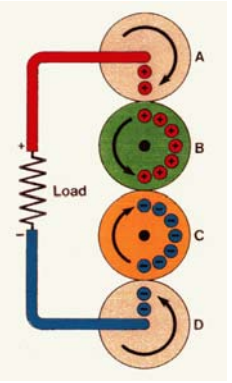
Orientation in Objects

Code Generierung mit XSLT



Sabine Winkler
winkler@oio.de

Thomas Bayer
bayer@oio.de

Orientation in Objects GmbH
Weinheimer Str. 68
68309 Mannheim
www.oio.de

Orientation in Objects

Quelle: http://www.taylorrentalstaug.com/contractor.htm

Inhalt

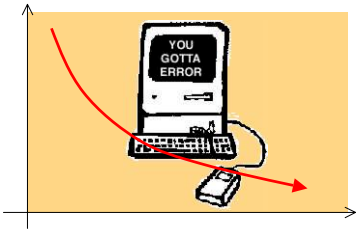
Orientation in Objects

- Einführung
- Code generieren mit XSLT
- Technische Generatoren
  - Erzeugen eines Graphical User Interfaces
  - Validator Generator
- Fachliche Generatoren
  - Prozessabbildung
  - Wertgrenzenbeispiel

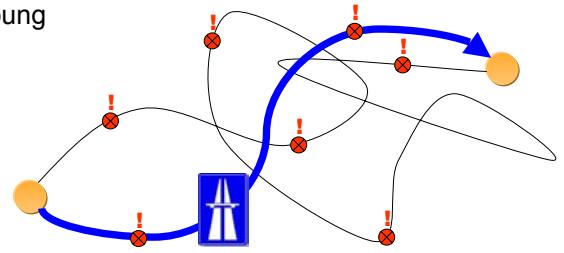
Vorteile von Code Generatoren

Orientation in Objects

- Reduzieren der Entwicklungszeit
- Eindämmen von Fehlerquellen



- Einfache Anpassung
- Bequeme Handhabung



Code Generatoren

Orientation in Objects

Einteilung

- einmalige Generierung
- mehrmalige Generierung
- Generatoren mit graphischem Interface
- dynamische Generatoren

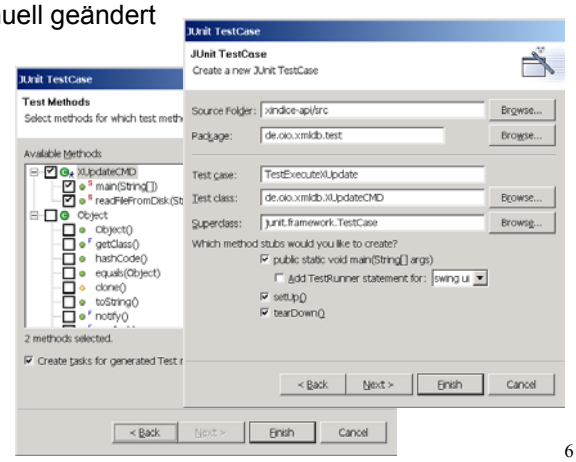
Anwendbarkeit

- Zusammenfassen von ähnlichen Aufgaben
- Abstraktionen
- Technisch
  - EJB, Persistenz, Proxys,...
- **Auch für Fachlichkeit !**

Einmalige Generierung

Orientation in Objects

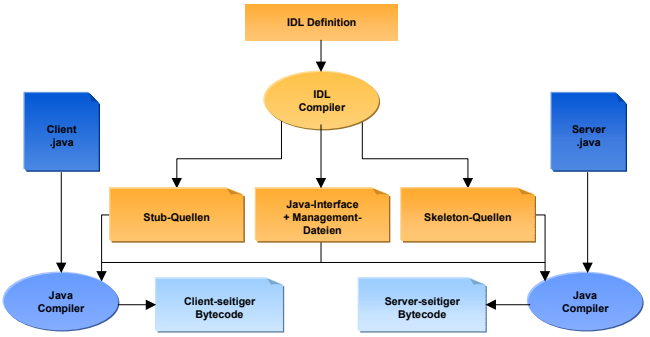
- Quellcode wird zu Beginn erzeugt
- Ergebnis wird manuell geändert
- z.B. Wizzards für
  - Java Klassen,
  - EJB, JMS



Mehrmalige Generierung

Orientation in Objects

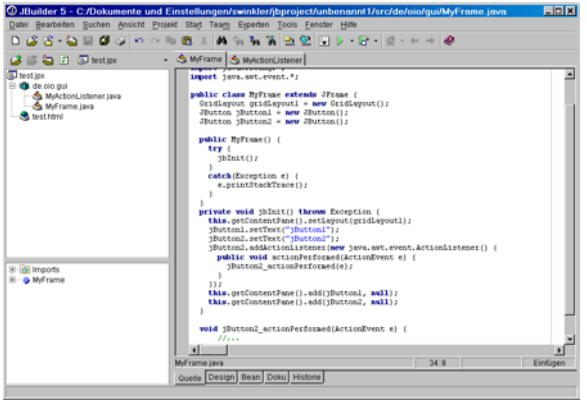
- Generierter Code wird nicht verändert
- Handgeschriebener Code verwendet generierten Code
- z.B. CORBA



Generatoren mit graphischem Interface

Orientation in Objects

- Oft Bestandteil von IDE‘s z.B. GUI Editoren
- Arbeit mit Platzhaltern, Roundtrip ...



Dynamische Generatoren

Orientation in Objects

Beispiel Server Pages:

```
<%@page language="java"%>
<html>
  <head>
    <title>Simple JSP
    </title>
  </head>
  <body>
  <h1>Hello World !</h1>
  It's
<%= new java.util.Date()%>
  </body>
</html>
```

Java-Compiler

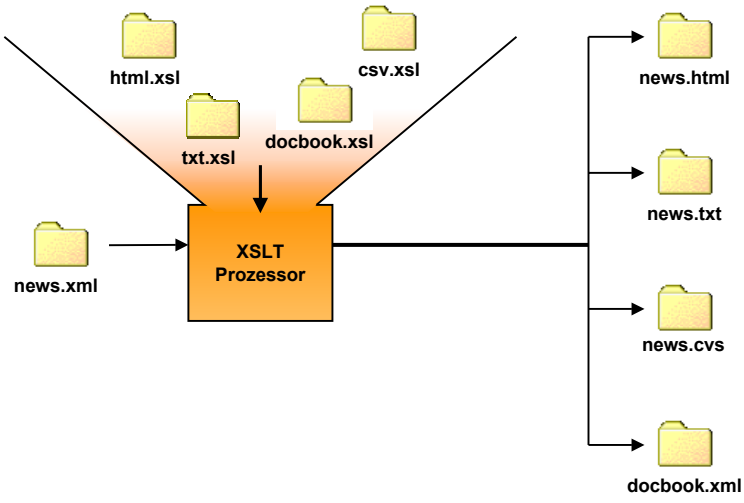
```
import...;

public class _0002fSimpleJsp_0002ejspSimpleJsp_jsp_0
        extends HttpJspBase {

    //....

  public void _jspService(HttpServletRequest request,
                          HttpServletResponse  response)
      throws IOException, ServletException {

    //...
    ServletConfig config = null;
    JspWriter out = null;

      // ...
    out = pageContext.getOut();
    out.write("<html>\r\n<head>\r\n<title>Simple
              JSP</title>\r\n</head>\r\n<body>\r\n
              <h1>Hello World</h1>\r\n It's ");
    out.print( new java.util.Date() );
    out.print("</body>\r\n</html>\r\n");
        // end
      // ...
  }
}
```


