

**Example using Annotation Based AOP**



**Lab54:**

**Annotation based AOP with Autoproxying**

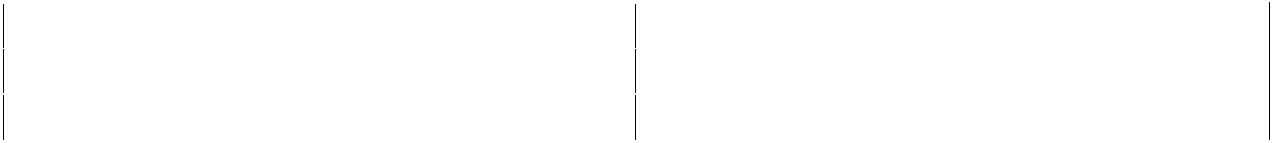
**@Before, @AfterReturning, @AfterThrowing**

**AspectJ Pointcuts Expressions based on annotations with @PointCut**

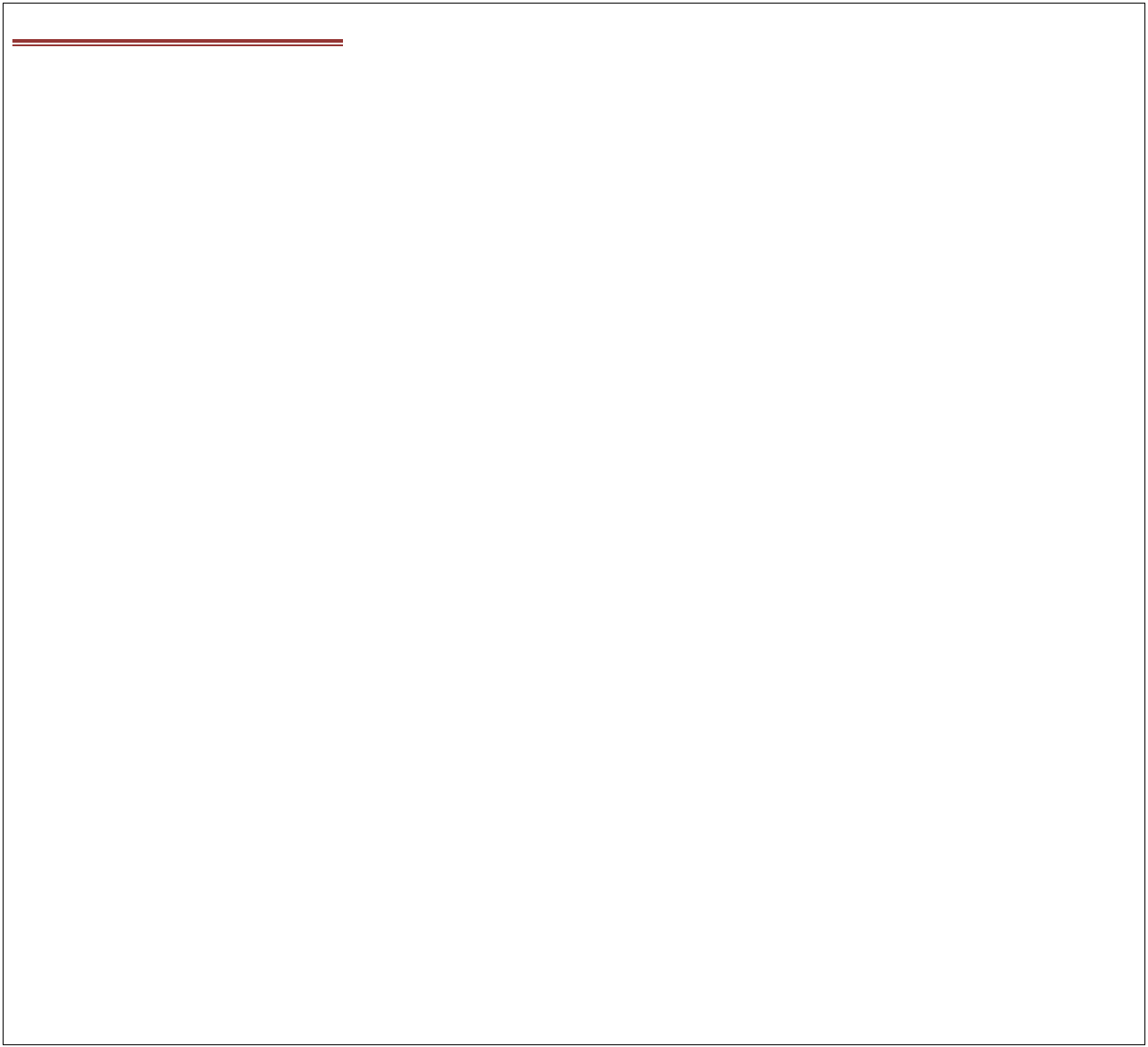
**Passing Arguments and target object with AspectJ Pointcuts Expressions Handling Return Values and Exceptions thrown.**

