

Wuja

**Eine Java Erweiterung für
Baumtransformationen**

Was ist Wuja?

- Ein Java Präprozessor
- Ein Tool, um baumartige Strukturen zu bearbeiten
- Ein möglicher Ersatz für XSLT

Wo kann man Wuja einsetzen?

- abstrakte Syntaxbäume
- DOM
- TypedDOM
- jeder Baum der `wuja.tree.Tree` implementiert

Was kann man mit Wuja machen ?

- In Bäumen suchen
- Beim Traversieren neue Bäume erzeugen

.js1t Dateien sind normale .java Dateien, die
Möglichkeit Regeln anzugeben erweitert wur

Das Rule Konstrukt

```
Trafo test = rule {  
  pattern{ [...] }  
  condition(X) { [...] }  
  body { [...] }  
};
```

Der pattern Teil:

- Spezifikation der gesuchten Baumstruktur.
- Variablenzuweisungen.
- Optionale Typkonvertierung der Variablen.
- **Beispiel:** `t = @"PLAY" \ @"TITLE" \ x = (Node) @`

Die `condition` Blöcke (I):

- Conditions sind optional.
- Conditions müssen die verwendeten Variablen im Scope haben.
- Conditions müssen ein `boolean` zurückliefern.
- Die Ergebnisse der Conditions werden `geANDet`.
- Conditions können alles machen was eine normale Methode darf.

Die condition Blöcke (II):

Beispiel:

```
condition(x) {  
    if(x.getNodeValue().length() > 1) {  
        return(true);  
    } else {  
        return(false);  
    }  
}
```

Der body:

- Wird aufgerufen wenn das Pattern matched und `conditions true` geliefert haben.
- Alle Variablen aus dem Pattern stehen mit dem `name` zur Verfügung.
- Der Body muß ein Objekt vom Typ `wuja.tree.Tree` liefern.

Anwendung der Transformation:

- Eine `rule` wird zu einem `Trafo` Objekt übersetzt
- `wuja.compiler.Trafo` bietet drei Methoden:
 - `Tree apply(Tree t)` transformiert einen Baum
 - `Tree[] applyAll(Tree t)` liefert alle transformierten Bäume in einem Array.
 - `Stream applyLazyAll(Tree t)` liefert einen Stream, der alle transformierten Bäume liefert.

Praktisches Beispiel

Operator	Description
<i>element</i>	Klassenname
@' ' <i>element</i> ' '	Beliebiger String
\	Pfadsuche
\\	Tiefe Pfadsuche
<i>X=pattern</i>	Variablenbindung
<i>X=(type)pattern</i>	Variablenbindung mit
?	Jokerzeichen
[<i>pattern-list</i>]	Liste von Pattern
<i>element</i> *	Findet <i>pattern</i> belieb
<i>element</i> +	Findet <i>pattern</i> belieb mindestens ein Mal
<i>pattern</i> , <i>pattern-list</i>	Sequenz von Pattern
(<i>element</i> <i>element</i> ...)	Alternative

Der Matcher

- ClassNodeMatcher / StringNodeMatcher
- VarNodeMatcher
- PathMatcher / DeepPathMatcher
- ChildlistMatcher

```
public abstract class Matcher {  
    public abstract Stream match (VarSet varSet, Tre  
}
```

```
public abstract class NodeMatcher extends Matcher
    public NodeMatcher ();
    public NodeMatcher (String name);
    public NodeMatcher (ChildlistMatcher childmatcher
    public NodeMatcher (String name, ChildlistMatcher
}
```

```
public class VarNodeMatcher extends Matcher {  
    public VarNodeMatcher (int index, NodeMatcher node)  
}
```

Der ChildlistMatcher

Greedy Listenkonsumierer (globbing)

- eatItem (optionale Zuweisung)
- eatWhile
- eatUntil
- alternation (optionale Zuweisung des ersten Tre

Es werden immer NodeMatcher konsumiert.

Der Compiler (wujac)

```
int i = 0;
Trafo t = rule{
    pattern{
        y = @"html" \\ x = (Node)@"img"
    }
    condition(x){
        return 0 == 0;
    }
    condition(x, y){
        return 1 == 1;
    }
    body{
        i++;
        return null;
    }
};
t.applyAll(myTree);
```

Übersetzung von Hand

```
new wuja.compiler.Trafo(Pattern, new wuja.matcher.  
    Conditions  
    Body  
});
```

Das Pattern

```
new wuja.matcher.DeepPathMatcher(  
    new wuja.matcher.VarNodeMatcher(  
        0,  
        new wuja.matcher.StringNodeMatcher("html")  
    ),  
    new wuja.matcher.VarNodeMatcher(  
        1,  
        new wuja.matcher.StringNodeMatcher("img")  
    )  
)
```

Die Conditions

```
private boolean condition_0(Node x){
    return 0==0;
}
private boolean condition_1(wuja.tree.Tree y, Node
    return 1==1;
}
public boolean testVariable(int index, wuja.tree.T
    if(index == 1){// x
        return true && condition_0((Node)value);
    }
    if(index == 0){// y
        return true;
    }
    return true;
}
```

Der Body

```
public wuja.tree.Tree apply(){
    Node x = (Node)vars[1];
    wuja.tree.Tree y = (wuja.tree.Tree)vars[0];
    if(!condition_1(y, x))
        throw new ConditionFailedException("condit
    i++;
    return null;
}
```

Ende