Code generieren mit XSLT

Orientation in Objects

- Einführung
- Code generieren mit XSLT
- Technische Generatoren
  - Erzeugen eines Graphical User Interfaces
  - Validator Generator
- Fachliche Generatoren
  - Prozessabbildung
  - Wertgrenzenbeispiel

XSLT Prozessor



Erzeugen einer Klasse

Quelle:

```
<?xml version="1.0" encoding="UTF-8"?>
<Foo>
   <classname>Bar</classname>
   <message>Hossale</message>
</Foo>
```

Stylesheet:

```
<xsl:template match="/Foo">

    public class <xsl:value-of select="classname"/>  {
              ...
    }

</xsl:template>
```

Erzeugter Code:

```
public class Bar {

}
```

Erzeugen von Attributen

Stylesheet:

```
<xsl:template match="/Foo">

      public class <xsl:value-of select="classname"/>  {

          <xsl:apply-templates/>
      }

</xsl:template>
```

```
<xsl:template match="*">
      String <xsl:value-of select="name()"/> = "<xsl:value-of select="."/>";
</xsl:template>
```

Erzeugter Code:

```
public class Bar {
     String class-name = "Bar";
     String message = "Hossale";
}
```


Erzeugen von Methoden

Stylesheet:

```
<xsl:template match="/Foo">
     public class <xsl:value-of select="classname"/> {

          <xsl:apply-templates/>
     }
</xsl:template>
```

```
<xsl:template match="*">
    public String get<xsl:value-of select="name()"/>() {
        return "<xsl:value-of select="."/>";
    }
</xsl:template>
```

Erzeugter Code:

```
public class Foo {
    public String getclassname() {
        return "Bar";
    }
    public String getmessage() {
        return "Hossale";
    }
}
```

Regel-basierte Erzeugung

Orientation in Objects

Stylesheet:

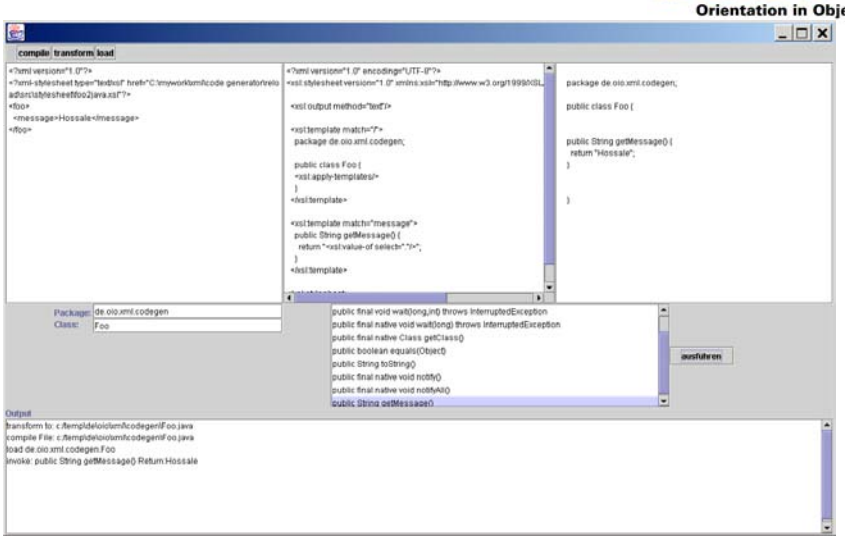
```
<xsl:template match="/foo">
  public class <xsl:value-of select="foo/class-name"/>  {

    <xsl:apply-templates/>

  }
</xsl:template>


<xsl:template match="fenster">
    ...
</xsl:template>


<xsl:template match="*[@type='text']">
    ...
</xsl:template>


<xsl:template match="*[@id]">
    ...
</xsl:template>
```

SEP

not (**.//** )

Dynamische Generierung

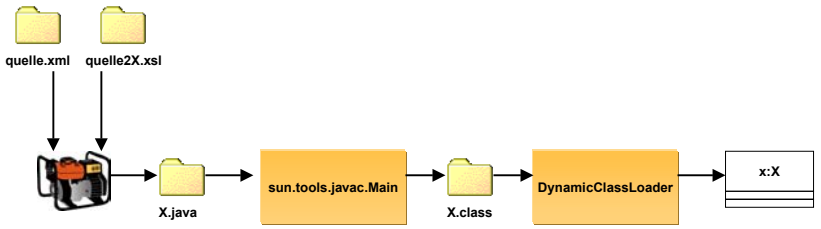
Orientation in Objects



Dynamische Generierung

Orientation in Objects

- Während der Laufzeit aus XSLT Code erzeugen
  - JAXP Transformer
- Compilieren
- Laden (Problem: Cachen von Klassen)
- Instanziieren
- Aufrufen



Dynamischer ClassLoader

Orientation in Objects

```
public class DynamicClassLoader extends ClassLoader {

  public Class loadClass(String name) throws ClassNotFoundException {

    if ( name.startsWith("java"))
      return findSystemClass(name);

    byte bytes[] = loadClassBytes(new File( FileUtil.getTempDir() +
      "/" + FileUtil.getFileNameFromClassName(name)));

    return defineClass( name, bytes, 0, bytes.length);
  }

  private byte[] loadClassBytes(File file) {
    ...
  }
}
```

Code Generator in Cocoon 2

Orientation in Objects

Cocoon Code:

```
ProgramGenerator programGenerator =
              (ProgramGenerator)this.manager.lookup(ProgramGenerator.ROLE);
Object obj = programGenerator.load( this.manager,
                                    "apus/person.form",
                                    "swal",
                                    "java",
                                    resolver);
```

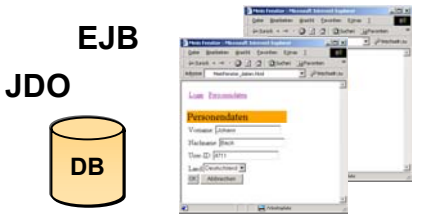
Stylesheet:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:swal="http://oio.de/swal"
                version="1.0">

