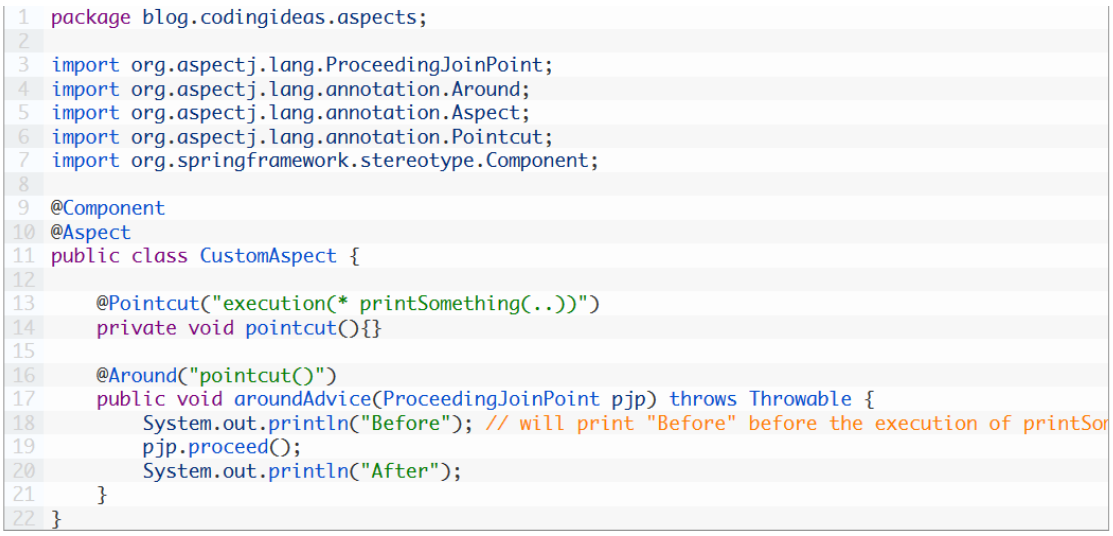
What is a ProceedingJoinPoint?

Around advice runs "around" a matched method execution. It has the opportunity to do work both before and after the method executes, and to determine when, how, and even if, the method actually gets to execute at all. Around advice is often used if you need to share state before and after a method execution in a thread-safe manner (starting and stopping a timer for example). Always use the least powerful form of advice that meets your requirements; don’t use around advice if simple before advice would do.

Around advice is declared using the aop:around element. The first parameter of the advice method must be of type ProceedingJoinPoint. Within the body of the advice, calling proceed() on the ProceedingJoinPoint causes the underlying method to execute. The proceed method may also be calling passing in an Object[] - the values in the array will be used as the arguments to the method execution when it proceeds. See the section called “Around advice” for notes on calling proceed with an Object[].



```java
package blog.codingideas.aspects;

import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Pointcut;
import org.springframework.stereotype.Component;

@Component
@Aspect
public class CustomAspect {

    @Pointcut("execution(* printSomething(..))")
    private void pointcut(){}

    @Around("pointcut()")
    public void aroundAdvice(ProceedingJoinPoint pjp) throws Throwable {
        System.out.println("Before"); // will print "Before" before the execution of printSo
        pjp.proceed();
        System.out.println("After");
    }
}
```