

Flexible Codemodifikation im Elucidative Programming

Martin Heinzerling

TU Dresden, Fakultät Informatik

22. Oktober 2009

Inhaltsverzeichnis

Einleitung

IST-Zustand

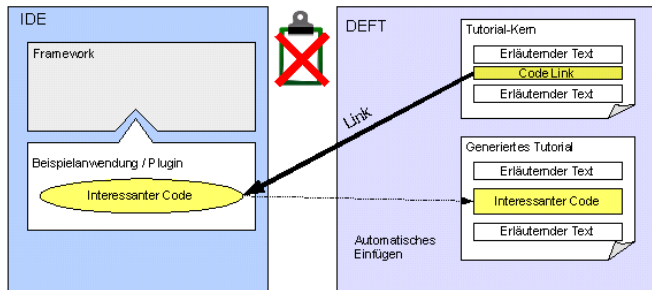
Erweiterungen

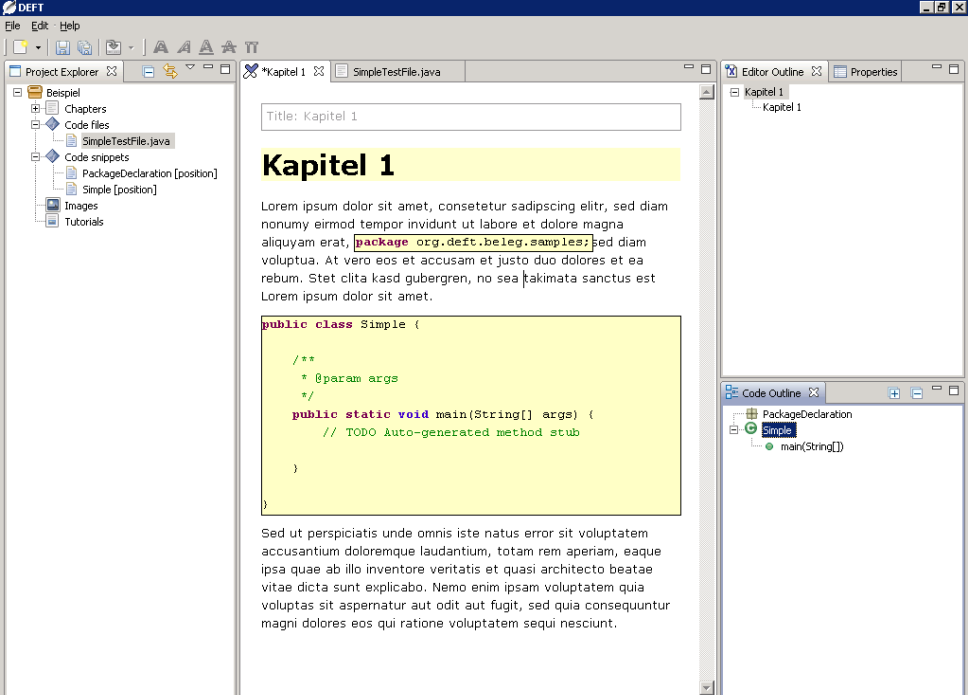
Umsetzung

Ausblick

Appendix

Elucidative Programming





Kapitel 1

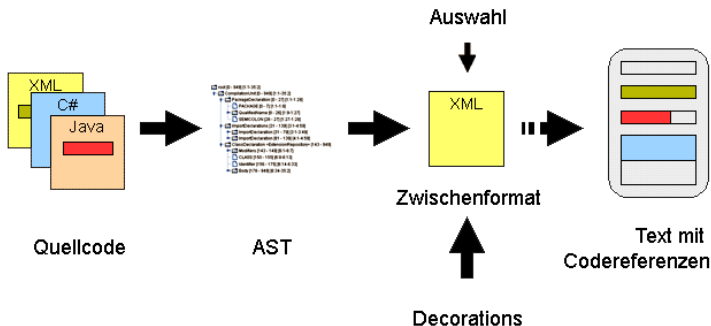
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, **package org.deft.beleg.samples;** sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

```
public class Simple {  
  
    /**  
  
     * @param args  
  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
    }  
}
```