  <xsl:template match="swal:form">
     ...
  </xsl:template>
  ...
</xsl:stylesheet>
```

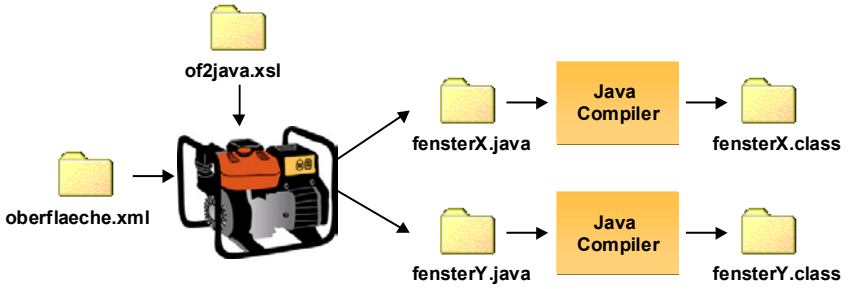

Erzeugen einer GUI

Orientation in Objects

- Einführung
- Code generieren mit XSLT
- Technische Generatoren
  - Erzeugen eines Graphical User Interfaces
  - Validator Generator
- Fachliche Generatoren
  - Prozessabbildung
  - Wertgrenzenbeispiel



Architektur

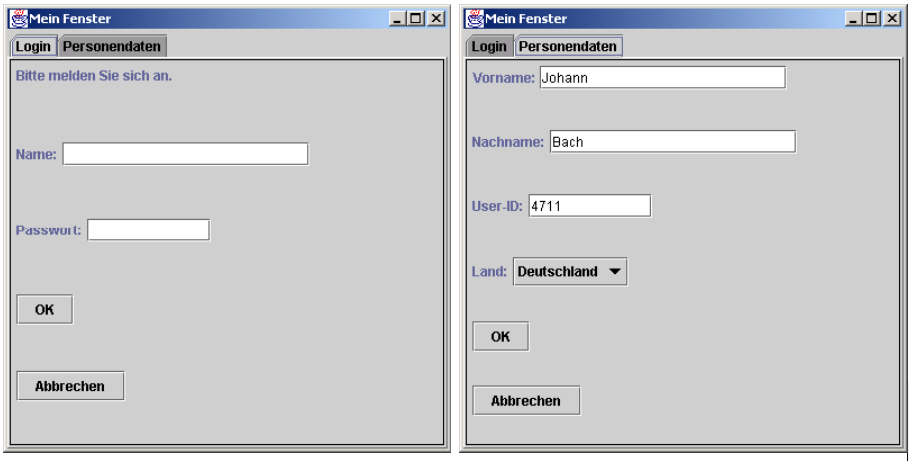


Quelle: oberflaeche.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<oberflaeche>
  <fenster id="MeinFenster" titel="Mein Fenster">
    <register id="register" titel="register1">
      <container id="login" titel="Login">
        <text id="text1">Bitte melden Sie sich an.</text>
        <feld id="Name" typ="text" titel="Name:" laenge="20"/>
        <feld id="Passwd" typ="text" titel="Passwort:" laenge="10"/>
        <knopf id="ok1" titel="OK"/>
        <knopf id="abbrechen1" titel="Abbrechen"/>
      </container>
      <container id="daten" titel="Personendaten">
        <feld id="vorname" typ="text" default="Joe" titel="Vorname" laenge="20"/>
        <feld id="nachname" typ="text" default="Bach" titel="Nachname" laenge="20"/>
        <feld id="userId" typ="text" default="4711" titel="User-ID:"/>
        <auswahlbox id="Land" titel="Land:">
          <auswahl>Deutschland</auswahl>
          <auswahl>Frankreich</auswahl>
          <auswahl>Schweiz</auswahl>
        </auswahlbox>
        <falschesElement/>
        <knopf id="ok2" titel="OK"/>
        <knopf id="abbrechen2" titel="Abbrechen"/>
      </container>
    </register>
  </fenster>
</oberflaeche>
```

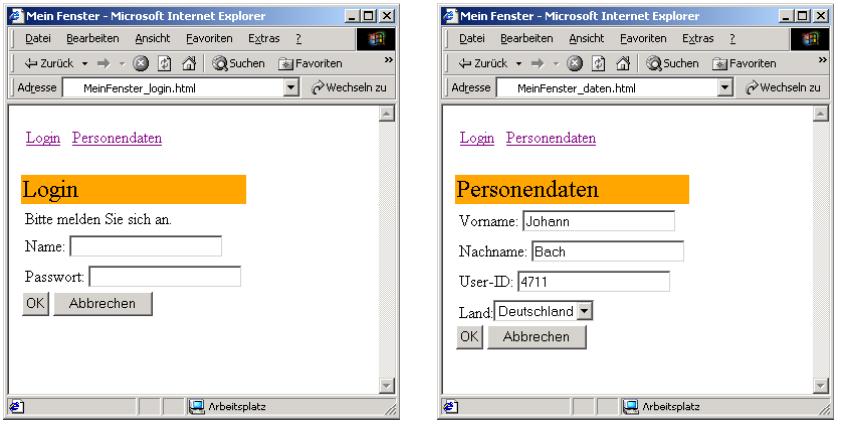
Generierte GUI in Swing

Orientation in Objects



HTML bzw. JSP Ausgabe

Orientation in Objects



Klassennamen ermitteln

Orientation in Objects

```
<xsl:template match="fenster">
   <xsl:document href="{@id}Impl.java">
   import java.awt.*;
   import java.awt.event.*;
   import javax.swing.*;
   import javax.swing.event.*;
   import javax.swing.border.*;

   public class <xsl:value-of select="@id"/>Impl extends JFrame {

   // Alle Instanzvariablen definieren
   <xsl:apply-templates mode="instancevariables"/>

   // Konstruktor
   <xsl:value-of select="@id"/>Impl()
   {
     // Allgemeines Layout
     super.setTitle("<xsl:value-of select="@titel"/>");
     setSize(400, 400);
     // Auf closing event reagieren
     addWindowList
             publ

     // Root-Panel
     JPanel <xsl:value-of select="@id"/> = new JPanel();
     <xsl:value-of select="@id"/>.setLayout(new GridLayout(0,1));

     // Aufbau des Fensters
     <xsl:apply-templates/>

     // Top-Level-Elemente in den Root_Panel hängen
     <xsl:for-each select="*">
             <xsl:value-of select="../@id"/>.add(<xsl:value-of select="@id"/>);
     </xsl:for-each>

     // Root-Panel in den ContentPane haengen
     getContentPane().add(<xsl:value-of select="@id"/>, "Center");
     }
   }
   </xsl:document>
</xsl:template>
```

```
public class <xsl:value-of select="@id"/>Impl
extends JFrame
```


Zu den Elementen

Orientation in Objects

```
<xsl:template match="fenster">
   <xsl:document href="{@id}Impl.java">
   import java.awt.*;
   import java.awt.event.*;
   import javax.swing.*;
   import javax.swing.event.*;
   import javax.swing.border.*;

   public class <xsl:value-of select="@id"/>Impl extends JFrame {

   // Alle Instanzvariablen definieren
   <xsl:apply-templates mode="instancevariables"/>

   // Konstruktor
   <xsl:value-of select="@id"/>Impl()
   {
     // Allgemeines Layout
     super.setTitle("<xsl:value-of select="@titel"/>");
     setSize(400, 400);
     // Auf closing event reagieren
     addWindowListener(new WindowAdapter(){
             public void windowClosing(WindowEvent e){
                     System.exit(0);}});

     // Root-Panel anlegen
     JPanel <xsl:value-of select="@id"/> = new JPanel();
     <xsl:value-of select="@id"/>.setLayout(new GridLayout(0,1));

     // Aufbau des Fensters
     <xsl:apply-templates/>

     // Top-Level-Elemente in den Root_Panel hängen
     <xsl:for-each select="*">
             <xsl:value-of select="../@id"/>.add(<xsl:value-of select="@id"/>);
     </xsl:for-each>

     // Root-Panel in den ContentPane haengen
     getContentPane().add(<xsl:value-of select="@id"/>, "Center");
     }
   }
   </xsl:document>
</xsl:template>
```

```
<xsl:apply-templates/>
```

Code für Register erzeugen

Orientation in Objects

```
<xsl:template match="register">

      <xsl:value-of select="@id"/> = new JTabbedPane ();

      <xsl:apply-templates/>

      // Hinzufuegen der einzelnen Unterelemente des Registers
      <xsl:for-each select="*">
            <xsl:value-of select="../@id"/>.addTab(
                  "<xsl:value-of select="@titel"/>",
                  <xsl:value-of select="@id"/>);
      </xsl:for-each>

      // Register in das uebergeordnete Panel haengen
      <xsl:value-of select="../@id"/>.add(<xsl:value-of select="@id"/>);
</xsl:template>
```


