

**Spring Container Callbacks or Lifecycle methods**

* 1. **Initialization callbacks**
  2. **Disposable callbacks**
  3. **Knowing who you are and where you are.**
  4. **Extending Spring Container Functionality.**

1. **Initialization Callbacks:**



**You can initialize the Resources required for your bean by using wiring process.**

**Sometimes, you may get the requirement to initialize the Resources required for your bean explicitly or you may want to verify the resources injected by the Spring Container.**

**In this case, you have to write the Resources Initialization code or verification code inside the Initialization Callbacks.**

**Spring supports following ways for Initialization.**

1. **Write your own Initialization method and mark that with @PostConstruct annotation.**
2. **Implement InitalizingBean interface and override the following method. a. public void afterPropertiesSet()**
3. **Write your own Initialization method and specify the method name in @Bean annotation using initMethod attribute. (In Java Configuration class).**

**Ex:**

**class Hello implements InitializingBean{**

**..**

**@PostConstruct**

**public void init(){**

**// Initialization code**

**}**

**public void afterPropertiesSet(){**

**// Initialization code**

**}**

**public void myInit(){**

**// Initialization code**

**}**

**}**

**In Java Configuration Class**

**@Bean(name="myhello",initMethod = "myInit")**

**public Hello createHello() {**

**return new Hello();**

**}**



**www.coursecube.com** **76** **Spring-5 Study Guide**



**2) Disposable Callbacks:**



**You may get the requirement to clean up the Resources which are initialized at the time of creating the bean instance.**

**You write the Resources clean up code inside the Disposable Callbacks.**

**Spring supports following ways for cleanup the Resources.**

1. **Write your own Clean up method and mark that with @PreDestroy annotation.**
2. **Implement DisposableBean interface and override the following method. a. public void destroy()**
3. **Write your own Clean up method and specify the method name in @Bean annotation using destroyMethod attribute (In Java Configuration class).**

**Ex:**

**class Hello implements DisposableBean {**

**...**

**@PreDestroy**

**public void cleanup(){**

**// cleanup code**

**}**

**public void destroy(){**

**// cleanup code**

**}**

**public void myCleanup(){**

**// cleanup code**

**}**

**}**

**In Java Configuration Class**

**@Bean(name="myhello",destroyMethod = "myCleanup")**

**public Hello createHello() {**

**return new Hello();**

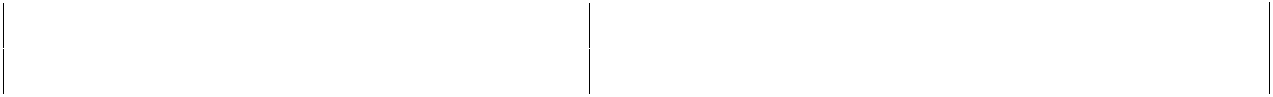
**}**



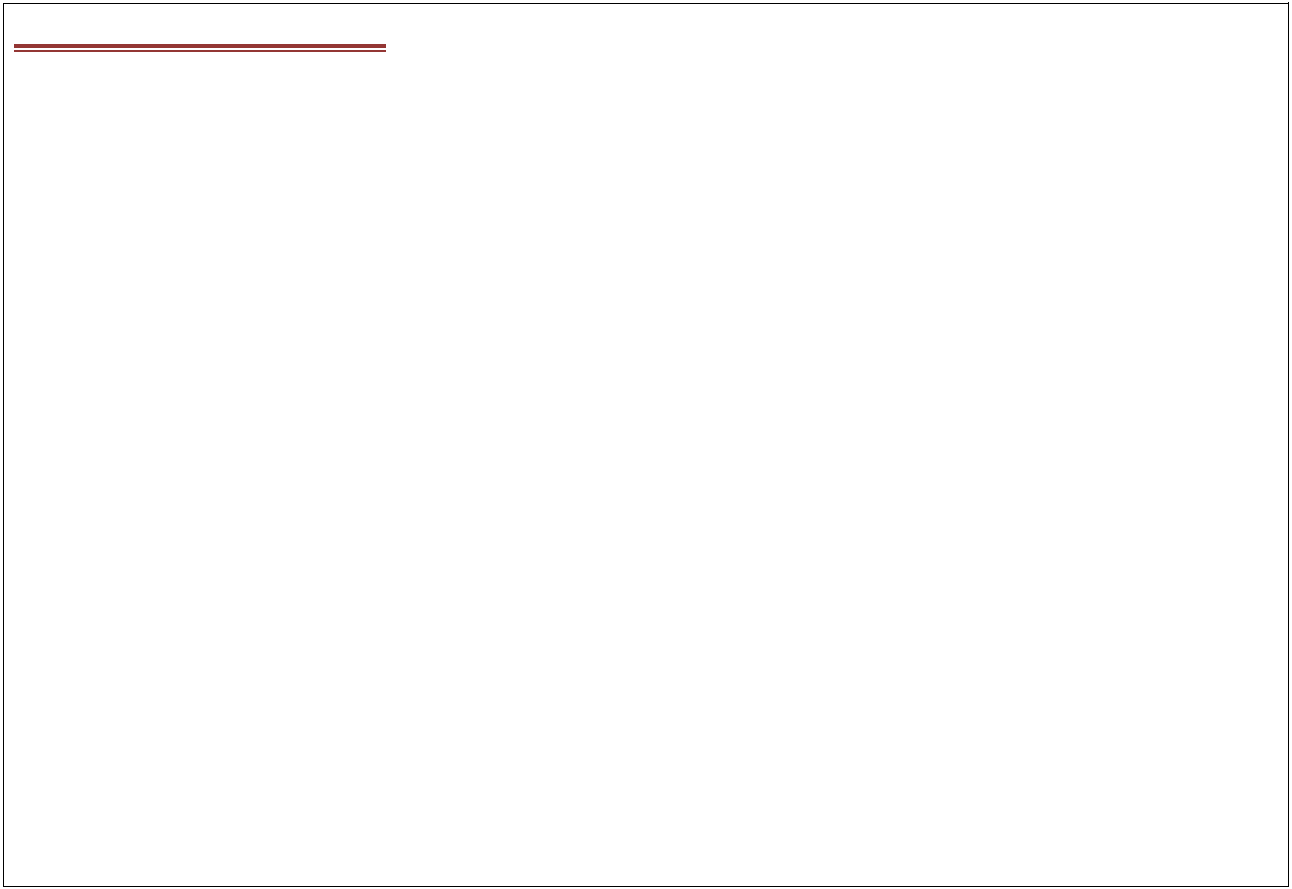
**www.coursecube.com** **77** **Spring-5 Study Guide**



**Lab39: Files required**



|  |  |  |  |
| --- | --- | --- | --- |
| **1.** | **Lab39.java** | **2.** | **Hai.java** |
|  |  |  |  |
| **3.** | **Hello.java** | **4.** | **JLCAppConfig.java** |
|  |  |  |  |



**1. Lab39.java**

**package com.coursecube.spring;**

**import org.springframework.context.annotation.AnnotationConfigApplicationContext; import org.springframework.context.support.AbstractApplicationContext; /\***

* **@Author : Srinivas Dande**
* **@Company : CourseCube**
* **@Website : www.coursecube.com**
* **\*/**

**public class Lab39 {**

**public static void main(String[] args) {**

**AbstractApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig.class);**

**System.out.println("---------Now Spring Container is Ready-----");**

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

**hello.show();**

**System.out.println("---------Now Going to Shut-Down Spring Container-----");**

**ctx.close();**

**ctx.registerShutdownHook();**

**}**

**}**



**2. Hai.java**

**package com.coursecube.spring;**

**/\***

* **@Author : Srinivas Dande**
* **@Company : CourseCube**
* **@Website : www.coursecube.com**
* **\*/**

**public class Hai {**

**String msg;**

**public void setMsg(String msg) {**

**this.msg = msg;**

**}**

**public String toString() {**

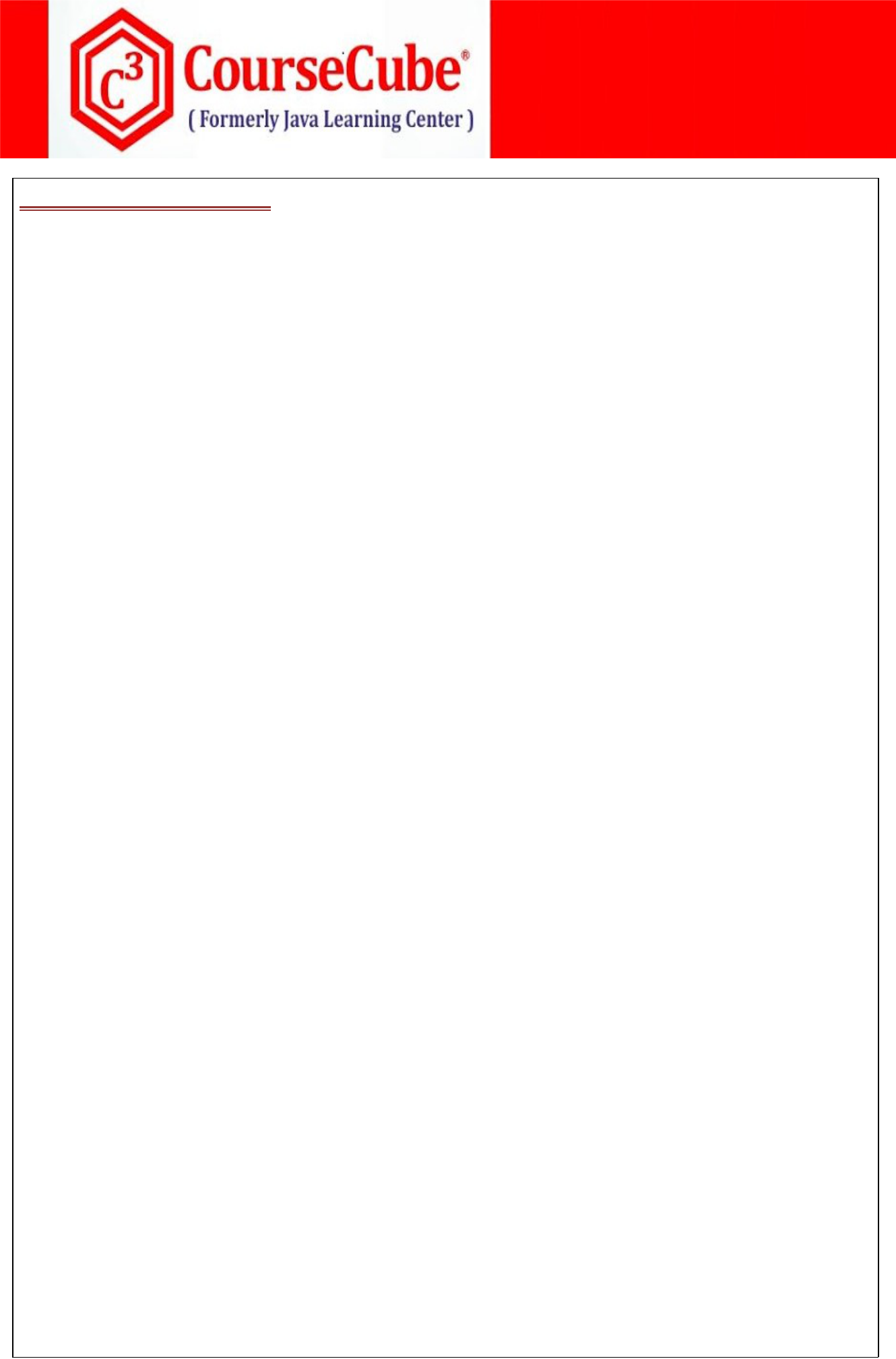
**return msg;**

**}**

**}**



**www.coursecube.com** **78** **Spring-5 Study Guide**



**3. Hello.java**

**package com.coursecube.spring;**

**import javax.annotation.PostConstruct;**

**import javax.annotation.PreDestroy;**

**import org.springframework.beans.factory.DisposableBean; import org.springframework.beans.factory.InitializingBean; import org.springframework.beans.factory.annotation.Autowired; /\***

