Advice Deep Dive

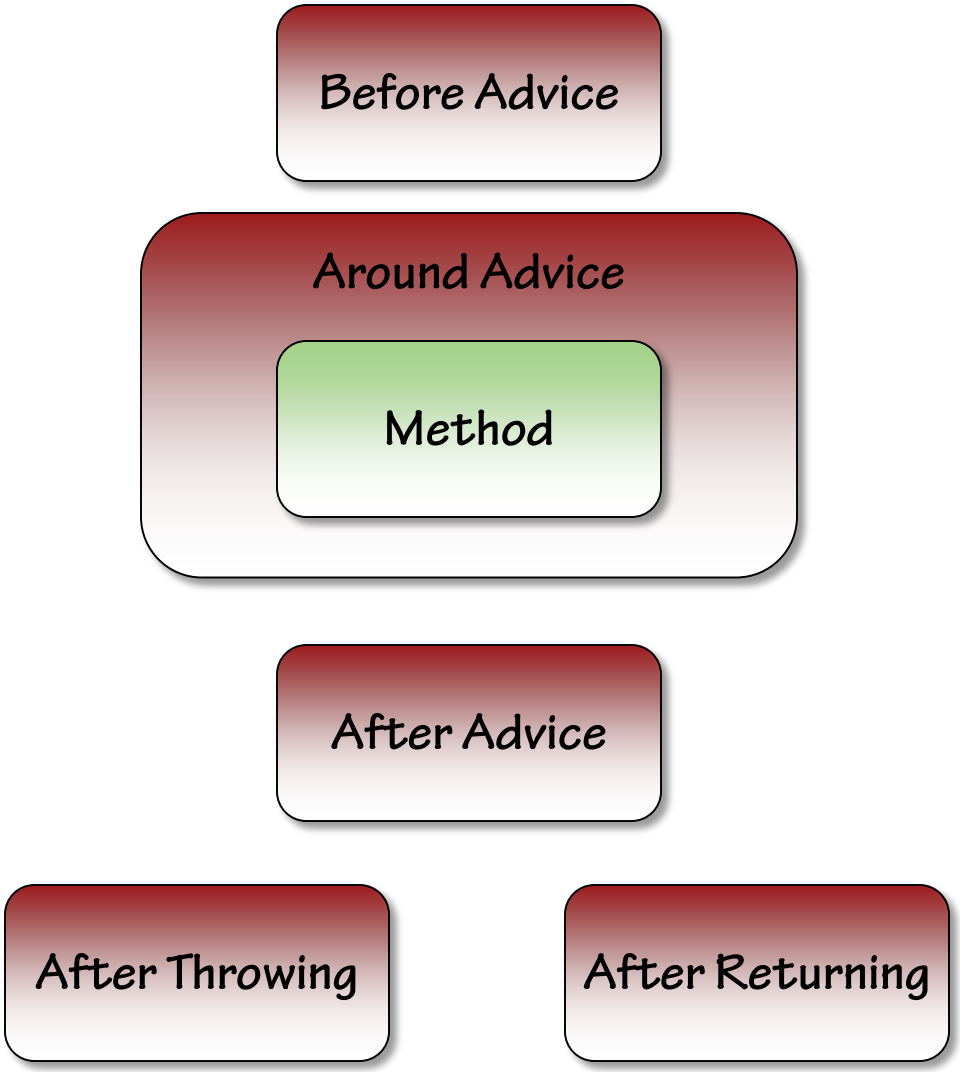
Eberhard Wolff
http://ewolff.com
eberhard.wolff@gmail.com

