

AspectJ: swiss army knife for binary patching, analysis and debugging of Java software

**15 minutes of relatively mild pain and a lifetime of comfort.
Yes, this is a long title.**

Dawid Weiss

Carrot Search s.c.
dawid.weiss@carrotsearch.com

Java Users Group, Poznań, 02/2011



— part I —

Begin From the End

A very real-life example of a

Nasty Bug

```
System.out.println("Parsing: " + args[0]);  
System.out.println(" => " + Double.parseDouble(args[0]));  
System.out.println("Done.");
```

A very real-life example of a

Nasty Bug

```
System.out.println("Parsing: " + args[0]);  
System.out.println(" => " + Double.parseDouble(args[0]));  
System.out.println("Done.");
```

```
Double.parseDouble("2.2250738585072012e-308")
```

Corporate-Desperate

Hot Fixing

Finding all the `parseDouble()` calls?

Eclipse search, boot class path manipulations, aspects...

Corporate-Desperate

Hot Fixing

Finding all the parseDouble() calls?

Eclipse search, boot class path manipulations, aspects...

Until a sensible workaround appears...

...protect the service.

Aspect-based quickfix

```
AGENT=-javaagent:[...]/aspectjweaver-1.6.10.jar  
ASPECTS=[...]/hotfix-aspects
```

```
java $AGENT -cp $ASPECTS:$PREV_CLASSPATH main.Class
```

```
> ... jug.demos.aspectj.Example1 2.2250738585072012e-308

[] info AspectJ Weaver Version 1.6.10 built on Friday Oct 22, 2010 [...]
[] info register classloader sun.misc.Launcher$AppClassLoader@4aad3ba4
[] info using configuration ...
[] info register aspect jug.demos.aspectj.aspects.ParseDoubleHotFix
[] weaveinfo Join point
  'method-call(double java.lang.Double.parseDouble(java.lang.String))'
  in Type 'jug.demos.aspectj.Example1' (Example1.java:17)
  advised by around advice from
    'jug.demos.aspectj.aspects.ParseDoubleHotFix' (ParseDoubleHotFix.aj:8)

Parsing: 2.2250738585072012e-308

Exception in thread "main" java.lang.IllegalArgumentException:
  We apologize for inconvenience, but this number is temporarily
  not parseable by Oracle: 2.2250738585072012e-308
  at jug.demos.aspectj.aspects.ParseDoubleHotFix.ajc$around$jug_demos_aspectj_aspects_
    ParseDoubleHotFix$1$9a9a9b99(ParseDoubleHotFix.aj:13)
  at jug.demos.aspectj.Example1.main(Example1.java:17)
```

The aspect that did it?

Think in terms of

What and Where

```
package jug.demos.aspectj.aspects;

public aspect ParseDoubleHotFix
{
    double around(String s):
        !within(jug.demos.aspectj.aspects..*) &&
        call(double java.lang.Double.parseDouble(String)) &&
        args(s)
    {
        if (s.indexOf("2250738585072012") >= 0) {
            throw new IllegalArgumentException(
                "We apologize for inconvenience, but this number is"
                " temporarily not parseable by Oracle: " + s);
        } else {
            return Double.parseDouble(s);
        }
    }
}
```

Aspects

Worth of Notice

Aspects may be written after the code is compiled.

And are completely independent of the codebase.

Full project scope application.

Aspects apply to all binaries, including those we don't have source access to.

Exercise:

