FrOSCon
Free and Open Source Software Conference

anderScore
trust in competence

Project Lombok in Action

25.08.2012, Vitaly Markin

whoami

anderScore trust in competence

- Vitaly Markin
- Diplom-Informatiker (Uni Bonn)
- Seit > 12 Jahren Java-Programmierer
- Software Engineer bei anderScore GmbH
  - Unternehmen mit ~20 Mitarbeitern
  - Entwicklung von Individual-Software im Rahmen von Kundenprojekten
  - Focus Java-Technologien
  - Agile Vorgehensweise
  - meist Großunternehmen als Kunden

Mail: vitaly.markin@anderscore.com

Inhalt

anderScore trust in competence

Longbock?

Quelle: © http://fcfanpage.fc.funpic.de

Longbock != Lombok

Quelle: © http://fcfanpage.fc.funpic.de

anderScore trust in competence

Longbow?

Quelle: © http://de.wikipedia.org

anderScore trust in competence

Longbow != Lombok

Quelle: © http://de.wikipedia.org

Einführung

- Lombok = Indonesische Insel
- Bei-nahe Java, aber eben nicht ganz!

Quelle: © Google Maps

Project Lombok

- Was ist es nun wirklich?

→OpenSource Projekt* gegen
(unnötige) Geschwätzigkeit
von Java

→Schlanke syntaktische
Erweiterung der Sprache

* Gestartet von Reinier Zwitserloot, Roel Spilker

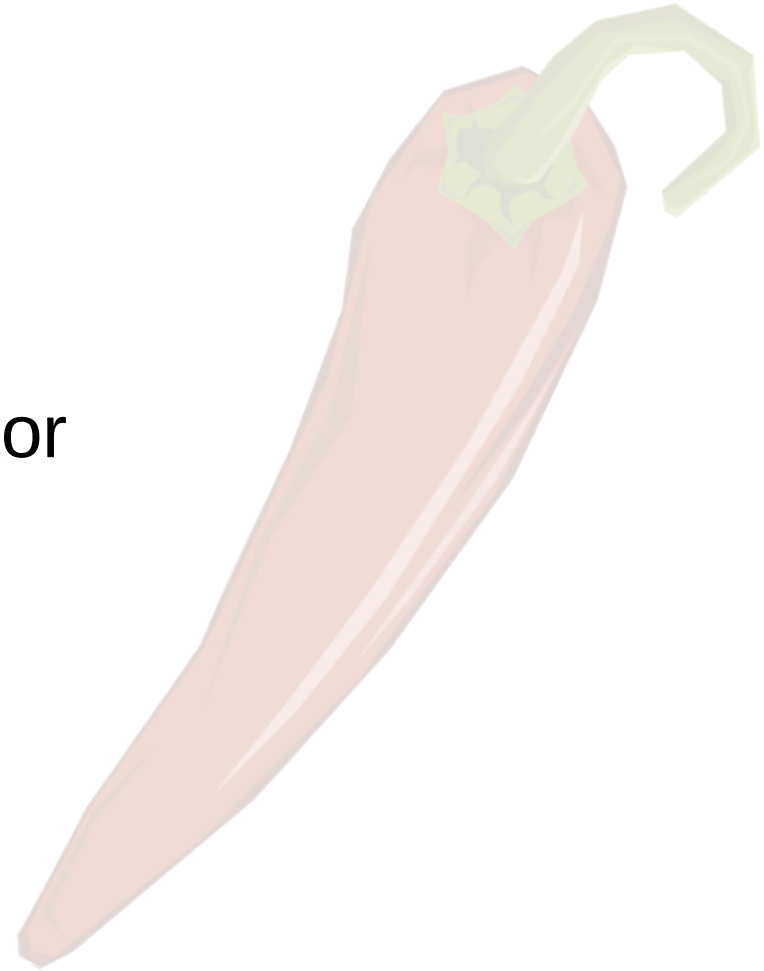
Project Lombok

anderScore trust in competence

- Was ist es nun wirklich?