Erzeugter Konstruktor Code

Orientation in Objects

```
...

register = new JTabbedPane ();

// Anmelden der Elemente im Registerfeld
login = new JPanel() ;
login.setLayout(new GridLayout(0, 1));
login.setBorder(LineBorder.createBlackLineBorder());

...

// Anmelden der Elemente im Registerfeld
daten = new JPanel() ;
daten.setLayout(new GridLayout(0, 1));
daten.setBorder(LineBorder.createBlackLineBorder());

  ...

// Hinzufuegen der einzelnen Unterelemente des Registers
register.addTab("Login", login);
register.addTab("Personendaten", daten);

// Register in das uebergeordnete Panel haengen
MeinFenster.add(register);

...
```

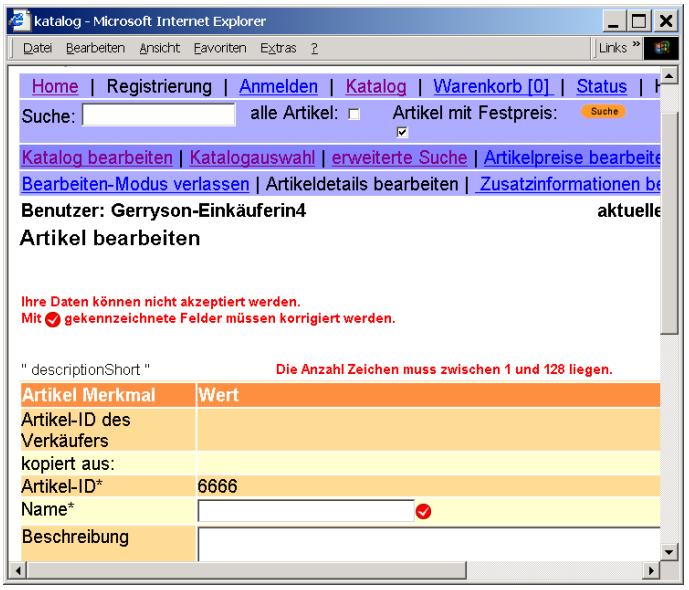
Validator Generator

Orientation in Objects

- Einführung
- Code generieren mit XSLT
- Technische Generatoren
  - Erzeugen eines Graphical User Interfaces
  - Validator Generator
- Fachliche Generatoren
  - Prozessabbildung
  - Wertgrenzenbeispiel

Validieren von Benutzereingaben

Orientation in Objects



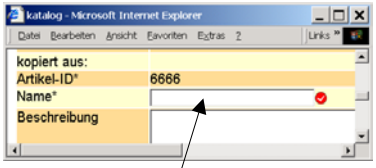
Konfiguration

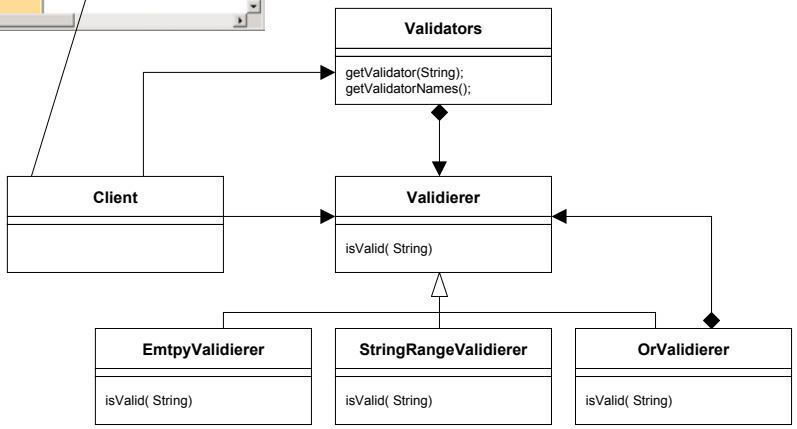
Orientation in Objects

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Validators>
  <Or name="Alter">
    <NumberRange>
       <param name="min" value='"0"'/>
       <param name="max" value='"99"'/>
    </NumberRange>
    <Empty/>
  </Or>
</Validators>
```

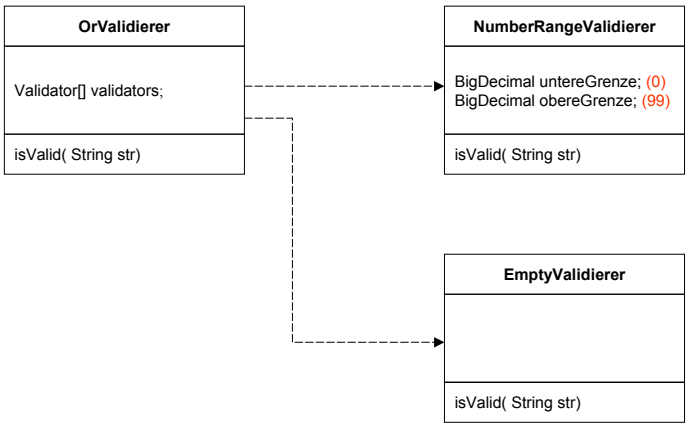
Validator Framework

Orientation in Objects





Validator Objekt „Alter“



Erzeugte Validator Fabrik

```
package de.oio.jax.validation;

import java.util.*;
import de.oio.util.validation.*;

public class Validators {

  private static Hashtable validatorsMap = new Hashtable();

  static {

    validatorsMap.put( "Alter",
                       new OrValidator( new Validator[]
                       { new NumberRangeValidator("0", "99"), new EmptyValidator()}
                      ));
  }

  public static Validator getValidator(String validatorName) {
    return (Validator) validatorsMap.get(validatorName);
  }

  public static Set getValidatorNames() {
    return validatorsMap.keySet();
  }
}
```

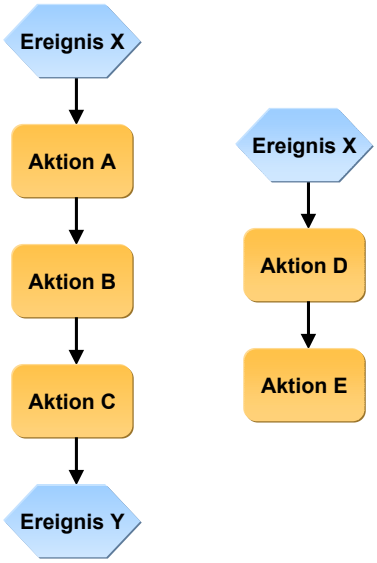
- Einführung
- Code generieren mit XSLT
- Technische Generatoren
  - Erzeugen eines Graphical User Interfaces
  - Validator Generator
- Fachliche Generatoren
  - Prozessabbildung
  - Wertgrenzenbeispiel

Fachliche Generatoren

- Prozesse
  - Folge logisch zusammenhängender Aktivitäten zur Erstellung einer Leistung oder Veränderung eines Objektes (Transformation)
- Workflow
  - Bezeichnung für arbeitsteilige Prozesse zur Abwicklung von Geschäftsvorfällen
- Entscheidungstabellen
  - zur Darstellung einfacher und komplexer Entscheidungsfolgen
- Businessregeln
- Textbausteingeneratoren

Ereignis Prozess Ketten EPK

Orientation in Objects



XML Repräsentation

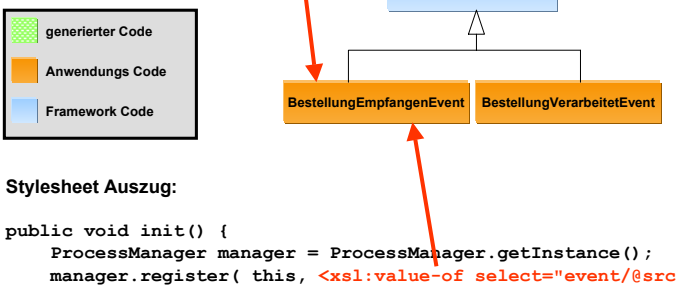
Orientation in Objects

