

AOP

Aspect Oriented Programming

Class

ExternalServices

UI Layer

Services Layer

DAO Layer

Cross Cutting Concerns

Business Class

External Service

class OrderServiceImpl

class ExternalService

{

{

placeOrder() → createTransaction()

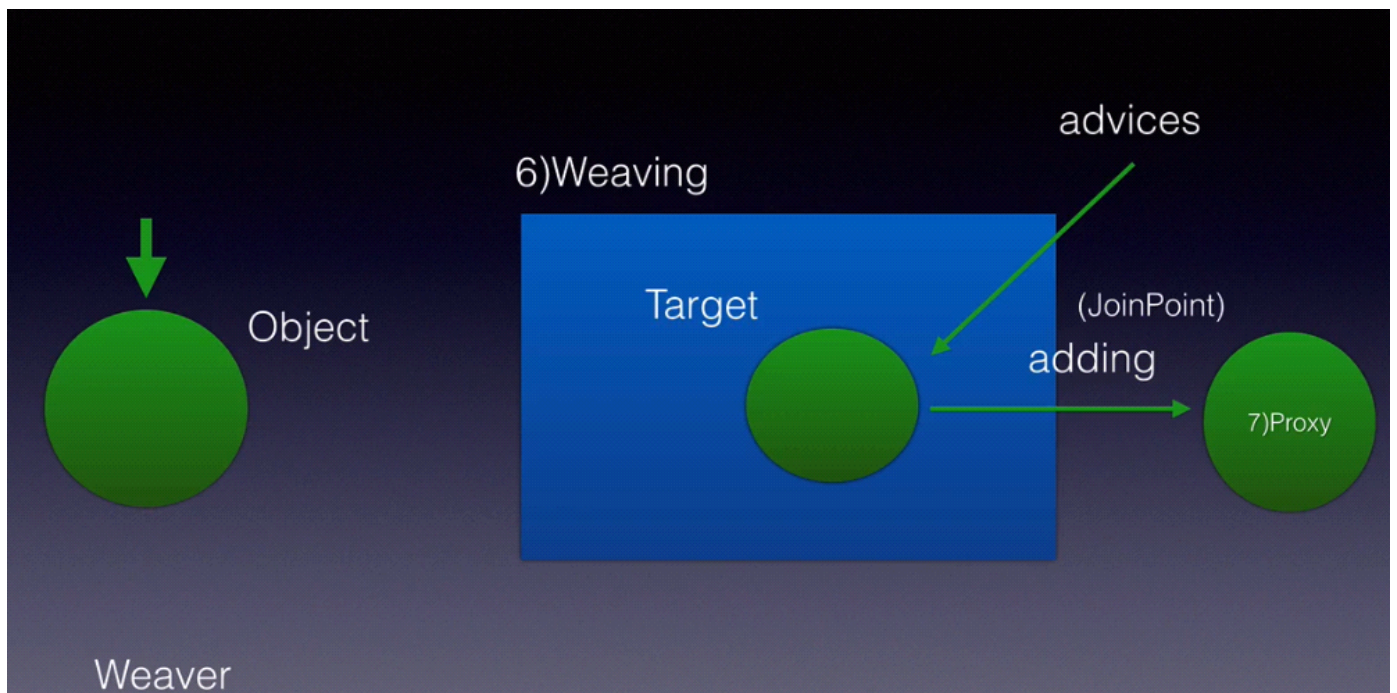
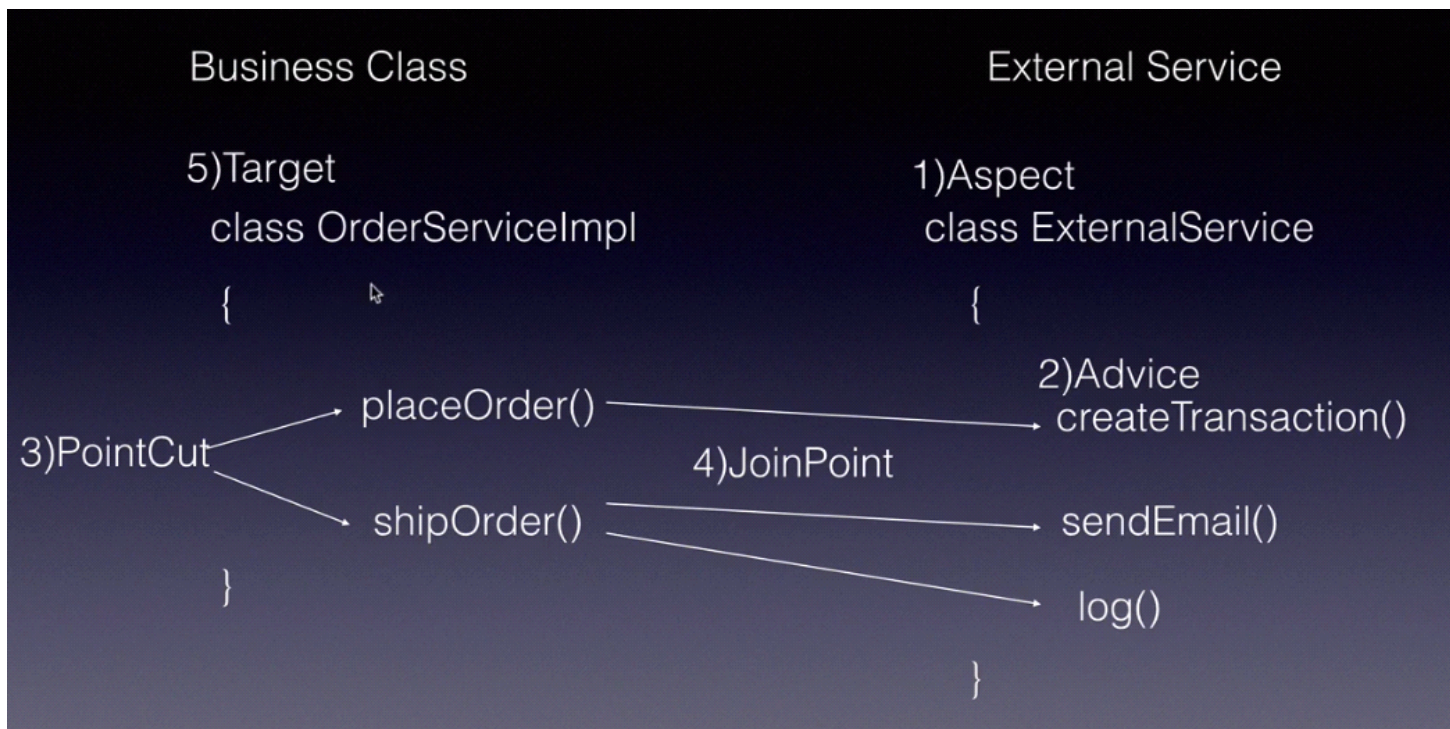
shipOrder() → sendEmail()

