AOP Pointcut İfadeleri

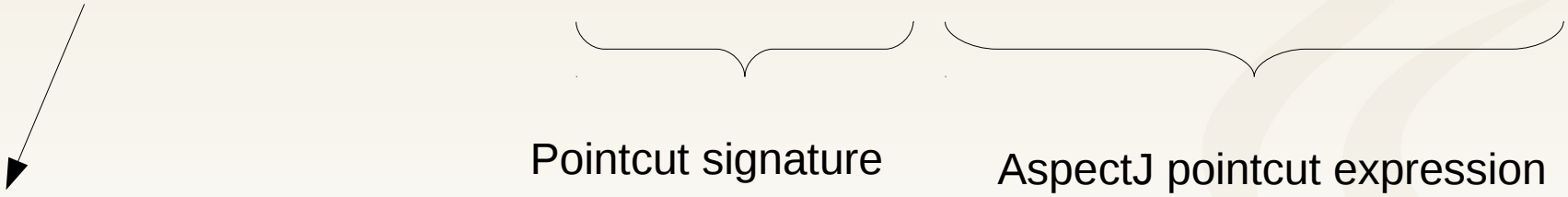
harezmi bilişim çözümleri

Pointcut İfadeleri

- Program içerisindeki **join point'lerin tespit edilip yakalanmaları**nı sağlar
- **Regular expression** benzeri ifadelerdir
- İki türlüdür
  - **Anonim**
    - Bir defa kullanılabilir
  - **İsimlendirilmiş** (named)
    - Başka yerlerden de erişilebilirler

Pointcut Tanımları

```
access_modifier pointcut name(args): pointcut_expression;
```

