



Motivationsvortrag :) Sopra

Christoph Schied, Philipp Güttler | 2010-02-04 | Universität Ulm,
Abt. SGI

Progwerkstatt

Programmorschlag

- Design Pattern

 - Observer Pattern

 - Model-View-Controller (MVC)

- GUI

- Drucken und PDF-Export in Java

 - Java Print Service API

 - Apache FOP

- Koordination des Teams untereinander

 - Coding Standards

 - Versionskontrolle

- Qualitätssicherung

 - FindBugs

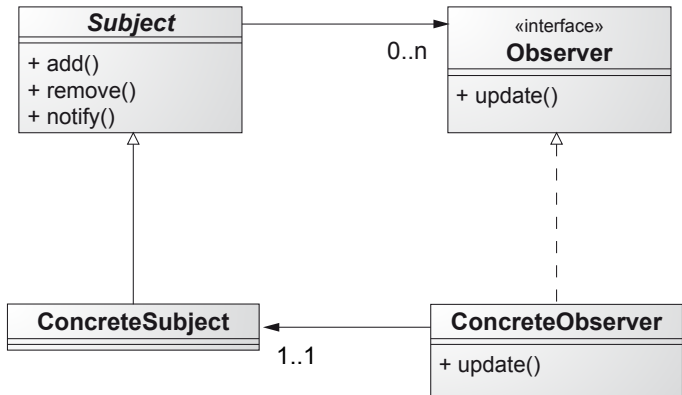
 - Modultests

Design Patterns

- ▶ bewährte Lösung für wiederkehrendes Problem
- ▶ *Creational Patterns*
 - ▶ Erzeugung von Objekten
 - ▶ Bsp: Singleton, Factory, Builder, Prototype. . .
- ▶ *Structural Patterns*
 - ▶ Aufbau von Klassen- und Objektstrukturen
 - ▶ Bsp: Adapter, Proxy, Composition
- ▶ *Behavioral Patterns*
 - ▶ Kommunikation zwischen Objekten
 - ▶ Bsp: Observer, Iterator
- ▶ Architectural Patterns, Antipatterns, Bugpatterns. . .

Observer Pattern

- ▶ ein Subjekt benachrichtigt gelistete Objekte über Veränderung

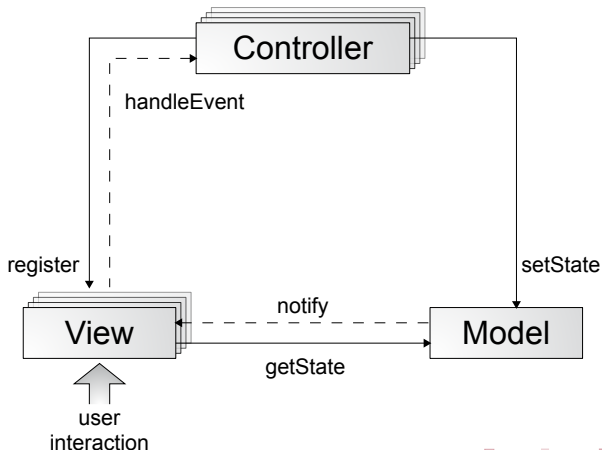


Observer Pattern

- ▶ Nutzen
 - ▶ automatische Aktualisierung
 - ▶ unabhängige Variation von Subjekten und Beobachtern
 - ▶ Referenzimplementierung: `java.util.Observable`,
`java.util.Observer`
- ▶ Probleme
 - ▶ Kommunikationskosten
 - ▶ Schleifengefahr
 - ▶ keine Information über konkrete Änderung

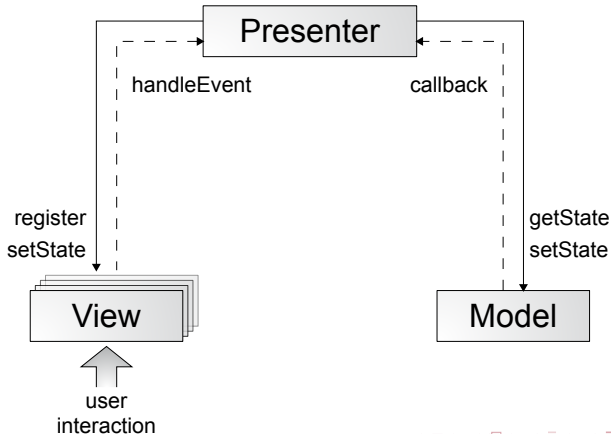
Model-View-Controller (MVC)

- Trennung von Geschäftslogik, Präsentation und Interaktion



Alternative Ansätze (MVC/MVP)