Patching Catalina/Tomcat

Simple JSP that calls parseDouble

Tomcat 6.0.24

Hotfix patching with aspects

Aspect classes, CATALINA_OPTS, inspect the logs

```
dweiss@idss-dw:~/Applications/apache-tomcat-6.0.24/bin$ ./catalina.sh run
Using CATALINA_BASE: /home/dweiss/Applications/apache-tomcat-6.0.24
Using CATALINA_HOME: /home/dweiss/Applications/apache-tomcat-6.0.24
...
[StandardClassLoader@348bdcd2] info register aspect jug.demos.aspectj.aspects.ParseDoubleHotFix
...
[StandardClassLoader@348bdcd2] weaveinfo Join point 'method-call(double Double.parseDouble(java.lang.String))'
in Type 'org.apache.catalina.connector.Request' (Request.java:2704)
advised by around advice from 'jug.demos.aspectj.aspects.ParseDoubleHotFix' (ParseDoubleHotFix.java:10)
...
[StandardClassLoader@348bdcd2] weaveinfo Join point 'method-call(double Double.parseDouble(java.lang.String))'
in Type 'org.apache.jasper.compiler.JspConfig' (JspConfig.java:68)
advised by around advice from 'jug.demos.aspectj.aspects.ParseDoubleHotFix' (ParseDoubleHotFix.java:10)
...
[JasperLoader@19bfcd9f] weaveinfo Join point 'method-call(double Double.parseDouble(java.lang.String))'
in Type 'org.apache.jsp.index_jsp' (index_jsp.java:63)
advised by around advice from 'jug.demos.aspectj.aspects.ParseDoubleHotFix' (ParseDoubleHotFix.java:10)
...
```

— part II —

Aspect Oriented Programming



Large white feathered hat

Coca-Cola

DRINK
Coca-Cola
5¢

See Page 10
The Coca-Cola Co.
Atlanta, Ga.
Birmingham, Chicago,
Philadelphia, Los Angeles,
Dallas

AOP's

Core Idea

Inject arbitrary code at well-defined execution points.

```
/**  
 * A typical code structure.  
 */
```

```
public class Example2
```

```
{
```

```
    public int value = 2; within
```

```
    static {
```

```
    }
```

static initializer

```
    public Example2() {
```

```
    }
```

execution(constructor)

```
    public void method() {
```

```
    }
```

execution(method)

```
    public static void main(String [] args) {
```

```
    }
```

execution(method)

```
}
```

```

/**
 * A typical code structure.
 */
public class Example2
{
    public int value = 2;

    static {
        System.out.println("[code] Class init.");
    }

    public Example2() {
        super(); // bad style, but hey...
    }

    public void method() {
        try {
            System.out.println("[code] Current value: " + value);
            value = value + 1;
            value++;
        } catch (Exception e) {
            throw new RuntimeException();
        }
    }

    public static void main(String [] args) {
        new Example2().method();
    }
}

```

field store

field read

method call

constructor call

exception handler

?

How many **JoinPoints?**

static

method or constructor calls, execution

dynamic

cflow, cflowbelow

contextual

this, target, args

advanced

adviceexecution, preinitialization

Logical combinations of join point declarations.

A few simple

PointCuts & JoinPoints

```
/** All join points from Example2. */  
pointcut targets(): within(jug.demos.aspectj.Example2);  
  
/** Any field writes, anywhere. */  
pointcut fieldWrite(): set(* *.*);  
  
/** Field writes (of any type) if done within Example2... */  
pointcut fieldWritesInExample2() : targets() && fieldWrite();  
  
/** Field writes if the <b>target</b> is Example2. */  
pointcut fieldWritesToExample2() : set(* Example2.*);  
  
/** Int-type field writes if the <b>target</b> is Example2. */  
pointcut intFieldWritesToExample2() : set(int Example2.*);
```

Exercise:

Field Read/Stores

Print a log on any field access (read/store) within Example2.

Digression:

Eclipse Setup

AJDT

Very useful for experimenting.

m2eclipse

Doesn't play well with AJDT, add nature manually.

AOP's

Terminology Summary

Pointcut — the “where”

Method call, constructor call, field access, static initializer...

Advice — the “what”

Before, after, around?

Aspect — the “big picture”

A collection of pointcuts and advices.

More concerning

Advices, Pointcuts, Aspects

Not during this talk.

AspectJ has a lot of useful documentation.

I will explain the examples as we go.

99% of practical applications are dead-simple.

AJDT

Using AJDT and a scratchpad project helps a lot for complex aspects.

— part III —

Weaving

Static, compilation-time

Aspect Weaving (iajc)

Compilation of aspects and source code.

Compilation of aspects and binary classes.

Aspects woven in the result.

Eclipse's AJDT.

Dynamic, load-time

Aspect Weaving (agent)

Weaves precompiled aspects into classes.

Binary-to-binary.

Aspects configured using XML.

META-INF/aop-ajc.xml

VM must be instructed to use it.

