Architectures Logicielles

Olivier Caron

Polytech Lille

Programmation par aspects
Aspect-Oriented Programming


GILBERT DELAHAYE - MARCEL MARLIER
martine
Ã©crit en UTF-8
casterman

Programmation orienté aspects

- Une nouvelle manière de programmer
- Nouveau découpage d'une application : un programme + aspects
- Réutiliser des aspects pour d'autres applications : vers des composants d'aspects
- Pallier au problèmes de fonctions transverses

Programmation orienté aspects

- Une nouvelle manière de programmer
- Nouveau découpage d'une application : un programme + aspects
- Réutiliser des aspects pour d'autres applications : vers des composants d'aspects
- Pallier au problèmes de fonctions transverses

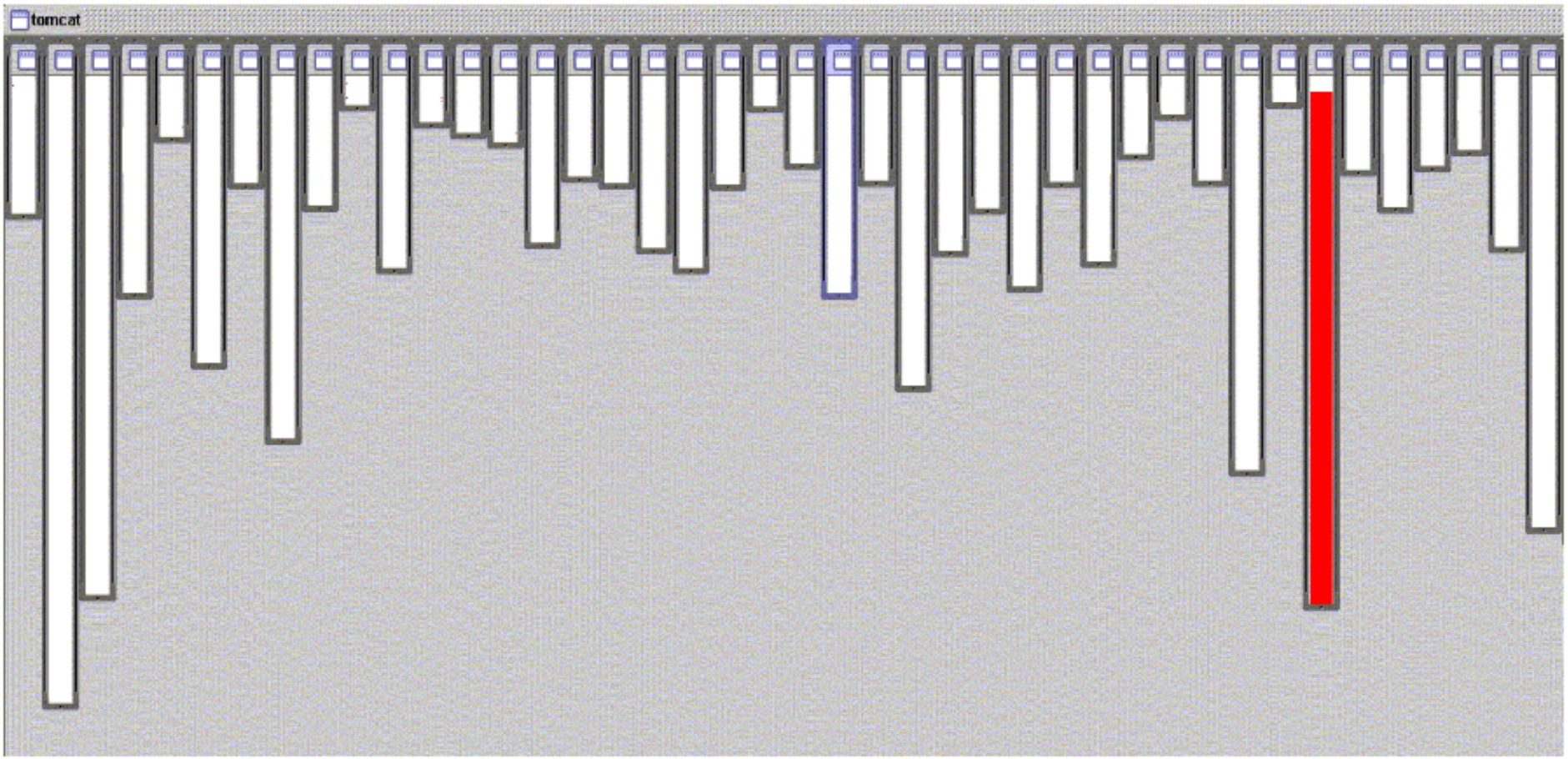
Programmation orienté aspects

- Une nouvelle manière de programmer
- Nouveau découpage d'une application : un programme + aspects
- Réutiliser des aspects pour d'autres applications : vers des composants d'aspects
- Pallier au problèmes de fonctions transverses

