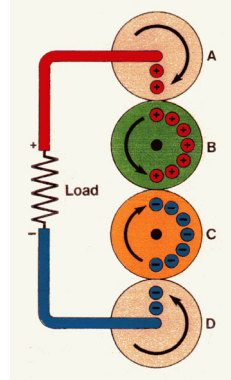


Code Generatoren und Frameworks Eine Traumhochzeit?

Thomas Bayer
bayer@oio.de

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



1

Let us pave the way
to your next
project



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

2

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

Einteilung

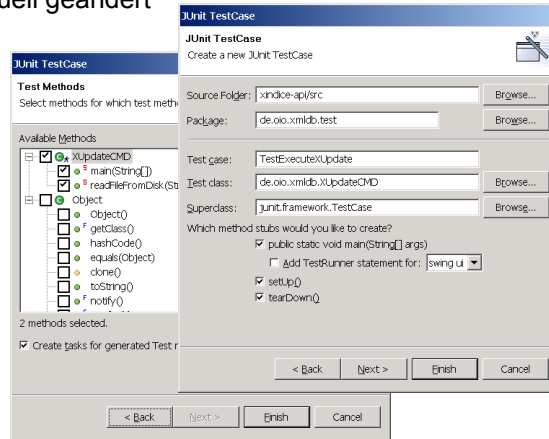
- einmalige Generierung
- mehrmalige Generierung
- Generatoren mit graphischem Interface
- dynamische Generatoren

Anwendbarkeit

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

Einmalige Generierung

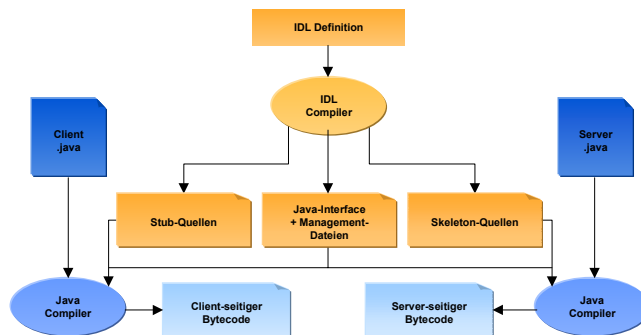
- Quellcode wird zu Beginn erzeugt
- Ergebnis wird manuell geändert
- z.B. Wizzards für
 - Java Klassen,
 - EJB, JMS



5

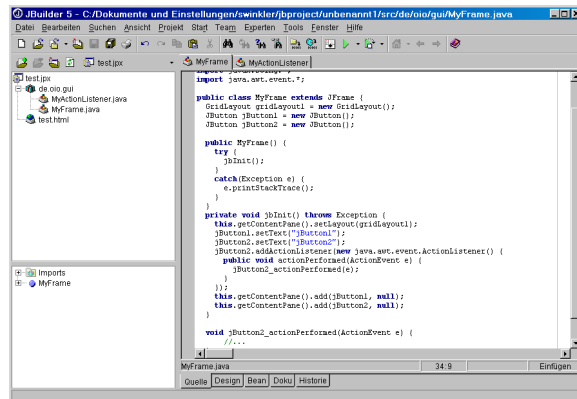
Mehrmalige Generierung

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



6

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



7

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
    // ...
  }
}
```

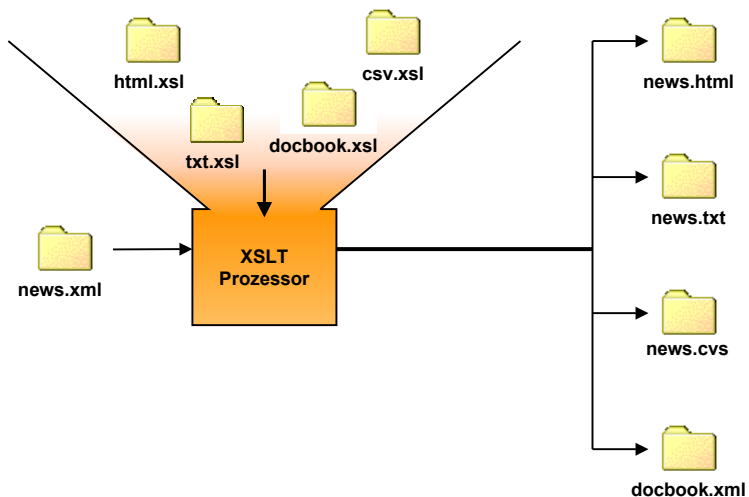
8

Code generieren mit XSLT

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

9

XSLT Prozessor



10

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 {

}
```