```
<processes>
   <process>
      <event/>
        <action/>
        <action/>
        <action/>
   </process>
  ...
<processes>
```

Verweis auf Java Klassen

Orientation in Objects

```
<processes>
  <process dest="de.oio.jax.process.demo.BestellungVerarbeitenProcess">
    <event src="de.oio.jax.process.demo.BestellungEmpfangenEvent"/>
    <action src="de.oio.jax.process.demo.HoleLagerbestandAction"/>
    <action src="de.oio.jax.process.demo.SendeBestandsmeldungAction"/>
    <action src="de.oio.jax.process.demo.VerbucheBestellungImERPAction"/>
  </process>
  ...
<processes>
```



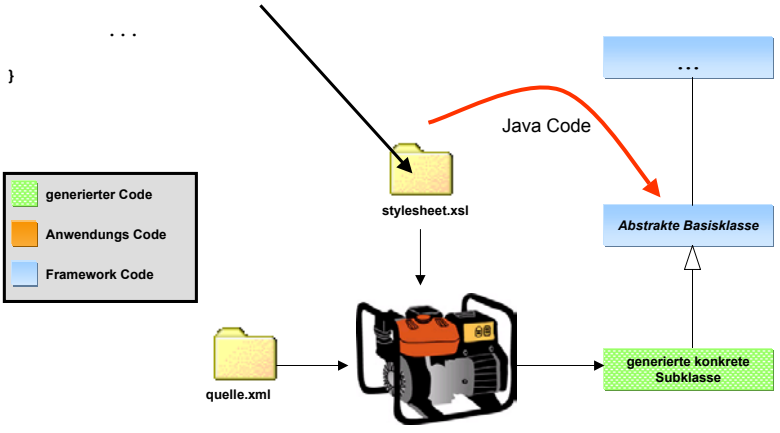
Stylesheet Auszug:

```
public void init() {
    ProcessManager manager = ProcessManager.getInstance();
    manager.register( this, <xsl:value-of select="event/@src"/>.class );
}
```


Code mit Framework verbinden

Orientation in Objects

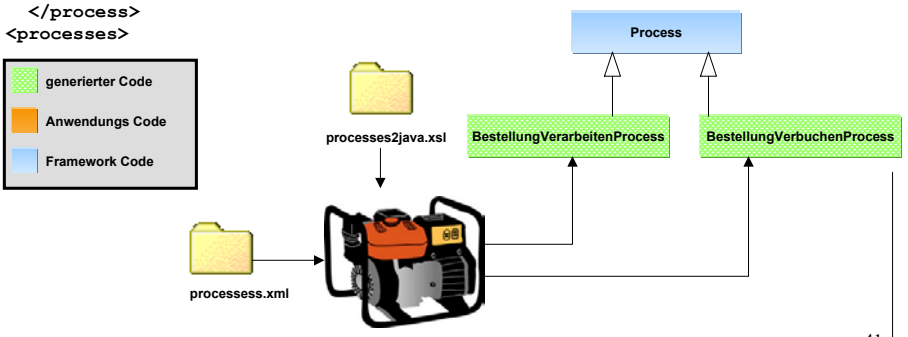
```
public class <xsl:call-template name="get-class-name">
               <xsl:with-param name="class-name" select="@dest"/>
             </xsl:call-template> extends de.oio.jax.process.Process {

        ...

}
```



Erzeugung mehrerer Klassen

Orientation in Objects

```
<processes>
  <process dest="de.oio.jax.process.demo.BestellungVerarbeitenProcess">
    <event/>
    <action/>
    ...
  </process>
  <process dest="de.oio.jax.process.demo.BestellungVerarbeitenProcess">
    <event/>
    <action/>
    ...
  </process>
<processes>
```



Xalan Erweiterung <redirect:write>

Orientation in Objects

```
<xsl:template match="process">

    <!-- Pfad für die neu anzulegende Datei ermitteln -->
    <xsl:variable name="path">
      <xsl:call-template name="get-path">
        <xsl:with-param name="class-name" select="@dest"/>
      </xsl:call-template>
    </xsl:variable>

    <redirect:write file="../../src/{$path}.java">
      package <xsl:call-template name="get-package-name">
        <xsl:with-param name="class-name" select="@dest"/>
      </xsl:call-template>;

      import java.util.Hashtable;
      ...

      public class <xsl:call-template name="get-class-name">
        <xsl:with-param name="class-name" select="@dest"/>
      </xsl:call-template> extends de.oio.jax.process.Process {

      ...

      }
    </redirect:write>
</xsl:template>
```

Nützliche Helfer

Orientation in Objects

`de.oio.jax.process.demo.BestellungVerarbeitenProcess`

Template: `get-path`

`=> de/oio/jax/process/demo/BestellungVerarbeitenProcess`

Template: `get-package-name`

`=> de.oio.jax.process.demo`

Template: `get-class-name`

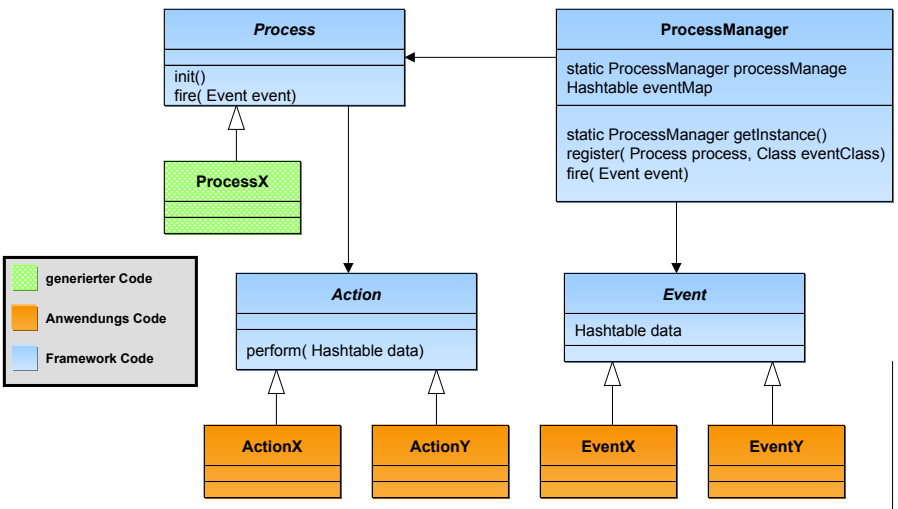
`=> BestellungVerarbeitenProcess`

Beispiel:

```
<xsl:call-template name="get-path">
   <xsl:with-param name="class-name" select="@dest"/>
 </xsl:call-template>
```


Prozess Framework

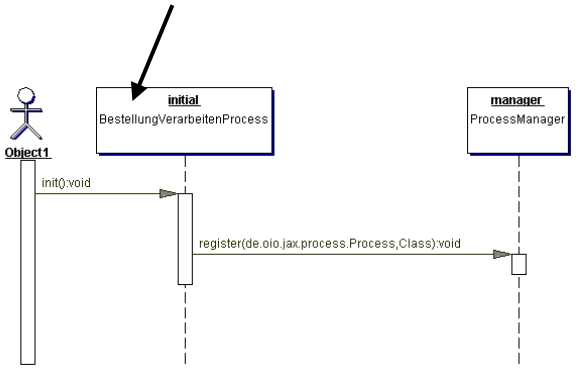
Orientation in Objects



Registrierung

Orientation in Objects

Ist generiert und dem Framework unbekannt!

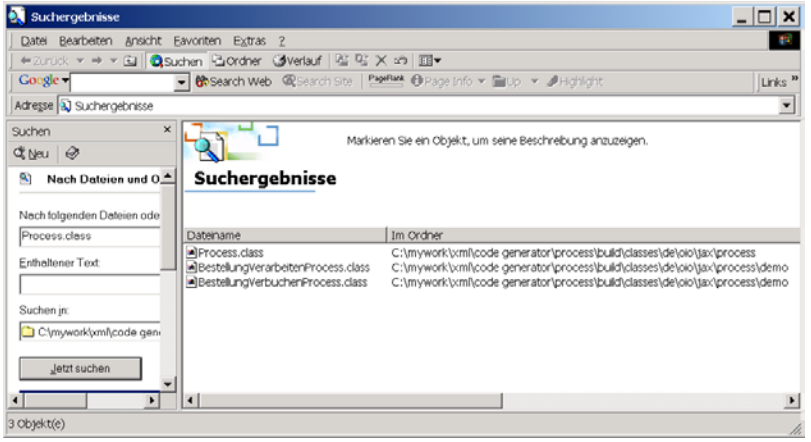


Problem:
Alle Prozesse müssen beim ProcessManager registriert werden,
damit der Sie später bei entsprechenden Events benachrichtigen kann.

Lösung mit Reflection

Orientation in Objects

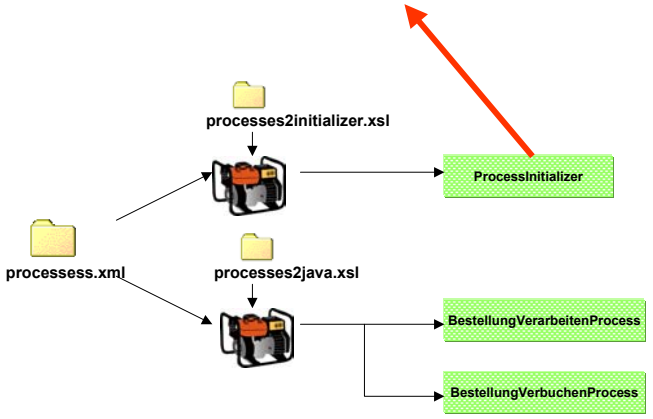
1.) Suche nach *Process.class
2.) Laden und Instanz erzeugen



Lösung mit Generator

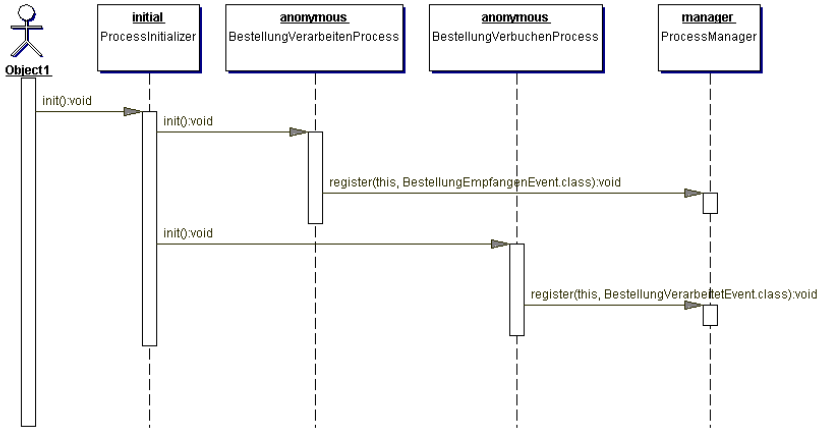
Orientation in Objects

```
public class ProcessInitializer {
   public static void init() {
        (new de.oio.jax.process.demo.BestellungVerarbeitenProcess()).init();
        (new de.oio.jax.process.demo.BestellungVerbuchenProcess()).init();
   }
}
```

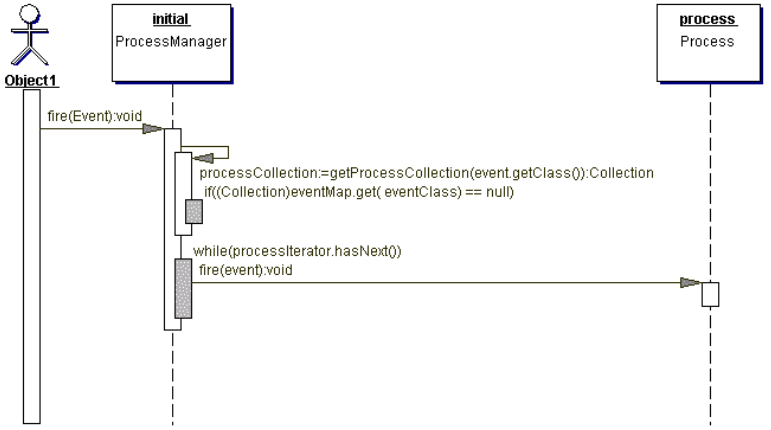


Registrierung beim ProcessManager

Orientation in Objects



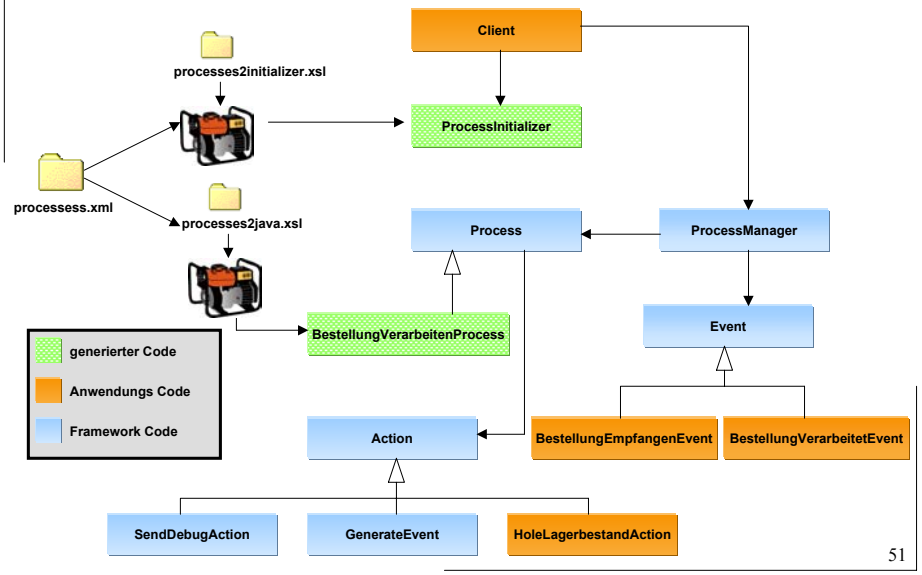
Feuern von Events



Client Code