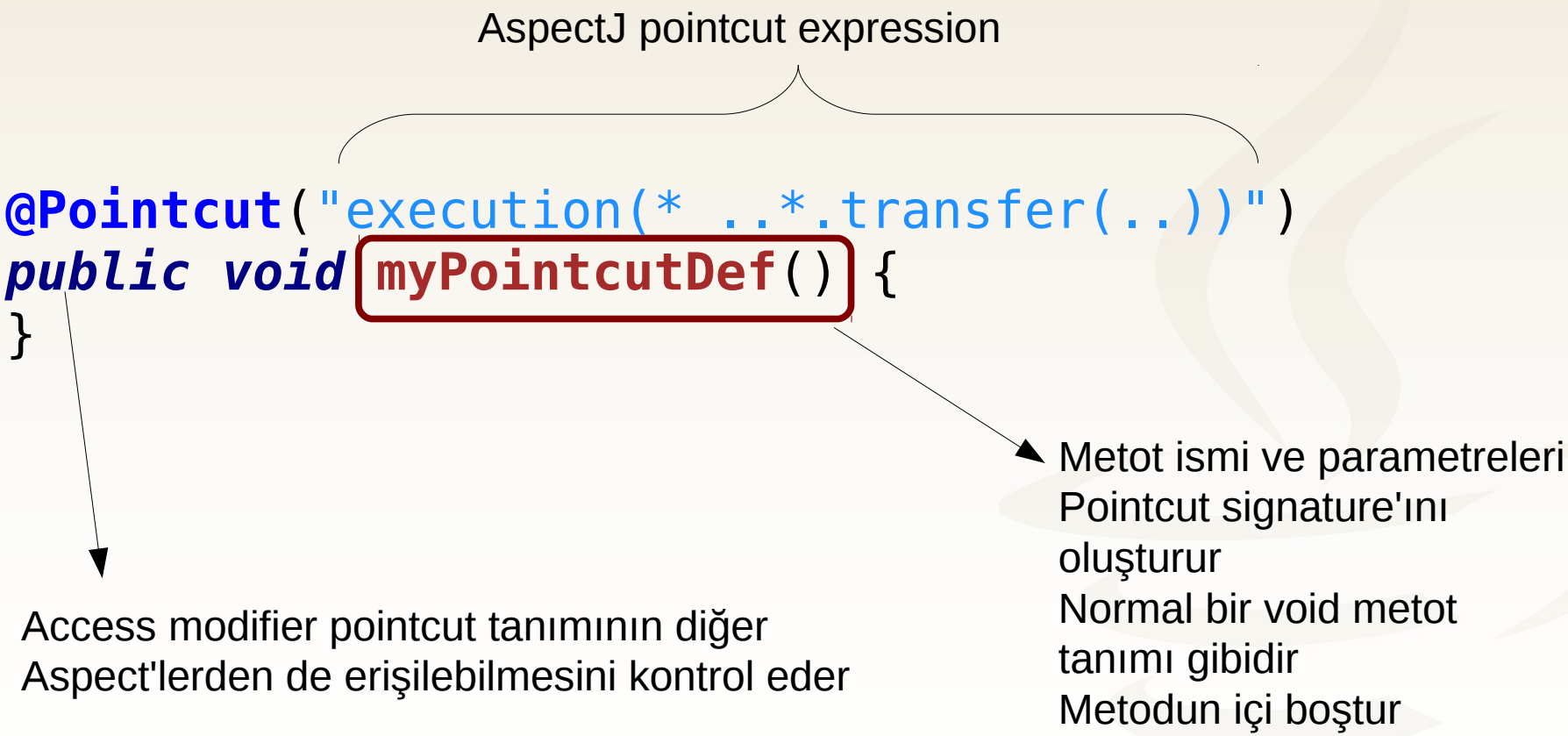


Pointcut signature

AspectJ pointcut expression

Public tanımlanan Pointcut'lar diğer aspect'lerden erişilebilir

Örnek Bir Pointcut Tanımı



Metot Execution Pointcut Tanımları

- **execution/call**
  - En yaygın kullanılan pointcut ifadesidir
  - **metot join point**leri yakalar
  - execution(* *(..))
  - execution(* set*(..))
  - execution(* com.xyz.service.*.*(..))
  - execution(* com.xyz.service..*.*(..))
  - call(* com.xyz.service.AccountService.*(..))

Execution/Call Pointcut Tanımının Yapısı

- *execution*(*modifiers-pattern?* ***return-type-pattern*** *declaring-type-pattern?.***name-pattern(param-pattern)** *throws-pattern?*)

Sadece AspectJ için anlamlıdır, Spring AOP için anlamı yoktur

Spring AOP proxy tabanlı bir framework olduğu için sadece public metotlarda devreye girebilir

Turuncu renkteki kısımlar opsiyoneldir

@annotation Pointcut Tanımı

- **@annotation**
  - Join point'in eşleneceği **metodun belirtilen annotasyona sahip olması** beklenir
  - @annotation(org.springframework.transaction.annotation.Transactional)

Field Access Pointcut Tanımı

- **get/set**
  - Field değerlerine erişim ve yeni değer set etmeyi kapsar
  - set(* *..Bar.*)
  - get(int *..Bar.*)
- *set*(*modifiers-pattern?* ***field-type-pattern*** *declaring-type-pattern?.****name-pattern***)

Yapısal Pointcut Tanımları

- Kaynak **kodun belirli bir bölümü**nü ele alan pointcut tanımlarıdır
  - Sınıf, aspect veya metot düzeyinde olabilir
- **within** ve **withincode** şeklinde iki türlüdür
  - *within**(TypePattern)***
  - *withincode**(MethodSignature)***
  - *withincode**(ConstructorSignature)***
    - Metot veya constructor içerisini yakalar
    - Metot içindeki **lokal sınıfları** da kapsar

Yapısal Pointcut Tanımları

- Genellikle **başka bir pointcut ifadesi ile birlikte** kullanılırlar
  - call(* java.io.PrintStream.print*(..)) && ! within(com.xyz.TraceAspect)
- Spring AOP **sadece within**'i destekler ve sadece within ile yakalanan tipin içindeki public metot invokasyonlarını kapsar

Yapısal Pointcut Tanımları

- **within**
  - within(com.xyz.service.*)
  - within(com.xyz.service..*)
- **@within**
  - Join pointlerin tanımlı oldukları **tipte belirtilen annotasyon**un olması istenir
  - @within(org.springframework.transaction.annotation.Transactional)

Execution Object Pointcut Tanımları

- Execution anındaki **nesnelerin tipleri**ne göre yakalama yapar
- Yakalanan joint point ile ilgili **context bilgisini** de transfer etmeye yararlar
- İki türlüdür
  - *this**(Type)**/ this**(ObjectIdentifier)***
  - *target**(Type)**/target**(ObjectIdentifier)***