11

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";
}
```

12

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";
  }
}
```

13

Regel-basierte Erzeugung

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>
```

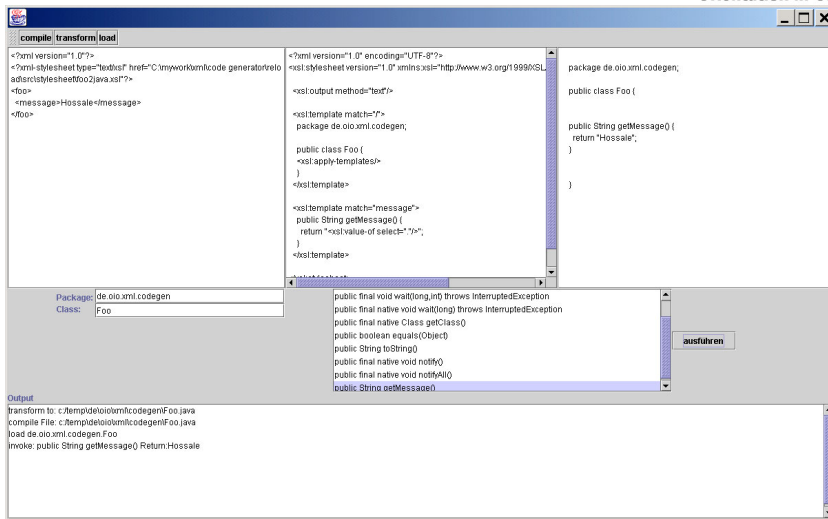
SEP

not



14

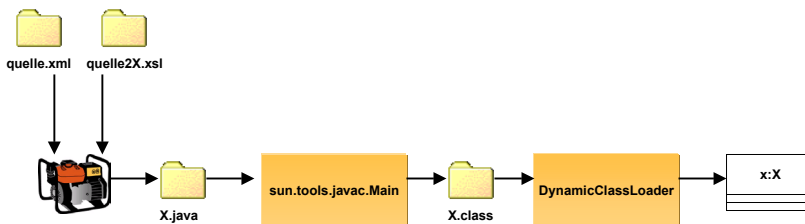
Dynamische Generierung



15

Dynamische Generierung

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



16

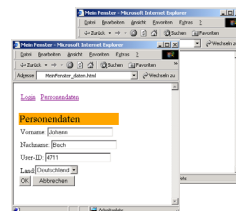
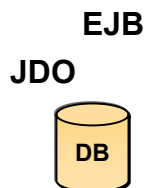
Dynamischer ClassLoader