→Compilezeit-Generator

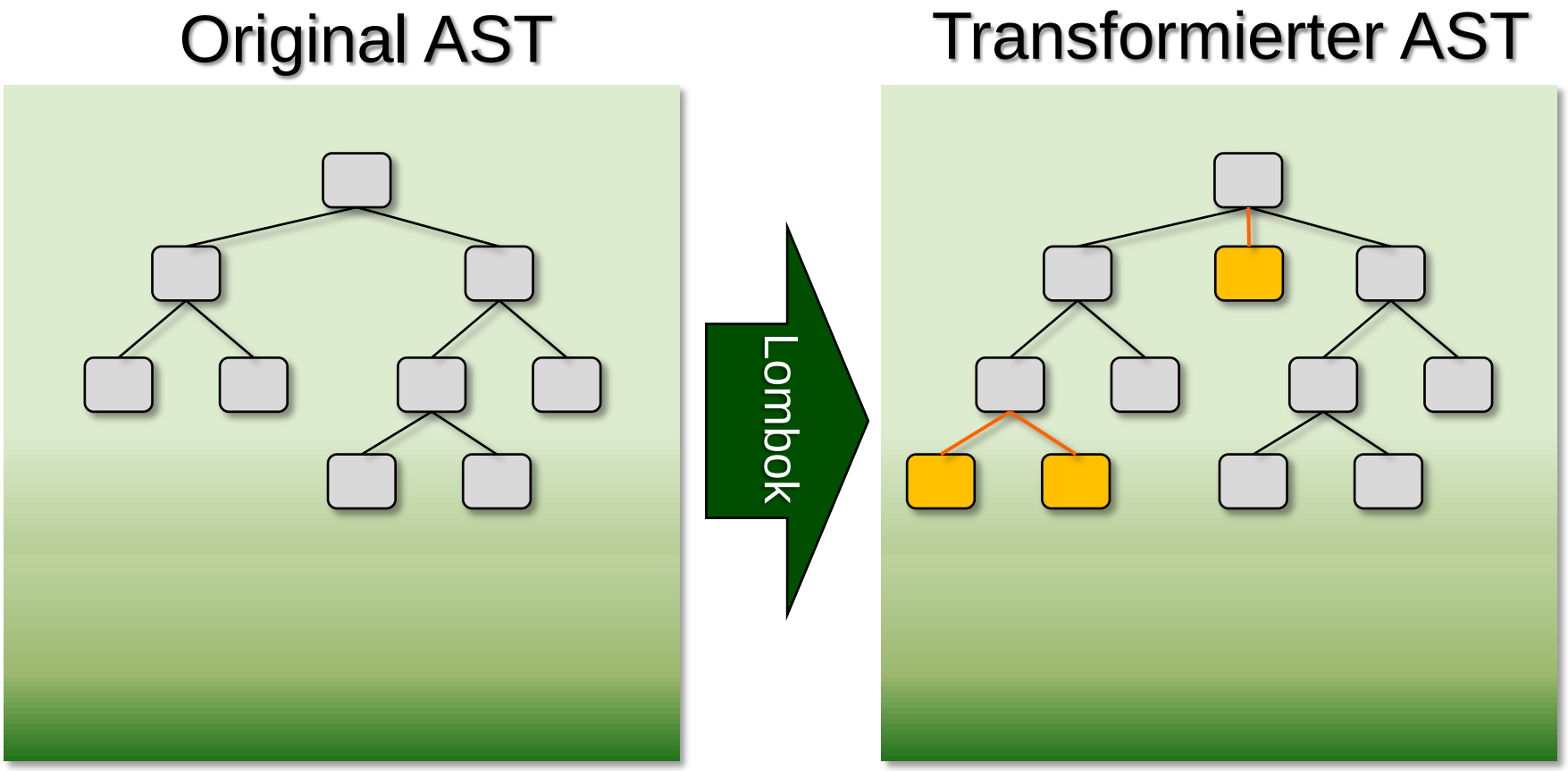
→„Entwicklungszeit“-Generator



Compilezeit-Generator

anderScore trust in competence

→ AST-Tranformator



„Entwicklungszeit“-Generator

anderScore trust in competence

- Für die Eclipse IDE
- NetBeans (eingeschr.)

```
@Data
public class Person {
  private String firstName;
  private String lastName;
  private int age;
}
```

Outline

Features: Überblick

anderScore trust in competence

- Annotationen zur Reduktion von Boilerplate Code
- „De-lombok“ – Funktion, um standard Java-Code zu erzeugen
- Framework für eigene Erweiterungen
  - Eclipse & javac AST-Transformationen

Getters / Setters

anderScore trust in competence

- @Getter / @Setter
  - Generierung von Standard-Getters / Setters
  - Anwendung auf (static) Felder, Klassen
  - AccessLevel Argument
- @Getter(lazy=true)
  - Initializierung beim ersten Aufruf
  - Anwendung auf **private final** Felder
- Vorteil: Resistent gegenüber Renaming-Refactorings

Constructors

- @NoArgsConstructor
  - Leerer Default-Konstruktor
- @RequiredArgsConstructor
  - mit allen nicht initialisierten final Feldern
  - mit allen als @NonNull gekennzeichneten Feldern
- @AllArgsConstructor
  - mit allen nicht initialisierten Feldern
- staticConstructor="of"

Anwendung auf Klassen

Contract & Convenience

anderScore trust in competence

- @EqualsAndHashCode
  - exclude={"xxx"}
  - callSuper=true
  - Anwendung auf Klassen
- @ToString
  - includeFieldNames
  - exclude={"xxx"}
  - callSuper=true
  - Anwendung auf Klassen