**One Busniess Services and Two Middle Level Service**

**Lab54: Files required**



|  |  |  |  |
| --- | --- | --- | --- |
| **1.** | **Lab54.java** | **2.** | **CustomerService.java** |
|  |  |  |  |
| **3.** | **SecurityService.java** | **4.** | **LogService.java** |
|  |  |  |  |
| **5.** | **CustomerNotFoundException.java** | **6.** | **JLCAppConfig.java** |
|  |  |  |  |



**1. Lab54.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 Lab54 {**

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

**ApplicationContext ctx = new AnnotationConfigApplicationContext(JLCAppConfig.class); CustomerService cs = (CustomerService) ctx.getBean("mycs"); int x = cs.addCustomer(101, "Sri", "Sri@jlc");**

**System.out.println("In Main() : " + x);**

**System.out.println("=============== ");**

**String str = cs.updateCustomer(102, "Vas", "Vas@jlc", 999);**

**System.out.println("In Main() : " + str);**

**System.out.println("=============== ");**

**try {**

**String email = cs.getCustomerEmail(99);**

**System.out.println("In Main() : " + email);**

**} catch (Exception ex) { System.out.println("In Main() : " + ex);**

**}**

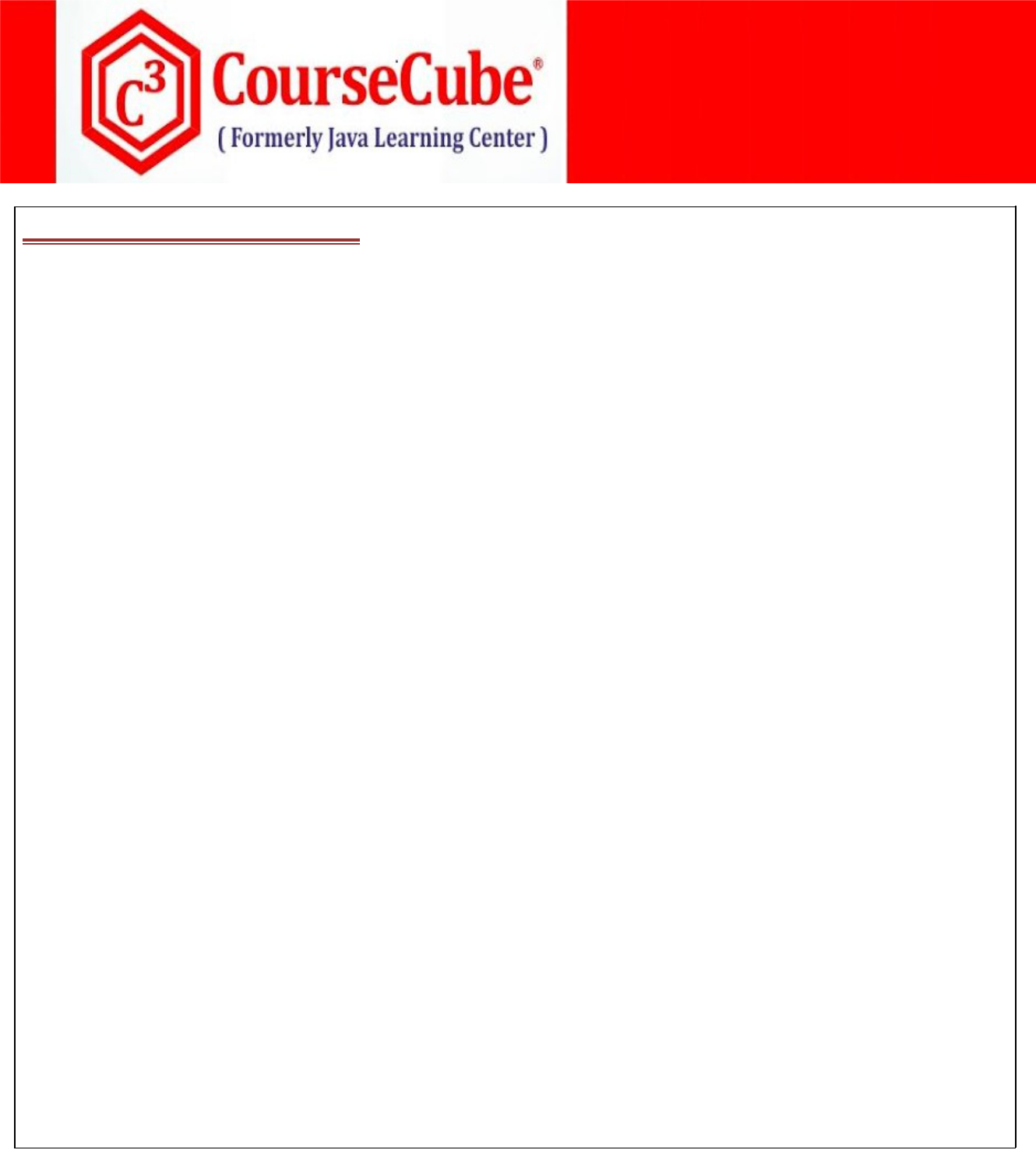
**System.out.println("=============== ");**

**}**

**}**



**www.coursecube.com** **164** **Spring-5 Study Guide**



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

**import org.springframework.stereotype.Component; /\***

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

**@Service("mycs")**

**public class CustomerService {**

**public int addCustomer(int cid, String cname, String email) { System.out.println("addCustomer - begin"); System.out.println("addCustomer - done"); System.out.println("addCustomer - end");**

**return 123;**

**}**

**public String updateCustomer(int cid, String cname, String email, long phone) { System.out.println("updateCustomer - begin"); System.out.println("updateCustomer - done"); System.out.println("updateCustomer - end");**

**return "Updated Successfully";**

**}**

**public String getCustomerEmail(int cid) { System.out.println("getCustomerEmail - begin"); if (1 == 2)**

**System.out.println("getCustomerEmail - done");**

**else**

**throw new CustomerNotFoundException(); System.out.println("getCustomerEmail - end"); return "srinivas@jlc.com";**

**}**

**}**



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

**import org.aspectj.lang.annotation.\*;**

**import org.springframework.core.annotation.Order; import org.springframework.stereotype.Component; /\***

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

**@Component**

**@Aspect**

**@Order(value = 1)**



**www.coursecube.com** **165** **Spring-5 Study Guide**



**public class SecurityService {**

**@Before("execution(\* com.coursecube.spring.\*Service.\*(..)) and args(cid) && target(bean)") public void verifyUser1(Object bean, int cid) {**