```
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) {  
        ...  
    }  
}
```

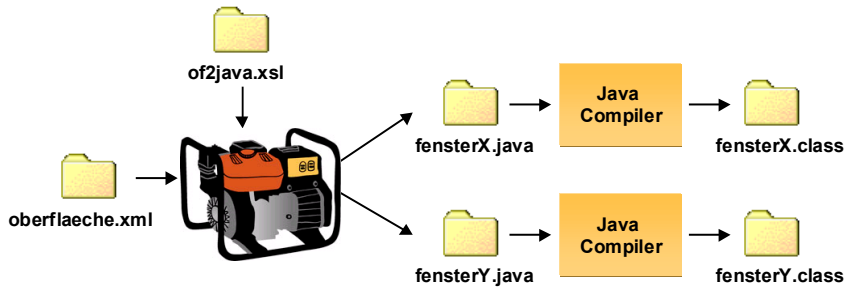
17

Erzeugen einer GUI

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



18



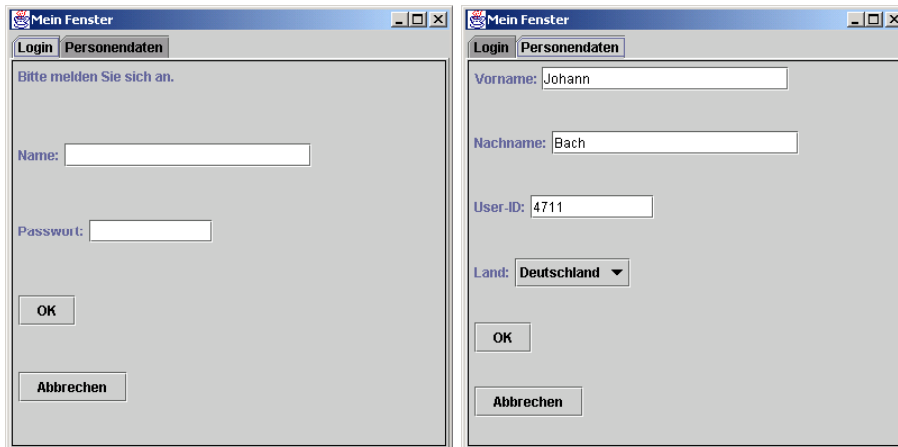
Quelle: oberflaeche.xml

```

<?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>

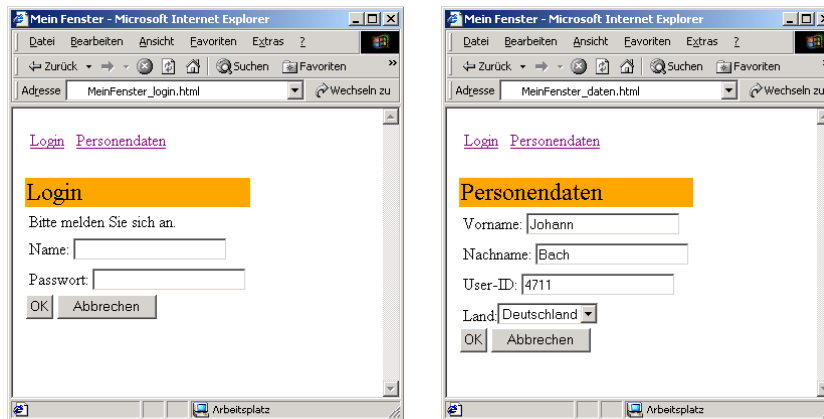
```

Generierte GUI in Swing



21

HTML bzw. JSP Ausgabe



22

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
        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>
```

23

Code Generierung mit XSLT © 2000-2002 Orientation in Objects GmbH

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>
```

24

Code Generierung mit XSLT © 2000-2002 Orientation in Objects GmbH

Code für Register erzeugen

```
<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>
```

25

Erzeugter Konstruktor Code

```
...

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());

...

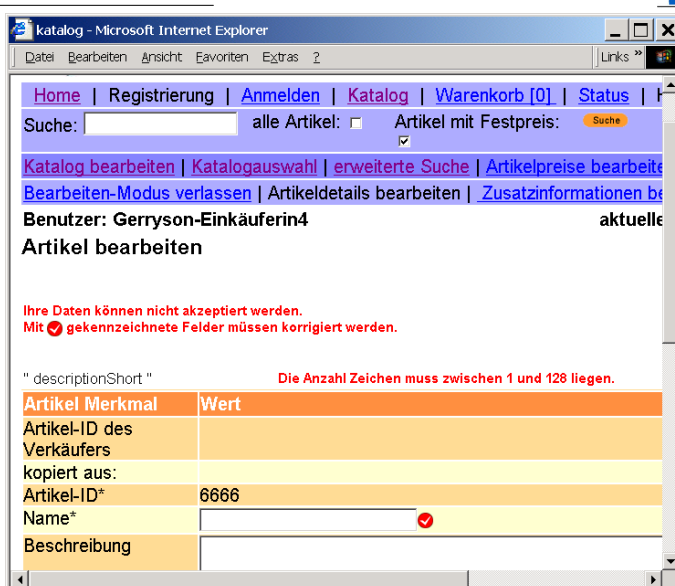
// Hinzufuegen der einzelnen Unterelemente des Registers
register.addTab("Login", login);
register.addTab("Personendaten", daten);

