

AspectJ

Von Objektorientierung zu Aspektorientierung

Michael Kanis

TU-München

01.07.2008

Agenda

- Einleitung
- Was ist Aspektorientierung?
- Überblick AspectJ
- AspectJ in der Praxis
- Teilnehmerübung
- Vorstellung Demoprogramm
- Fragen

Einleitung

Aspektorientierung

Motivation

Aspekte

Weaving

AspectJ

Statische Joinpoints

Dynamische
Joinpoints

Praxis

Übung

Demo

Fazit

Das Problem

Software wird immer komplexer und damit schwerer wartbar.

Ziel

- Modularisierung
- Wiederverwendung
- Separation of Concerns
- **Bessere Software**

Modularisierung

Objektorientierung ist gut für Modularisierung, aber reicht oft nicht!

- Module werden von “querschnittlichen Belangen” durchsetzt, sog. **Cross-Cutting Concerns**
- Module enthalten verschiedene Belange (**Tangling**)
- Belange sind über Module verstreut (**Scattering**)

Aspekte

Aspektorientierte Programmierung (AOP) lagert Cross-Cutting Concerns in **Aspekte** aus und verbessert so die Wartbarkeit durch Modularisierung.

Example

```
public aspect FooBar {  
    ...  
}
```

Einleitung

Aspektorientierung

Motivation

Aspekte

Weaving

AspectJ

Statische Joinpoints

Dynamische
Joinpoints

Praxis

Übung

Demo

Fazit

Die getrennten Belange müssen wieder zusammengefügt werden!

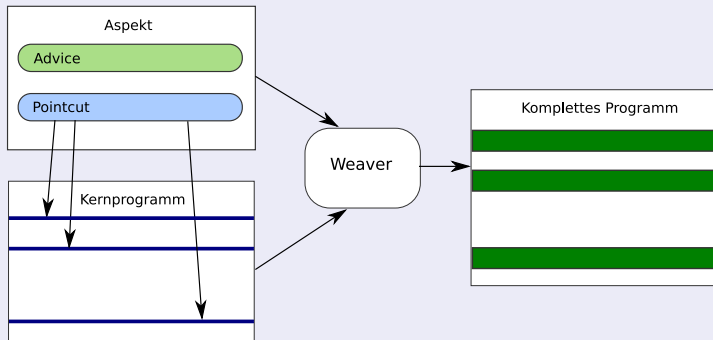
Strategien

- Precompiling (kommt bei AspectJ nicht zum Einsatz)
- **Compile Time Weaving** (direkt nach Compile)
- **Load Time Weaving** (beim Laden der Klasse)

Bestandteile von Aspekten

- **Joinpoints** sind die Menge der Programmstellen, an denen ein Aspekt eingewoben werden kann.
- **Pointcuts** sind eine Menge von Joinpoints. Sie bestimmen, an welchen Stellen im Java-Code ein bestimmter Advice ausgeführt wird.
- **Advices** beinhalten den einzuwebenden Code. Sie definieren also das Verhalten eines Aspektes.

Codeweaving



Das fertige Programm besteht aus Standard Bytecode und kann auf jeder JVM ausgeführt werden.

Einleitung

Aspektorientierung

Motivation

Aspekte

Weaving

AspectJ

Statische Joinpoints

Dynamische
Joinpoints

Praxis

Übung

Demo

Fazit

Statische Joinpoints

AspectJ

Michael Kanis

Einleitung

Aspektorientierung

Motivation

Aspekte

Weaving

AspectJ

Statische Joinpoints

Dynamische
Joinpoints

Praxis

Übung

Demo

Fazit

... werden immer zur Compile-Zeit ausgewertet.

Überblick

- Eigene Compiler-Warnings & Errors
- Verändern der Klassenhierarchie
- Klassen (oder Interfaces!) verändern

Compiler-Meldungen

- Zugriff auf bestimmte APIs unterbinden
- Zugriff zwischen Architekturschichten steuern

Example

```
public aspect NoDirectSql {
    declare error:
        withincode(* com.example.foo.*.*(..)) &&
        call(* java.sql.*.*(..)) :
            "SQL not allowed";

    declare warning: ...
}
```

Einleitung

Aspektorientierung

Motivation

Aspekte

Weaving

AspectJ

Statische Joinpoints

Dynamische
Joinpoints

Praxis

Übung

Demo

Fazit

Klassenhierarchie

- Interfaces und Superklassen in Hierarchie einfügen
- Nur eine Superklasse, aber mehrere Interfaces

Example

```
public aspect ParentExtender {  
    declare parents: C implements I, J, K;  
  
    declare parents: B extends A;  
}
```

Einleitung

Aspektorientierung

Motivation

Aspekte

Weaving

AspectJ

Statische Joinpoints

Dynamische
Joinpoints

Praxis

Übung

Demo

Fazit

Klassen erweitern

- Funktionalität von Klassen erweitern, ohne deren Quellcode zu verändern
- Standardimplementierung für Methoden von Interfaces

Example

```
public aspect MethodInjector {
    private String C.message = "Hello_world!";

    public void C.sayHello() {
        System.out.println(message);
    }
}
```

Einleitung

Aspektorientierung

Motivation

Aspekte

Weaving

AspectJ

Statische Joinpoints

Dynamische
Joinpoints

Praxis

Übung

Demo

Fazit

Dynamische Joinpoints

... werden direkt nach dem Compile-Vorgang (Compile-Time) oder beim Laden der Klasse (Load-Time) in den Bytecode eingewoben.