```
ProcessInitializer.init();

Hashtable data = new Hashtable();

data.put( "Nr", "20020505-001");
data.put( "Ware", "Lutscher");
data.put( "Menge", new Long(10));
data.put( "MailAddrCustomer", "bayer@oio.de");

BestellungEmpfangenEvent bestellungEmpfangenEvent
                         = new BestellungEmpfangenEvent();

bestellungEmpfangenEvent.setData( data);

ProcessManager.getInstance().fire( bestellungEmpfangenEvent);
```

Gesamte Anwendung



Fazit

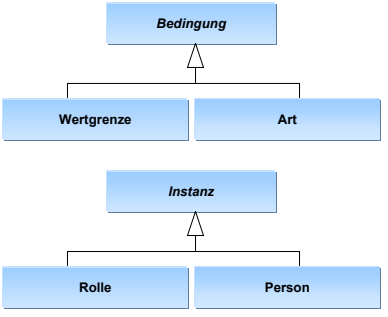
- Stylesheet kann mehrere Klassen erzeugen
- „Java“ Code vom Generator ins Framework verschieben
- Reflection durch Generator ersetzen
- Auf eine Quelle können mehrere Stylesheets angewandt werden

- Einführung
- Code generieren mit XSLT
- Technische Generatoren
  - Erzeugen eines Graphical User Interfaces
  - Validator Generator
- Fachliche Generatoren
  - Prozessabbildung
  - Wertgrenzenbeispiel

Wertgrenzenbeispiel einfach nur XML

Wert bis EUR 400	bis EUR 1000	bis *	Art Investition/Kosten	Genehmigung durch Keine	Abteilungsleiter	Chef
x			Investition	x		
x			Kosten	x		
	x		Investition		x	
	x		Kosten	x		
		x	Investition		x	x
		x	Kosten		x	

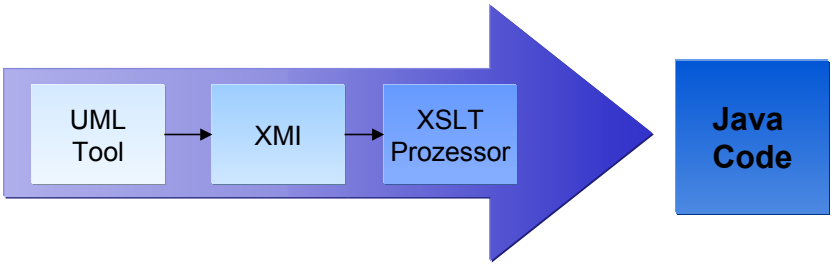
```
<vorfall>
 <bedingung>
   <wertgrenze wert="1000"/>
   <art typ="I"/>
 </bedingung>
 <genehmigung>
  <instanz>
    <rolle>Abteilungsleiter</rolle>
  </instanz>
 </genehmigung>
</vorfall>
```



Code aus UML Diagrammen

Orientation in Objects

- Generierung von Klassen ist einfach



Fazit

Orientation in Objects

- Ausgereifte Technologien
  - Objektorientierung
  - Java
  - Design Pattern
  - XML und XSLT
- Alles ist heute einsetzbar!

Quellen

Orientation in Objects

- Tobias Kieninger, Thomas Bayer, „Codegenerierung mit XML“, Java Spektrum 1/2002, S. 52 ff
- http://www.oio.de/codegenerator/
- Erich Gamma et al., „Design Patterns Elements of Reusable Object-Oriented Software“
- XPath, http://www.w3.org/TR/xpath
- XSLT, http://www.w3.org/Style/XSL/
- XSLT Entwurfsmuster
  - http://www.xslt-patterns.com
- Struts XSLT Code Generator, http://xsltgenerator.sourceforge.net/
- http://www.uiml.org

The End

Orientation in Objects

Vielen Dank für Ihre Aufmerksamkeit !

Thomas Bayer
bayer@oio.de

Sabine Winkler
winkler@oio.de

Dynamische Anwendungen

Orientation in Objects

- Reflection API von Java
- Erzeugter Code kann zur Laufzeit geladen werden
  - Problem: Erneutes Laden der gleichen Klasse
    => Eigener Classloader
- Beispiel mit `Class.forName()`
- sitemap von Cocoon

Beispiel (5)

Orientation in Objects

```
<xsl:template match="fenster">
   <xsl:document href="{@id}Impl.java">
   import java.awt.*;
   import java.awt.event.*;
   import javax.swing.*;
   import javax.swing.event.*;
   import javax.swing.border.*;

   public class <xsl:value-of select="@id"/>Impl extends JFrame {

   // Alle Instanzvariablen definieren
   <xsl:apply-templates mode="instancevariables"/>

   // Konstruktor
   <xsl:value-of select="@id"/>Impl()
   {
     // Allgemeines Layout
     super.setTitle("<xsl:value-of select="@titel
     setSize(400, 400);
     // Auf closing event reagieren
     addWindowListener(new WindowAdapter(){
             public void windowClosing(WindowEver
                     System.exit(0);}});
     // Root-Panel anlegen
     JPanel <xsl:value-of select="@id"/> = new JPanel();
     <xsl:value-of select="@id"/>.setLayout(new GridLayout(0,1));

     // Aufbau des Fensters
     <xsl:apply-templates/>

     // Top-Level-Elemente in den Root-Panel hängen
     <xsl:for-each select="*">
             <xsl:value-of select="../@id"/>.add(<xsl:value-of select="@id"/>);
     </xsl:for-each>

     // Root-Panel in den ContentPane haengen
     getContentPane().add(<xsl:value-of select="@id"/>, "Center");
     }
   }
   </xsl:document>
</xsl:template>
```

Auswählen der XML-Elemente und Hinzufügen als Top-Level-Elemente:

`<xsl:for-each select="*">`

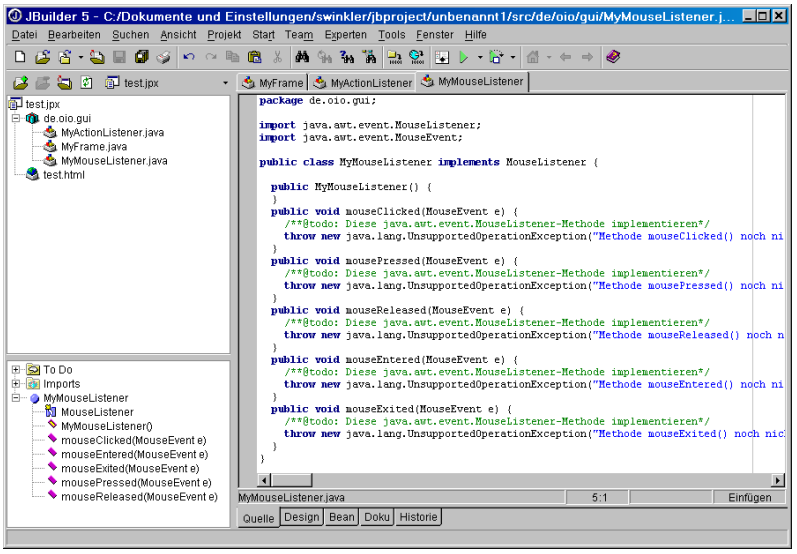
Ausgabe 1

Orientation in Objects

```
...