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

...
```

26

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



The screenshot shows a web browser window titled "katalog - Microsoft Internet Explorer". The page content includes a navigation menu, a search bar, and a user profile section. A red error message is displayed: "Ihre Daten können nicht akzeptiert werden. Mit gekennzeichnete Felder müssen korrigiert werden." Below this, a specific error is shown: "descriptionShort Die Anzahl Zeichen muss zwischen 1 und 128 liegen." A table with two columns, "Artikel Merkmal" and "Wert", lists several fields. The "Name*" field has a red checkmark icon next to its value, indicating it is the source of the error.

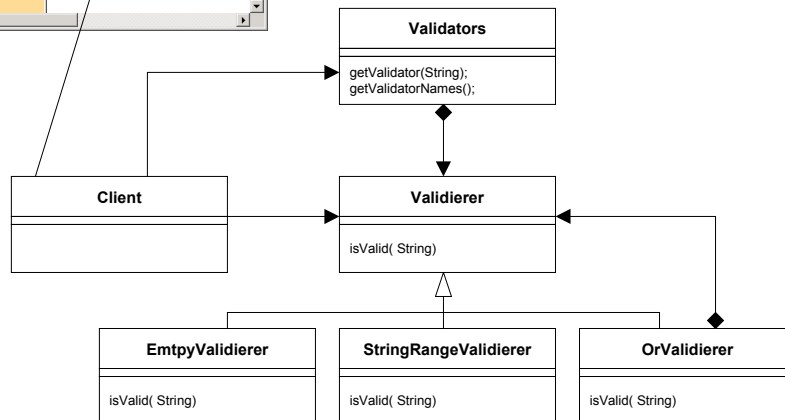
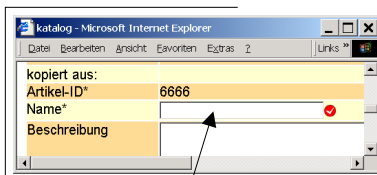
Artikel Merkmal	Wert
Artikel-ID des Verkäufers	
kopiert aus:	
Artikel-ID*	6666
Name*	<input type="text" value=""/>
Beschreibung	<input type="text" value=""/>

Konfiguration

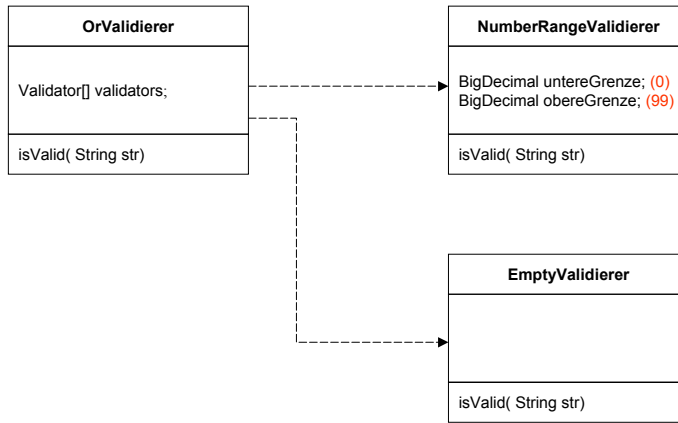
```
<?xml version="1.0" encoding="ISO-8859-1"?>  
<Validators>  
  <Or name="Alter">  
    <NumberRange>  
      <param name="min" value="0" />  
      <param name="max" value="99" />  
    </NumberRange>  
    <Empty />  
  </Or>  
</Validators>
```



Validator Framework



Validator Objekt „Alter“



31

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();
    }

}
```