pluralsight
hardcore developer training

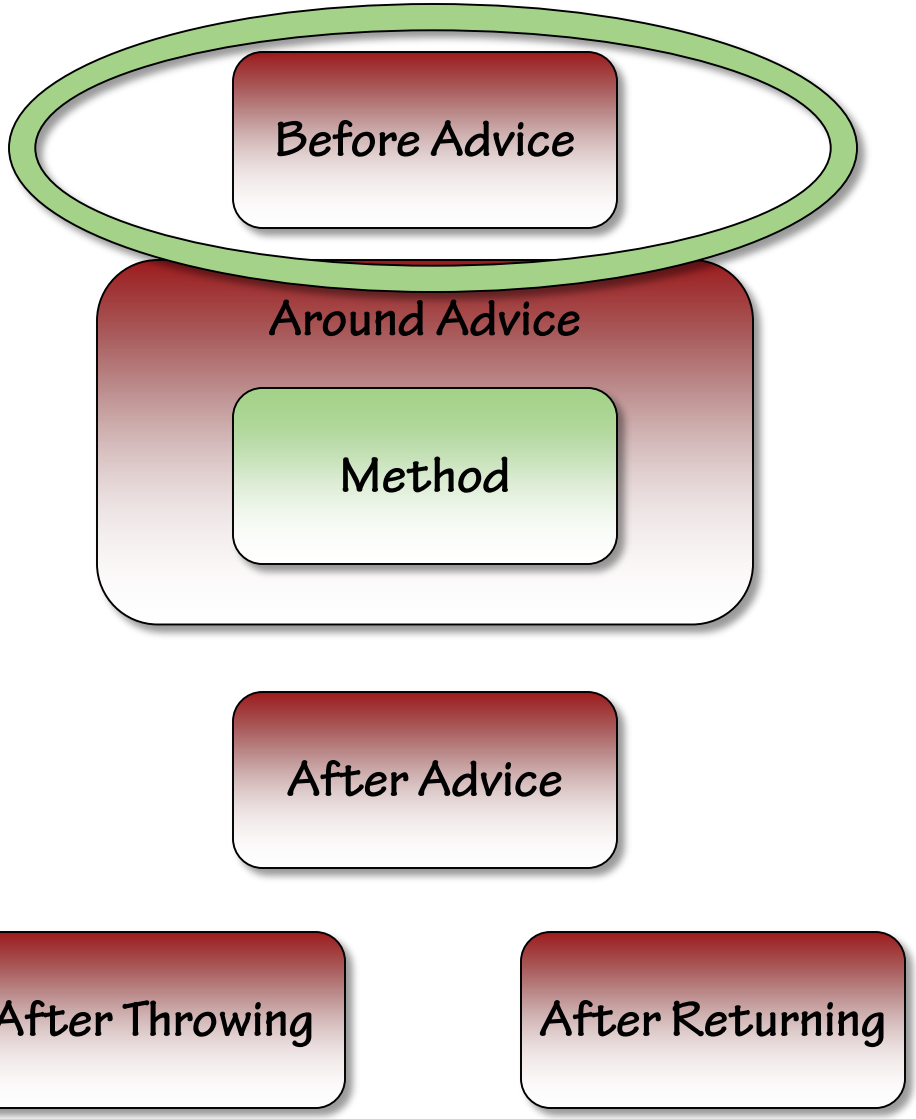
Advice Deep Dive

- So far: only before advice
- There is much more
- More possibilities

Types of Advices



Types of Advices



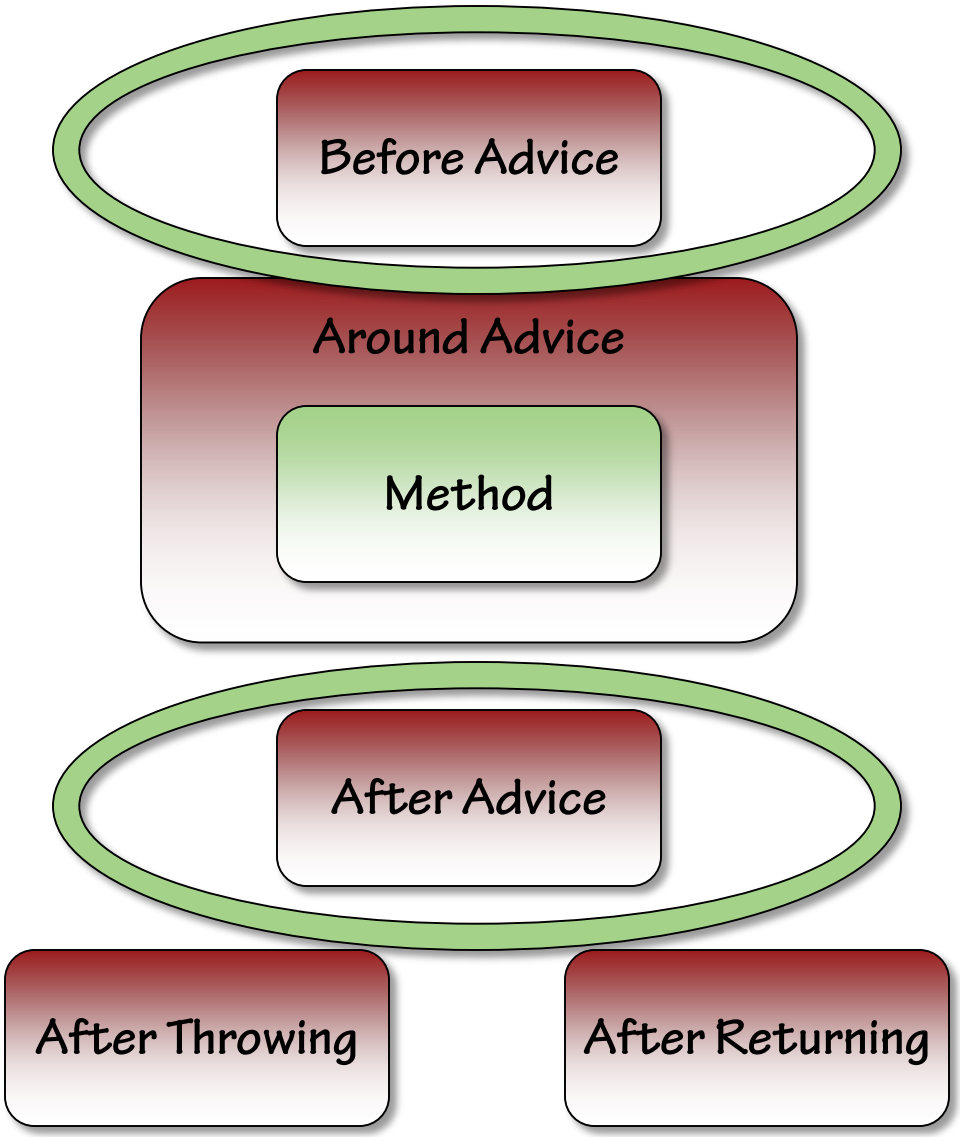
Before Advice

- Executed before the method

- Exception prevents method to be executed
- Exception is propagated to the caller

```
@Before(
  "execution(void doSomething())"
)
public void entering() {
  logger.trace("entering method");
}
```