* **@Author : Srinivas Dande**
* **@Company : CourseCube**
* **@Website : www.coursecube.com**
* **\*/**

**public class Hello implements InitializingBean,DisposableBean {**

**@Autowired**

**Hai hai;** **//3**

**static {**

**System.out.println("1.Hello-S.B"); //1**

**}**

**public Hello() {**

**System.out.println("2.Hello-D.C"); //2**

**}**

**@PostConstruct**

**public void init1() {**

**System.out.println("4.Hello-init1()");**

**//Initialization Code**

**}**

**@PostConstruct**

**public void init2() {**

**System.out.println("4.Hello-init2()");**

**//Initialization Code**

**}**

**public void afterPropertiesSet() throws Exception { System.out.println("5.Hello-afterPropertiesSet()"); //Initialization Code**

**}**

**public void myInit() {**

**System.out.println("6.Hello-myInit()");**

**//Initialization Code**

**}**

**@PreDestroy**

**public void cleanup1() {**

**System.out.println("A.Hello-cleanup1()");**

**//Cleanup -Code**

**}**



**www.coursecube.com** **79** **Spring-5 Study Guide**



**@PreDestroy**

**public void cleanup2() {**

**System.out.println("A.Hello-cleanup2()");**

**//Cleanup -Code**

**}**

**public void destroy() throws Exception {**

**System.out.println("B.Hello-destroy()");**

**//Cleanup -Code**

**}**

**public void myCleanup() { System.out.println("C.Hello-myCleanup()"); //Cleanup -Code**

**}**

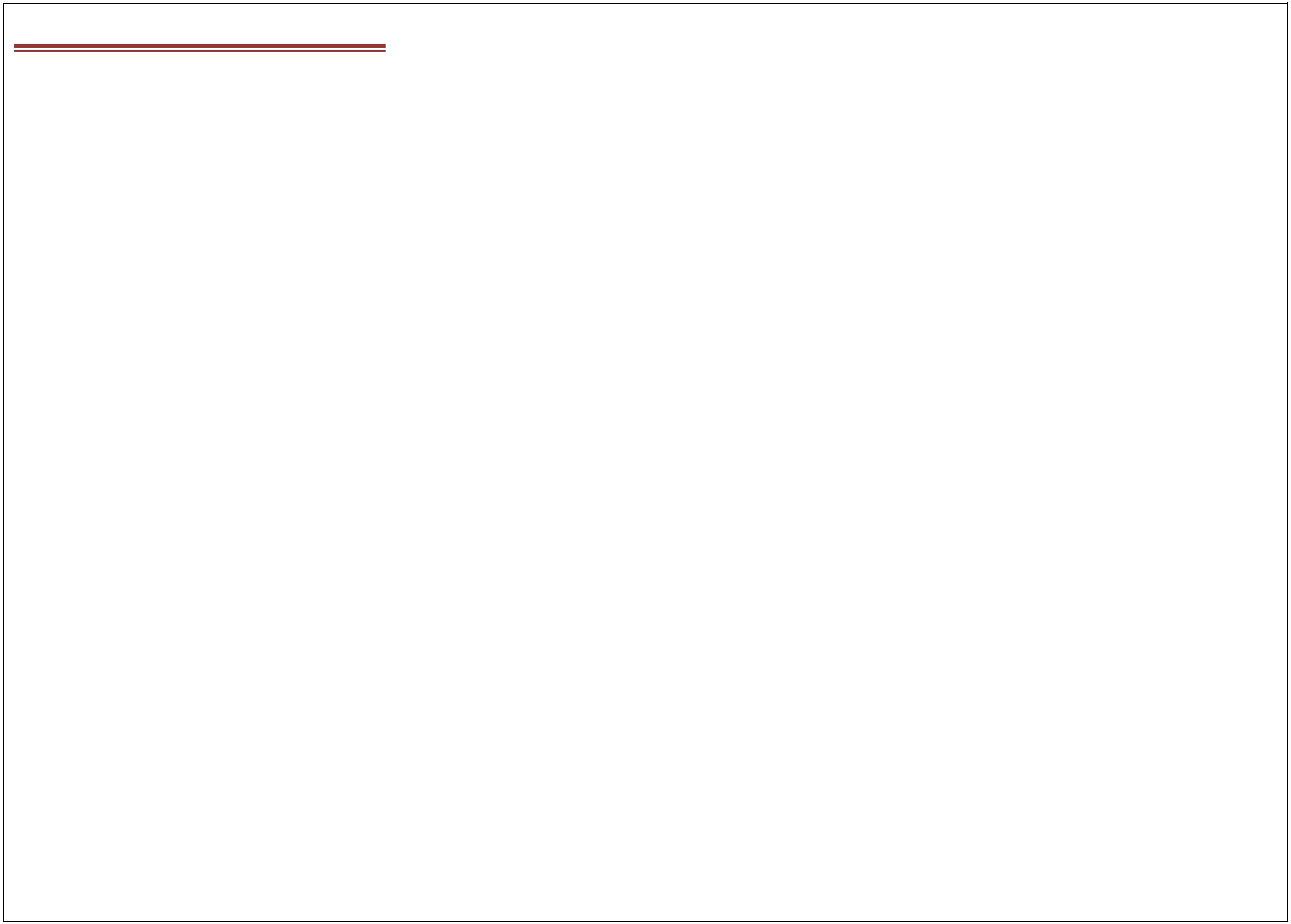
**public void show() {**

**System.out.println("Hello-show()");**

**System.out.println(hai);**

**}**

**}**



1. **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(name="myhai")**

**public Hai createHai() {**

**Hai hai=new Hai();**

**hai.setMsg("I am Hai Bean");**

**return hai;**

**}**

**@Bean(name="myhello",initMethod = "myInit",destroyMethod = "myCleanup")**

**public Hello createHello() {**

**return new Hello();**

**}**

**}**



**www.coursecube.com** **80** **Spring-5 Study Guide**



1. **Knowing who you are and where you are**

**Sometimes Bean wants to know what its name is and where it is running. For this, you can use the following 3 Aware interfaces.**

* + 1. **BeanNameAware**
    2. **BeanFatoryAware**
    3. **ApplicationContextAware**

**When Bean wants to know its Name then Bean class has to implement BeanNameAware interface and has to override the following method.**



* + **public void setBeanName(String bname)**

**When Bean wants to get the BeanFactory reference then Bean class has to implement BeanFatoryAware interface and has to override the following method.**

* + **public void setBeanFactory(BeanFactory factory)**

**When Bean wants to get the ApplicationContext reference then Bean class has to implement ApplicationContextAware interface and has to override the following method.**

* + **public void setApplicationContext(ApplicationContext ctx)**

**Note:**

**BeanFactory and ApplicationContext objects can be injected into the bean in two ways.**

* + 1. **Using Aware interfaces.**
    2. **Using @Autowired**

1. **Extending Container Functionality**

**You can extend the Spring Container Functionality using BeanPostProcessor interface.**



**Steps to Write Custom BeanPostProcessor**

**Write your own Custom class by implementing BeanPostProcessor interface.**

**Override the following 2 methods.**

* + **public Object postProcessBeforeInitialization(Object bean, String bname)**
  + **public Object postProcessAfterInitialization(Object bean, String bname)**

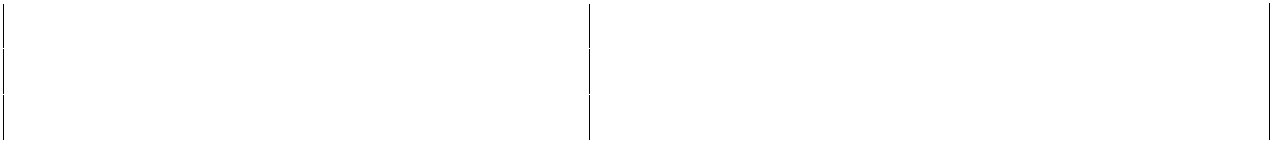
**Register your BeanPostProcessor with the Spring Container.**



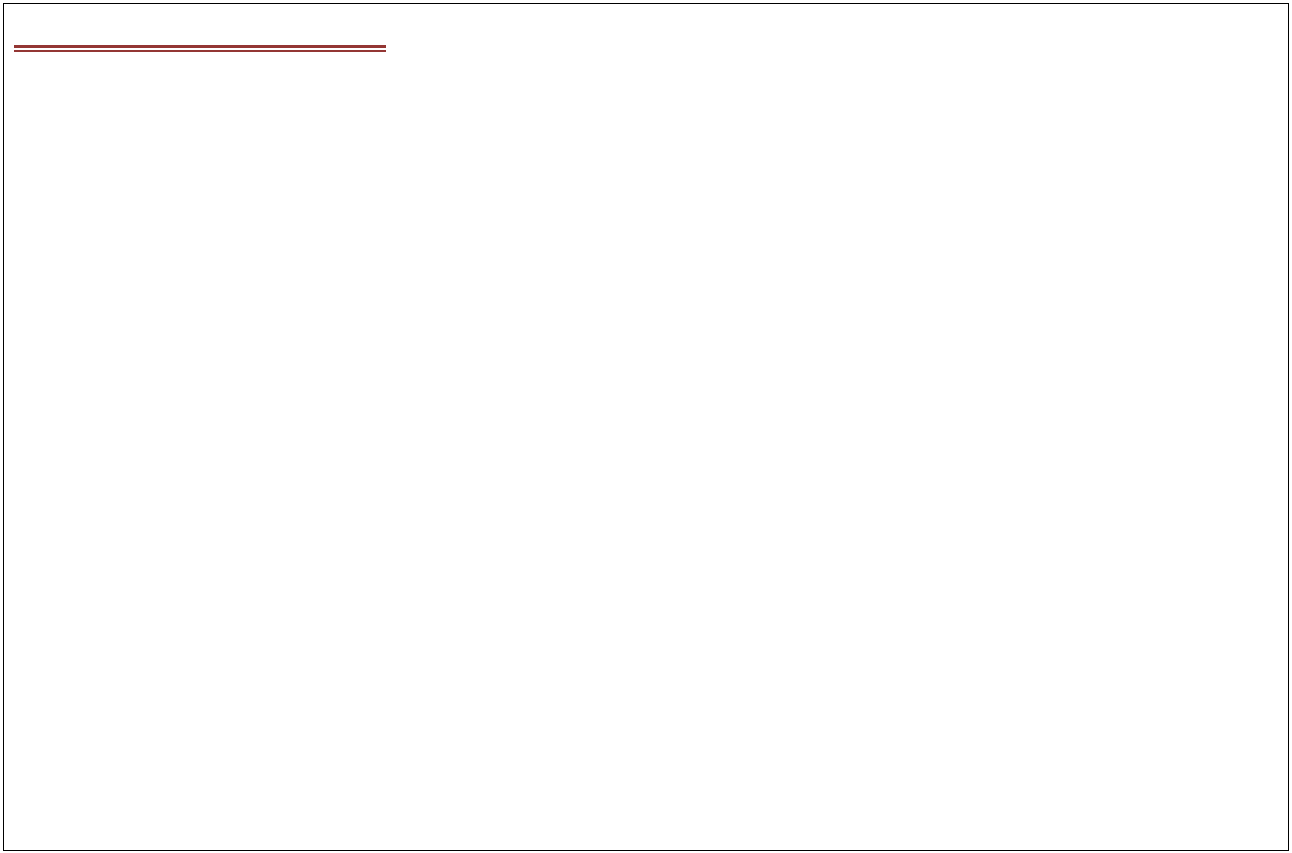
**www.coursecube.com** **81** **Spring-5 Study Guide**