Execution Object Pointcut Tanımları

- target() genellikle **metot call join point** ile birlikte kullanılır ve hedef nesneyi yakalar
- this() ise çağrıyı yapan nesneleri döndürür
- target() Spring AOP'da **target bean**'i döner
- this() Spring AOP'da **proxy nesne**yi döner

Execution Object Pointcut Tanımları

- **this**
  - Join point eşleşmesi için current nesnenin belirli bir tipte olmasını ister
  - this(com.xyz.service.AccountService)
- **target**
  - Join point eşleşmesi için hedef nesnenin belirli bir tipte olmasını ister
  - target(com.xyz.service.AccountService)

Execution Object Pointcut Tanımları

- **@this**
  - Join point eşleşmesi için current nesnenin sınıfında belirtilen annotasyonun olması gerekir
  - @this(org.springframework.transaction.annotation.Transactional)
- **@target**
  - Join point eşlemesi için **hedef nesnenin sınıfında belirtilen annotasyon**un olması gerekir
  - @target(org.springframework.transaction.annotation.Transactional)

Argüman Pointcut Tanımları

- Join point'in argüman tipine bakarak yakalar
- Metot ve constructor'lar için input, argümanların tipidir
- Exception catch joint point için exception tipidir
- Field set join point için ise yeni değerin tipidir
  - *args**(TypePattern)**,*
  - *args**(ObjectIdentifier)***

Argüman Pointcut Tanımları

- **args**
  - args(java.io.Serializable)
    - *Metot parametresinin runtime daki tipine bakar*
  - execution(* *(java.io.Serializable))
    - *Metot singature'undaki parametre tipine bakar*
- **@args**
  - **Argümanların runtime tiplerinde belirtilen annotasyon**un olması gerekir
  - @args(com.xyz.security.Classified)

Spring'e Özel Pointcut Tanımı: bean

- **bean**
  - Spring managed bean isimlerine göre eşleme yapılır
    - bean(beanIdveyaName)
    - * wildcard kullanılabilir
  - **AspectJ'de mevcut değil**dir
  - bean(petClinicService)
  - bean(*Service)

Pointcut'ların Paylaşılması

- Poincut tanımları başka aspectler içerisinden isimleri ile erişilebilir ve yeniden kullanılabilir

```
@Aspect
public class SystemArchitecture {
  @Pointcut("within(x.y.web..*)")
  public void inWebLayer() {}

  @Pointcut("within(x.y.service..*)")
  public void inServiceLayer() {}

  @Pointcut("within(x.y.dao..*)")
  public void inDataAccessLayer() {}

  @Pointcut("execution(* x.y.service.*.*(..))")
  public void businessService() {}

  @Pointcut("execution(* x.y.dao.*.*(..))")
  public void dataAccessOperation() {}
}
```

Pointcut'ların Paylaşılması

```
@Aspect
public class SecurityAspect {

@Around("x.y.SystemArchitecture.dataAccessOperation()")
  public Object
doAccessCheck(ProceedingJoinPoint pjp) throws
Throwable {
    // ...
  }

}
```

Pointcut Operatörleri

- ***** : Nokta hariç herhangi bir sayıda karakteri yakalar
- **..** : Noktalar dahil herhangi bir sayıda karakteri yakalar
- **+** : Belirtilen tipin alt sınıflarını veya alt interface'lerini yakalar
- **!** : Belirtilen pointcut'ın dışındaki joinpoint'leri yakalar
- **||** : iki pointcut'ı birleştirir, herhangi biri ile eşleşen joinpoint'leri yakalar
- **&&** : iki pointcut'ı birleştirir, her ikisi ile de eşleşen joinpoint'leri yakalar

Bileşke Pointcut Tanımları

```
@Pointcut("execution(public * *(..))")
private void anyPublicOperation() {}

@Pointcut("within(x.y.service..*)")
private void inService() {}

@Pointcut("anyPublicOperation() && inService()")
public void publicServiceOperations() {}
```

Pointcut tanımlarına isimleri ile erişmek mümkündür, bu diğer aspectler içinden erişim için de geçerlidir

Pointcut ifadeleri &&, || ile **birleştirilebilir**
Daha kompleks pointcut tanımlarını, basit pointcut'ları bir araya getirerek oluşturmayı sağlar

İletişim

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