Types of Advices



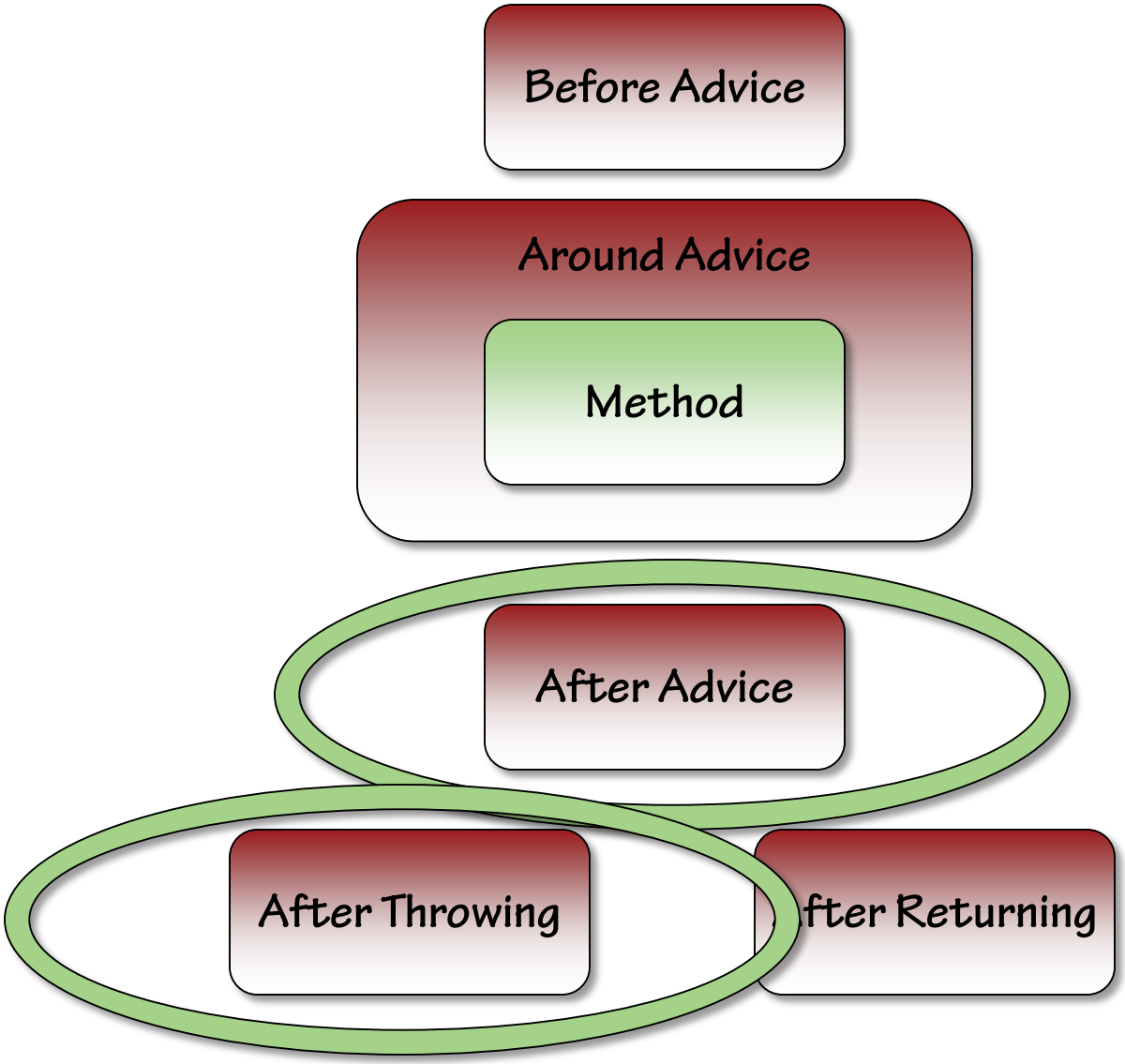
After Advice

- Executed after the method is executed

- Exception could have been thrown…
- …or method could have been executed successfully

```
@After(
 "execution(* *(..))"
)
public void exiting(JoinPoint joinPoint) {
  logger.trace("exiting " + joinPoint.getSignature());
}
```

Types of Advices



After Throwing

- Executed if method threw an exception
- Exception will be propagated to the caller

