AOP Advice Tipleri

Around Advice

- **En yaygın** kullanılan advice tipidir
- Join point'i **wrap** eder, bütün diğer advice tiplerini kapsar
- Join point execution'ını **istenildiği gibi yönetmek** mümkündür
- İstenirse asıl metot çalıştırılmayabilir, input parametreleri değiştirilerek çağrılabilir ya da farklı bir metot return değeri dönülebilir

Around Advice

```aspectj
@Aspect
public class AroundExample {
  @Around("execution(* x.y.service.*.*(..))")
  public Object doBasicProfiling(ProceedingJoinPoint pjp) throws Throwable {
    //asıl metot çağrılmadan önce...
    Object retVal = pjp.proceed();
    //asıl metot çağrıldıktan sonra...
    return retVal;
  }
}
```

Around advice metotlarında ilk parametre **ProceedingJoinPoint** olmak **zorundadır**

```aspectj
public aspect AroundExample {
    Object around() : execution(* x.y.service.*.*(..)) {
        //asıl metot çağrılmadan önce..
        Object retVal = proceed();
        //asıl metot çağrıldıktan sonra...
        return retVal;
    }
}
```

Before Advice

```
@Aspect
public class BeforeExample {
  @Before("execution(* x.y.service.*.*(..))")
  public void doAccessCheck() {
    // ...
  }
}
```

Eğer advice içinde **exception meydana** gelirse yakalanan joint point execute edilmeyecektir

```
public aspect BeforeExample {

    before() : execution(* x.y.service.*.*(..)) {
    //...
    }

}
```

After Returning Advice

```
@Aspect
public class AfterReturningExample {
  @AfterReturning(pointcut="execution(* x.y.service.*.*(..))",
returning="retVal")
  public void doAccessCheck(Object retVal) {
    // ...
  }
}
```

Dönen değeri advice içerisine parametre olarak geçmek mümkündür

```
public aspect AfterReturningExample {

    after() returning(Object retVal) :execution(* x.y.service.*.*(..)) {
    //...
    }

}
```

After Throwing Advice

```
@Aspect
public class AfterThrowingExample {

  @AfterThrowing(pointcut="execution(* x.y.service.*.*(..))",
throwing="ex")
  public void doRecoveryActions(DataAccessException ex) {
    // ...
  }
}
```

Fırlatılan exception'ı da advice içerisine Parametre olarak geçmek mümkündür

```
public aspect AfterThrowingExample {

    after() throwing(DataAccessException ex) : execution(*
x.y.service.*.*(..)) {
        //...
    }

}
```

After (finally) Advice

```
@Aspect
public class AfterFinallyExample {
@After(pointcut="execution(* x.y.service.*.*(..))")
  public void doActionsAlways() {
    // ...
  }
}
```

```
public aspect AfterFinallyExample {
    after() : execution(* x.y.service.*.*(..)) {
        //...
    }
}
```

Advice Parametreleri

- args() pointcut tanımı ile
  - Hem **eşlenecek join point'leri** sınırlanabilir
  - Hem de **metot parametreleri advice'a** input argüman olarak geçilebilir
- this, target, args, @within, @target, @args, @annotation hepsi **metot parametrelerini advice'a geçmek** için kullanılabilir

Advice Parametreleri

```
@Around("execution(List<Account> find*(..)) && args(accountHolderName)")
public Object preProcessQuery(ProceedingJoinPoint pjp,
                String accountHolderName) throws Throwable {

    String newName = "act_" + accountHolderName + "_user";

    return pjp.proceed(new Object[] { newName });

}
```

args() pointcut'daki değişken ismi ile metot parametresindeki değişken ismi eşleşmelidir. Böylece metot input argümanı hem joinpoint eşleşmesinde kullanılmakta hem de advice metoduna değer olarak geçilebilmektedir

Advice Parametreleri

```
public aspect ArgsExample {

        List around(String name) : execution(List find*(..)) && args(name) {
        String newPattern = preProcess(name);

        return proceed(newPattern);
    }

    private String preProcess(String name) {
        return name;
    }
}
```

Advice Parametreleri

```
 @Before("execution(* *.*(..)) && args(account,..)")
     public void validateAccount(Account account) {
     // ...
 }
```

```
aspect ArgsExample {

    before(Account account) : execution(* *.*(..)) && args(account,..) {
        //...
    }
}
```

