCAppConfig.java	**New One**
7. jlc1.xml	**New One**
8. jlc2.xml	**New One**

1. Lab7.java

```
package com.coursecube.spring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab7 {
public static void main(String[] args) {

ApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig.class);
System.out.println("---------Now Spring Container is Ready-----");

Hello hello=(Hello)ctx.getBean("myhello");
hello.show();

Hai hai=(Hai) ctx.getBean("myhai");
hai.show();
}
}
```

6. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.ImportResource;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
@ImportResource({"jlc1.xml","jlc2.xml"})
public class JLCAppConfig {

	@Bean("myhello")
	public Hello createHello(A ao,B bo) {
		Hello h=new Hello(ao,bo);
		return h;
	}

	@Bean("myhai")
	public Hai createHai(A ao,B bo) {
		Hai hai=new Hai();
		hai.setAobj(ao);
		hai.setBobj(bo);
		return hai;
	}

}
```

7. jlc1.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="aobj" class="com.coursecube.spring.A">
<property name="a" value="99"/>
<property name="msg" value="I am A"/>
</bean>

</beans>
```

8. jlc2.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="bobj" class="com.coursecube.spring.B">
<constructor-arg value="88"/>
<constructor-arg value="I am B"/>
</bean>

</beans>
```

Injecting Various Types of Bean Properties:

You can Inject the following types of Bean Properties.

1. Simple Types(Primitives,Wrappers,Strings,Dates).
2. List type
3. Set type
4. Map type
5. java.util.Properties type
6. other Beans
7. Collection of Other Beans

All of these types of Bean Properties can be Injected with Setter Injection or Constrcutor Injection.

Lab8: Injecting Various Types of Properties
Lab9: Injecting Two List Types

Lab8: Files required

1. **Lab8.java**	2. **Address.java**
3. **Account.java**	4. **Customer.java**
5. **JLCAppConfig.java**	

1. Lab8.java

```
package com.coursecube.spring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab8 {
public static void main(String[] args) {

ApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig.class);
System.out.println("---------Now Spring Container is Ready-----");

Customer cust=(Customer) ctx.getBean("mycust");
System.out.println(cust);
System.out.println(cust.getEmails());
System.out.println(cust.getPhones());
System.out.println(cust.getRefs());
System.out.println(cust.getAddress());

for(Account acc:cust.getAccounts()) {
		System.out.println(acc);
}
}
}
```

2. Address.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Address {
 private String street;
 private String city;
 private String state;
 public Address() {}
 public void setStreet(String street) {
          this.street = street;
}
public void setCity(String city) {
          this.city = city;
}
public void setState(String state) {
          this.state = state;
}
@Override
public String toString() {
          return  street + ", " + city + ", " + state ;
}
}
```

3. Account.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Account {
private int accno;
private String atype;
private double bal;
public Account() {}
public Account(int accno, String atype, double bal) {
          super();
          this.accno = accno;
          this.atype = atype;
          this.bal = bal;
}
@Override
public String toString() {
          return accno + ", " + atype + ", " + bal ;
}
}
```

4. Customer.java

```
package com.coursecube.spring;

import java.util.*;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
```

```
public class Customer {
        private int cid;//1
        private String cname;//1
        private String email;//1
        private long phone;//1
        private List<String> emails;//2
        private Set<Integer> phones;//3
        private Map<String,Integer> refs;//4
        private Properties myprops;//5
        private Address address;//6
        private List<Account> accounts;//7
        public Customer() {}
        public Customer(int cid, String cname, String email, long phone) {
                super();
                this.cid = cid;
                this.cname = cname;
                this.email = email;
                this.phone = phone;
        }
       //Setters and Getters
        @Override
        public String toString() {
                return cid + ", " + cname + ", " + email + ", " + phone;
        }
}
```

5. JLCAppConfig.java

```
package com.coursecube.spring;

import java.util.*;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {
        @Bean(name="myemails")
        public List<String> getEmails(){
                System.out.println("JLCAppConfig -getEmails() ");
                List<String> ems=new ArrayList<>();
                ems.add("sri@jlc");
                ems.add("vas@jlc");
                ems.add("sd@jlc");
                return ems;
        }
        @Bean(name="myphones")
        public Set<Integer> getPhones(){
                System.out.println("JLCAppConfig -getPhones() ");
                Set<Integer> phs=new TreeSet<>();
                phs.add(111);
                phs.add(222);
                phs.add(333);
                return phs;
        }