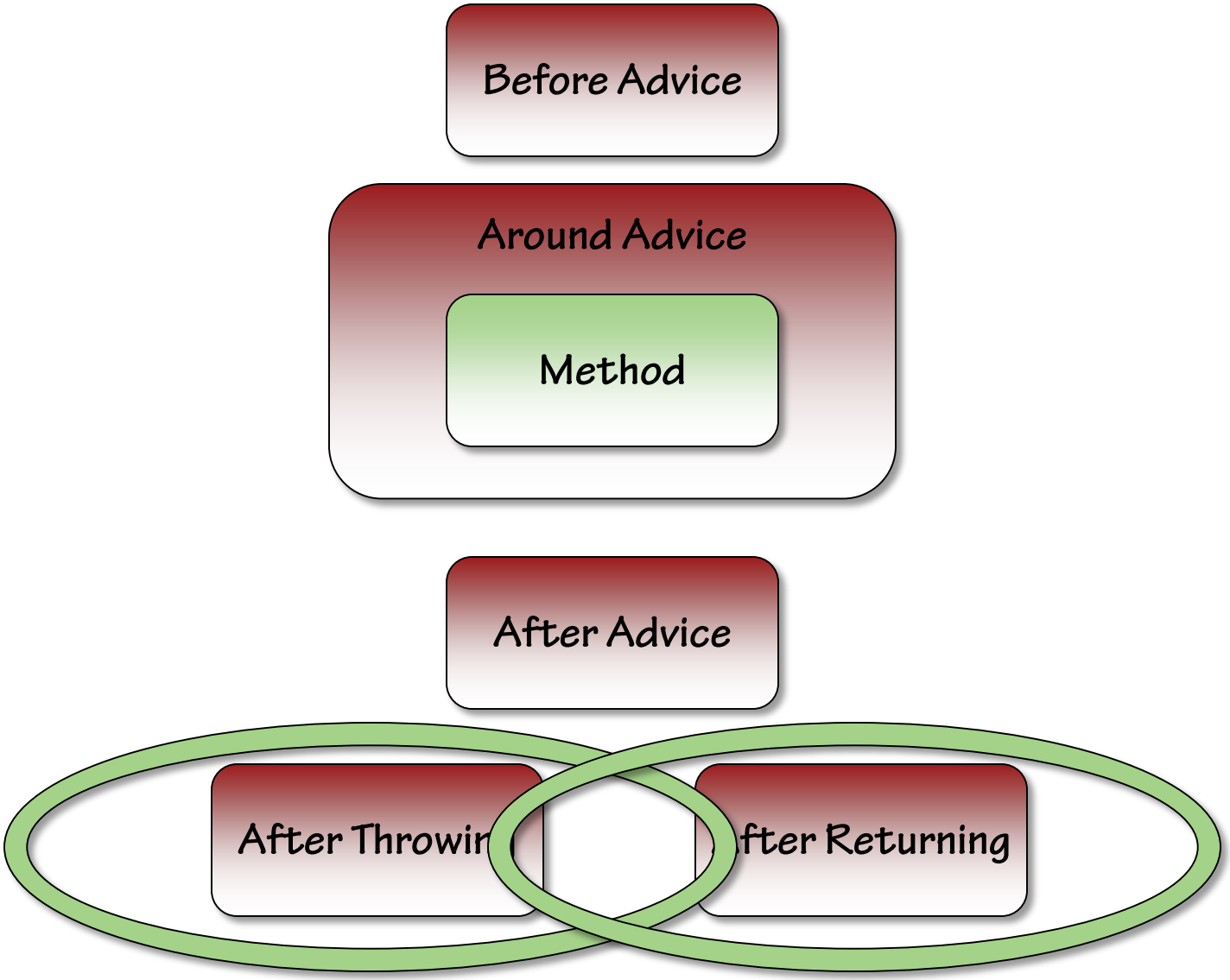
```
@AfterThrowing(pointcut =
  "execution(* *(..))")
public void logException() {
  logger.error("Exception");
}
```

After Throwing

- Thrown exception can be accessed
- Type safe i.e. method only executed if a RuntimeException is thrown

```
@AfterThrowing(pointcut =
  "execution(* *(..))",
  throwing = "ex")
public void logException(RuntimeException ex) {
  logger.error("Exception ", ex);
}
```