- ▶ teilweise unterschiedliche Notation und Modellierung

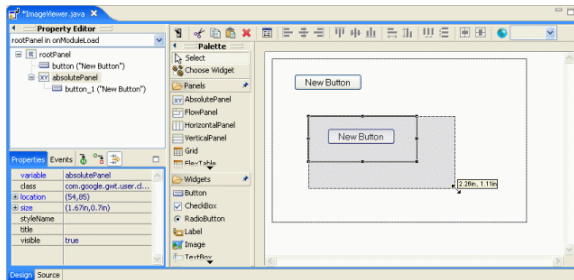


Richtlinien für gute GUIs

- ▶ Keine hartkodierte Größenangaben
- ▶ Vorhandene Funktionalität des Toolkits nutzen
- ▶ Andere Programme analysieren und sich inspirieren lassen
- ▶ Typische GUI Konventionen umsetzen
- ▶ Benutzertests durchführen
- ▶ Sich ans MVC-Pattern erinnern

GUI Editoren

- ▶ GUI zusammen klicken, WYSIWYG
- ▶ zwei Arten von GUI Editoren
 - ▶ Code-generierung
 - ▶ Beschreibung von GUI (XML)
- ▶ Keine Hilfe bei dynamischen GUIs
- ▶ Grosse Auswahl, Suchmaschine der Wahl quälen



Java Print Service API - javax.print

1. Suche nach Druckdiensten (z.B. Drucker, PDF, XPS, etc.)
 - ▶ `PrintServiceLookup` erlaubt *gezielte* Suche
 - ▶ `DocFlavor` definiert Datenformat (MIME und Klasse)
 - ▶ `AttributeSet` setzt Ausgabeformat (Ausrichtung)
 - ▶ `ServiceUI.printDialog(..)`: Auswahl durch Benutzer

2. Druckauftrag erzeugen

```
DocPrintJob job = service.createPrintJob();
```

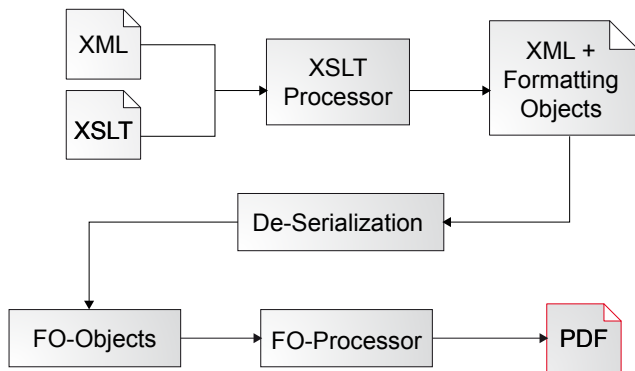
3. Datenhandler erzeugen

```
SimpleDoc doc = new SimpleDoc(docObj, flavor, attribs);
```

4. Drucken! `job.print(doc, attribs);`

Apache FOP

- Umwandlung von XSL-FO in *PDF*, *PNG*, *RTF* etc.



Koordination

- ▶ Nutzt die Macht der Technik
 - ▶ Versionskontrolle
 - ▶ Multi-User Chat (jabber, irc, ...)
 - ▶ Bug/Task-Tracker
 - ▶ Wiki
 - ▶ Teamlaufwerke bei der SGI
- ▶ Häufige Treffen
- ▶ Austausch mit anderen Teams



[Login](#) | [Settings](#) | [Help/Guide](#) | [About Trac](#)

Wiki | Timeline | Roadmap | Browse Source | View Tickets | New Ticket | Search

This report: [Edit](#) | [Copy](#) | [Delete](#) | [New Report](#) | [Custom Query](#)

{9} Time Tracking (7 matches)

Ticket	Planned	Spent	Remaining	Accuracy	Customer	Summary	Component	Status
#6	10h		10h	0.0	milestone1	asdf	component1	new
#5	2h	4h	0h	2.0	milestone1	234	component1	new
#4				0.0	milestone1	yxcv	component1	new
#3	4h	4h		0.0	milestone1	test3	component1	closed
#2	4h	2h	2h	0.0	milestone1	test2	component1	new
#1	8h	7.0h	3.0h	2.0	milestone1	test 1	component1	new

Coding Standards

```

class Div{static $1 $_;class $1{void _$(String
$_){System.out. print($_);}$1(){_();}void _(){int
_, $, _$, $$, __, ä=(1<<5), å=100, ö=12, åö=ä*ö;å-=1<<4;
while(åö>0){for($$=_$_=__$_=(int)å;_$_>($$-(1<<2))
;_$( ""+(char)(_$_), _$_-=1<<1)for(_$_=__$_-9;_>($-6);
_-=1<<1, _$( ""+(char)(_$_!=å?_:ä));char S$=(char)
(å+ö+1);_$( "te"+S$+(char)(ö+(int)S$));åö--;}}Div
(){$_=new $1();} public static void main(String
[]$) {Div å=new Div();}}

```

Coding Standards

- ▶ Auf Sprache einigen
- ▶ Keine Rechtschreibfehler bei Bezeichnern
- ▶ Code-Formatierungsstil festlegen und einhalten
- ▶ Sinnvoll kommentieren
- ▶ Dokumentation
- ▶ Suns Java-Styleguide