**System.out.println("\*\* SS - verifyUser1 : " + cid); System.out.println(bean.getClass().getSimpleName()); }**

**@Before("execution(\* com.coursecube.spring.\*Service.\*(..)) and args(cid)")**

**public void verifyUser2(int cid) {**

**System.out.println("\*\* SS - verifyUser2 : " + cid);**

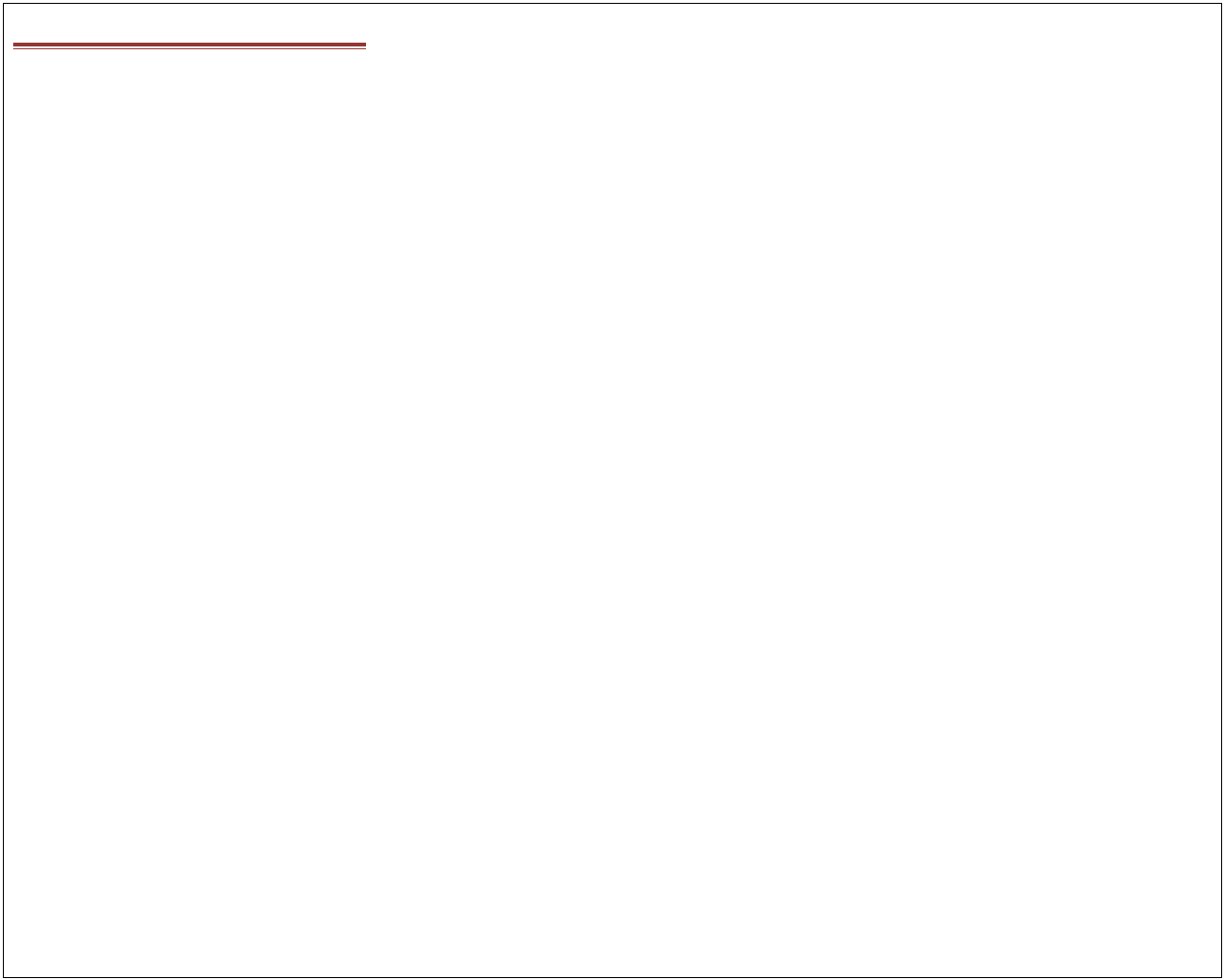
**}**

**@Before("execution(\* com.coursecube.spring.\*Service.\*(..)) and args(cid,cname,email)") public void verifyUser3(int cid, String cname, String email) {**

**System.out.println("\*\* SS - verifyUser3 : " + cid + "\t" + cname + "\t" + email); }**

**@Before("execution(\* com.coursecube.spring.\*Service.\*(..)) and args(cid,cname,email,phone)")**

**public void verifyUser4(int cid, String cname, String email, long phone) { System.out.println("\*\* SS - verifyUser4 : " + cid + "\t" + cname + "\t" + email + "\t" + phone); } }**