Types of Advices

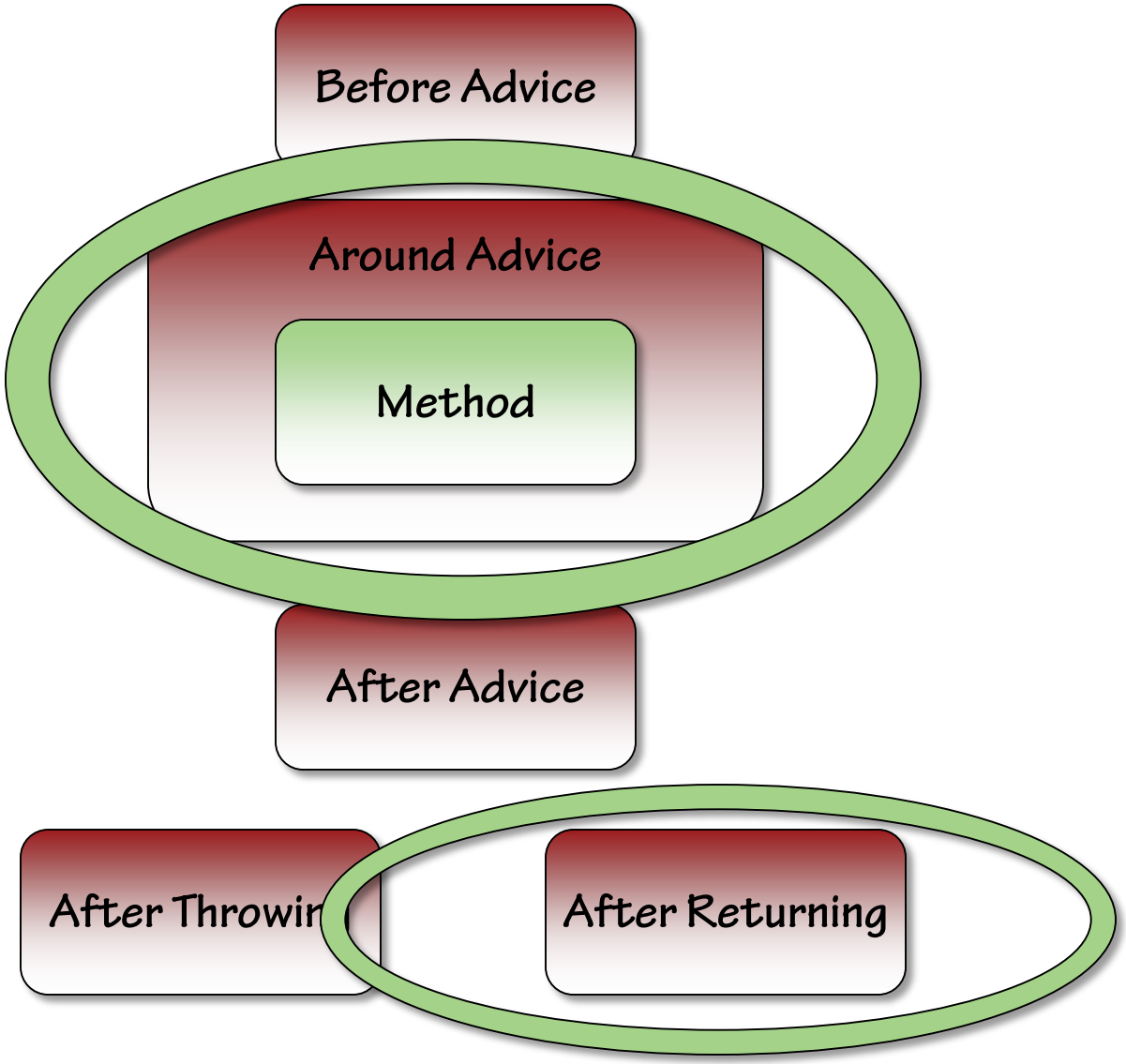


After Returning

- Executed if the method returned successfully
- Can access the result
- Type safe i.e. only called if a String is returned

```
@AfterReturning(pointcut =
"execution(* *(..))",
 returning = "string"
)
public void logResult(String string) {
  logger.trace("result "+string);
}
```

Types of Advices



Around Advice

- Wraps around the method

- Can prevent the original method from being called
- …without throwing an exception like the before advice
- Only advice that can catch exceptions
- Only advice that can modify return value

- Current method call is passed to the Advice
- ProceedingJoinPoint
- Can be executed or skipped