JavaDoc

- ▶ Dokumentationswerkzeug für Java Quelltexte
- ▶ Generiert HTML Dokumentation
- ▶ Dokumentiert Klassen sowie Funktionen und deren Parameter
- ▶ Eclipse zeigt JavaDoc bei Autovervollständigung an
- ▶ Siehe JavaDoc-Referenz für genaue Beschreibung der Tags

```

1  /**
2   * Implements a List
3   * @author Hans Dampf
4   */
5  class List {
6     /**
7      * Print out the list
8      * @param limit max amount of printed Elements
9      * @return number of printed Elements
10     */
11     int printList(int limit) {

```



Versionskontrolle

- ▶ Protokolliert Änderungen in Dokumenten
- ▶ Fügt Änderungen der Nutzer zusammen
- ▶ Repository repräsentiert aktuellen Stand
- ▶ Hilft Fehler auf Commits zurückzuführen
- ▶ Verschiedene Modelle:
 - ▶ Zentrale Versionsverwaltung (cvs, svn, perforce, ...)
 - ▶ Verteilte Versionsverwaltung (git, mercurial, ...)

Versionskontrolle - Good Practices

- ▶ NIEMALS nicht-kompilierenden Code einchecken
- ▶ Commit ist kein Save-Button
- ▶ Ordentliche Commit-Nachrichten schreiben
- ▶ Commits in sinnvolle Teile aufspalten

Qualität

- ▶ EN ISO 8402
 - ▶ *“Qualität ist die Gesamtheit von Merkmalen einer Einheit bezüglich ihrer Eignung, festgelegte und vorausgesetzte Erfordernisse zu erfüllen.”*
- ⇒ Programmierung
- ▶ Unterstützung durch IDE, Debugger
 - ▶ Spezielle Programme, z.B. FindBugs
 - ▶ Modultests, z.B. mit JUnit-Framework
 - ▶ Code Reviews, Bots, Usability Evaluierung ...

The screenshot shows the FindBugs application window. The top menu bar includes File, Edit, Navigation, Designation, and Help. Below the menu is a toolbar with icons for Package, Priority, Category, Bug Kind, and Bug Pattern. The left pane displays a project tree with the following structure:

- edu.umd.cs.findbugs.config (3)
- edu.umd.cs.findbugs.filter (1)
- edu.umd.cs.findbugs.util (1)
 - Medium (1)
 - Bad practice (1)
 - Stream not closed on all paths (1)
 - Method may fail to close stream (1)
 - edu.umd.cs.findbugs.util.Util.getXMLType (1) - **Selected**
- edu.umd.cs.findbugs.visitclass (1)
- edu.umd.cs.findbugs.workflow (2)
- java.util (2)

The right pane shows the source code for `Util.java` in `edu.umd.cs.findbugs.util`. Line 108 is highlighted in yellow:

```

97         assert true;
98     }
99 }
100     static final Pattern tag = Pattern.compile("^\\s*<\\w+\"
101     public static String getXMLType(InputStream in) throws IO
102         if (!in.markSupported())
103             throw new IllegalArgumentException("Input stream
104
105     in.mark(5000);
106     BufferedReader r = null;
107     try {
108         r = new BufferedReader(Util.getReader(in), 2000);
109
110     String s;
111     int count = 0;
112     while (count < 4) {
113         s = r.readLine();
114         if (s == null)
115             break;
116         Matcher m = tag.matcher(s);
117         if (m.matches())
  
```

Below the code editor is a search bar with buttons for Find, Find Next, and Find Previous. The bottom pane displays the following error message:

edu.umd.cs.findbugs.util.Util.getXMLType(InputStream) may fail to close stream
 At Util.java:[line 108]
 In method edu.umd.cs.findbugs.util.Util.getXMLType(InputStream) [Lines 102 - 123]
 Need to close java.io.Reader

Method may fail to close stream
 The method creates an IO stream object, does not assign it to any fields, pass it to other methods that might close it, or return it, and does not appear to close the stream on all paths out of the method. This may result in a file descriptor leak. It is generally a good idea to use a `finally` block to ensure that streams are closed.

<http://findbugs.sourceforge.net/>

UNIVERSITY OF MARYLAND

Abbildung: FindBugs GUI

Modultests

- ▶ einzelne Methoden auf Funktionalität testen
- ▶ Wird erwarteter Output bei definiertem Input erzeugt?
- ▶ unnötig für triviale Methoden
- ▶ einzelne Tests zusammenfassen
- ▶ Ablauf automatisieren
- ▶ ersetzt nicht User Tests

Modultests mit JUnit

```
1 public class SopraTeamTester {
2     @org.junit.Before
3     public void setUp() { ... }
4
5     @org.junit.Test
6     public void testTeamMergeCount() {
7         assertTrue(merged.size() ==
8                     team1.size() + team2.size());
9     }
10
11     public static void main(String[] args) {
12         JUnitCore.main(SopraTeamTester.class.getName());
13     }
14 }
```