**4. LogService.java**

**package com.coursecube.spring;**

**import org.aspectj.lang.annotation.\*;**

**import org.springframework.core.annotation.Order; import org.springframework.stereotype.Component; /\***

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

**@Component**

**@Aspect**

**@Order(value = 2)**

**public class LogService {**

**@Before(value = "execution(\* com.coursecube.spring.\*Service.\*(..)) and target(bean)", argNames = "bean")**

**public void logBefore(Object bean) {**

**System.out.println("\*\* LS- logBefore");**

**System.out.println("Target Bean Class : " + bean.getClass().getSimpleName()); }**

**@AfterReturning(pointcut = "execution(\* com.coursecube.spring.\*Service.\*(..)) and target(bean)", returning = "returnVal")**

**public void logReturning(Object bean, Object returnVal) { System.out.println("\*\* LS- logReturning");**

**System.out.println("Target Bean Class : " + bean.getClass().getSimpleName()); System.out.println("Return Value : " + returnVal); }**



**www.coursecube.com** **166** **Spring-5 Study Guide**



**@AfterThrowing(pointcut = "execution(\* com.coursecube.spring.\*Service.\*(..)) and target(bean)", throwing = "ex")**

**public void logThrowing(Object bean, Exception ex) { System.out.println("\*\* LS- logThrowing");**

**System.out.println("Target Bean Class : " + bean.getClass().getSimpleName());**

**System.out.println("Exception Thrown : " + ex);**

**}**

**}**



**5. InsufficientFundsException.java**

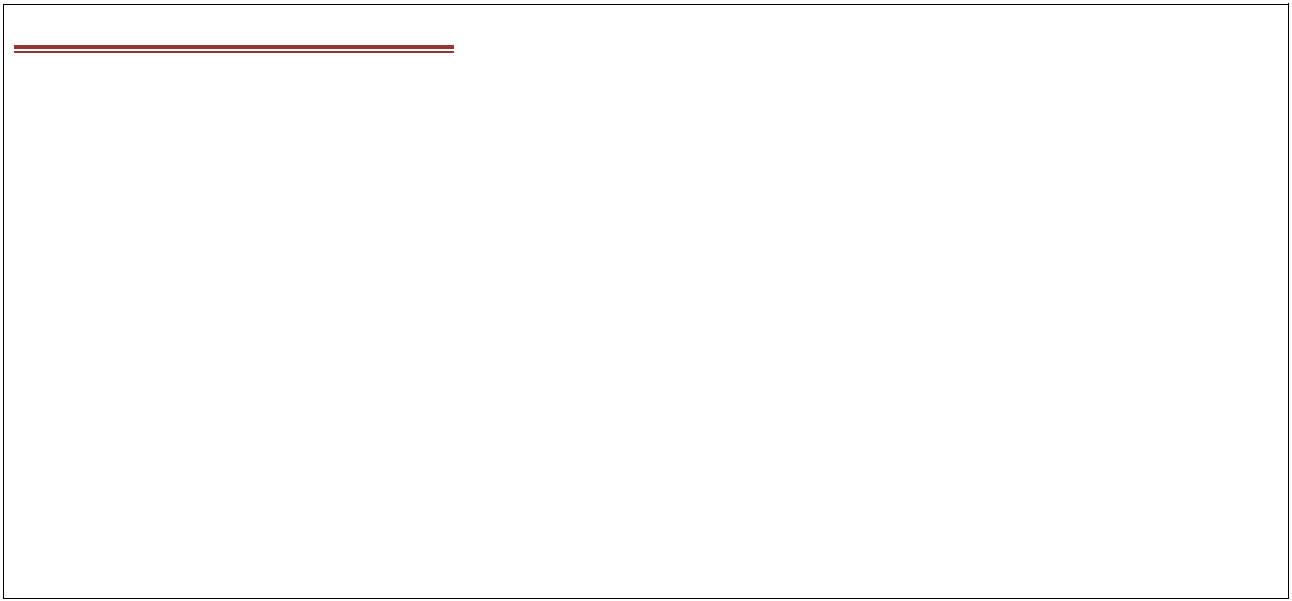
**package com.coursecube.spring;**

**/\***

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

**public class CustomerNotFoundException extends RuntimeException {**

**}**



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

**import org.springframework.context.annotation.\*;**

**import org.springframework.context.annotation.EnableAspectJAutoProxy; /\***

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