**Lab40: Files required**



|  |  |  |  |
| --- | --- | --- | --- |
| **1.** | **Lab40.java** | **2.** | **Hai.java** |
|  |  |  |  |
| **3.** | **Hello.java** | **4.** | **MyPostProcessor.java** |
|  |  |  |  |
| **5.** | **JLCAppConfig.java** |  |  |
|  |  |  |  |



**1. Lab40.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 Lab40 {**

**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");**

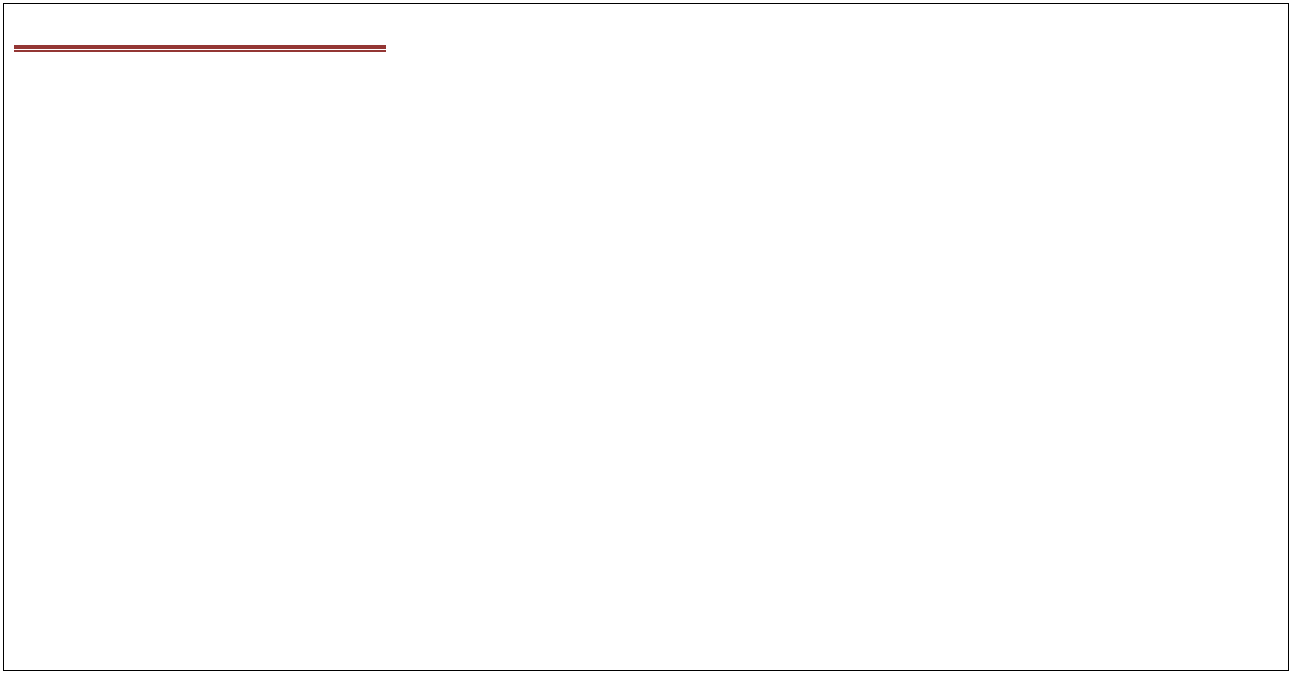
**hello.show();**

**Hai hai=(Hai) ctx.getBean("myhai");**

**hai.show();**

**}**

**}**



**2. Hai.java**

**package com.coursecube.spring;**

**import org.springframework.beans.factory.BeanNameAware; /\***

* **@Author : Srinivas Dande**
* **@Company : CourseCube**
* **@Website : www.coursecube.com**
* **\*/**

**public class Hai implements BeanNameAware{**

**String bname; //1 S.I**

**public void setBeanName(String bname) {**

**System.out.println("Hai-setBeanName()");**

**this.bname=bname;**

**}**



**www.coursecube.com** **82** **Spring-5 Study Guide**



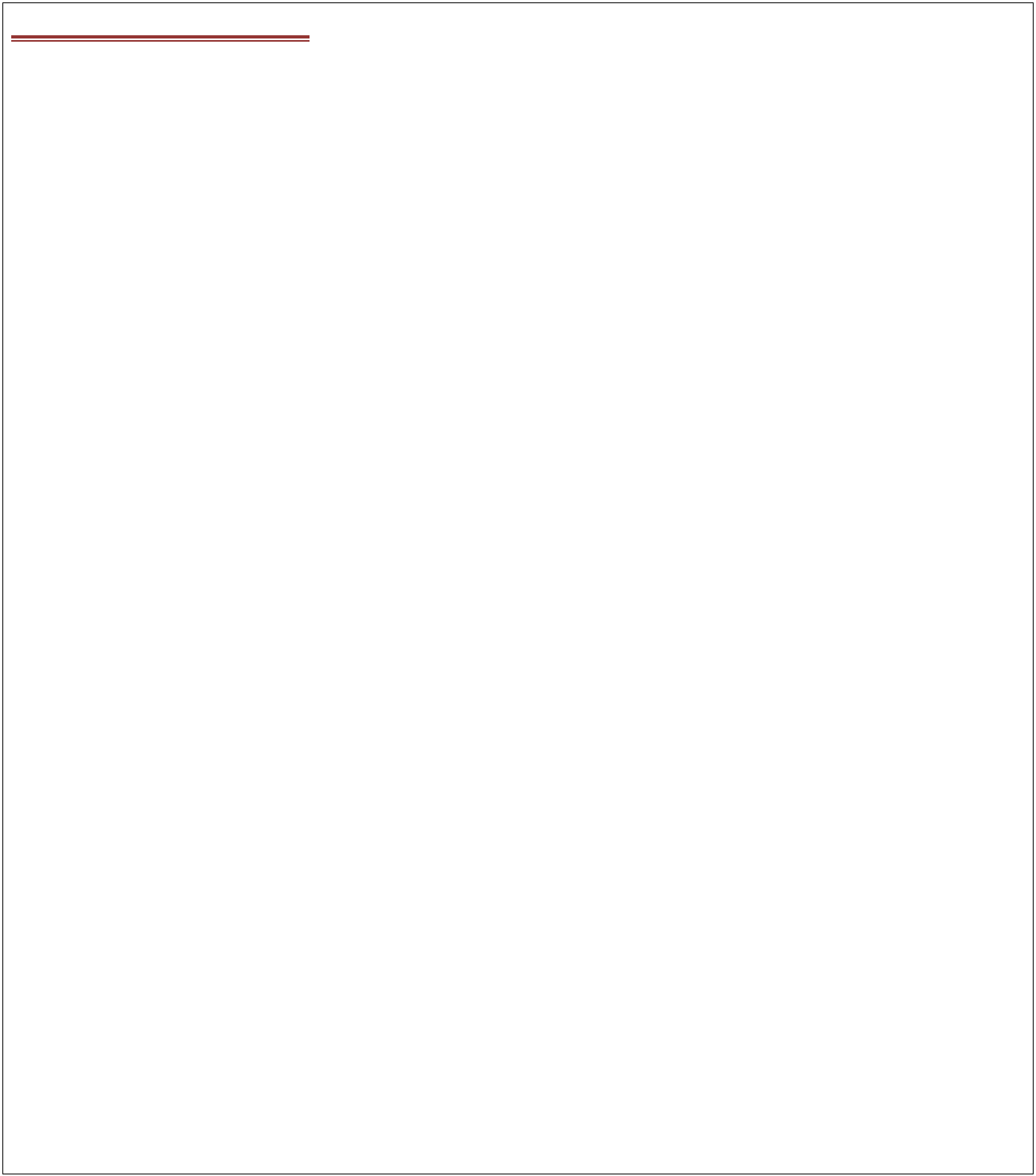
**public void show() {**

**System.out.println("Hai-show()");**

**System.out.println("Bean Name : "+bname);**

**}**

**}**



**3. Hello.java**

**package com.coursecube.spring;**

**import javax.annotation.Resource;**

**import org.springframework.beans.BeansException; import org.springframework.beans.factory.BeanFactory; import org.springframework.beans.factory.BeanFactoryAware; import org.springframework.beans.factory.BeanNameAware; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.context.ApplicationContext; import org.springframework.context.ApplicationContextAware; /\***

* **@Author : Srinivas Dande**
* **@Company : CourseCube**
* **@Website : www.coursecube.com**
* **\*/**

