© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

Hermod Opstvedt

Chief Architect

DnB NOR ITUD

Common components

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- What is AspectJ?
  - Aspect-oriented programming (AOP) is a technique for improving separation of concerns.
    - Crosscutting concerns.
      - ✓ Logging
      - ✓ Exception handling
      - ✓ Authorization
      - ✓ *etc*.
  - Compiles into standard Java bytecode.

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- Has been available since Java 1.1
- Spring and other API's use it.
- Is a vital part of many J2EE containers (WebSphere, *etc*.).

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- AspectJ is an aspect-oriented extension to Java.
  - Files are named .aj
- Provides a means of modularization.
- Reduction in code size.
- You can delay certain parts of your implementation.
- Code is easier to reuse.

© Copyright 2008, DnB NOR ITUD

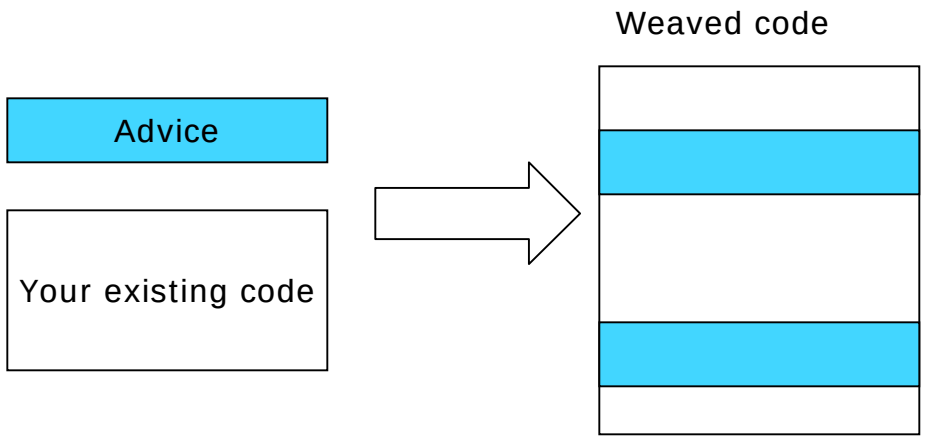
Take Control with AspectJ

- How to use AspectJ:
  - Identify the concern.
    - May need to decompose your code.
  - Implement.
    - Write the concern.
  - Identify the places where it should be implemented.
    - Compose.

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- How does it work?
  - The concern (Advice) is weaved into your existing code (Joinpoints) using rules (Pointcuts).



© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- Using AspectJ
  - Available as an Eclipse plugin.
    - Currently supports Java 5 (AspectJ 5)
    - Supports annotations.
  - Can be used standalone using the ajc compiler.

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- Advice
  - An advice is the peace of code that you want to execute under a given condition.
  - Three different types of advices:
    - Before
    - Around
    - After

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- Joinpoints
  - Identifiable points in a program's execution.
    - Method call.
    - Assignment to variables.
    - Constructor call.

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- Pointcuts
  - Selects Joinpoints and gives you access to the running context at those Joinpoints
    - Execution
      - ✓ On the method body being executed.
    - Call
      - ✓ In the methods calling the method.

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- A HelloWorld Aspect

```
public aspect HelloWorldAspect {

// Pointcuts define where an advice is to be executed
public pointcut methodExecution() : call(* css2008..*.*(..)) && !within(HelloWorldAspect);

// Before Advise
before() : methodExecution() {
System.err.println("HelloWorldAspect(before): Just before calling");
}

// After Advice (returns normally)
after() returning : methodExecution() {
System.err.println("HelloWorldAspect(after): Just after calling");
}

// After Advice (method threw exception)
after() throwing : methodExecution() {
System.err.println("HelloWorldAspect(after): call threw an exception");
}
}
```

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- Testclass

```
public class HelloWorldClass {

public static void main(String[] args)
{
HelloWorldClass me=new HelloWorldClass();
me.sayHello();
me.throwIt();

}

private void sayHello()
{
System.err.println("Hello World!");
}

private void throwIt()
{
throw new RuntimeException();
}
}
```

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- When run, produces:

```
HelloWorldAspect(before): Just before calling
Hello World!
HelloWorldAspect(after): Just after calling
HelloWorldAspect(before): Just before calling
HelloWorldAspect(after): call threw an exception
Exception in thread "main" java.lang.RuntimeException
at css2008.aspectj.test.HelloWorldClass.throwIt(HelloWorldClass.java:20)
at css2008.aspectj.test.HelloWorldClass.main(HelloWorldClass.java:9)
```

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- Weaving
  - Static
    - Code is run through a compiler that joins the Aspects and the code.
    - New code is created and is permanently changed (*i.e.* new jar).
  - Loadtime
    - Code is weaved when it is loaded into memory
    - Original code is untouched (*i.e.* same jar)

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- Introduction
  - A way of adding methods or variables into existing code statically.
    - Instead of rewriting a class or decompiling a class that you did not write, you simply write the code and then Introduce it into that class.
    - Simple case: You have a class that has two members and a sum method. Using introduction, you can add a difference method, and in that method have access to the members.