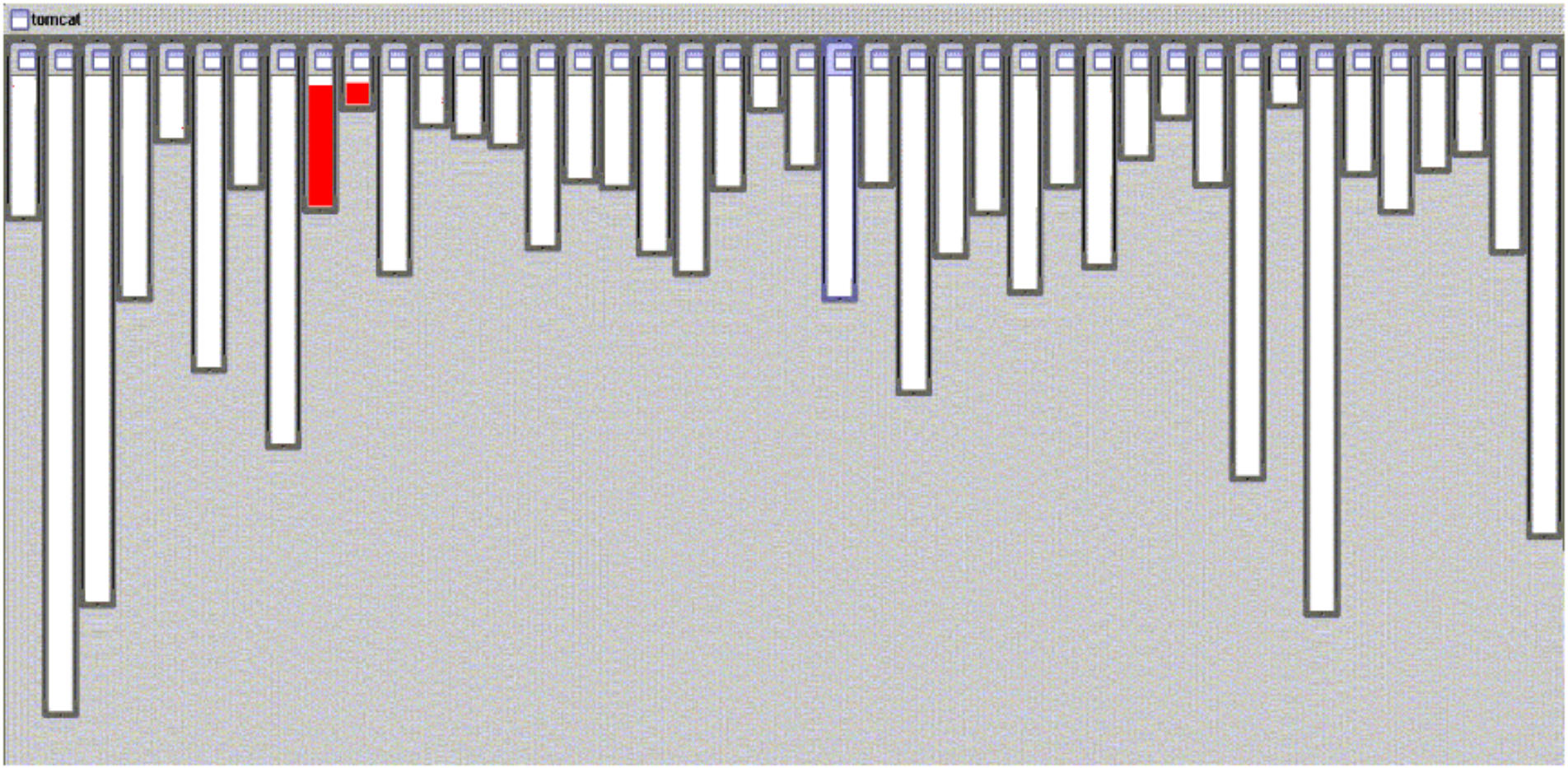
Programmation orienté aspects

- Une nouvelle manière de programmer
- Nouveau découpage d'une application : un programme + aspects