**public class Hello implements BeanNameAware,BeanFactoryAware,ApplicationContextAware{**

**String bname; //1 S.I**

**BeanFactory myfactory1; //2 S.I**

**@Resource**

**BeanFactory myfactory2; //2 F.I**

**ApplicationContext mycontext1;//3 S.I**

**@Autowired**

**ApplicationContext mycontext2;//3 F.I**

**public void setBeanName(String bname) { System.out.println("Hello-setBeanName()"); this.bname=bname;**

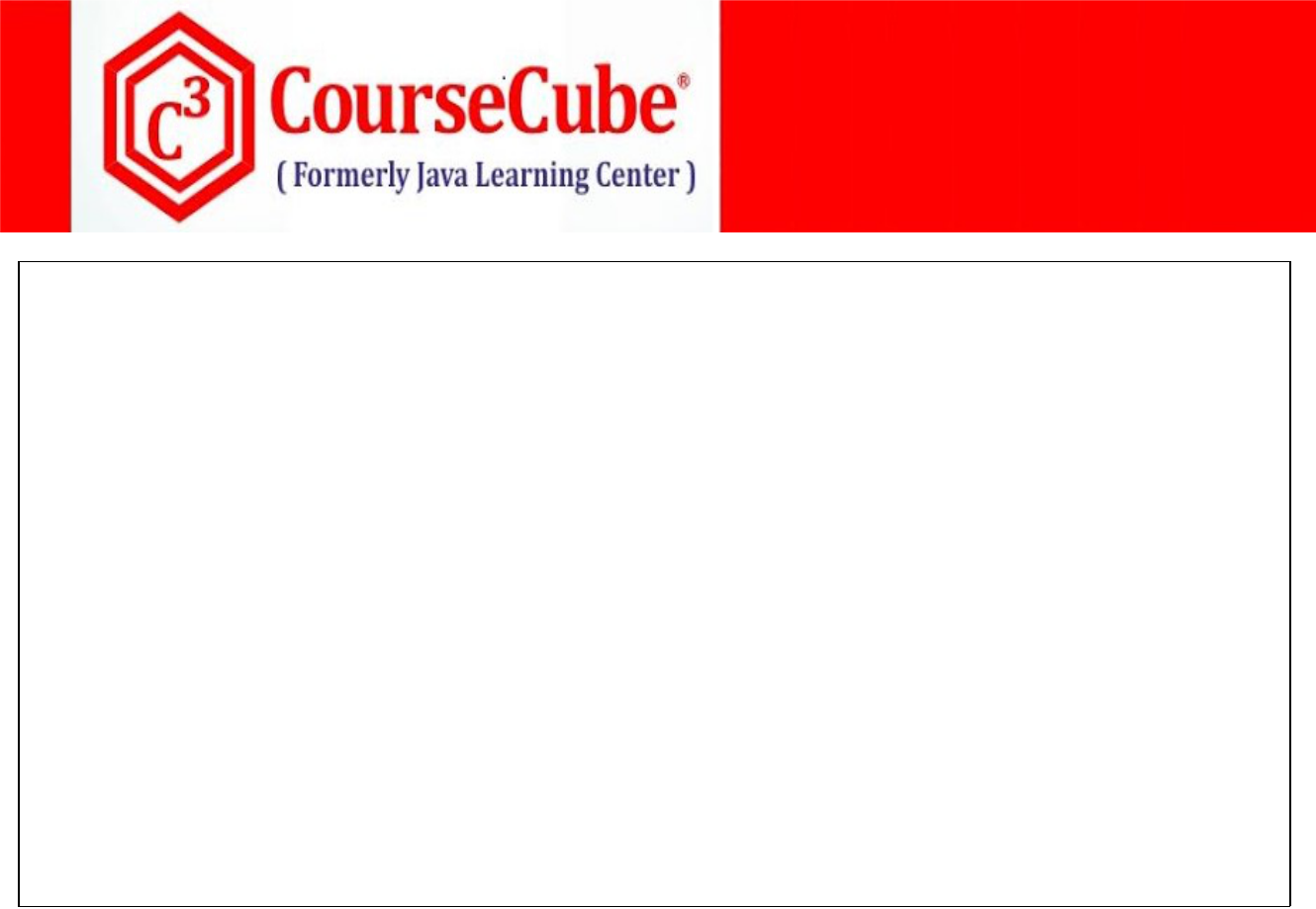
**}**

**public void setBeanFactory(BeanFactory myfactory1) throws BeansException { System.out.println("Hello-setBeanFactory()"); this.myfactory1=myfactory1;**

**}**



**www.coursecube.com** **83** **Spring-5 Study Guide**



**public void setApplicationContext(ApplicationContext mycontext1) throws BeansException { System.out.println("Hello-setApplicationContext()"); this.mycontext1=mycontext1;**

**}**

**public void show() {**

**System.out.println("Hello-show()");**

**System.out.println("Bean Name : "+bname);**

**System.out.println(myfactory1);**

**System.out.println(myfactory2);**

**System.out.println(myfactory1==myfactory2);**

**System.out.println(mycontext1);**

**System.out.println(mycontext2);**

**System.out.println(mycontext1==mycontext2);**

**}**

**}**



1. **MyPostProcessor.java package com.coursecube.spring;**

**import org.springframework.beans.BeansException;**

**import org.springframework.beans.factory.config.BeanPostProcessor; /\***

* **@Author : Srinivas Dande**
* **@Company : CourseCube**
* **@Website : www.coursecube.com**
* **\*/**

**public class MyPostProcessor implements BeanPostProcessor{**

**public Object postProcessBeforeInitialization(Object obj,String bname) throws BeansException { System.out.println("BeforeInitialization : "+bname+" - "+obj.getClass().getSimpleName()); return obj;**

**}**

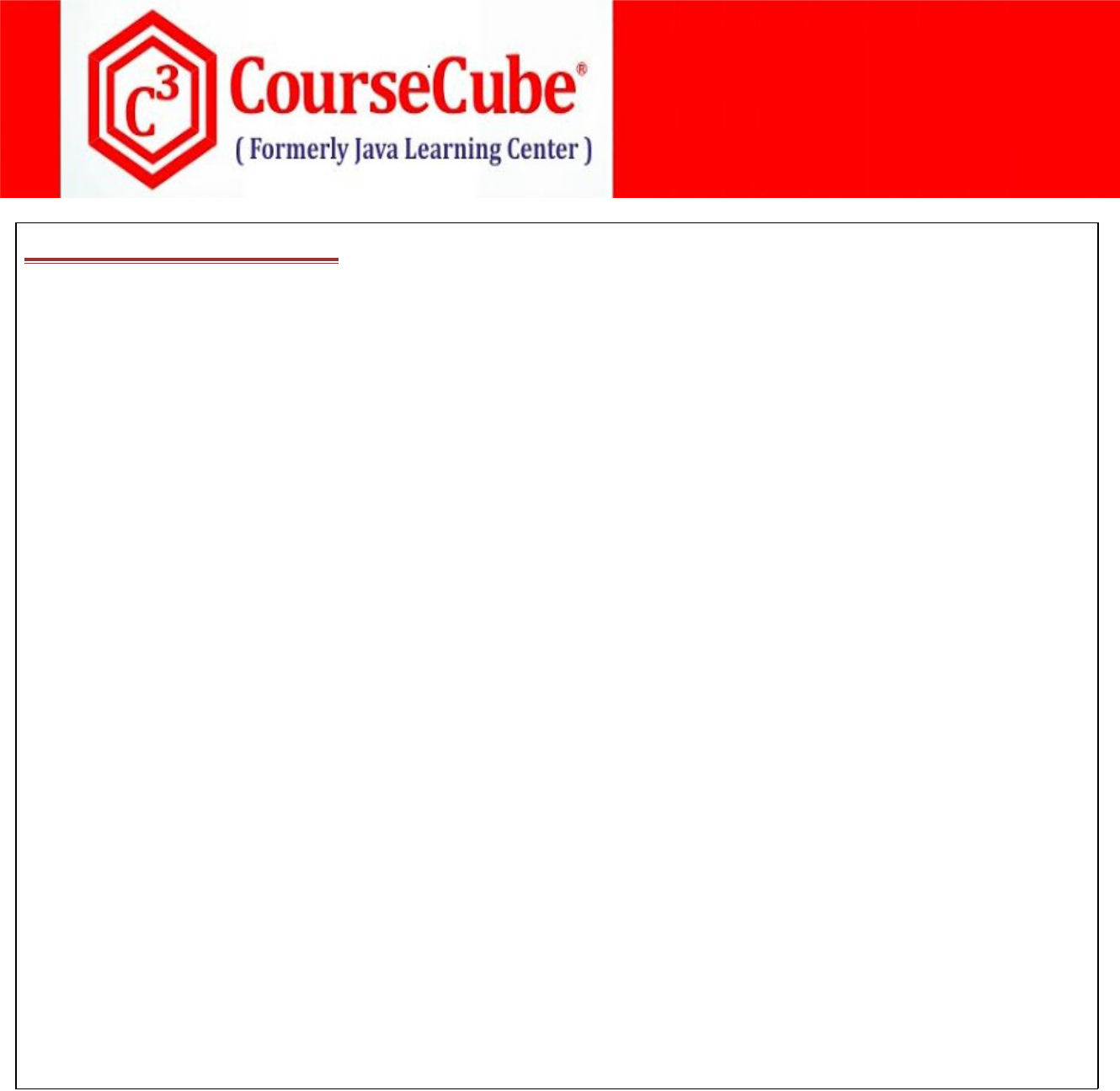
**public Object postProcessAfterInitialization(Object obj,String bname) throws BeansException { System.out.println("AfterInitialization : "+bname+" - "+obj.getClass().getSimpleName()); return obj;**

**}**

**}**



**www.coursecube.com** **84** **Spring-5 Study Guide**



1. **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(name="myhello")**

**public Hello createHello() {**

**return new Hello();**

**}**

**@Bean(name="myhai")**

**public Hai createHai() {**

**return new Hai();**

**}**

**@Bean**

**public MyPostProcessor getPostProcessor() {**

**return new MyPostProcessor();**

**}**

**}**



**www.coursecube.com** **85** **Spring-5 Study Guide**



**Spring Containers**

**There are two types of Spring containers provided.**

* 1. **BeanFactory Container**
  2. **ApplicationContext Container**

**1) BeanFactory Container**



**You can create the BeanFactory Container as follows in two steps.**

* 1. **Create the Resource object.**

**o Resource res=new ClassPathResource("jlcindia.xml");**

* + **Resource res=new FileSystemResource("D:/Spring/Labs/IOC/Lab21/src/jlcindia.xml");**

1. **Create the XmlBeanFactory object with Resource created above**
   * **BeanFactory factory=new XmlBeanFactory(res);**

**Life of Bean in the BeanFactory container**

1. **Bean Class will be loaded into memory.**
2. **Bean Instance will be created by using corresponding constructor (Constructor injection)**
3. **Bean dependencies will be injected with the following ways**
   1. **XML based Explicit Wiring**
   2. **XML based Auto Wiring**
4. **When bean class is implementing BeanNameAware interface then setBeanName() method will be called by the container.**
5. **When bean class is implementing BeanFactoryAware interface then setBeanFactory() method will be called by the container.**
6. **When bean class is implementing InitializingBean interface then afterPropertiesSet() method will be called by the container.**
7. **When bean definition contains init-method attribute then that specified method will be called.**
8. **Fully initialized Bean instance will be ready to use in the Bean Factory container.**

**At Spring Container Shutdown Time**

**At Spring Container Shutdown Time, It will destroy all the Bean instances. When container is destroying one bean instance, it will do the following tasks.**