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt.

```
package org.deft.beleg.samples;  
  
public class Simple {  
  
    /**  
  
     * @param args  
  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
    }  
}
```

Architektur



Zeilennummerierung

```
3 public class Simple {  
4  
5     /**  
6      * @param args  
7      */  
8     public static void main(String[] args) {  
9         // TODO Auto-generated method stub  
10  
11     }  
12  
13 }
```

Ausblenden von Codeabschnitten

```
public class Simple { ... }
```

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

```
public class Simple {  
  
    public static void main(String[] )  
  
}
```


Hervorhebungen

```
public class Simple {  
  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
    }  
}
```

Anfügen und Ersetzen

```
public class Simple {  
  
    public static void main(String[] args) {  
        System.out.println("A");  
        System.out.println("B");  
        System.out.println("C");  
  
        notImportant();  
    }  
}
```



```
public class Simple {  
    //New comment  
    public static void main(String[] args) {  
        someOutput();  
  
        ...  
    }  
}
```

Entfernen von unerwünschten Leerzeichen

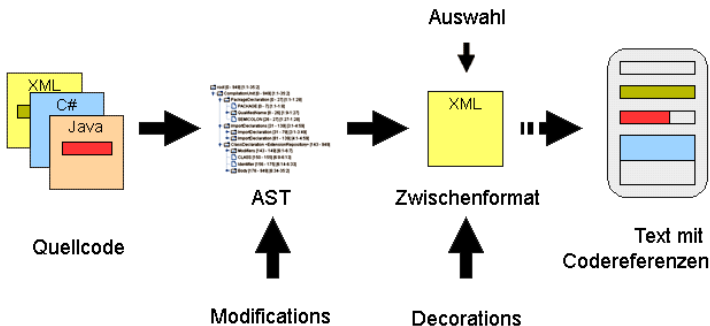
```
public class Simple {  
  
    public void main(String[] args)  
}
```

```
public class Simple {  
  
    public static void main(String[] )  
}
```

Zusammenfassung

- ▶ Codeformatierung in beliebiger Auszeichnungssprache (z.B. CSS)
- ▶ Einfügen von Token/ Freitext/ Markern vor/ nach beliebigen Token bzw. Ersetzung von Token
- ▶ Markierung von invaliden Code für weitere Verarbeitung
- ▶ Automatisches entfernen unnötiger Leerzeichen
- ▶ Einfügen/ Entfernen von Leerzeichen/ Leerzeilen/ Zeilenumbrüchen
- ▶ Umwandeln von Tokenlisten in Kommentare

Architektur



ASTLayouter

- ▶ Einfügen einer Anzahl von Leerzeichen vor/nach einem Token
- ▶ Entfernen aller Leerzeichen vor/nach einem Token
- ▶ Einfügen einer Anzahl von Leerzeilen vor/nach einem Token
- ▶ Entfernen aller Leerzeilen vor/nach einem Token
- ▶ Einfügen eines Zeilenumbruchs innerhalb einer Zeile vor/nach einem Token
- ▶ Entfernen eines Zeilenumbruchs vor/nach einem Token
- ▶ Einrückung einer Menge von Token

ASTModifier

- ▶ Einfügen von Knoten oder Teilbäumen vor/nach einem Token
- ▶ Ersetzen eines einzelnen oder einer Liste von Knoten
- ▶ Umwandeln von Knoten in Kommentartoken

Beispiel

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<codesnippet>
  <modifications>
    <modification class="org.deft...ASTLayouterInvoker">
      <param name="method">insertSpacesAfter</param>
      <param name="location">/CompilationUnit/PackageDeclaration/PACKAGE</param>
      <param name="offset">3</param>
    </modification>

    <modification class="org.deft...ASTLayouterInvoker">
      <param name="method">trim</param>
      <param name="location">/CompilationUnit/ClassDeclaration/Modifiers</param>
    </modification>