- Réutiliser des aspects pour d'autres applications : vers des composants d'aspects
- Pallier au problèmes de fonctions transverses

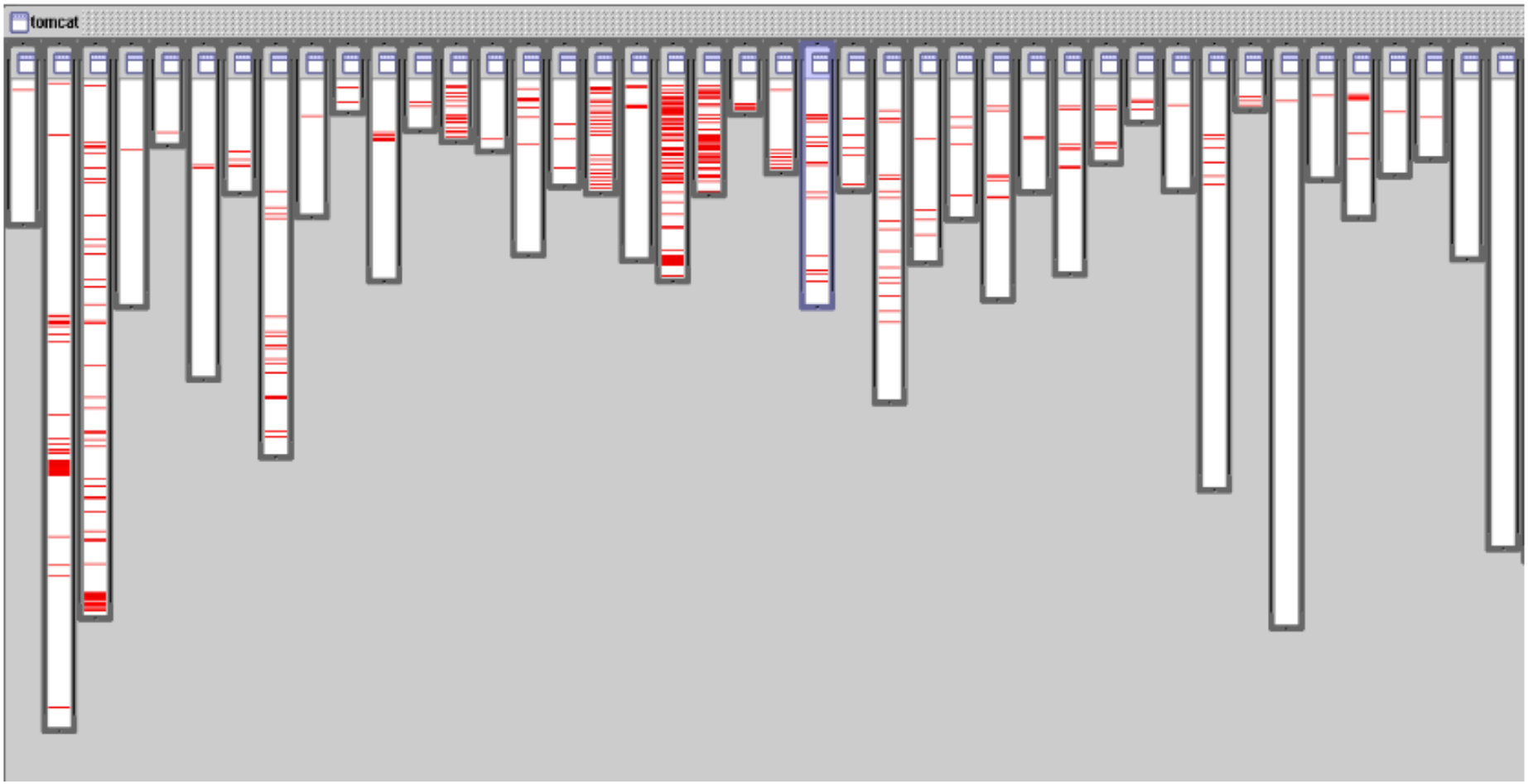
Exemple : serveur Tomcat et Parsing (1/3)