**@Configuration**

**@EnableAspectJAutoProxy**

**@ComponentScan(basePackages = "com.coursecube.spring")**

**public class JLCAppConfig {**

**}**

**Example using Annotation Based AOP**



**Lab55:**

**Annotation based AOP with Autoproxying**

**@Around,@AfterThrowing**

**AspectJ Pointcuts Expressions based on annotations with @PointCut**

**Passing Arguments and target object with AspectJ Pointcuts Expressions Handling Return Values and Exceptions thrown.**

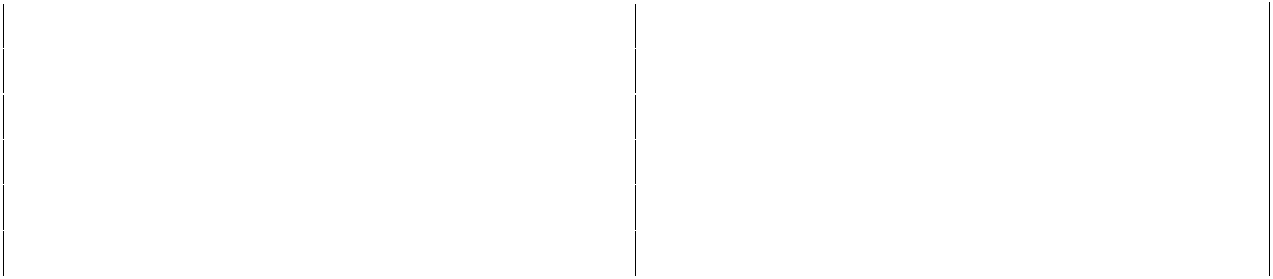
**One Busniess Services and Two Middle Level Services**



**www.coursecube.com** **167** **Spring-5 Study Guide**



**Lab55: Files required**



|  |  |  |
| --- | --- | --- |
| **1.** | **Lab55.java** | **Same as Lab54** |
|  |  |  |
| **2.** | **CustomerService.java** | **Same as Lab54** |
|  |  |  |
| **3.** | **SecurityService.java** | **Updated in Lab55** |
|  |  |  |
| **4.** | **LogService.java** | **Updated in Lab55** |
|  |  |  |
| **5.** | **CustomerNotFoundException.java** | **Same as Lab54** |
|  |  |  |
| **6.** | **JLCAppConfig.java** | **Same as Lab54** |
|  |  |  |



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

**import org.aspectj.lang.ProceedingJoinPoint;**

**import org.aspectj.lang.annotation.\*;**

**import org.springframework.core.annotation.Order; import org.springframework.stereotype.Component; /\***

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

**@Component**

**@Aspect**

**@Order(value = 1)**

**public class SecurityService {**

**@Around("execution(\* com.coursecube.spring.\*Service.\*(..)) and args(cid) && target(bean)")**

**public Object verifyUser1(ProceedingJoinPoint pjp, Object bean, int cid) throws Throwable {**

**System.out.println("\*\* SS - verifyUser1 : " + cid);**

**System.out.println(bean.getClass().getSimpleName());**

**Object returnVal = pjp.proceed();**

**return returnVal;**

**}**

**@Around("execution(\* com.coursecube.spring.\*Service.\*(..)) and args(cid)") public Object verifyUser2(ProceedingJoinPoint pjp, int cid) throws Throwable { System.out.println("\*\* SS - verifyUser2 : " + cid); Object returnVal = pjp.proceed();**

**return returnVal;**

**}**

**@Around("execution(\* com.coursecube.spring.\*Service.\*(..)) and args(cid,cname,email)") public Object verifyUser3(ProceedingJoinPoint pjp, int cid, String cname, String email) throws Throwable {**

