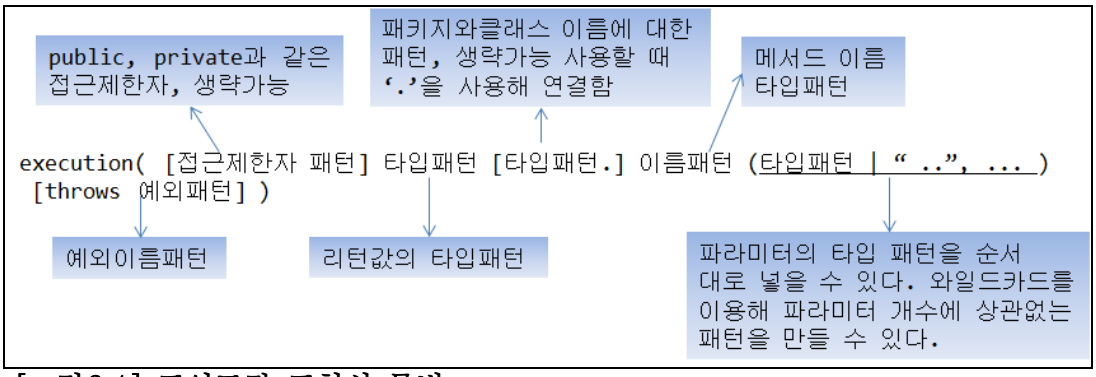
6.1.1 AspectJ 포인트컷 표현식

AspectJ 포인트컷 표현식은 포인트컷 지시자를 이용하여 작성한다.
포인트컷 지시자 중에서 가장 대표적으로 사용되는 것은 execution()이다.
execution() 지시자를 사용한 포인트컷 표현식의 문법구조는 기본적으로 다음과 같다. [] 괄호는 옵션항목이기 때문에 생략이 가능하다는 의미이며, |는 OR 조건이다.



[그림6.4] 포인트컷 표현식 문법

예를 들어 리플렉션으로 Target 클래스의 minus()라는 메서드의 풀시그너처(full signature)를 가져와 출력해 보자

```
package springworkbook.aop.pointcut;

public class Target {

   public int minus(int num1, int num2) throws RuntimeException {

      return 0;

   }

}
```

[코드6.10] Target 클래스

```
System.out.println(Target.class.getMethod("minus", int.class,
    int.class));
```

출력된 결과는 다음과 같다.

```
public int springworkbook.pointcut.Target.minus(int, int)
      throws java.lang.RuntimeException;
```

■ 접근제한자 패턴 : public
접근제한자이다. public, protected, private 등이 올 수 있다. 포인트컷 표현식에서는 생략할 수 있다. 생략이 가능하다는 건 이 항목에 대해서는 조건을 부여하지 않는다는 의미이다.

■ 타입 패턴 : int
리턴값의 타입을 나태는 패턴이다. 포인트컷의 표현식에서 리턴값의 타입패턴은 필수항목이다. 따라서 반드시 하나의 타입을 지정해야 한다. 또는 * 를 써서 모든 타입을 다 선택하겠다고 해도 된다. 생략은 불가능하다.

■ 타입 패턴 : springworkbook.aop.pointcut.Target
패키지와 타입이름을 포함한 클래스의 타입 패턴이다. 생략가능하다. 생략하면 모든 타입을 다 허용하겠다는 뜻이다. 뒤에 이어나오는 메서드 이름패턴과 '.' 으로 연결되기 때문에 작성할 때 잘 구분해야 한다. 패키지 이름과 클래스 또는 인터페이스 이름에 * 를 사용할 수 있다.
또 '..' 를 사용하면 여러 개의 패키지를 선택할 수 있다.

■ 이름 패턴 : minus
메서드 이름 패턴이다. 필수항목이기 때문에 반드시 적어야 한다. 모든 메서드를 다 선택하겠다면 * 를 넣으면 된다.

■ 파라미터 타입 패턴 : (int, int)
메서드 파라미터 타입 패턴이다. 메서드 파라미터의 타입 패턴이다. 메서드 파라미터의 타입을 '.' 로 구분하면서 순서대로 적으면 된다. 파라미터가 없는 메서드를 지정하고 싶다면 ()로 적는다. 파라미터 타입과 개수에 상관없이 모두 다 허용하는 패턴으로 만들려면 '..'을 넣으면 된다. '...' 을 이용해서 뒷부분의 파라미터 조건만 생략할 수도 있다. 필수항목이므로 반드시 넣어야 한다.