    // Register
    private JTabbedPane register;

    // container
    private JPanel login;

    // container
    private JPanel daten;

...
```

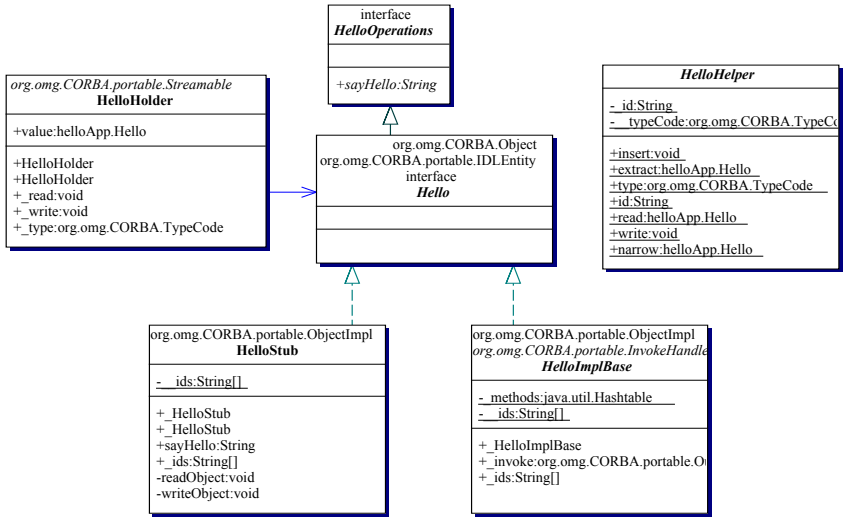

Erzeugte Code von JBuilder GUI Editor

Orientation in Objects



Klassendiagramm CORBA

Orientation in Objects



Listing 5

Orientation in Objects

```
<!-- Buttons -->
<xsl:template match="knopf">

    // Untergeordnetes Panel und Button anlegen
    JPanel pan_<xsl:value-of select="@id"/> = new JPanel();
    pan_<xsl:value-of select="@id"/>.setLayout(new FlowLayout(FlowLayout.LEFT));

    <xsl:value-of select="@id"/> =
           new JButton("<xsl:value-of select="@titel"/>");

// Button, Panel und container anhaengen.
    pan_<xsl:value-of select="@id"/>.add(<xsl:value-of select="@id"/>);
    <xsl:value-of select="../@id"/>.add(pan_<xsl:value-of select="@id"/>);

</xsl:template>
```

Hello World - In ‘n‘ Out

Orientation in Objects

```
<?xml version="1.0" encoding="UTF-8"?>
<greetings>
        <hello-message>Hello</hello-message>
        <name>Tobias</name>
</greetings>
```


```
public class Greetings{
    public static void main(String[] args){
            String name = "Tobias";
            System.out.print("Hello ");
            System.out.println(name);
    }
}
```


Ergebnis

Orientation in Objects

```
package de.oio.xmldb.test;

import junit.framework.TestCase;

public class TestExecuteXUpdate extends TestCase {

    public TestExecuteXUpdate(String name) {
       super(name);
    }
    /**
     * @see TestCase#setUp()
     */
    protected void setUp() throws Exception {
        super.setUp();
    }
    /**
     * @see TestCase#tearDown()
     */
    protected void tearDown() throws Exception {
        super.tearDown();
    }

    public void testMain() {}

    public void testReadFileFromDisk() {}
}
```

Hello World - XSLT


```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="....."> 
   <xsl:output method="text" version="1.0" encoding="UTF-8" indent="yes"/>
   <xsl:template match="greetings">
   public class Greetings{
       public static void main(String[] args){
           String name = "<xsl:value-of select="name"/>";
           <xsl:apply-templates select="hello-message"/>
       }
   }
   </xsl:template>

   <xsl:template match="hello-message">
           System.out.print("<xsl:value-of select="."/> ");
           System.out.println(name);
   </xsl:template>

</xsl:stylesheet>
```


Listing 3


```
<xsl:template match="register" mode="instancevariables">
        // Register
        private JTabbedPane <xsl:value-of select="@id"/>;
        <xsl:apply-templates mode="instancevariables"/>
</xsl:template>

<xsl:template match="container" mode="instancevariables">
        // container
        private JPanel <xsl:value-of select="@id"/>;
        <xsl:apply-templates mode="instancevariables" />
</xsl:template>
```

Beispiel (3)

Orientation in Objects

```
<xsl:template match="fenster">
   <xsl:document href="{@id}Impl.java">
   import java.awt.*;
   import java.awt.event.*;
   import javax.swing.*;
   import javax.swing.event.*;
   import javax.swing.border.*;

   public class <xsl:value-of select="@id"/>Impl extends JFrame {

   // Alle Instanzvariablen definieren
   <xsl:apply-templates mode="instancevariables"/>

   // Konstruktor
   <xsl:value-of select="@id"/>Impl()
   {
     // Allgemeines Layout
     super.setTitle("<xsl:value-of select="@titel"/>");
     setSize(400, 400);
     // Auf closing event reagieren
     addWindowListener(new WindowAdapter(){
             public void windowClosing(WindowEvent e){
                     System.exit(0);}});
     // Root-Panel anlegen
     JPanel <xsl:value-of select="@id"/> = new JPanel();
     <xsl:value-of select="@id"/>.setLayout(new GridLayout(0,1));

     // Aufbau des Fensters
     <xsl:apply-templates/>

     // Top-Level-Elemente in den Root_Panel hängen
     <xsl:for-each select="*">
             <xsl:value-of select="../@id"/>.add(<xsl:value-of select="@id"/>);
     </xsl:for-each>

     // Root-Panel in den ContentPane haengen
     getContentPane().add(<xsl:value-of select="@id"/>, "Center");
     }
   }
   </xsl:document>
</xsl:template>
```

Einfügen eines Konstruktors:

<xsl:value-of select="@id"/>

Definition der Instanzvariablen

Orientation in Objects

```
<xsl:template match="fenster">
   <xsl:document href="{@id}Impl.java">
   import java.awt.*;
   import java.awt.event.*;
   import javax.swing.*;
   import javax.swing.event.*;
   import javax.swing.border.*;

   public class <xsl:value-of select="@id"/>Impl extends JFrame {

   // Alle Instanzvariablen definieren
   <xsl:apply-templates mode="instancevariables"/>

   // Konstruktor
   <xsl:value-of select="@id"/>Impl()
   {
     // Allgemeines Layout
     super.setTitle("<xsl:value-of select="@titel"/>");
     setSize(400, 400);
     // Auf closing event reagieren
     addWindowListener(new WindowAdapter(){
             public void windowClosing(WindowEvent e){

     // Root-Panel anlegen
     JPanel <xsl:value-of select="@id"/> = new JPanel();
     <xsl:value-of select="@id"/>.setLayout(new GridLayout(0,1));

     // Aufbau des Fensters
     <xsl:apply-templates/>

     // Top-Level-Elemente in den Root_Panel hängen
     <xsl:for-each select="*">
             <xsl:value-of select="../@id"/>.add(<xsl:value-of select="@id"/>);
     </xsl:for-each>

     // Root-Panel in den ContentPane haengen
     getContentPane().add(<xsl:value-of select="@id"/>, "Center");
     }
   }
   </xsl:document>
</xsl:template>
```

<xsl:apply-templates mode="instancevariables"/>