SPRING FRAMEWORK

SIMPLE ASPECTS

APRIL, 2015

ADD DEPENDENCIES

DIY

```
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjweaver</artifactId>
    <version>${aspectj.version}</version>
</dependency>
```

ADD DEPENDENCIES

```
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjweaver</artifactId>
    <version>${aspectj.version}</version>
</dependency>


<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjrt</artifactId>
    <version>${aspectj.version}</version>
</dependency>
```

AUTOPROXY CONFIG

DIY

```
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/
                                    spring-beans-3.2.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/
                                    spring-aop-3.2.xsd">

        <aop:aspectj-autoproxy />

</beans>
```

```
@Configuration
@EnableAspectJAutoProxy
```

ASPECT

DIY

```
@Aspect
public class LoggingAspect {

}
```

```
<bean id="logaspect"
        class="... .LoggingAspect" />
```

or

```
@Component
```

POINTCUT

DIY

```
@Pointcut("execution(* *.logEvent(..))")
private void allLogEventMethods() {}
```

POINTCUT

DIY

```
@Pointcut("execution(* *.logEvent(..))")
private void allLogEventMethods() {}
```

```
@Pointcut("allLogEventMethods() &&
                within(*.*File*Logger)")
private void logEventInsideFileLoggers() {}
```

ADVICE BEFORE

DIY

```
@Before("allLogEventMethods()")
public void logBefore(JoinPoint joinPoint) {

    LOG.info("BEFORE : " +
        joinPoint.getTarget()
            .getClass().getSimpleName()
        + " " +
        joinPoint.getSignature()
            .getName());
}
```

ADVICE BEFORE

DIY

```
@Before("allLogEventMethods()")
public void logBefore(JoinPoint joinPoint) {

    LOG.info("BEFORE : " +
        joinPoint.getTarget()
            .getClass().getSimpleName()
        + " " +
        joinPoint.getSignature()
            .getName());
}
```

```
@Before("execution(* *.logEvent(..))")
public void logBefore(JoinPoint joinPoint)
```

ADVICE AFTER

DIY

```
@AfterReturning(
    pointcut="allLogEventMethods()",
    returning="retVal")

public void logAfter(Object retVal) {
    LOG.info("Returned value: " + retVal);
}
```

ADVICE AFTER

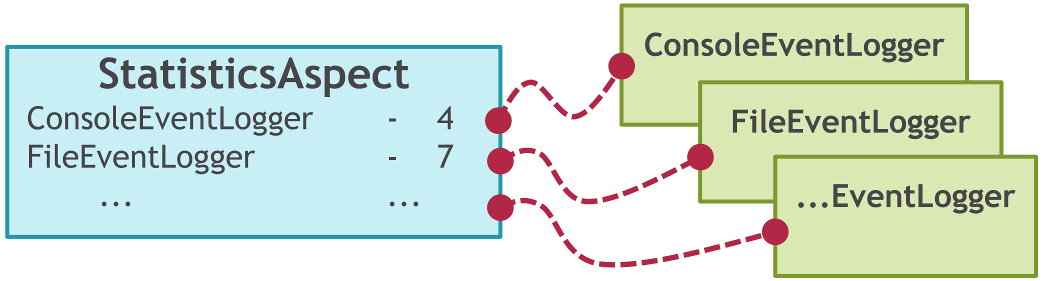
DIY

```
@AfterThrowing(
    pointcut="allLogEventMethods()",
    throwing="ex")

public void logAfterThrow(Throwable ex) {
    LOG.warn("Thrown: " + ex);
}
```

STATISTICS ASPECT

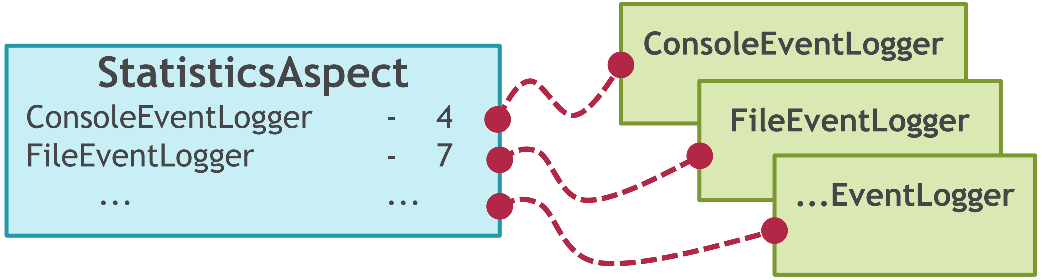
DIY



Statistics **aspect** that counts how many times `logEvent()` method was called in each logger.

STATISTICS ASPECT

DIY



```
@Aspect
public class StatisticsAspect {
    private Map<Class<?>, Integer> counter;

    @AfterReturning("allLogEventMethods()")
    public void count(JoinPoint jp) {
        Class<?> clazz = jp.getTarget().getClass();
        if (!counter.containsKey(clazz)) {
            counter.put(clazz, 0);
        }
        counter.put(clazz, counter.get(clazz)+1);
     }
```

ADVICE AROUND

DIY

```
@Around("consoleLoggerMethods()")
public void aroundLogEvent(
      ProceedingJoinPoint jp) {
```

ADVICE AROUND

DIY

```java
@Around("consoleLoggerMethods()")
public void aroundLogEvent(
        ProceedingJoinPoint jp) {

    // ... getting count

    if (count < MAX_ALLOWED) {

        jp.proceed();

    } else {
        // ... do other stuff
    }
}
```

ADVICE WITH ARGS

DIY

```
@Around("consoleLoggerMethods() &&
         args(evt)")
public void aroundLogEvent(
      ProceedingJoinPoint jp, Object evt) {

   if (...) {
      jp.proceed(new Object[] {evt});
   } else {
      otherLogger.logEvent(evt);
   }
}
```

XML BASED CONFIGURATION

```
<aop:config>

    <aop:aspect id="myAspect" ref="aBean">

        <aop:pointcut id="consoleLogging"
            expression="execution(* *.logEvent(..))
            &amp;&amp; within(*.Console*Logger)"/>

        <aop:before pointcut-ref="consoleLogging"
                        method="logBefore"/>

    </aop:aspect>

</aop:config>


<bean id="aBean" class="..."></bean>
```

THANK YOU

YURIY_TKACH@EPAM.COM

APRIL, 2015