Exemple : serveur tomcat et Pattern Matching (2/3)

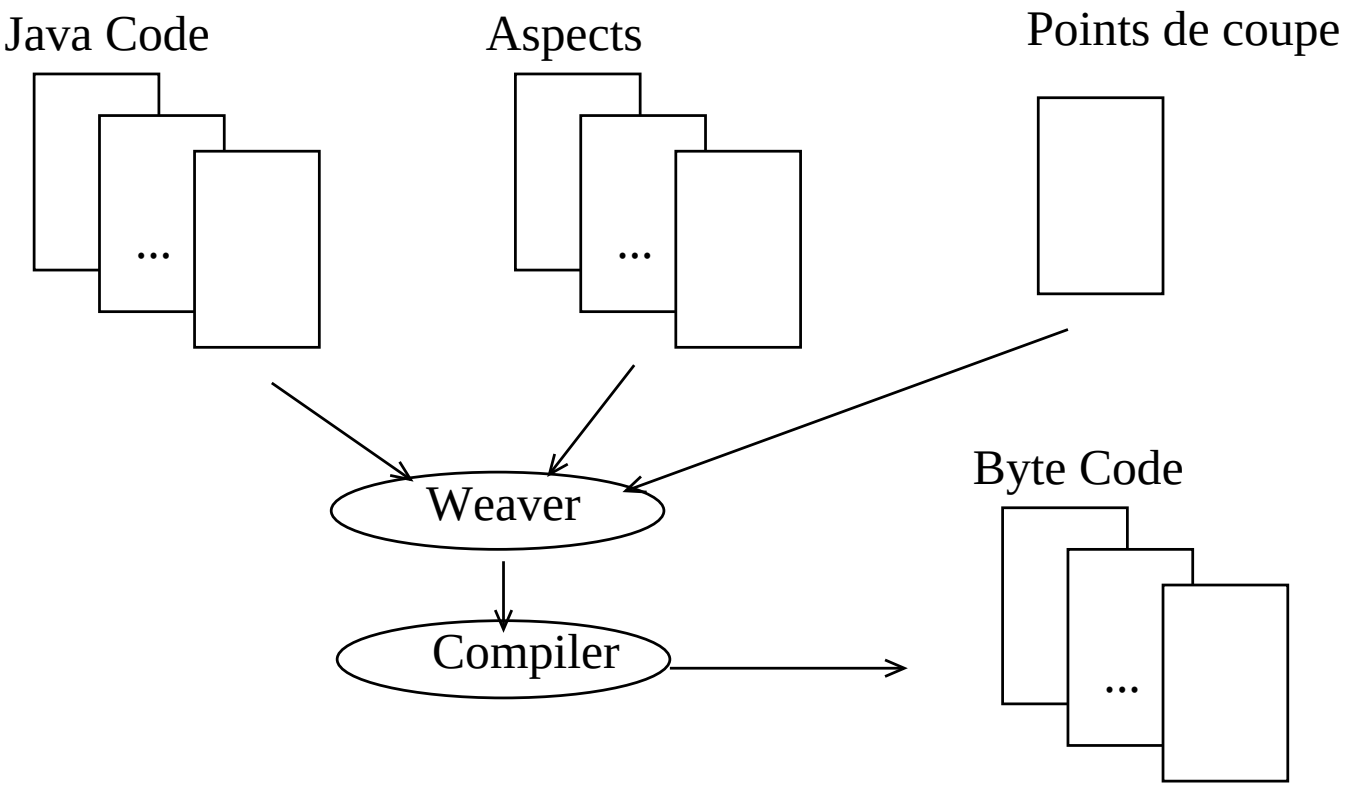


Exemple : serveur tomcat et Logging (3/3)