* 1. **When bean class is implementing DisposableBean interface then destroy() method will be called by the container.**
  2. **When bean definition contains destroy-method attribute then that specified method will be called.**



**www.coursecube.com** **86** **Spring-5 Study Guide**



1. **ApplicationContext Container**

**ApplicationContext interface has three concrete implementations.**

* + 1. **ClassPathXmlApplicationContext**
    2. **FileSystemXmlApplicationContext**
    3. **AnnotationConfigApplicationContext**
    4. **WebApplicationContext**
    5. **XmlWebApplicationContext**
    6. **AnnotationConfigWebApplicationContext**

**You can create the ApplicationContext Container as follows.**



* **ApplicationContext ctx=new ClassPathXmlApplicationContext("jlcindia.xml");**
* **ApplicationContext ctx=new**

**FileSystemXmlApplicationContext("D:/Spring/Labs/IOC/Lab20/src/jlcindia.xml");**

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

**Life of Bean in the ApplicationContext container**

1. **Bean Class will be loaded into memory.**
2. **Bean Instance will be created by using corresponding constructor (Constructor injection)**
3. **Bean dependencies will be injected with the following ways**
   1. **Annotation based Auto Wiring.(Filed Injection )**
   2. **autowire attribute Auto Wiring (Setter Injection)**
   3. **Explicit Wiring (Setter Injection)**
4. **When bean class is implementing BeanNameAware interface then setBeanName() method will be called by the container.**
5. **When bean class is implementing BeanFactoryAware interface then setBeanFactory() method will be called by the container.**
6. **When bean class is implementing ApplicationContextAware interface then setApplicationContext() method will be called by the container.**
7. **When BeanPostProcessor is registered then postProcessBeforeInitialization() method will be called by the container.**
8. **When any method is found with @PostConstruct annotation then that method will be called.**
9. **When bean class is implementing InitializingBean interface then afterPropertiesSet() method will be called by the container.**



**www.coursecube.com** **87** **Spring-5 Study Guide**



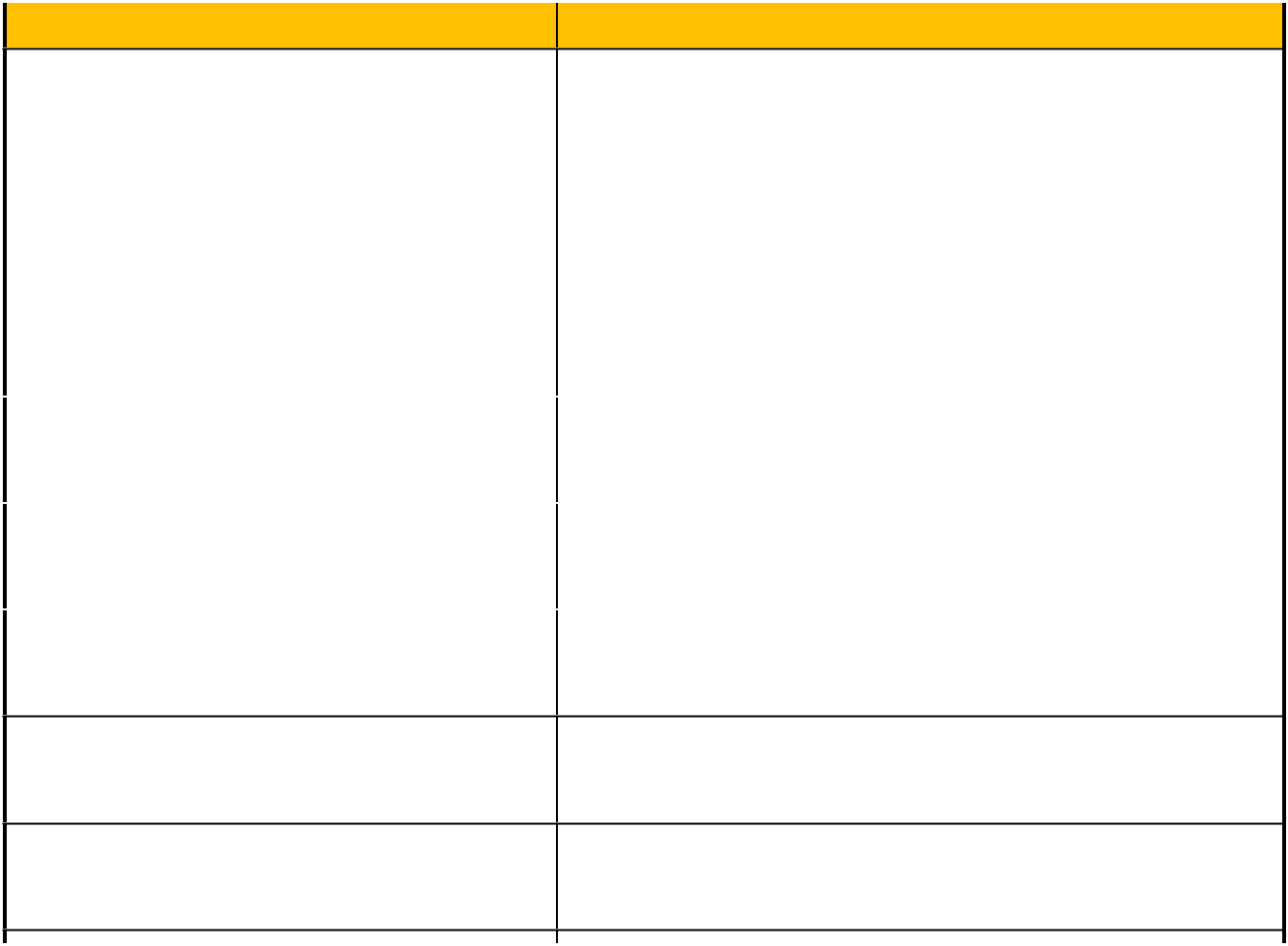
1. **When bean defintion contains init-method attribute then that specified method will be called.**
2. **When BeanPostProcessor is registered then postProcessAfterInitialization() method will be called by the container.**
3. **Fully initialized Bean instance will be ready to use in the ApplicationContext container.**

**At Spring Container Shutdown Time**

**At Container Shutdown Time, It will destroy all the Bean instances.**

**When container is destroying one bean instance, it will do the following tasks.**

* 1. **When any method is found with @PreDestroy annotation then that method will be called.**
  2. **When bean class is implementing DisposableBean interface then destroy() method will be called by the container.**
  3. **When bean defintion contains destroy-method attribute then that specified method will be called.**



**BeanFactory** **ApplicationContext**

**BeanFactory is an interface which has one concrete sub class called XmlBeanFactory.**

**ApplicationContext is an interface which has following sub class called**

**ClassPathXmlApplicationContext**

**FileSystemXmlApplicationContext**

**AnnotationConfigApplicationContext**

**WebApplicationContext**

**XmlWebApplicationContext**

**AnnotationConfigWebApplicationContext**

|  |  |
| --- | --- |
| **Beans configured with BeanFactory** | **Beans configured with ApplicationContext container will b** |
| **container will be loaded lazily by default.** | **loaded aggressively by default.** |
|  |  |
| **BeanFactory container does not support** | **ApplicationContext container supports annotations.** |
| **annotations.** |  |
|  |  |
| **BeanFactory container does not support** | **ApplicationContext container supports** |
| **BeanPostProcessor** | **BeanPostProcessor.** |

**BeanFactory container does not support Event Publishing.**

**ApplicationContext container supports Event Publishing.**

**BeanFactory container does not provide the way to resolve message bundles.**

**ApplicationContext container provides the way to resolve message bundles.**



**www.coursecube.com** **88** **Spring-5 Study Guide**



**Similarities between BeanFactory container and ApplicationContext container**

**Both the containers will do the following common tasks:**

1. **Loads bean classes.**
2. **Create the bean instances.**
3. **Injects the bean Dependencies.**
4. **Maintains the Bean Instances as long as container is running.**
5. **Destroys the Bean Instances at shut-down time.**