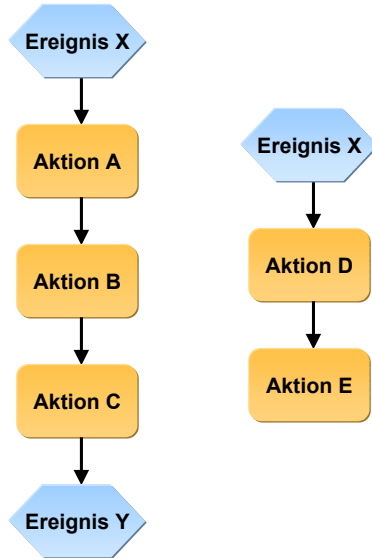
32

- 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



XML Repräsentation

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



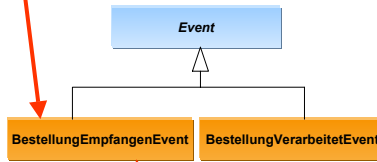
Verweis auf Java Klassen



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>
```

	generierter Code
	Anwendungs Code
	Framework Code



Stylesheet Auszug:

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

37

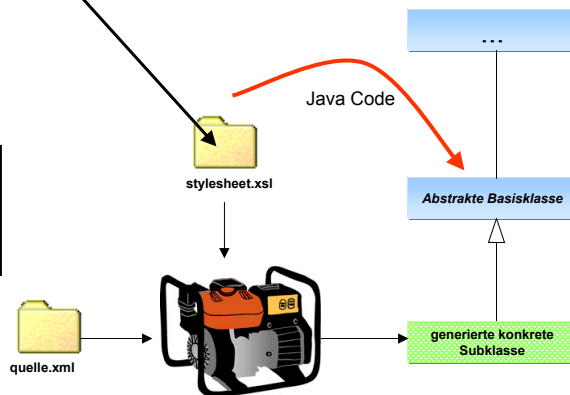
Code mit Framework verbinden



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 {
  ...
}
```

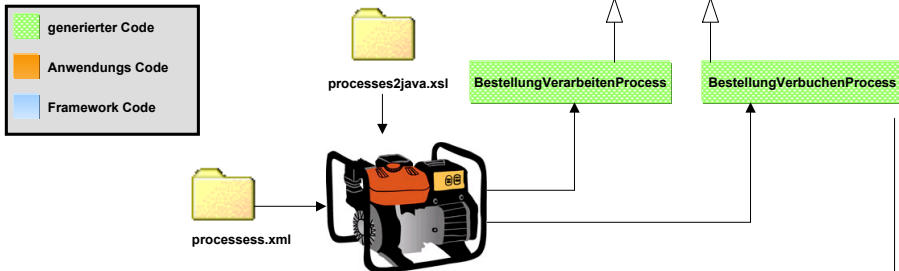
	generierter Code
	Anwendungs Code
	Framework Code



38

Erzeugung mehrerer Klassen

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



39

Code Generierung mit XSLT © 2000-2002 Orientation in Objects GmbH

Xalan Erweiterung <redirect:write>

```
<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>
```