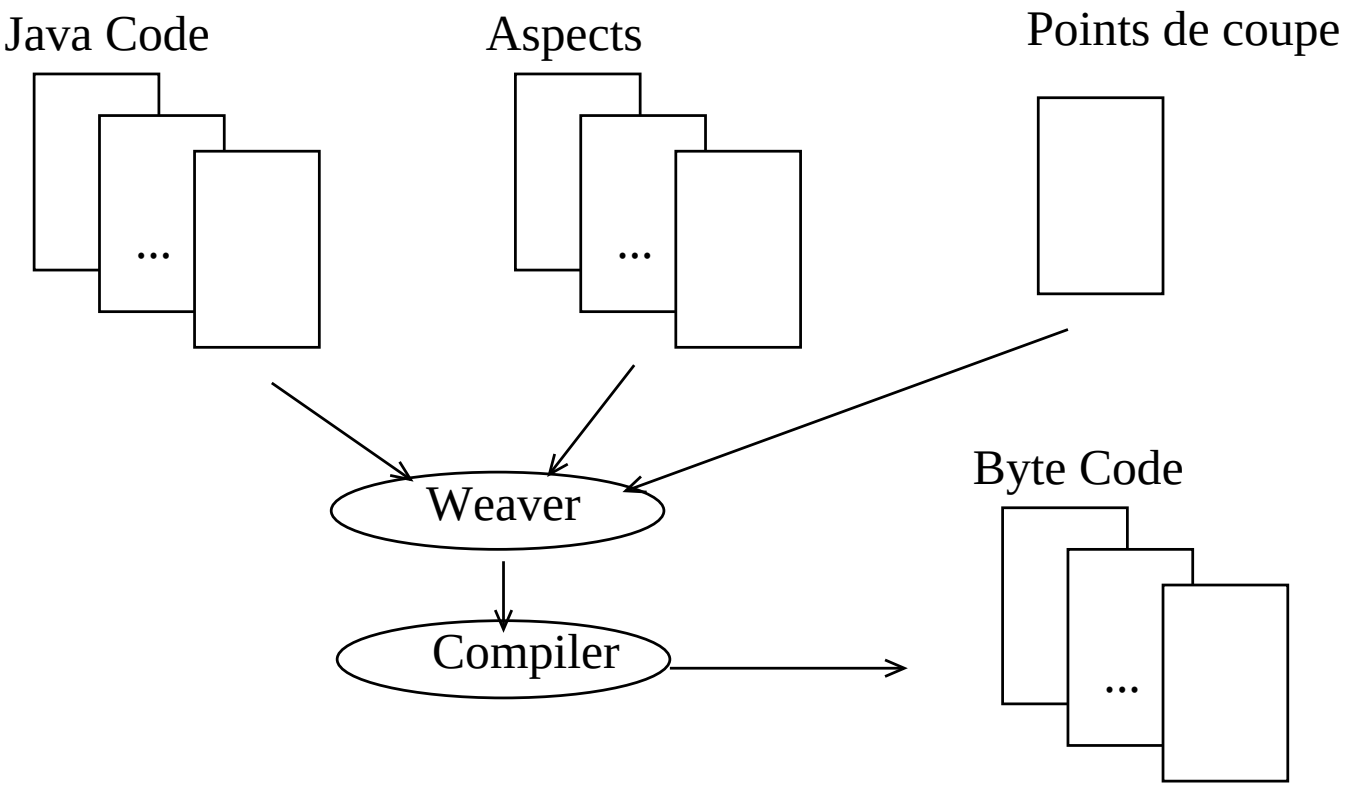
Approche AOP

- disposer d'un langage d'aspect
- définir les points de jonction
- Réaliser le tissage des aspects sur l'application (weaver)