**System.out.println("\*\* SS - verifyUser3 : " + cid + "\t" + cname + "\t" + email);**

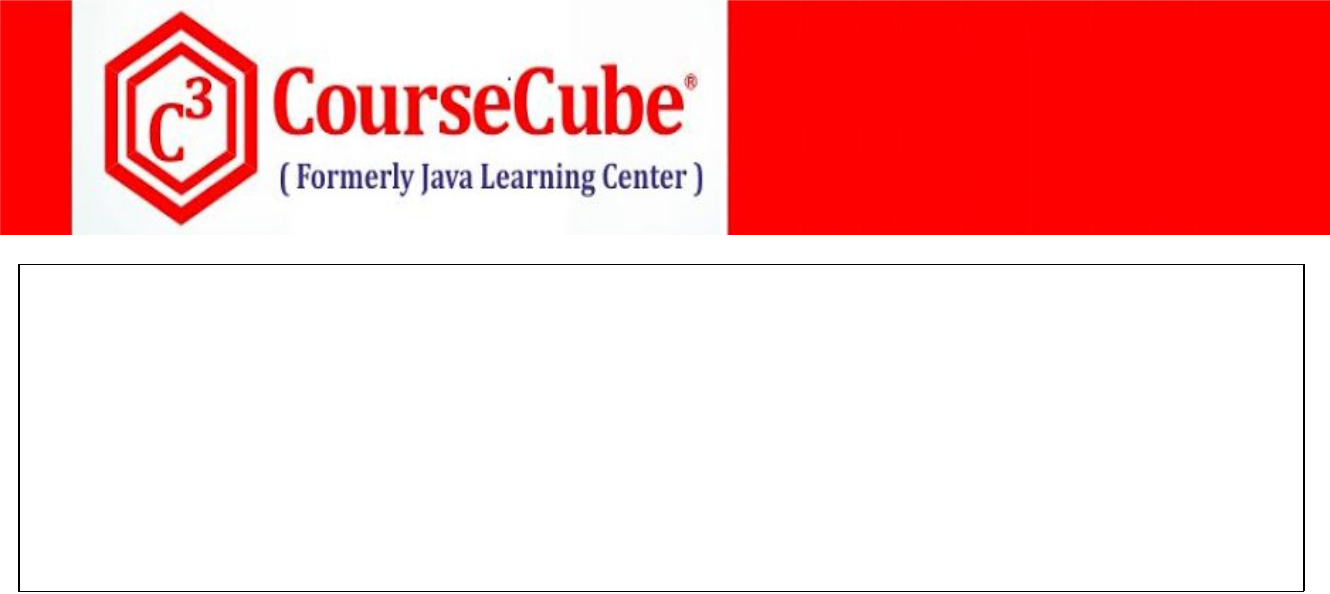
**Object returnVal = pjp.proceed();**

**return returnVal;**

**}**



**www.coursecube.com** **168** **Spring-5 Study Guide**



**@Around("execution(\* com.coursecube.spring.\*Service.\*(..)) and args(cid,cname,email,phone)")**

**public Object verifyUser4(ProceedingJoinPoint pjp, int cid, String cname, String email, long phone) throws Throwable {**