        @Bean(name="myrefs")
        public Map<String,Integer> getRefs(){
                System.out.println("JLCAppConfig -getRefs() ");
                Map<String,Integer> refs=new TreeMap<>();
                refs.put("A",11);
                refs.put("B",22);
                refs.put("C",33);
                refs.put("D",44);
                return refs;
        }

        @Bean(name="myprops")
        public Properties getProps(){
                System.out.println("JLCAppConfig -getProps() ");
                Properties props=new Properties();
                props.put("A",11);
                props.put("B",22);
                props.put("C",33);
                props.put("D",44);
                return props;
        }

        @Bean(name="myadd")
        public Address getAdd() {
                System.out.println("JLCAppConfig -getAdd() ");
                Address add=new Address();
                add.setStreet("BTM Layout");
                add.setCity("Bangalore");
                add.setState("KA");
                return add;
        }

        @Bean(name="myaccs")
        public List<Account> getAccounts(){
                System.out.println("JLCAppConfig -getAccounts() ");

                List<Account> myaccs=new ArrayList<>();
                myaccs.add(new Account(101,"SA",15000));
                myaccs.add(new Account(102,"CA",25000));
                myaccs.add(new Account(103,"DA",35000));
                return myaccs;
        }
        @Bean(name="mycust")
        public Customer createCustomer(List<String> myemails,Set<Integer> myphones,
                Map<String,Integer> myrefs,Properties myprops,Address myadd,
                List<Account> myaccs) {

                System.out.println("JLCAppConfig -createCustomer() ");
                 Customer cust=new Customer(101,"Srinivas","Sri@jlc.com",12345);
                cust.setEmails(myemails);
                cust.setPhones(myphones);
                cust.setRefs(myrefs);
                cust.setMyprops(myprops);
                cust.setAddress(myadd);
                cust.setAccounts(myaccs);
                return cust;
        }
}
```

Lab9: Files required

1. Lab9.java	**2. Customer.java**
3. JLCAppConfig.java	

1. Lab9.java

```
package com.coursecube.spring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab9 {
public static void main(String[] args) {

ApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig.class);
System.out.println("---------Now Spring Container is Ready-----");

Customer cust=(Customer) ctx.getBean("mycust");
System.out.println(cust);
System.out.println(cust.getEmails());
System.out.println(cust.getPhones());
}
}
```

2. Customer.java

```
package com.coursecube.spring;

import java.util.*;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Customer {
	private int cid;
	private String cname;
	private String email;
	private long phone;
	private List<String> emails;
	private List<String> phones;
	public Customer() {}
	public Customer(int cid, String cname, String email, long phone, List<String> emails,
List<String> phones) {
		super();
		this.cid = cid;
		this.cname = cname;
		this.email = email;
		this.phone = phone;
		this.emails = emails;
		this.phones = phones;
	}

	//Setters and Getters
	@Override
	public String toString() {
		return cid + ", " + cname + ", " + email + ", " + phone;
	}
}
```

3. JLCAppConfig.java

```
package com.coursecube.spring;

import java.util.*;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {
	@Bean(name="myemails")
	public List<String> getEmails(){
		List<String> ems=new ArrayList<>();
		ems.add("sri@jlc");
		ems.add("vas@jlc");
		ems.add("sd@jlc");
		return ems;
	}
	@Bean(name="myphones")
	public List<String> getPhones(){
		List<String> phs=new ArrayList<>();
		phs.add("111");
		phs.add("222");
		phs.add("333");
		return phs;
	}
	@Bean(name="mycust")
	public Customer createCustomer(List<String> myemails,List<String> myphones) {
	Customer cust=new Customer(101,"Srinivas","Sri@jlc.com",12345,myemails,myphones);
	return cust;
	}
}
```

Wiring

- Wiring is the Process of Inejecting Bean Dependecies.

- Wiring Can be done in two ways.
  1) Explicit Wiring
  2) Implicit Wiring (or) AutoWiring

1)Explicit Wiring:

- In the case of Explicit Wiring, Bean Dependencies has to be specified by you explicitly.

Ex:

```
class Hello {
Hai hai;
....
}

@Bean("myhello")
public Hello createHello(Hai hai) {
	Hello h=new Hello();
	h.setHai(hai);
	return h;
}
```

- In the above Example, We are explicitly specifying the Hello dependecy called Hai

Case 1: What happens when 0 Matching Beans found.(Refer Lab10)

- Exception will be thrown
- NoSuchBeanDefinitionException: No qualifying bean of type 'com.coursecube.spring.Hai' available: expected at least 1 bean

Note: In the case of Collections Only , Container creates the Empty Object and passes that as paramter **.(Ref. Lab9)**

Case 2: What happens when exactly 1 Matching bean found.(Refer Lab11)

- Identified single bean will be injected with Setter method.

Case 3: What happens when two or more Matching beans found.

- If Any bean name is matching with local variable name then that will be injected with setter method**.(Refer Lab12)**

- If Any bean name is not matching with local variable name then Exception will be thrown.**(Ref.Lab13)**
- NoUniqueBeanDefinitionException: No qualifying bean of type ……..expected single matching bean but found 2: myhai1,myhai2

Lab10: Files required

1. Lab10.java	**2. Hai.java**
3. Hello.java	**4. JLCAppConfig.java**

1. Lab10.java

```
package com.coursecube.spring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab10 {
public static void main(String[] args) {

ApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig.class);
System.out.println("---------Now Spring Container is Ready-----");

Hello hello=(Hello)ctx.getBean("myhello");
hello.show();
}
}
```

2. Hai.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hai {
        static {
                System.out.println("Hai - S.B");
        }
        public Hai() {
                System.out.println("Hai() - D.C");
        }

      public String toString() {
       return "Hai Guys, I am Hai Bean";
      }
}
```

3. Hello.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {
       Hai hai;  //1. Dependency

        static {
                System.out.println("Hello - S.B");
        }
        public Hello() {
                System.out.println("Hello() - D.C");
        }

       public void setHai(Hai hai) {  //2. Setter Injection
               System.out.println("Hello - setHai()");
               this.hai = hai;
       }

      public void show() {
              System.out.println("Hello-show()");
              System.out.println(hai);
      }
}
```

4. JLCAppConfig.java

```
package com.coursecube.spring;

import java.util.*;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {
	@Bean("myhello")
	public Hello createHello(Hai hai) { //1. Dependency
		Hello h=new Hello();
		h.setHai(hai);	//2. Setter Injection
		return h;
	}
}
```

Lab11: Files required

1. Lab11.java	**Same as Lab10**
2. Hai.java	**Same as Lab10**
3. Hello.java	**Same as Lab10**
4. JLCAppConfig.java	**Updated Here**

4. JLCAppConfig.java

```
package com.coursecube.spring;

import java.util.*;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

	@Bean("myhai")
	public Hai createHai1() {
		Hai hai=new Hai();
		return hai;
	}
```

```
        @Bean("myhello")
        public Hello createHello(Hai hai) { //1. Dependency
                Hello h=new Hello();
                h.setHai(hai);    //2. Setter Injection
                return h;
        }
}
```

Lab12: Files required

1. **Lab12.java**	**Same as Lab10**
2. **Hai.java**	**Same as Lab10**
3. **Hello.java**	**Same as Lab10**
4. **JLCAppConfig.java**	**Updated Here**

4. JLCAppConfig.java

```
package com.coursecube.spring;

import java.util.*;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

        @Bean("myhai1")
        public Hai createHai1() {
                Hai hai=new Hai("I am First Bean");
                return hai;
        }

        @Bean("myhai2")
        public Hai createHai2() {
                Hai hai=new Hai("I am Second Bean");
                return hai;
        }

        @Bean("myhello")
        public Hello createHello(Hai myhai2) { //1. Dependency
                Hello h=new Hello();
                h.setHai(hai);    //2. Setter Injection
                return h;
        }
}
```

Lab13: Files required

1. **Lab13.java**	**Same as Lab10**
2. **Hai.java**	**Same as Lab10**
3. **Hello.java**	**Same as Lab10**
4. **JLCAppConfig.java**	**Updated Here**

4. JLCAppConfig.java

```
package com.coursecube.spring;

import java.util.*;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

	@Bean("myhai1")
	public Hai createHai1() {
		Hai hai=new Hai("I am First Bean");
		return hai;
	}