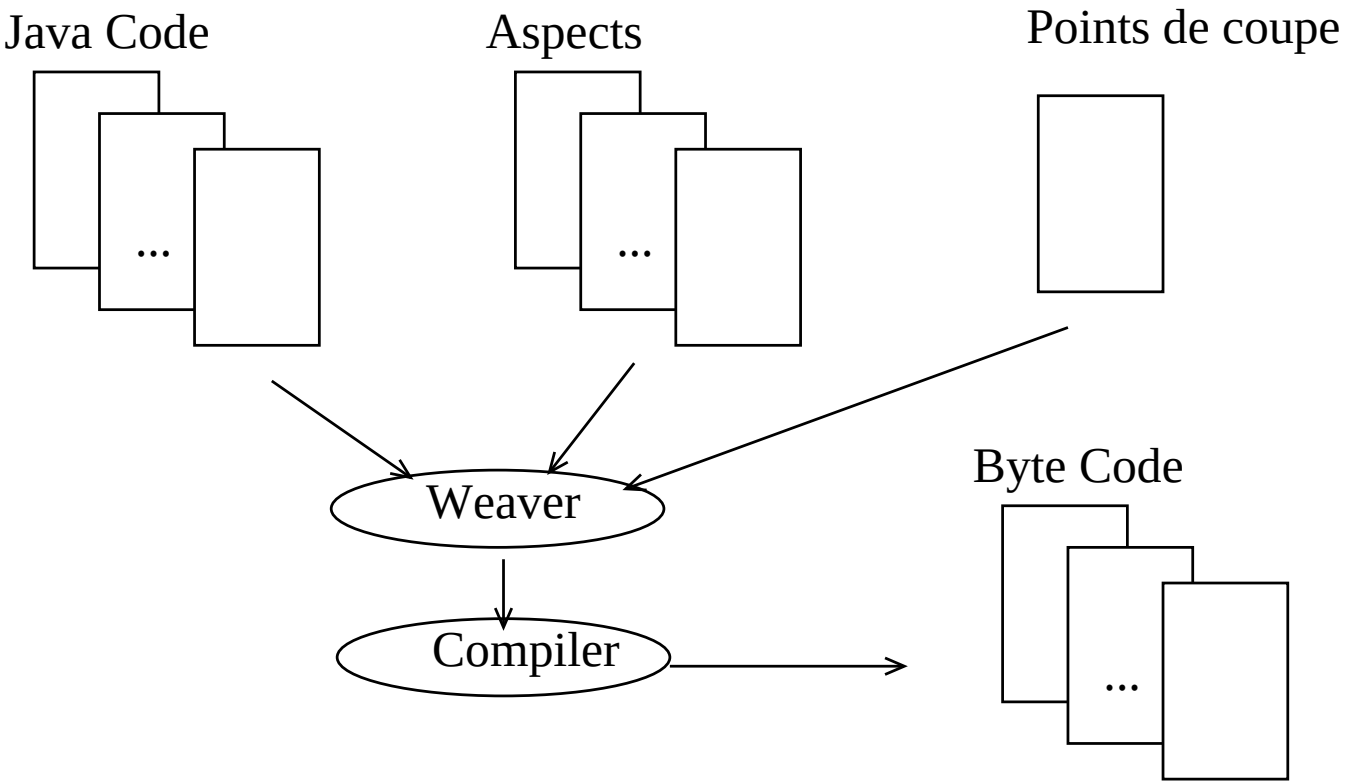
Approche AOP

- disposer d'un langage d'aspect
- définir les points de jonction
- Réaliser le tissage des aspects sur l'application (weaver)



Approche AOP

- disposer d'un langage d'aspect
- définir les points de jonction
- Réaliser le tissage des aspects sur l'application (weaver)



Aspect J et JBoss AOP

- www.eclipse.org/aspectj (Xerox Lab puis IBM eclipse)
- Utilisation du langage Java
- Origine : compilateur statique java (Xerox) dynamique (IBM)
- Autre implémentation AOP : projet JBoss AOP

Aspect J et JBoss AOP

- www.eclipse.org/aspectj (Xerox Lab puis IBM eclipse)
- Utilisation du langage Java
- Origine : compilateur statique java (Xerox) dynamique (IBM)
- Autre implémentation AOP : projet JBoss AOP