-javaagent:[...]/aspectjweaver-1.6.10.jar

Exercise: Eclipse and

Dynamic Weaving

Integrate aspects into a non-AspectJ project launch.

Eclipse: weaving launcher for Example1.

The dirty

Weaving Internals

Generated code depends on pointcut complexity.

static vs. dynamic, context exposure

Generated code may require aspectjrt.jar.

Even if aspects don't use any AspectJ infrastructure.

Most embedded code is still clean and fast.

Thinking of

Performance

Avoid dynamic pointcuts.

Avoid generic arg captures (use specific type).

Weave into exact pointcut locations.

Inspect weaving logs.

Thinking of

Performance

Avoid dynamic pointcuts.

Avoid generic arg captures (use specific type).

Weave into exact pointcut locations.

Inspect weaving logs.

Only applicable to performance-critical aspects, of course.

Exercise:

Bare Woven Code

```
public class Example3 {  
    public int methodA() { return 1; }  
    public int methodB() { return 1; }  
}
```

```
public aspect Example3Pointcuts {  
    Object around(): execution(* *.methodA()) {  
        return proceed();  
    }  
  
    int around(): execution(int *.methodB()) {  
        return proceed();  
    }  
}
```

```
public int methodA() {
    return Conversions.intValue(methodA_aroundBody1$advice(
        this, Example3Pointcuts.aspectOf(), null));
}

private static Object methodA_aroundBody1$advice(...) {
    AroundClosure aroundclosure = ajc$aroundClosure;
    return Conversions.intObject(methodA_aroundBody0(ajc$this));
}

private static int methodA_aroundBody0(Example3 ajc$this) {
    return 1;
}

public int methodB() {
    return methodB_aroundBody3$advice(
        this, Example3Pointcuts.aspectOf(), null);
}

private static int methodB_aroundBody3$advice(...) {
    AroundClosure aroundclosure = ajc$aroundClosure;
    return methodB_aroundBody2(ajc$this);
}

private static int methodB_aroundBody2(Example3 ajc$this) {
    return 1;
}
```

— part IV —

Applications

Program

Tracing and Performance

Much like inserting logging statements.

But easily adjustable.

Like fine-grained category filters.

Possible to trace third-party libraries

Exact call path of an error, perf. measurements.

Exercise

Tracing Google Guava

Print a trace of all of Guava's methods for the code below:

```
ConcurrentMap<Integer, Integer> map =
    new MapMaker().makeComputingMap(
        new Function<Integer, Integer>()
        {
            public Integer apply(Integer input) {
                return input * 2;
            }
        }
    );

map.get(10);
System.out.println("Done.");
```

```
before() : guava() && (methods() || constructors()) {  
    System.out.println("> " +  
        thisJoinPointStaticPart.toShortString());  
}
```

With a very simple method-level time tracing:

```
Object around(): guava() && (methods() || constructors()) {
    long start = System.currentTimeMillis();
    try {
        return proceed();
    } finally {
        long end = System.currentTimeMillis();
        System.out.println("> " +
            thisJoinPointStaticPart.toShortString() +
            " => " + (end - start) / 1000.d + "sec.");
    }
}
```

Detecting

Concurrency Bugs

Next to impossible to debug in step-by-step mode.

Happen infrequently.

Exercise: looking for real

Concurrency Problems

Is your class accessed by a single thread at a given moment in time (exclusively)?

Exercise: looking for real

Concurrency Problems

Is your class accessed by a single thread at a given moment in time (exclusively)?

```
@Target({TYPE})  
@Retention(RetentionPolicy.RUNTIME)  
public @interface ExclusiveAccess {  
}
```

```
@ExclusiveAccess
public static class EvenCounter
{
    private int value;

    public synchronized void increment()
    {
        value++;
        value++;
    }

    public int getValue()
    {
        return value;
    }
}
```

```
Object around(Object t):  
    within(@ExclusiveAccess *) && execution(* *.*(..)) && target(t) {  
        try {  
            enter(t);  
            return proceed(t);  
        } finally {  
            leave(t);  
        }  
    }  
}
```

```
Object around(Object t):  
    within(@ExclusiveAccess *) && execution(* *.*(..)) && target(t) {  
        try {  
            enter(t);  
            return proceed(t);  
        } finally {  
            leave(t);  
        }  
    }  
}
```

```
private final IdentityHashMap<Object, Thread> allocated =  
    Maps.newIdentityHashMap();  
  
public synchronized void enter(Object target) {  
    Thread other = allocated.get(target);  
    if (other != null) {  
        System.out.println("Same object shared by threads: "  
            + Thread.currentThread() + " and "  
            + other + ", object=" + target);  
    }  
    allocated.put(target, Thread.currentThread());  
}  
  
public synchronized void leave(Object target) {  
    allocated.remove(target);  
}
```

On-site debugging

Live JVM

Write your aspects.