**Bean definition with XML Configuration**

**<bean id="myhello"**

**name="myhello"**

**class="com.jlc.Hello"**

**lazy-init="true"**

**scope="prototype"**

**autowire="byName"**

**init-method=" myInit"**

**destroy-method=" myCleanup"**

**…/>**

**Bean definition with Java Configuration**

**@Bean( name="myhello",**

**autowire =Autowire.BY\_NAME,**

**initMethod = "myInit",**

**destroyMethod = "myCleanup")**

**@Scope("prototype")**

**@Lazy(true)**

**public Hello hello() {**

**return new Hello();**

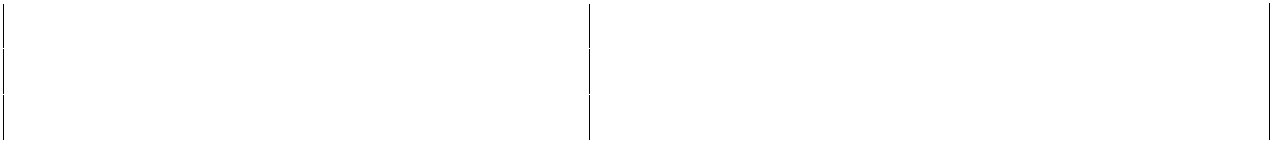
**}**



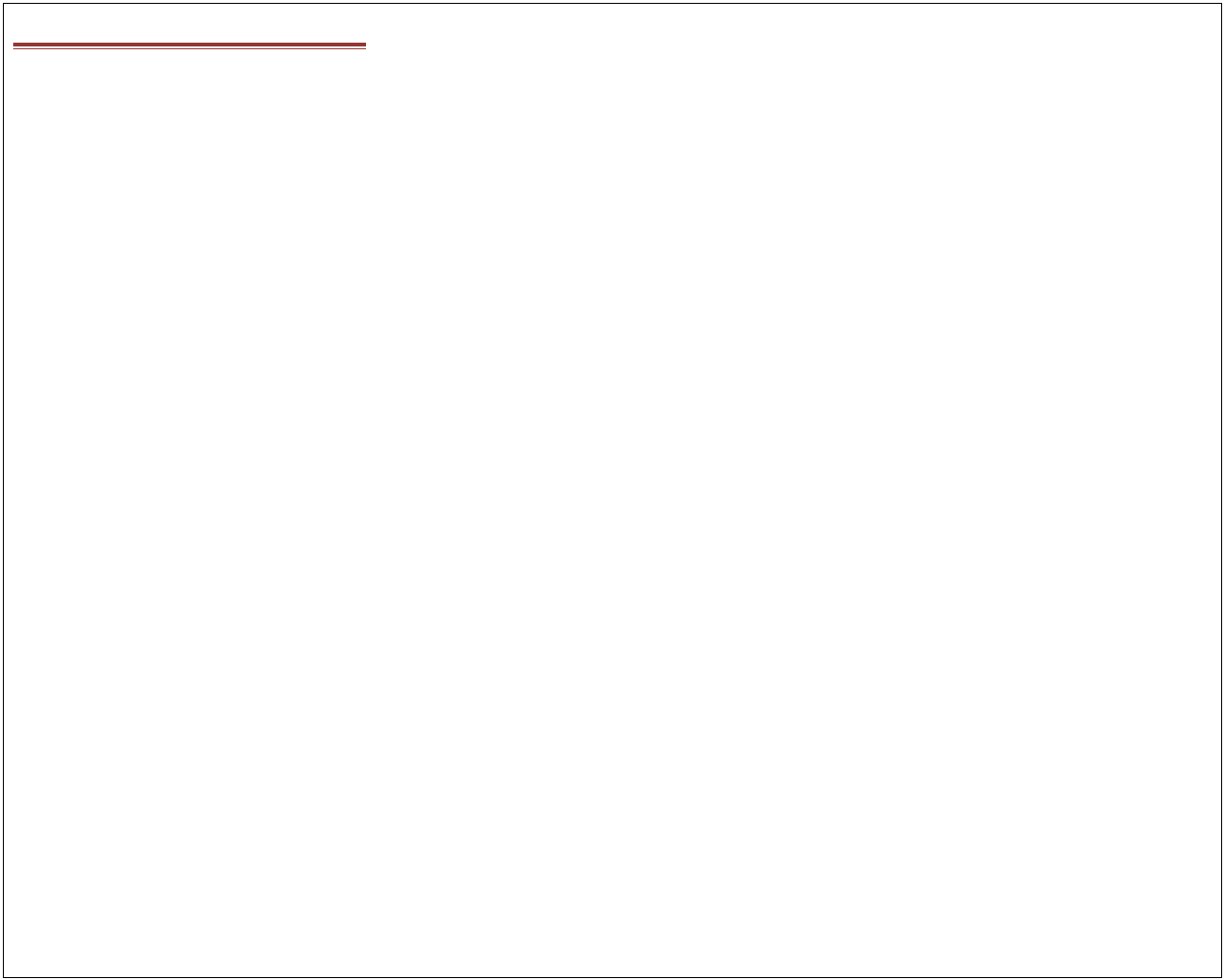
**www.coursecube.com** **89** **Spring-5 Study Guide**



**Lab41: Files required**



|  |  |  |  |
| --- | --- | --- | --- |
| **1.** | **Lab41.java** | **2.** | **A.java** |
|  |  |  |  |
| **3.** | **B.java** | **4.** | **Hello.java** |
|  |  |  |  |
| **5.** | **MyPostProcessor.java** | **6.** | **JLCAppConfig.java** |
|  |  |  |  |



**1. Lab41.java**

**package com.coursecube.spring;**

**import org.springframework.context.annotation.AnnotationConfigApplicationContext; import org.springframework.context.support.AbstractApplicationContext; /\***

* **@Author : Srinivas Dande**
* **@Company : CourseCube**
* **@Website : www.coursecube.com**
* **\*/**

**public class Lab41 {**

**public static void main(String[] args) {**

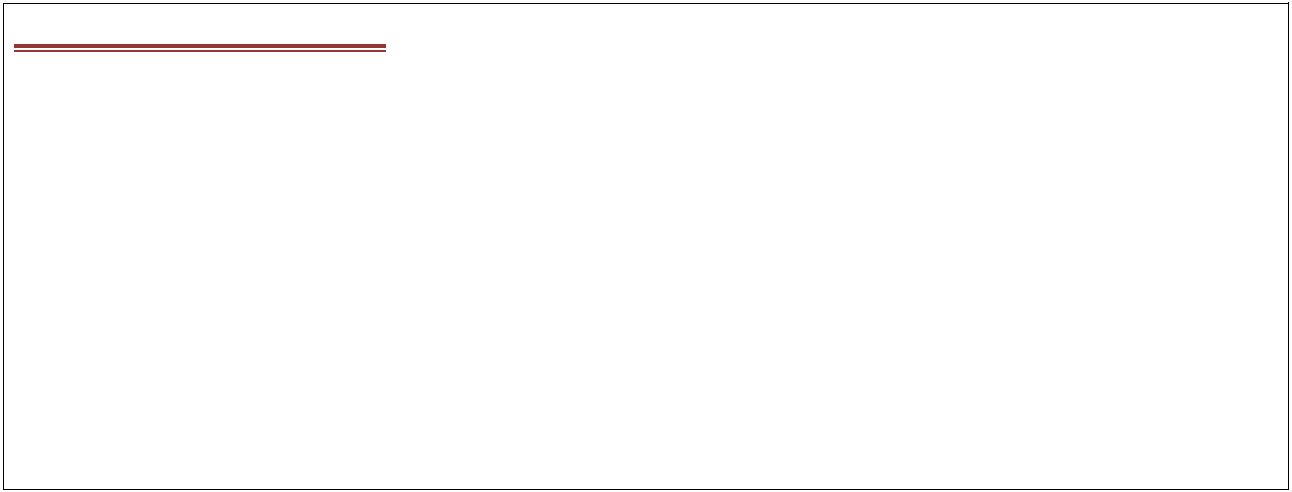
**AbstractApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig.class);**

|  |  |  |  |
| --- | --- | --- | --- |
| **System.out.println("--------** | **Now Spring Container is Ready** | **--------");** |  |
| **System.out.println("-----------** | **I am going to Use beans---------------** | **");** |  |
| **Hello hello=(Hello)ctx.getBean("myhello");** | |  |  |
| **hello.show();** |  |  |  |
| **System.out.println("-----------** | **Bean Usage completed ---------------** | **");** |  |
| **System.out.println("---------** | **Now Going to Shut-Down Spring Container-----** | | **");** |
| **ctx.close();** |  |  |  |

**ctx.registerShutdownHook();**

**}**

**}**



**2. A.java**

**package com.coursecube.spring;**

**public class A {**

**String msg; //Dependency**

**public void setMsg(String msg) {**

**this.msg = msg;**

**}**

**public String toString() {**

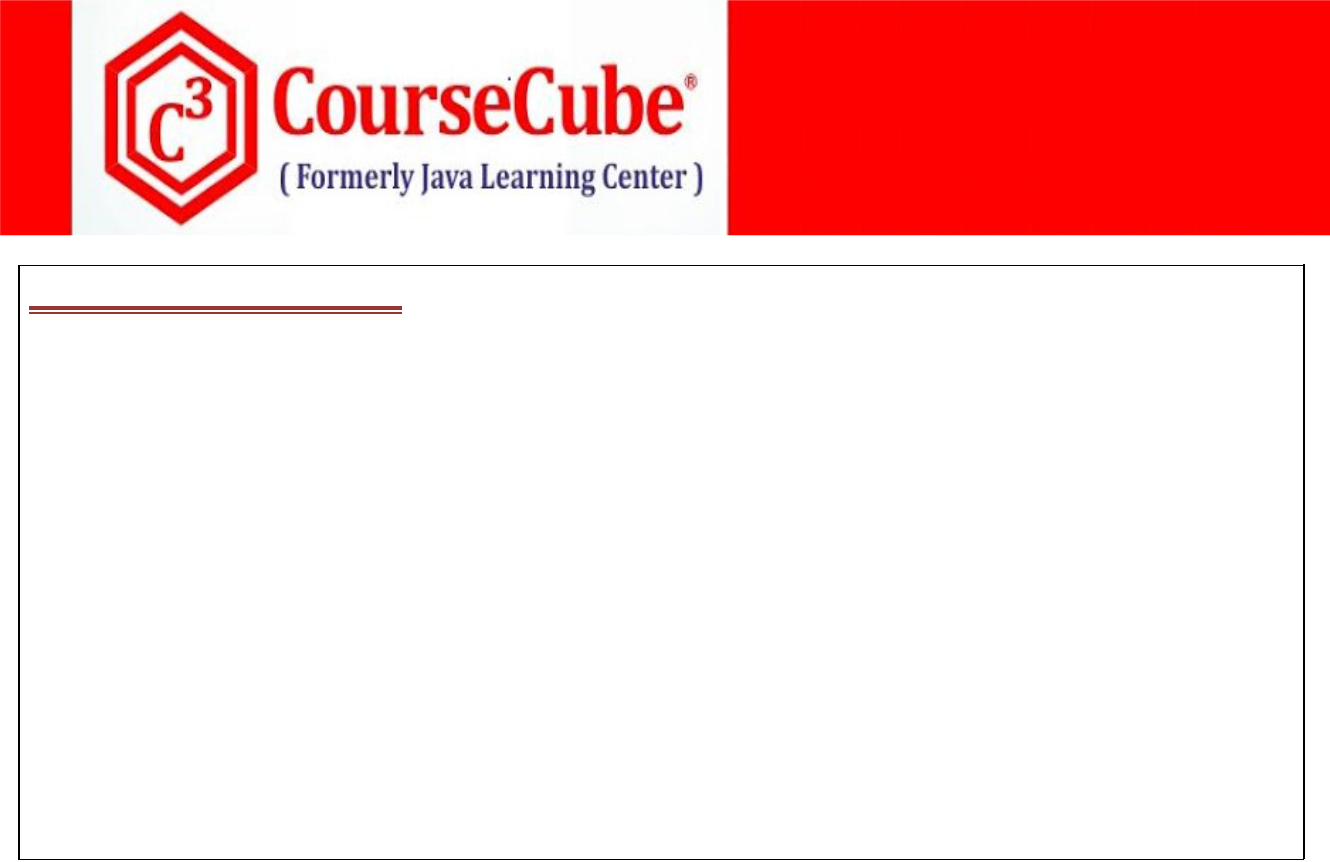
**return msg;**

**}**

**}**



**www.coursecube.com** **90** **Spring-5 Study Guide**



**3. B.java**

**package com.coursecube.spring;**

**/\***

* **@Author : Srinivas Dande**
* **@Company : CourseCube**
* **@Website : www.coursecube.com**
* **\*/**

**public class B {**

**String str; //Dependency**

**public void setStr(String str) {**

**this.str = str;**

**}**

**public String toString() {**

**return str;**

**}**

**}**



**4. Hello.java**

**package com.coursecube.spring;**

**import javax.annotation.\*;**

**import org.springframework.beans.BeansException; import org.springframework.beans.factory.\*;**

**import org.springframework.beans.factory.annotation.Autowired; import org.springframework.context.\*; /\***

* **@Author : Srinivas Dande**
* **@Company : CourseCube**
* **@Website : www.coursecube.com**
* **\*/**

**public class Hello implements InitializingBean,DisposableBean, BeanNameAware,BeanFactoryAware,ApplicationContextAware{**