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- AddDifference Aspect

```
import css2008.aspectj.test.SumClass;
public aspect AddDifference {

public int SumClass.difference()
{
return getA()-getB();
}

}
```

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- SumClass

```
public class SumClass {

private int a=0;
private int b=0;

public int sum()
{
return a+b;
}

public int getA() {
return a;
}
public void setA(int a) {
this.a = a;
}
public int getB() {
return b;
}
public void setB(int b) {
this.b = b;
}

}
```

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- TestSum class

```
public class TestSum {

public static void main(String[] args)
{
SumClass sc=new SumClass();
sc.setA(2);
sc.setB(2);
System.err.println("Sum:" + sc.sum());
// Call the introduced method
System.err.println("Difference:" + sc.difference());

}

}
```

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- When run, produces:

```
HelloWorldAspect(before): Just before calling
HelloWorldAspect(after): Just after calling
HelloWorldAspect(before): Just before calling
HelloWorldAspect(after): Just after calling
HelloWorldAspect(before): Just before calling
HelloWorldAspect(after): Just after calling
Sum:4
HelloWorldAspect(before): Just before calling
HelloWorldAspect(before): Just before calling
HelloWorldAspect(after): Just after calling
HelloWorldAspect(before): Just before calling
HelloWorldAspect(after): Just after calling
HelloWorldAspect(after): Just after calling
Difference:0
```

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- What you need to deploy
  - The jar file containing the advices if using Loadtime weaving.
  - Or the jar file that has been weaved statically.
  - The AspectJ jar files
    - aspectjrt.jar
    - aspectjweaver.jar if Loadtime weaving.

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- If using Loadtime weaving:
  - Tell the JVM that you want to use AspectJ
    - In Java 5 add the following as:

      -javaagent:pathto/aspectjweaver.jar
    - A file that contains a list of the advices, and also tells the loader which classes/packages to include when it searches for places to add the code, placed under META-INF in the classpath.
      - aop.xml

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- Sample aop.xml

```
<aspectj>

   <weaver options="-proceedOnError -Xset:weaveJavaPackages=true -Xset:weaveJavaxPackages=true
-Xlintfile:META-INF/Xlint.properties –showWeaveInfo -showWeaveInfo -verbose -debug ">
      <include within="*..*" />
      <exclude within="a.b.c.d.e..*" />
   </weaver>
   <aspects>
      <aspect name="a.b.c.d.SomeAspect" />
      <aspect name="a.b.c.d.AnotherAspect" />
   <aspects>
</aspectj>
```

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- A file that tells how to handle the process of weaving with respect to errors and logging placed under META-INF in the classpath.
  - ✓ Xlint.properties

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- Sample Xlint.properties

```
invalidAbsoluteTypeName = ignore
invalidWildcardTypeName = ignore
unresolvableMember = warning
typeNotExposedToWeaver = ignore
shadowNotInStructure = ignore
unmatchedSuperTypeInCall = ignore
canNotImplementLazyTjp = ignore
multipleAdviceStoppingLazyTjp=warning
noGuardForLazyTjp=ignore
uncheckedAdviceConversion = warning
needsSerialVersionUIDField = ignore
brokeSerialVersionCompatibility = ignore
....
```

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- Adding loadtime weaving to JDK prior to Java 5
- Download Glassbox from source
  - https://glassbox-inspector.dev.java.net/
- Run (Using the targeted JVM)
  - java -jar "pathtoinstall\glassbox\ltw13\dist\createJavaAdapter.jar"

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- Resulting jar:
  - aspectj14Adapter.jar.
- Add the following to your startup:
  - -Xbootclasspath/p:pathtoinstall/java14Adapter.jar -Xbootclasspath/a:pathtoinstall/createJavaAdapter.jar;pathtoinstall/aspectj14Adapter.jar;pathtoinstall/aspectjweaver.jar;pathtoinstall/sax-2.0.1.jar -Daspectwerkz.classloader.preprocessor=org.aspectj.ext.ltw13.ClassPreProcessorAdapter

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- Using AspectJ to handle problems.
  - There are numerous cases that can be handled in a common way.
    - Logging.
      - ✓ You want to standardize on a certain API.
    - Exception handling.
      - ✓ You want all exceptions to be handled the same way.
    - Handling communication errors.
      - ✓ An external call may hang forever (Ref HTTP)
      - ✓ A backend system may respond and loop forever.

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- A simple case:
  - Stop all calls to a certain method, by throwing an exception.
  - We have class SomeClass that has a method someMethod.
  - When class SomeCaller calls someMethod an Exception is thrown.
  - We implement it using a before advice.

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- Sample 2:
  - We want to alter the return value of a method because the State tax has changed and this class is in a jar file that some 3rd party vendor has provided us with.
  - We implement this using an after advise.

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- Sample 3:
  - We don't want a call to a certain method to be executed because there is a bug in it, so we want to execute the same but bug-free code. However we don't have time to wait for the programmer who wrote it to fix it.
  - We implement this using an around advice.

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- Sample 4:
  - Introducing new methods and members to an existing class.

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- Some more complex samples combined with JMX.
  - Building a system that handles error conditions gracefully, and enables you to control access to resources, *etc*.
  - Smart system that uses statistics in order to make “wise”" decisions.

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- Pitfalls
  - Beware of class loading issues in containers.
  - Do not over-implement, because you may lose track of what is actually going on in your code (typically advising advices).

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- References:
  - http://www.eclipse.org/aspectj/
  - http://hugunin.net/papers/2001-ecoop-overviewOfAspectj.pdf
  - http://aspectwerkz.codehaus.org

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

© Copyright 2008, DnB NOR ITUD

Take Control with AspectJ

- Remember to fill out evaluation forms
- All sample code will be on the post-conference CD.