log()

}

}


AOP Terminology





# Pointcut

public      int      com.bharath.MyClass.multiply(int,int)  
Access Specifier   Return Type   package.class.methodName ()



The diagram illustrates the components of a pointcut expression. It shows the expression `public int com.bharath.MyClass.multiply(int,int)` with labels below it: `Access Specifier` under `public`, `Return Type` under `int`, and `package.class.methodName ()` under `com.bharath.MyClass.multiply(int,int)`. Arrows point from each label to its corresponding part in the expression.

symbols	can be used at
*	AS,RT,PACK,CLASS,MN
..	pack,current and sub (1) Any Parameter(2)

```
public void *Id()
```

```
public int *e*(..)
```

```
public int get(..)
```

```
public * *()
```

```
public void get(..)
```

```
public int *(..)
```

```
public * com.app..*.get*()
```

```
public * *(..)
```

## AOP Implementations

AspectJ

Spring AOP

JBoss AOP



# Spring AOP

AspectJ Annotation Driven

AspectJ XML Driven

Classic Spring Proxy-Based AOP

## AspectJ Annotation Driven

@Aspect

@Before

@After

@AfterReturning

@Around

@AfterThrowing

springaop

ProductService

LoginAspect

logBefore

ProductServiceImpl

multiply(int num1,int num2)

Create the spring configuration

Create and Run the test