	@Bean("myhai2")
	public Hai createHai2() {
		Hai hai=new Hai("I am Second Bean");
		return hai;
	}

	@Bean("myhello")
	public Hello createHello(Hai hai) { //1. Dependency
		Hello h=new Hello();
		h.setHai(hai);    //2. Setter Injection
		return h;
	}
}
```

AutoWiring :

- In the case of AutoWiring, No need to specify the Bean Dependencies explicitly.
- Spring Container is responsbile for
    - ✓ Detecting Bean Dependencies
    - ✓ Injecting Bean Dependencies

- If Container is Detecting and Injecting Bean Dependencies automatically then it is called as AutoWiring.
- There are two ways to perform AutoWiring with **autowire attribute of @Bean Annotation**
    1) By Name Autowire Process
    2) By Type Autowire Process

- Autowire enum has two constants called **BY_NAME** and **BY_TYPE**

```
@Bean(name="myhello",autowire = Autowire.BY_NAME)
@Bean(name="myhello",autowire = Autowire.BY_BYTE)
```

1)By Name Autowire Process

- Container detects the Bean by Name.
- Container checks whether any bean is found whose name is matching property name.

Case 1: What happens when 0 beans found.(Refer Lab14)

- *Bean dependency remains Un-Injected.

Case 2: What happens when exactly 1 bean found.(Refer Lab15)

- Identified single bean will be injected with Setter method.

2)By Type Autowire Process

- Container detects the Bean by Data Type.
- Container checks whether any bean is found whose data Type is matching property data type.

Case 1: What happens when 0 beans found. (Ref.Lab16)

*Bean dependency remains Un-Injected.

Case 2: What happens when exactly 1 bean found.(Refer Lab17)
Identified single bean will be Injected with setter method.

Case 3: What happens when two or more beans found.(Refer Lab 18)

- Exception will be thrown
- Caused by: org.springframework.beans.factory.NoUniqueBeanDefinitionException: No qualifying bean of type 'com.coursecube.spring.Hai' available: expected single matching bean but found 3: myhai1,myhai2,hai

Lab14: Files required

1. Lab14.java	Same as Lab10
2. Hai.java	Same as Lab10
3. Hello.java	Same as Lab10
4. JLCAppConfig.java	Updated Here

4. JLCAppConfig.java

```
package com.coursecube.spring;

import java.util.*;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

	@Bean("myhai")
	public Hai createHai1() {
		Hai hai=new Hai("I am Hai Bean");
		return hai;
	}

	@Bean(name="myhello",autowire = Autowire.BY_NAME)
	public Hello createHello() {
		Hello h=new Hello();
		return h;
	}
}
```

Lab15: Files required

1. Lab15.java	Same as Lab10
2. Hai.java	Same as Lab10
3. Hello.java	Same as Lab10
4. JLCAppConfig.java	Updated Here

4. JLCAppConfig.java

```
package com.coursecube.spring;

import java.util.*;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
```

```
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

	@Bean("hai")
	public Hai createHai1() {
		Hai hai=new Hai("I am Hai Bean");
		return hai;
	}

	@Bean(name="myhello",autowire = Autowire.BY_NAME)
	public Hello createHello() {
		Hello h=new Hello();
		return h;
	}
}
```

Lab16: Files required

1. Lab16.java	**Same as Lab10**
2. Hai.java	**Same as Lab10**
3. Hello.java	**Same as Lab10**
4. JLCAppConfig.java	**Updated Here**

4. JLCAppConfig.java

```
package com.coursecube.spring;

import java.util.*;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

	@Bean(name="myhello",autowire = Autowire.BY_TYPE)
	public Hello createHello() {
		Hello h=new Hello();
		return h;
	}
}
```

Lab17: Files required

1. **Lab17.java**	**Same as Lab10**
2. **Hai.java**	**Same as Lab10**
3. **Hello.java**	**Same as Lab10**
4. **JLCAppConfig.java**	**Updated Here**

4. JLCAppConfig.java