Test them locally.

Move your (woven) code to the server.

Enable breakpoints in (local) Eclipse.

Start the service with remote debugger.

`dt_socket`

Exercise: on-site

Tomcat Debugging

webapp: example1-5.prep, breakp. in aspect

Becoming

Chuck Norris

Tracking arguments or inaccessible fields.

Exposing private or semi-private members.

Pushing arguments down the call stack (cflow).

(You name it.)

— part V —

Derivations

Politechnika Poznańska
Wydział Informatyki i Zarządzania
Instytut Informatyki

Praca dyplomowa magisterska

ASPECT ORIENTED PROGRAMMING IN J2ME

Tomasz Maćkowiak

Applying aspects to **J2ME**

On-device logging.

On-device exception stacks.

On-device profiling.

'No leftover code' assertions.

Thread-locals.

Java 1.5 syntax support (!).

Applying aspects to **J2ME: pipeline**

Weaving aspects and sources.

Result: Java 1.5+ binaries.

Post-processing binaries.

Tuning AspectJ-specific code for J2ME. Exception rethrow wrappers.

Retrotranslator.

Result: Java 1.4 binaries.

Preverification

Result: J2ME binaries.

This should be applicable to Android dev. pipeline.

An example aspect

Exception Stacks

```
pointcut methodOrConstructorExecution():  
    !within(aspectme.j2me..*)  
    && (execution (* *.*(..)) || execution(*.new(..)));
```

An example aspect

Exception Stacks

```
pointcut methodOrConstructorExecution():  
    !within(aspectme.j2me..*)  
    && (execution (* *.*(..)) || execution(*..new(..)));
```

```
before() : methodOrConstructorExecution() {  
    getStack(Thread.currentThread())  
        .push(thisJoinPointStaticPart.getSignature());  
}
```

An example aspect

Exception Stacks

```
pointcut methodOrConstructorExecution():  
    !within(aspectme.j2me..*)  
    && (execution (* *.*(..)) || execution(*.new(..)));
```

```
before() : methodOrConstructorExecution() {  
    getStack(Thread.currentThread())  
        .push(thisJoinPointStaticPart.getSignature());  
}
```

```
after() returning : methodOrConstructorExecution() {  
    getStack(Thread.currentThread()).pop();  
}
```

An example aspect

Exception Stacks

```
after() throwing (Throwable t): methodOrConstructorExecution() {
    Throwable augmented = t;
    if (!isAugmented(t)) {
        augmented = augment(getStack(Thread.currentThread()), t);
    }

    getStack(Thread.currentThread()).pop();
    if (augmented != t)
        Rethrower.rethrow(augmented);
}
```

TABLE 9.1: Evaluation results for stack trace aspect for Diet3D and NaviExpert.

Measurement		Diet3D	NaviExpert
Size (bytes)	Original	20 804	883 551
	With aspect	66 978	1 456 400
	Overhead (%)	222%	65%
Avg. emul. speed (FPS)	Original	75.56	—
	With aspect	65.42	—
	Overhead (%)	13.42%	—
Avg. dev. speed (FPS)	Original	52.44	—
	With aspect	45.37	—
	Overhead (%)	13.48%	—

— part VI —

Summary

Summary

Aspects are *very* useful.

Even if not for the original purposes.

Load time weaving a powerful *patching* system.

Static weaving useful for inspecting the woven output.

Summary

Aspects are *very* useful.

Even if not for the original purposes.

Load time weaving a powerful *patching* system.

Static weaving useful for inspecting the woven output.

Homework

Re-run the examples locally. Play with more advanced pointcuts (cflow).



dawid.weiss@carrot-search.com