**String myname; //2 . C.I**

**@Autowired**

**A aobj; //3.A F.I**

**B bobj; //3.B S.I**

**String bname;**

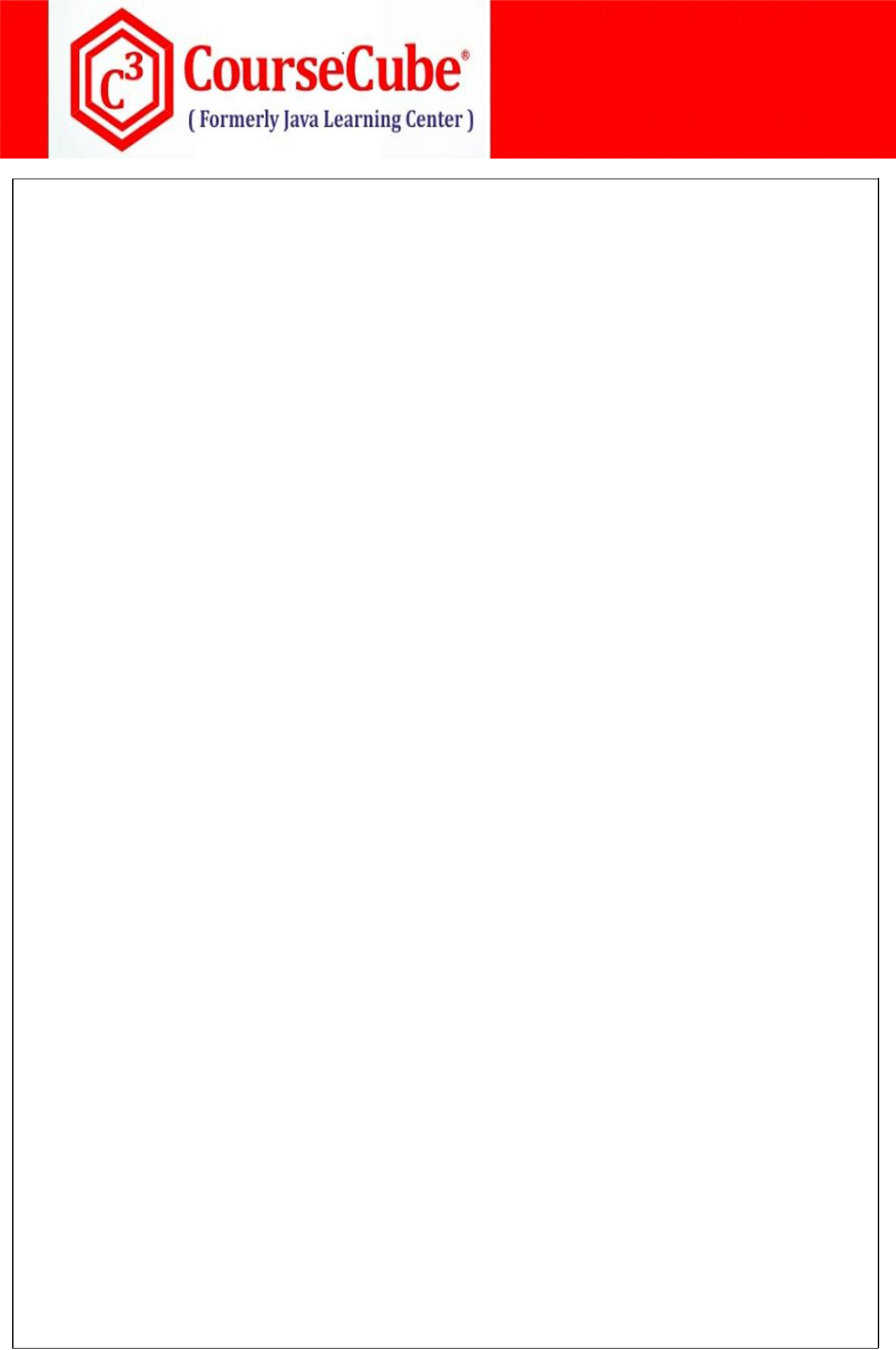
**BeanFactory myfactory1;**

**@Resource**

**BeanFactory myfactory2;**



**www.coursecube.com** **91** **Spring-5 Study Guide**



**ApplicationContext mycontext1;**

**@Autowired**

**ApplicationContext mycontext2;**

**static {**

**System.out.println("1.Hello-S.B"); //1**

**}**

**public Hello(String myname) {**

**System.out.println("2.Hello-1-Arg"); //2**

**this.myname=myname;**

**}**

**public void setBobj(B bobj) { System.out.println("3.B.Hello-setBobj"); //3 this.bobj=bobj;**

**}**

**public void setBeanName(String bname) { System.out.println("4.Hello-setBeanName()"); this.bname=bname;**

**}**

**public void setBeanFactory(BeanFactory myfactory1) throws BeansException { System.out.println("5.Hello-setBeanFactory()"); this.myfactory1=myfactory1;**

**}**

**public void setApplicationContext(ApplicationContext mycontext1) throws BeansException{ System.out.println("6.Hello-setApplicationContext()"); this.mycontext1=mycontext1;**

**}**

**@PostConstruct**

**public void init() {**

**System.out.println("8.Hello-init()");**

**//Initialization Code**

**}**

**public void afterPropertiesSet() throws Exception { System.out.println("9.Hello-afterPropertiesSet()"); //Initialization Code**

**}**

**public void myInit() {**

**System.out.println("10.Hello-myInit()");**

**//Initialization Code**

**}**

**@PreDestroy**

**public void cleanup() {**

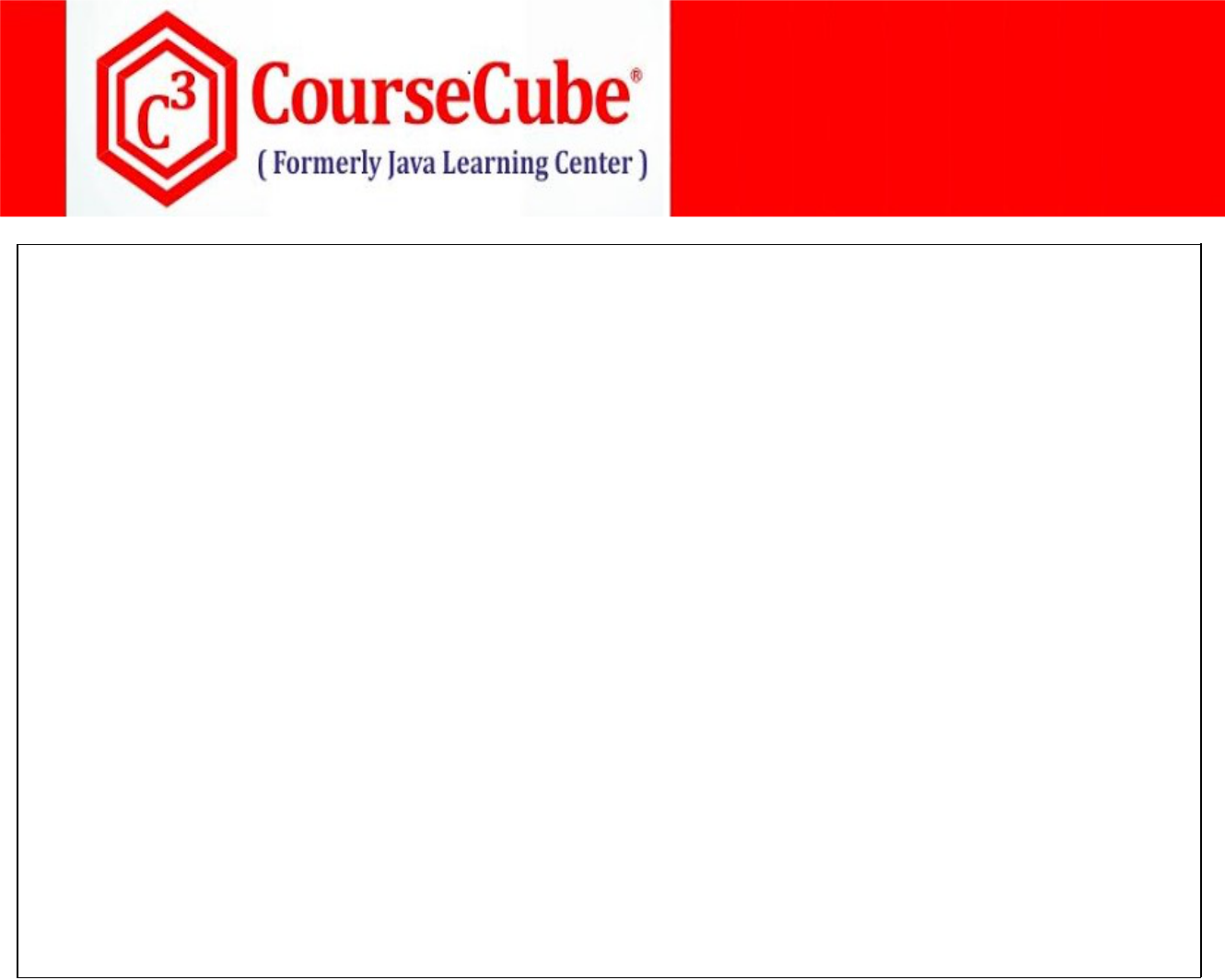
**System.out.println("A.Hello-cleanup()");**

**//Cleanup -Code**

**}**



**www.coursecube.com** **92** **Spring-5 Study Guide**



**public void destroy() throws Exception {**

**System.out.println("B.Hello-destroy()");**

**//Cleanup -Code**

**}**

**public void myCleanup() { System.out.println("C.Hello-myCleanup()"); //Cleanup -Code**

**}**

**public void show() {**

**System.out.println("Hello-show()");**

**System.out.println(myname);**

**System.out.println(aobj);**

**System.out.println(bobj);**

**System.out.println("Bean Name : "+bname);**

**System.out.println(myfactory1);**

**System.out.println(myfactory2);**

**System.out.println(mycontext1);**

**System.out.println(mycontext2);**

**}**

**}**



1. **MyPostProcessor.java package com.coursecube.spring;**

**import org.springframework.beans.BeansException;**

**import org.springframework.beans.factory.config.BeanPostProcessor; /\***

* **@Author : Srinivas Dande**
* **@Company : CourseCube**
* **@Website : www.coursecube.com**
* **\*/**

**public class MyPostProcessor implements BeanPostProcessor{**

**public Object postProcessBeforeInitialization(Object obj,String bname) throws BeansException { System.out.println("7.BeforeInitialization : "+bname+" - "+obj.getClass().getSimpleName()); return obj;**

**}**

**public Object postProcessAfterInitialization(Object obj,String bname) throws BeansException { System.out.println("11.AfterInitialization : "+bname+" - "+obj.getClass().getSimpleName()); return obj;**