Mr. @Data

anderScore trust in competence

- Und nun alles zusammen...
- Implizit für
  - @Getter/@Setter für alle Felder
  - @EqualsAndHashCode für alle Felder
  - @ToString
  - @NoArgConstructor
  - Anwendung auf Klassen
- staticConstructor="of"
- Explizites Overriding mit Annotations an einzelnen Feldern möglich

Log it, log it

anderScore trust in competence

- @Log → Erzeugt **private static final java.util.logging.Logger log = java.util.logging.Logger.getLogger(Employee.class.getName());**
- @Log4j → Erzeugt **private static final org.apache.log4j.Logger log = org.apache.log4j.Logger.getLogger(Employee.class);**

- Weitere
  - @Slf4j → slf4j - Logger
  - @CommonsLog → commons Logger

```
@Log
public class Employee extends Person {
                    ...
  private double calculateSalary() {
          ...
    log.warning("If you see this, everything is lost.");
          ...
  }
}
```

Don‘t hesitate, @Delegate

anderScore trust in competence

```
public class Foo {

                ...
  @Delegate
  private Collection<Employee> colleagues = new ArrayList<Employee>();
                ...

}
```

- Stellt alle öffentlichen Methoden der Klasse 'Collection' der Klasse 'Foo' zur Verfügung
- Leitet alle Aufrufe an das annotierte Feld weiter

Don‘t hesitate, @Delegate

anderScore trust in competence

```
public class Foo {

  @Delegate(types=SimpleAddRemove.class)
  private Collection<Employee> colleagues = new ArrayList<Employee>();
                          ...
  private interface SimpleAddRemove {
    public void add(Employee c);
    public void remove(Employee c);
  }
                          ...
}
```

- Include-Filter: Weiterleitung nur der im Interface `SimpleAddRemove` aufgeführter Methoden mit passender Signatur

Don‘t hesitate, @Delegate

anderScore trust in competence

```
@Delegate(types=SimpleAddRemove.class, excludes={SimpleAddAll.class})
private Collection<Employee> colleagues = new ArrayList<Employee>();
                              ...
private interface SimpleAddRemove {
  public void add(Employee c);
  public void remove(Employee c);
  public void addAll(Collection<Employee> all);
}

private class SimpleAddAll {
  public void addAll(Collection<Employee> all) {
    // irgendeine Implementierung
  }
}
```

- Exclude-Filter: Keine Weiterleitung der in der Klasse 'SimpleAddAll' aufgeführte Methoden

Du hast die **val**

anderScore trust in competence

- Neues „Schlüsselwort“: **val**
- Verzicht auf Typdeklaration für alle lokalen **final** Variablen

```
val someMap = new HashMap<String, Object>();
```

vs.

```
final HashMap<String, Object> someMap2 = new HashMap<String, Object>();
```

De-lombok

- Zurück zum WYSIWYG
- Maven Plug-in 3rd Party / Ant Build-Task
- Commandline Tool

Vor- / Nachteile und Grenzen

anderScore trust in competence

Vorteile

- Sauberere Datenklassen
- Schnellere Entwicklung
- GWT-Unterstützung
- Renaming von Feldern = Renaming von Getters/Setters
- Maven & Ant+Ivy Integration
- Eigene AST-Transformationen möglich

Nachteile / Grenzen

- Kein WYSIWYG mehr
- Keine IDEA-Unterstüzung
  - Nur experimentelle NetBeans-Unterstüzung
- Debugging erschwert
- (noch) keine eine copy-Getters für Immutables
- kein callSuper in Konstruktoren

Ausblick

anderScore trust in competence

- Projekt wird ständig weiterentwickelt
- Experimentelle Features
- Möglichkeit, eigene Features zu implementieren
  - z.Z. getrennt für EJC und javac
  - Abstraktes Framework für AST-Transformationen in Aussicht

(Haus-)Aufgabe

anderScore trust in competence

- Erstellen Sie eine Klasse '**SmallCompany**' mit folgenden Properties:
  - 'name' (Pflichtfeld)
  - 'anticipatedRevenue' (Mit Lazy-Getter und einer beliebig komplizierten Formel zur Berechnung des erw. Umsatzes
  - Liste von Angestellten ('employees')
    - Stellen Sie SmallCompany die Methoden 'add' und 'remove' unter Verwendung von @Delegate zur Verfügung
- Lassen Sie Lombok einen Konstruktor für alle Pflichtfelder von SmallCompany erzeugen
- Versehen Sie in der Klasse 'Person' die Felder 'firstName' und 'lastName' mit der Annotation @NonNull
  - Welches Problem ergibt sich dadurch?
  - Wie könnte man es beheben?

Dowload Beispielprojekte

anderScore trust in competence

- Maven Projekt für Eclipse

http://ftp2.anderscore.com/froscon/ProjectLombok/maven.zip

- Einfaches Eclipse Projekt

http://ftp2.anderscore.com/froscon/ProjectLombok/plain.zip