**System.out.println("\*\* SS - verifyUser4 : " + cid + "\t" + cname + "\t" + email + "\t" + phone);**

**Object returnVal = pjp.proceed();**

**return returnVal;**

**}**

**}**



**4. LogService.java**

**package com.coursecube.spring;**

**import org.aspectj.lang.ProceedingJoinPoint;**

**import org.aspectj.lang.annotation.\*;**

**import org.springframework.core.annotation.Order; import org.springframework.stereotype.Component; /\***

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

**@Component**

**@Aspect**

**@Order(value = 2)**

**public class LogService {**

**@Around(value = "execution(\* com.coursecube.spring.\*Service.\*(..)) and target(bean)", argNames = "pjp,bean")**

**public Object log(ProceedingJoinPoint pjp, Object bean) throws Throwable { System.out.println("\*\* LS- logBefore");**

**System.out.println("Target Bean Class : " + bean.getClass().getSimpleName());**

**Object returnVal = pjp.proceed();**

**System.out.println("\*\* LS- logReturning");**

**System.out.println("Return Value : " + returnVal);**

**return returnVal;**

**}**

**@AfterThrowing(pointcut = "execution(\* com.coursecube.spring.\*Service.\*(..)) and target(bean)", throwing = "ex")**

**public void logThrowing(Object bean, Exception ex) { System.out.println("\*\* LS- logThrowing");**

**System.out.println("Target Bean Class : " + bean.getClass().getSimpleName());**

**System.out.println("Exception Thrown : " + ex);**

**}**

**}**



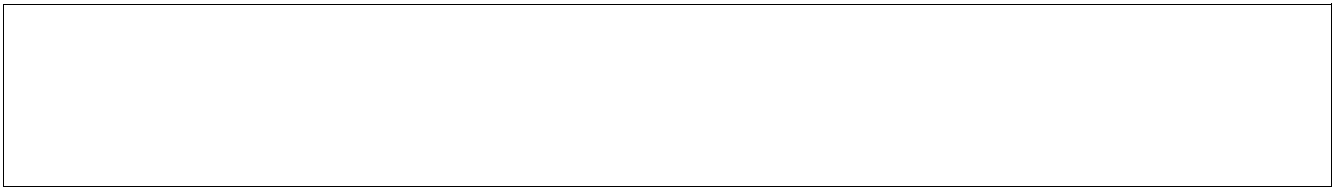
**www.coursecube.com** **169** **Spring-5 Study Guide**



**Exploring PointCut Expressions:**



**Following is the format to define AspectJ Pointcut expression**



**execution(**

**modifiers-pattern? ret-type-pattern declaring-type-pattern? name-pattern(param-pattern)**

**throws-pattern? )**

**Examples:**

1. **Apply Advice to any public method: execution(public \* \*(..))**
2. **Apply Advice to any method with a name that begins with set: execution(\* set\*(..))**
3. **Apply Advice to any method defined by the AccountService interface: execution(\* com.coursecube.service.AccountService.\*(..))**
4. **Apply Advice to any method defined in the service package: execution(\* com. coursecube.service.\*.\*(..))**
5. **Apply Advice to any method defined in the service package or one of its sub-packages: execution(\* com. coursecube.service..\*.\*(..))**
6. **Any join point (method execution only in Spring AOP) within the service package: within(com. coursecube.service.\*)**
7. **Any join point (method execution only in Spring AOP) within the service package or one of its sub-packages:**

**within(com. coursecube.service..\*)**

1. **Any join point (method execution only in Spring AOP) where the proxy implements the AccountService interface:**

**this(com. coursecube.service.AccountService)**



**www.coursecube.com** **170** **Spring-5 Study Guide**



1. **Any join point (method execution only in Spring AOP) where the target object implements the AccountService interface:**

**target(com. coursecube.service.AccountService)**

1. **Any join point (method execution only in Spring AOP) that takes a single parameter and where the argument passed at runtime is Serializable:**

**args(java.io.Serializable)**

1. **Any join point (method execution only in Spring AOP) where the target object has a @Transactional annotation:**

**@target(org.springframework.transaction.annotation.Transactional)**

1. **Any join point (method execution only in Spring AOP) where the declared type of the target object has an @Transactional annotation:**

**@within(org.springframework.transaction.annotation.Transactional)**

1. **Any join point (method execution only in Spring AOP) where the executing method has an @Transactional annotation:**

**@annotation(org.springframework.transaction.annotation.Transactional)**

1. **Any join point (method execution only in Spring AOP) which takes a single parameter, and where the runtime type of the argument passed has the @Classified annotation:**

**@args(com.coursecube.Hello)**

1. **Any join point (method execution only in Spring AOP) on a Spring bean named tradeService:**

**bean(tradeService)**

1. **Any join point (method execution only in Spring AOP) on Spring beans having names that match the wildcard expression \*Service:**

**bean(\*Service)**



**www.coursecube.com** **171** **Spring-5 Study Guide**