```
package com.coursecube.spring;

import java.util.*;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

         @Bean("myhai")
        public Hai createHai1() {
                Hai hai=new Hai("I am Hai Bean");
                return hai;
        }
        @Bean(name="myhello",autowire = Autowire.BY_TYPE)
        public Hello createHello() {
                Hello h=new Hello();
                return h;
        }
}
```

Lab18: Files required

1. **Lab18.java**	**Same as Lab10**
2. **Hai.java**	**Same as Lab10**
3. **Hello.java**	**Same as Lab10**
4. **JLCAppConfig.java**	**Updated Here**

4. JLCAppConfig.java

```
package com.coursecube.spring;

import java.util.*;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

	@Bean("myhai1")
	public Hai createHai1() {
		Hai hai=new Hai("I am First  Hai Bean");
		return hai;
	}

	@Bean("myhai2")
	public Hai createHai2() {
		Hai hai=new Hai("I am Second Hai Bean");
		return hai;
	}

	@Bean("hai")
	public Hai createHai3() {
		Hai hai=new Hai("I am Third Hai Bean");
		return hai;
	}

	@Bean(name="myhello",autowire = Autowire.BY_TYPE)
	public Hello createHello() {
		Hello h=new Hello();
		return h;
	}
}
```

Injecting Sub Types into Super Types.

Lab19: Files required

1. Lab19.java	**2. A.java**
3. B.java	**4. CustomerDAO.java**
5. CustomerDAOImpl.java	**6. Hello.java**
7. JLCAppConfig.java	

1. Lab19.java

```
package com.coursecube.spring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab19 {
public static void main(String[] args) {
ApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig.class);
System.out.println("---------Now Spring Container is Ready-----");

Hello hello=(Hello)ctx.getBean("myhello");
hello.show();
}
}
```

2. A.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class A  {
public String  toString() {
          return "I am Bean A";
}
}
```

3. B.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class B  extends A{
          public String  toString() {
                    return "I am Bean B";
          }
}
```

4. CustomerDAO.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public interface CustomerDAO {
public void addCustomer();
}
```

5. CustomerDAOImpl.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class CustomerDAOImpl implements CustomerDAO {
        @Override
        public void addCustomer() {
                System.out.println("----AddCustomer---");
        }
}
```

6. Hello.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {
        A aobj; //1
        CustomerDAO customerDAO; //2

        public void setAobj(A aobj) {
                this.aobj = aobj;
        }
        public void setCustomerDAO(CustomerDAO customerDAO) {
                this.customerDAO = customerDAO;
        }
        public void show() {
                System.out.println(aobj);
                customerDAO.addCustomer();
        }
}
```

7. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.beans.factory.annotation.Autowire;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {
	/*
	@Bean("ao")
	public A createA() {
		return new A();
	}
       */
	@Bean(name="bo")
	public B createB() {
		return new B();
	}
	@Bean(name="cdao")
	public CustomerDAO getCustDAO() {
		return new CustomerDAOImpl();
	}
	@Bean(name="myhello",autowire = Autowire.BY_TYPE)
	public Hello createHello() { //Hello Bean
		return new Hello();
	}
}
```

Cyclic Dependency Injection:

- Spring Supports Cyclic DI with Setter Injection.
- Spring does not support Cyclic DI with Constrcutor Injection.

Lab20: Files required

1. Lab20.java	**2. Hai.java**
3. Hello.java	**4. JLCAppConfig.java**

1. Lab20.java

```
package com.coursecube.spring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab20 {
public static void main(String[] args) {
ApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig.class);
System.out.println("---------Now Spring Container is Ready-----");
}
}
```

2. Hai.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hai {

        Hello hello;   //Dependency

        public Hai() {
                System.out.println("Hai-D.C");
        }
        public void setHello(Hello hello) {   //Setter Injection
                System.out.println("Hai - setHello()");
                this.hello = hello;
        }
}
```

3. Hello.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Hello {

        Hai hai;   //Dependency


        public Hello() {
                System.out.println("Hello-D.C");
        }
        public void setHai(Hai hai) {   //Setter Injection
                System.out.println("Hello-setHai()");
                this.hai = hai;
        }
}
```

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.beans.factory.annotation.Autowire;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

	@Bean(name="myhai",autowire = Autowire.BY_TYPE)
	public Hai createHai() {
		Hai hai=new Hai();
		return hai;
	}

	@Bean(name="myhello",autowire = Autowire.BY_TYPE)
	public Hello createHello() {
		Hello hello=new Hello();
		return hello;
	}
}
```