Around Advice Example

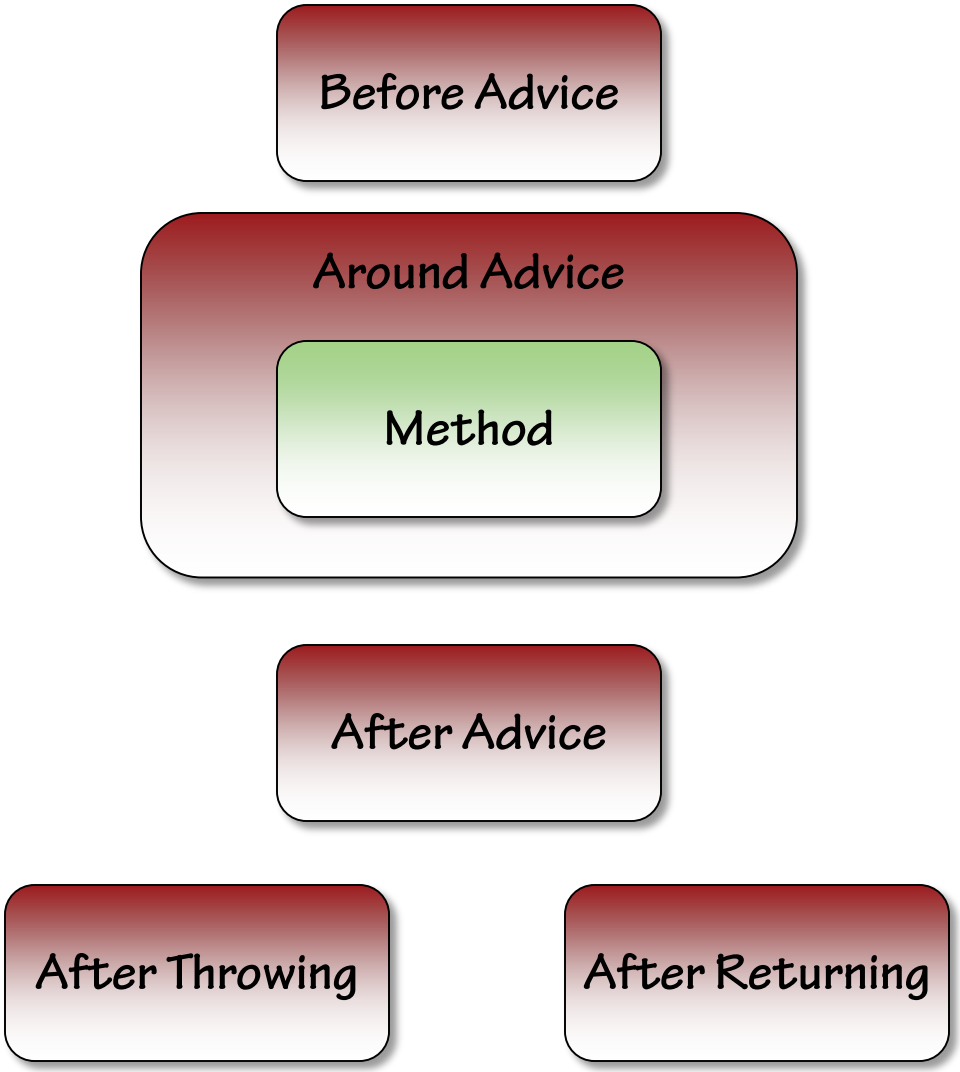
```
@Around(
 "execution(* *(..))"
)
public Object trace(ProceedingJoinPoint proceedingJP )
  throws Throwable {
  String methodInformation =
    proceedingJP.getStaticPart().getSignature().toString();
  logger.trace("Entering "+methodInformation);
  try {
    return proceedingJP.proceed();
  } catch (Throwable ex) {
    logger.error("Exception in "+methodInformation, ex);
    throw ex;
  } finally {
    logger.trace("Exiting "+methodInformation);
  }
}
```

Around Advice

- Most powerful advice
- i.e. can be used instead of Before and After
- Around is powerful but also complex

- Should use the appropriate advice

Summary


Before Advice
Around Advice
Method
After Advice
After Throwing
After Returning