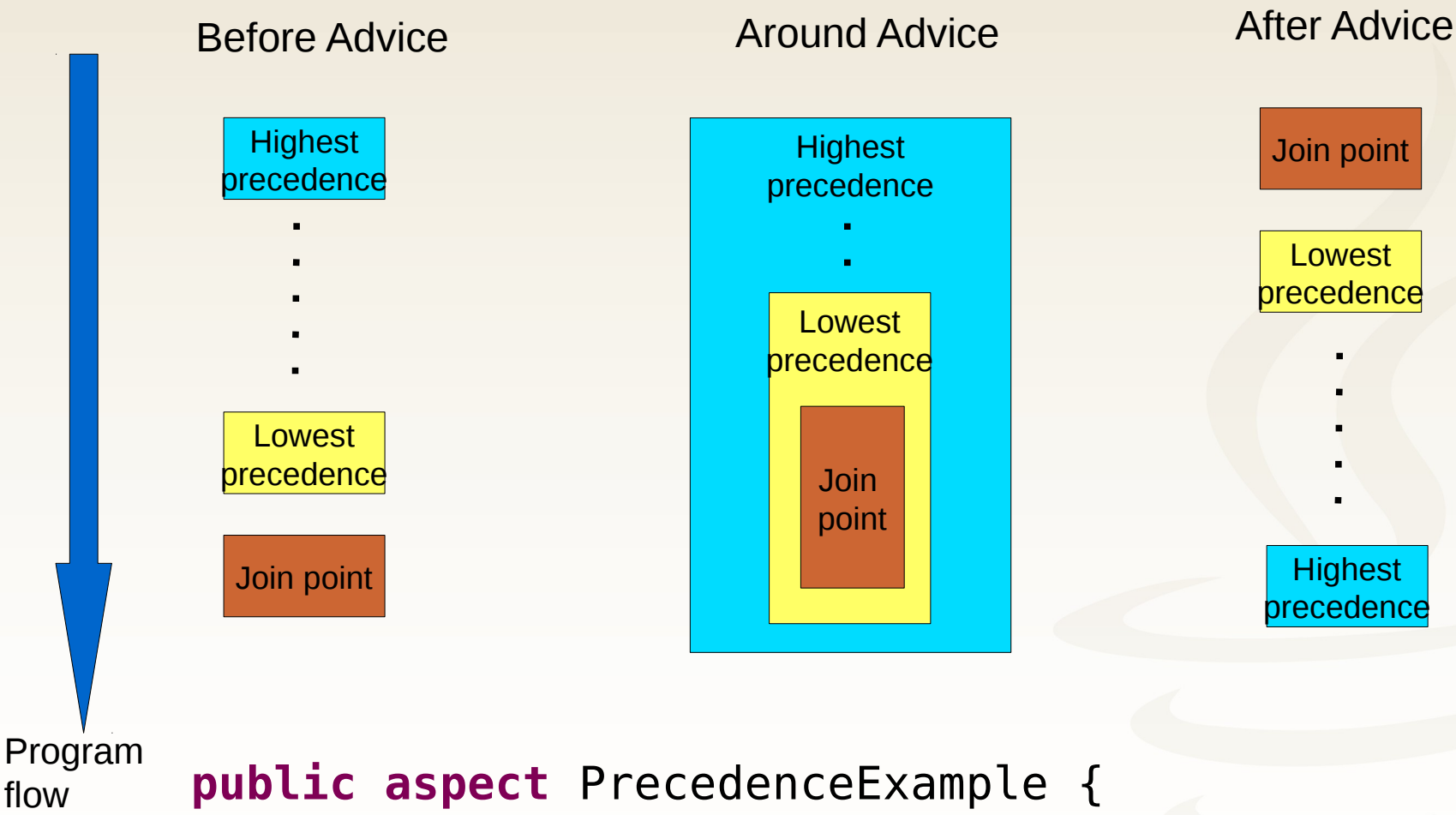
Advice Parametreleri

```
@Before(value = "execution(* *.*(..)) && target(bean) &&
@annotation(auditable)", argNames = "bean,auditable")
    public void audit(Object bean, Auditable auditable) {
AuditCode code = auditable.value();
// ...
}

@Before(value = "execution(* *.*(..)) && target(bean) &&
@annotation(auditable)", argNames = "bean,auditable")
    public void audit(JoinPoint jp, Object bean, Auditable auditable) {
AuditCode code = auditable.value();
// ...
}

@Before("execution(* *.*(..))")
    public void audit(JoinPoint jp) {
// ...
}
```

Advice Öncelik Sırası




```
public aspect PrecedenceExample {
   declare precedence: LoggingAspect,TxAspect;
}
```

İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com