■ 예외이름 패턴 : throws java.lang.RuntimeException
예외 이름에 대한 타입 패턴이다. 생략 가능하다.

```
package springworkbook.aop.pointcut;

public class Target implements TargetInterface {
	public void hello() { }

	public void hello(String a) { }

	public int minus(int a, int b) throws RuntimeException {
		return 0;
	}

	public int plus(int a, int b) {
		return 0;
	}

	public void method() { }
}
```

[코드6.11] AspectJ 표현식 테스트를 위한 Target.java

```
package springworkbook.aop.pointcut;
public class Bean {
	public void method() throw RuntimeException {
```

```
        }
}
```

[코드6.12] AspectJ 표현식 테스트를 위한 Bean.java

	포인트컷 표현식	Target					Bean
		hello()	hello(String)	plus(int, int)	minus(int, int)	method()	method()
1	execution(* hello(..))	O	O				
2	execution(* hello())	O					
3	execution(* hello(String))		O				
4	execution(* meth*(..))					O	O
5	execution(* *(int,int))			O	O		
6	execution(* *())	O				O	O
7	execution(* springworkbook.aop.pointcut.Target.*(..))	O	O	O	O	O	
8	execution(* springworkbook.aop.pointcut.*.*(..))	O	O	O	O	O	O
9	execution(* springworkbook.aop.pointcut..*.*(..))	O	O	O	O	O	O
10	execution(* springworkbook..*.*(..))	O	O	O	O	O	O
11	execution(* com..*.*(..))						
12	execution(* *..Target.*(..))	O	O	O	O	O	
13	execution(* *..Tar*.*(..))	O	O	O	O	O	
14	execution(* *..*get.*(..))	O	O	O	O	O	
15	execution(* * ..B*.*(..))						O
16	execution(* * ..TargetInterface.*(..))	O	O	O	O		
17	execution(* *(..) throws Runtime*)				O		O
18	execution(int *(..))			O	O		
19	execution(void *(..))	O	O			O	O

(1) execution(* hello(..))
1번은 hello라는 이름을 가진 메서드를 선정하는 것이다. 파라미터는 모든 종류를 다 허용한다.

(2) execution(* hello())
2번은 파라미터 패턴이 ()로 되어 있으니 hello 메서드 중에서 파라미터가 없는 것만 선

택한다.

(3) execution(* hello(String))
3번은 파라미터의 개수가 한 개이며 String 타입인 것을 찾는다.

(4) execution(* meth*(..))
4번은 메서드 이름에 와일드카드를 사용하는 경우이다. 이름의 시작과 끝에 모두 * 를 적용할 수 있다. meth로 시작하는 모든 클래스의 메서드가 다 허용된다.

(5) execution(* *(int,int))
5번은 파라미터 타입만으로 선정하는 방법이다. 메서드 이름에는 상관없이 두개의 정수 파라미터를 가진 메서드가 선정된다.

(6) execution(* *())
6번은 파라미터가 없는 모든 메서드를 선택하는 것이다.

7,8,9번은 패키지와 클래스 타입 패턴을 사용하는 방법이다.
(7) execution(* springworkbook.aop.pointcut.Target.*(..))
7번은 클래스를 직접지정한 것이다. 따라서 Bean 클래스의 메서드는 선정되지 않았다.

(8) execution(* springworkbook.aop.pointcut.*.*(..))
8번은 클래스 이름에 와일드카드를 사용한 것이다. 따라서 springworkbook.aop.pointcut 패키지의 모든 클래스에 적용된다. 하지만 서브패키지의 클래스는 포함되지 않는다.

(9) execution(* springworkbook.aop.pointcut..*.*(..))
9번은 '..' 를 사용해서 서브패키지의 모든 클래스까지 포함시키는 식이다.

(10) execution(* springworkbook..*.*(..))
10번은 특정 패키지의 모든 서브패키지를 다 지정하는 방법도 가능하다. springworkbook으로 시작하는 모든 패키지의 모든 클래스를 다 선정한다.

(11) execution(* com..*.*(..))
11번은 com으로 시작하는 패키지를 선정하는 식이기 때문에 어떤 메서드로 적용되지 않는다.

(12) execution(* *..Target.*(..))
12번은 패키지에는 상관없이 Target이라는 이름의 모든 클래스에 적용하는 식이다. 다른 패키지에 같은 이름의 클래스가 있어도 적용된다는 점에 주의해야 한다.

(13) execution(* *..Tar*.*(..))
(14) execution(* *..*get.*(..))
(15) execution(* * ..B*.*(..))
13,14,15번은 클래스 이름에 와일드 카드를 부여한 것이다. 모든 패키지이름, 클래스 이름, 메서드 이름에는 와일드 카드를 사용할 수 있다.

(16) execution(* * ..TargetInterface.*(..))
16번은 Target 클래스가 아니라, Target 클래스가 구현한 TargetInterface 인터페이스를 선정 조건으로 한 것이다. 이렇게 인터페이스를 사용하면 Target 클래스의 메서드 중에서 이 인터페이스를 구현한 메서드에만 포인트컷이 적용된다. 따라서 인터페이스에는 정의되지 않는 method() 메서드는 제외된다.

(17) execution(* *(..) throws Runtime*)
17번은 메서드와 클래스와는 상관없이 예외 선언만 확인해서 메서드를 선정하는 포인트컷이다. Runtime으로 시작하는 어떤 예외라도 던진다면 이 포인트컷의 조건을 만족할 것이다.

(18) execution(int *(..))
(19) execution(void *(..))
18,19번은 리턴타입을 이용해 메서드를 선정하는 예이다.