    <modification class="org.deft...ASTModifierInvoker">
      <param name="method">appendAfter</param>
      <param name="insert">resources/SimpleTestFile.java|||/PACKAGE</param>
      <param name="location">/CompilationUnit/PackageDeclaration/SEMICOLON</param>
    </modification>
    ...
  </modifications>
</codesnippet>
```

Beispiel

```
...
<modification class="org.deft...ASTModifierInvoker">
  <param name="method">changeToComment</param>
  <param name="location">/CompilationUnit/ClassDeclaration/Body/RBRACE</param>
  <param name="commentType">org.deft...JavaBlockCommentVisitor</param>
</modification>
</modifications>

<decorations>
  <decoration class="org.deft...StyleDecoratorInvoker">
    <param name="location">//Identifier</param>
    <param name="container">org.deft...CssDeclarationStyleContainer</param>
    <param name="value">font-weight:bold;</param>
  </decoration>
</decorations>
</codesnippet>
```

Beispiel

```
package org.deft.beleg.samples;

public class Simple {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }
}
```

```
package    org.deft.beleg.samples;package

publicclass Simple {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }
}*/+/*
```

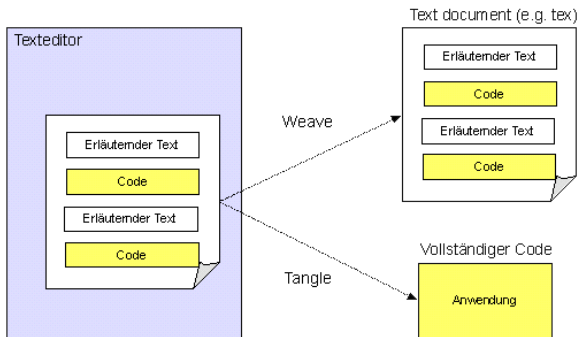
Ausblick

- ▶ Integration in aktuelle DEFT-Version
- ▶ Abstraktion auf allgemeinere Artefakte

Vielen Dank!

Download unter
<http://www.mheinzerling.de/tud.php>

Literate Programming



Schutz von invaliden Abschnitten

```
public class Simple { ... }
```

```
public class Simple {  
  
    public static void main(String[] )  
  
}
```

Zusammenziehen von Zeilen

```
public static void main(String[] args) {  
    ...  
  
    t=a;  
    a=b;  
    b=t;  
}
```

```
public static void main(String[] args) {  
    ...  
  
    t=a; a=b; b=t;  
}
```


Autotrimmer für Java 1.6

```
<?xml version="1.0" encoding="utf-8"?>
<autotrim language="Java" version="1.6">
  <!-- h=none/left/right/both v=none/before/after/both -->
  <trim h="left" v="none">
    ABSTRACT, FINAL, STATIC, SYNCHRONIZED, TRANSIENT, STRICTFP, VOLATILE, NATIVE,
    VOID, CONST, BYTE, SHORT, CHAR, INT, LONG, FLOAT, DOUBLE, BOOLEAN,
    CLASS, INTERFACE, ENUM, EXTENDS, IMPLEMENTS,
    ASSIGN, EXCLAM, AMPERSAND, PIPE, LOGICALAND, LOGICALOR,
    ...
    NULL, TRUE, FALSE, THROWS, THROW, NEW, INSTANCEOF, ASSERT,
    Identifier, VariableDeclarationFragment, SingleVariableDeclaration
  </trim>
  <trim h="right" v="none">
    PACKAGE, IMPORT, PRIVATE, PROTECTED, PUBLIC, Modifiers, SimpleType,
    IF, WHILE, SWITCH, FOR, TRY, CASE, ELSE, RETURN, DO, GOTO, CATCH, FINALLY, DEFAULT,
  </trim>
  <!-- Kann weggelassen werden -->
  <trim h="none" v="none">
    Body, Parameters, LPAREN, RPAREN, LBRACE, RBRACE, LBRACKET, RBRACKET,
    Annotations, ...
    CompilationUnit, InterfaceDeclaration, ...
    IfStatement, IfSection, ElseSection, SwitchStatement, ...

    AT, COMMA, DOT, SEMICOLON, ELLIPSE, QUESTION, COLON, ...
  </trim>
</autotrim>
```