**}**

**}**



**www.coursecube.com** **93** **Spring-5 Study Guide**



1. **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="myao")**

**public A createA() {**

**A ao=new A();**

**ao.setMsg("I am Bean - A");**

**return ao;**

**}**

**@Bean(name="mybo")**

**public B createB() {**

**B bo=new B();**

**bo.setStr("I am Bean - B");**

**return bo;**

**}**

**@Bean(name="myhello",autowire = Autowire.BY\_TYPE, initMethod = " myInit", destroyMethod = "myCleanup")**

**public Hello createHello() {**

**return new Hello("Srinivas Dande");**

**}**

**@Bean**

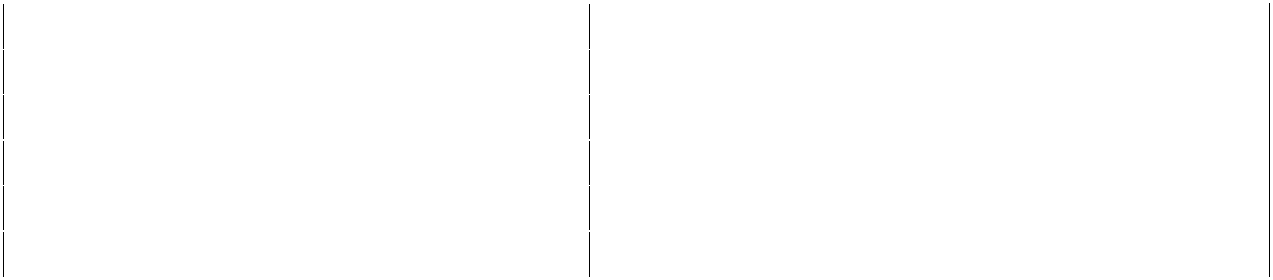
**public MyPostProcessor getPostProcessor() {**

**return new MyPostProcessor();**

**}**

**}**

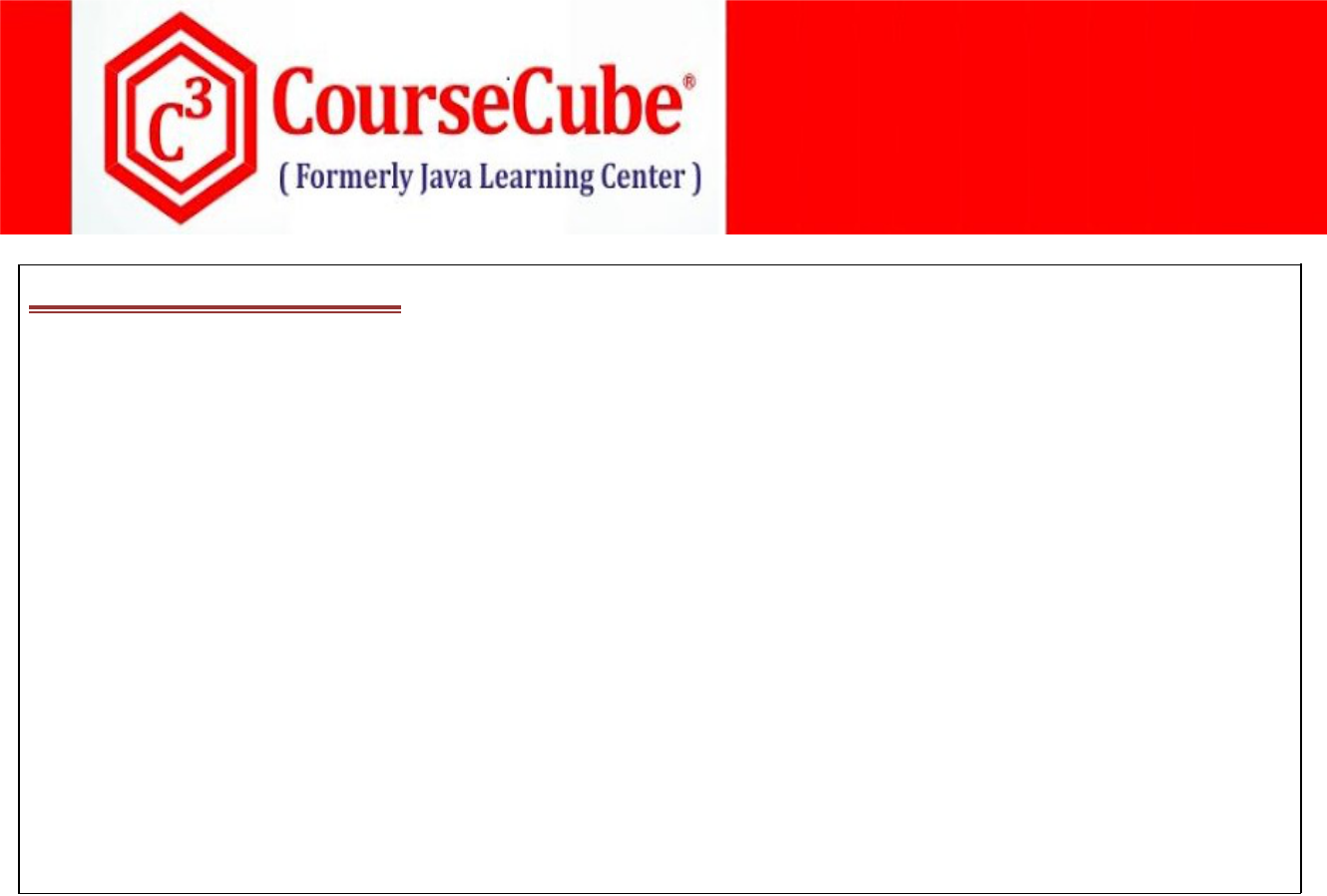
**Lab42: Files required**



|  |  |  |
| --- | --- | --- |
| **1.** | **Lab42.java** | **New in Lab42** |
|  |  |  |
| **2.** | **A.java** | **Same as Lab41** |
|  |  |  |
| **3.** | **B.java** | **Same as Lab41** |
|  |  |  |
| **4.** | **Hello.java** | **Same as Lab41** |
|  |  |  |
| **5.** | **MyPostProcessor.java** | **Same as Lab41** |
|  |  |  |
| **6.** | **Jlcindia.xml** | **New in Lab42** |
|  |  |  |



**www.coursecube.com** **94** **Spring-5 Study Guide**



**1. Lab42.java**

**package com.coursecube.spring;**

**import org.springframework.beans.factory.BeanFactory;**

**import org.springframework.beans.factory.xml.XmlBeanFactory; import org.springframework.core.io.\*;**

**public class Lab42 {**

**public static void main(String[] args) {**

**Resource res=new ClassPathResource("jlcindia.xml");**

**BeanFactory factory=new XmlBeanFactory(res);**

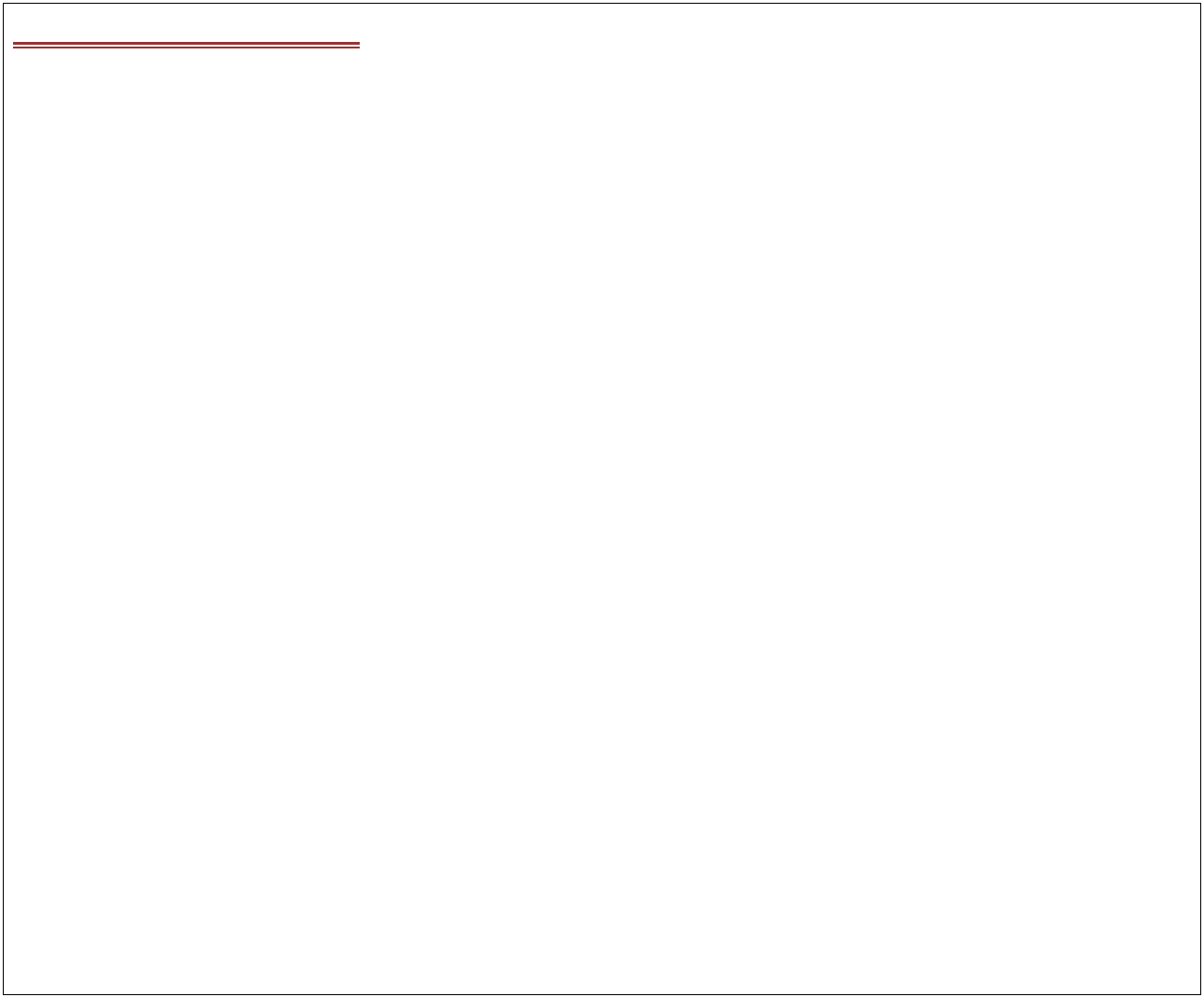
**System.out.println("--------Now Spring Container is Ready--------");**

**Hello hello=(Hello) factory.getBean("myhello");**

**hello.show();**

**}**

**}**



**6. jlcindia.xml**

**<?xml version="1.0" encoding="UTF-8"?>**

**<beans xmlns="http://www.springframework.org/schema/beans" xmlns:context="http://www.springframework.org/schema/context" 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 http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd">**

**<context:annotation-config/>**

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

**<bean id="bobj" class="com.coursecube.spring.B"> <property name="str" value="I am Bean - B" /> </bean>**

**<bean id="myhello"** **class="com.coursecube.spring.Hello"**

**init-method="myInit" destroy-method="myCleanup"**

**autowire="byName">**

**<constructor-arg value="Srinivas Dande" />**

**</bean>**

**<bean class="com.coursecube.spring.MyPostProcessor" />**

**</beans>**



**www.coursecube.com** **95** **Spring-5 Study Guide**