Überblick

- Pointcuts
- Advices

Syntax

```
pointcut Name (Parameter) : Pointcut Expression
```

Pointcut Expressions

- `call(Signature)` Methodenaufruf
- `execution(Signature)` Ausführung einer Methode
- `get(Signature) / set(Signature)` Feldzugriffe
- `handler(TypePattern)` Exception Handler (`call {}`)
- `this(Type or Id)` Aktuell ausführendes Objekt
- `target(Type or Id)` Zielobjekt
- `args(Type or Id, ...)` Argumente
- ...

Einleitung

Aspektorientierung

Motivation

Aspekte

Weaving

AspectJ

Statische Joinpoints

Dynamische
Joinpoints

Praxis

Übung

Demo

Fazit

Dynamische Joinpoints

Example

```
pointcut example():  
    call(void setText(String));
```

```
pointcut example(String s):  
    execution(void setText(String)) &&  
    args(s);
```

```
pointcut getter(): get(myVar);  
  
pointcut setter(int i): set(myVar) && args(i);
```

```
pointcut example():  
    handler(RuntimeException);
```

Einleitung

Aspektorientierung

Motivation

Aspekte

Weaving

AspectJ

Statische Joinpoints

Dynamische
Joinpoints

Praxis

Übung

Demo

Fazit

Dynamische Joinpoints

Syntax

```
Advice [ throws TypeList ] : Pointcut { Body }
```

Advices

- `before(Formals)`
- `after(Formals) returning [(Formal)]`
- `after(Formals) throwing [(Formal)]`
- `after(Formals)`
- `around(Formals)`

Wie kann ich AspectJ in der Praxis nutzen?

- Durchsetzen von Programmierrichtlinien
- **Logging / Tracing**
- **Performance-Messung**
- **Design by Contract**
- Design-Patterns
- Sicherheitsaspekte
- Synchronisation
- Bean-Bindings
- Caching

Einleitung

Aspektorientierung

Motivation

Aspekte

Weaving

AspectJ

Statische Joinpoints

Dynamische
Joinpoints

Praxis

Übung

Demo

Fazit

Tracing

- Alle Methodenaufrufe loggen

Example

```
before() : execution(* *.* (..)) {  
    System.out.println(  
        thisJoinPointStaticPart.getSignature());  
}
```

Performance-Messung

- Wo sind die Langläufer?

Example

```
Object around() : execution(* *.calc* (..)) {
    long start = System.currentTimeMillis();
    return proceed();
    long stop = System.currentTimeMillis();

    System.out.printf("%s: %dms",
        thisJoinPointStaticPart.getSignature(),
        (stop - start));
}
```

Einleitung

Aspektorientierung

Motivation

Aspekte

Weaving

AspectJ

Statische Joinpoints

Dynamische
Joinpoints

Praxis

Übung

Demo

Fazit

Design by Contract

- Vereinbarte Bedingungen zur Laufzeit prüfen

Example

```
public aspect NotNullPostCondition {
    pointcut notNull():
        call(@NotNull !void *(..));

    after() returning(Object o) : notNull() {
        String MSG = ...
        if (o == null) {
            throw new IllegalStateException(MSG);
        }
    }
}
```

Einleitung

Aspektorientierung

Motivation

Aspekte

Weaving

AspectJ

Statische Joinpoints

Dynamische
Joinpoints

Praxis

Übung

Demo

Fazit

Aufgabe

Im Workspace ist ein einfaches Programm, das mit Swing geometrische Figuren zeichnet. Die Ausgabe erfolgt ohne Kantenglättung.

- Schaltet Antialiasing an!
- Verändert **nicht** den bestehenden Code
- Verwendet **execution** Pointcut und **around** Advice

Problematik

- Swing ist nicht thread safe!
- SWT auch nicht, fängt aber falsche Zugriffe zur Laufzeit ab
- Ziel: ähnliches Verhalten für Swing, ohne dessen Code anzufassen
- Lösung: Aspekt mit before Advice

⇒ **Aspekt verändert nachträglich das Programm**

Vorgehen

- Überprüfung, ob der Thread aus dem eine Swing-Methode aufgerufen wird, EDT ist
- Wenn dies nicht der Fall ist, `IllegalStateException` (`RuntimeException`) werfen
- Dadurch wird die Ausführung gestoppt

Problematik

- Cache ist ein schneller Zwischenspeicher
- z.B. Ergebnisse langlaufender Berechnungen, Dateien von langsameren Speichern, Datenbankabfragen, ...
- Echter Cross-Cutting Concern
- Beispiel: cachen von Dateien, die auf Festplatte liegen

⇒ **Aspekt ist von Anfang an ins Design eingebunden**

Caching mit AspectJ

Vorgehen

- Pointcut auf Lade-Methode
- Überprüfung, ob Objekt im Cache
- Wenn ja, dieses zurückgeben und aufhören
- Sonst proceed(...)
- Ergebnis in den Cache legen

AspectJ rocks!

AspectJ erweitert Java um viele nützliche Features aus der aspektorientierten Programmierung. Sinnvoll eingesetzt

- unterstützen Sie den Entwickler
- verbessern die Lesbarkeit von Code und
- tragen somit zu besserer Software bei.

⇒ **Ausprobieren!**

Was noch?

- AJDT ist noch etwas buggy
- Alternativer AspectJ Compiler AspectBench (abc)
- Alternativen zu AspectJ (z.B. Spring AOP, JBoss AOP, ObjectTeams)
- Aspektorientierung auch für andere Sprachen (C, C++, Cocoa, Python, PHP, Ruby, ...)