Aspect J et JBoss AOP

- www.eclipse.org/aspectj (Xerox Lab puis IBM eclipse)
- Utilisation du langage Java
- Origine : compilateur statique java (Xerox) dynamique (IBM)
- Autre implémentation AOP : projet JBoss AOP

Aspect J et JBoss AOP

- www.eclipse.org/aspectj (Xerox Lab puis IBM eclipse)
- Utilisation du langage Java
- Origine : compilateur statique java (Xerox) dynamique (IBM)
- Autre implémentation AOP : projet JBoss AOP

JBoss AOP by example : programme principal

```
public class Main {
  public static void main(String args[]) {
    Compte c = new Compte() ;
    c.setId("Dupont") ;
    c.setSolde(200) ;
    System.out.println(c) ;
  }
}
```

JBoss AOP by example : classe Compte

```
public class Compte {
  private String id ; private double solde ;

  public Compte() {}

  public String getId() { return this.id ; }
  public void setId(String value) {
    this.id=value ;
  }

  public double getSolde() { return this.solde ; }
  public void setSolde(double value) {
    this.solde=value ;
  }

  public String toString() {
    return "compte de "+this.id+" : "+this.solde ;
  }
}
```

JBoss AOP by example : aspect-Interceptor

```
package org.jboss.aop.advice;

import org.jboss.aop.joinpoint.Invocation;

   public interface Interceptor {
      public String getName();
      public Object invoke(Invocation invocation)
             throws Throwable;
   }
}
```

JBoss AOP by example : point de coupe

- fichier de coupe : jboss-aop.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<aop>
    <bind pointcut="execution(public␣*␣Compte->*(..))">
        <interceptor class="demoAspect.DemoTrace"/>
    </bind>
</aop>
```

JBoss AOP by example : DemoInterceptor (1/2)

```
package demoAspect ;

import org.jboss.aop.joinpoint.Invocation;
import org.jboss.aop.joinpoint.MethodInvocation;
import org.jboss.aop.advice.Interceptor;

public class DemoTrace implements Interceptor
{
    public String getName() { return "DemoTrace"; }
```

JBoss AOP by example : DemoInterceptor (2/2)

```
    public Object invoke(Invocation invocation)
    throws Throwable {
       String nomMethode="" ;
       try {
          MethodInvocation mi = (MethodInvocation)invocation;
          nomMethode = mi.getMethod().getName() ;
          System.out.println("Interception de : "+ nomMethode) ;
          return invocation.invokeNext();
       } finally {
                System.out.println("Quitter :"+nomMethode);
       }
    }
}
```

JBoss AOP by example : exécution

```
runWithoutAOP:
    [java] compte de Dupont : 200.0
runWithAOP:
    [java] Interception de : setId
    [java] Quitter :setId
    [java] Interception de : setSolde
    [java] Quitter :setSolde
    [java] Interception de : toString
    [java] Quitter :toString
    [java] compte de Dupont : 200.0
```

Exemple AOP et sécurité

```
public Object invoke(Invocation invocation)
 throws Throwable {
    String nomMethode="" ;
    try {
      MethodInvocation mi = (MethodInvocation)invocation;
      nomMethode = mi.getMethod().getName() ;
      System.out.println("Interception de : "+ nomMethode) ;
      System.out.println("Permission Denied");
      return "" ;
      // return invocation.invokeNext();
      // ici, pas d'appel à la méthode interceptée
    } finally {
            System.out.println("Quitter :"+nomMethode);
    }
 }
```

JBoss AOP

- Langage de points de coupe :
  - expressions rationnelles, attributs, flot d'exécution,...
  - interception de méthodes annotées (utile JEE)
    Exemple : `(* *->@UneAnnotation(..))`
- Des aspects réutilisables : exemple aspect de trace d'exécution.
- JBoss Application Server :
  - Architecture interne AOP
  - Persistance, transaction, sécurité sont des aspects
  - Possibilité d'ajouter de nouveaux aspects !

I never finish anyth