40

Code Generierung mit XSLT © 2000-2002 Orientation in Objects GmbH

Nützliche Helfer

```
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

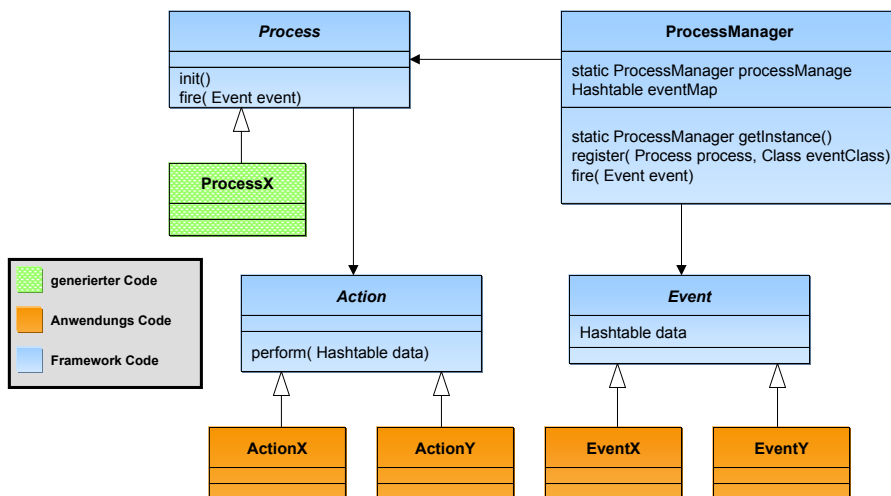
```
=> BestellungVerarbeitenProcess
```

Beispiel:

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

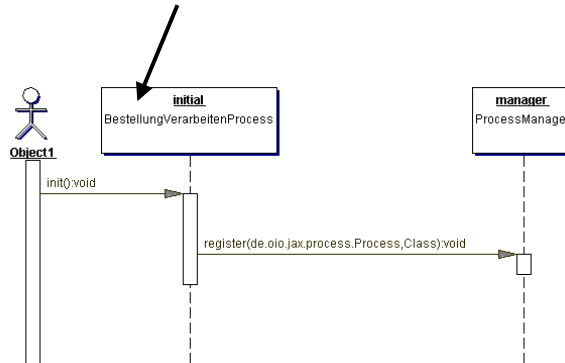
41

Prozess Framework



42

Ist generiert und dem Framework unbekannt!

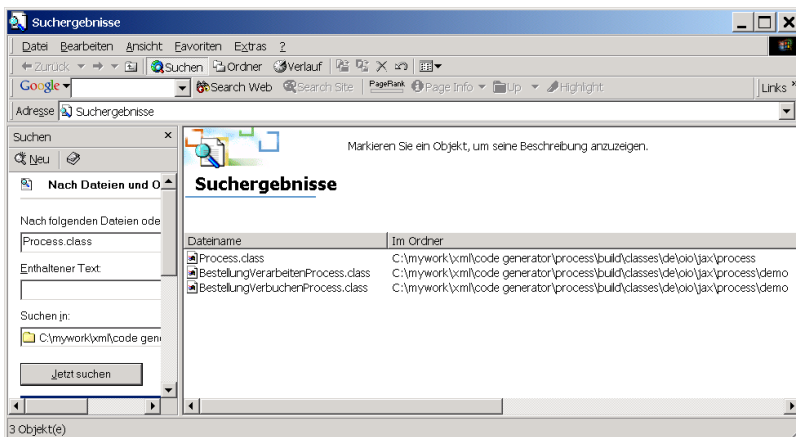


Problem:

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

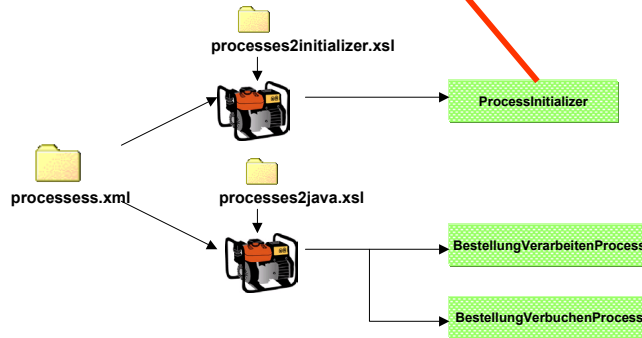
Lösung mit Reflection

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

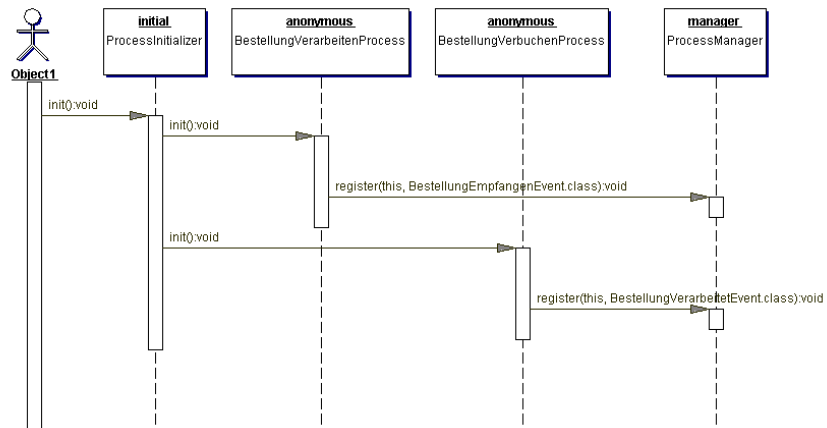


Lösung mit Generator

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



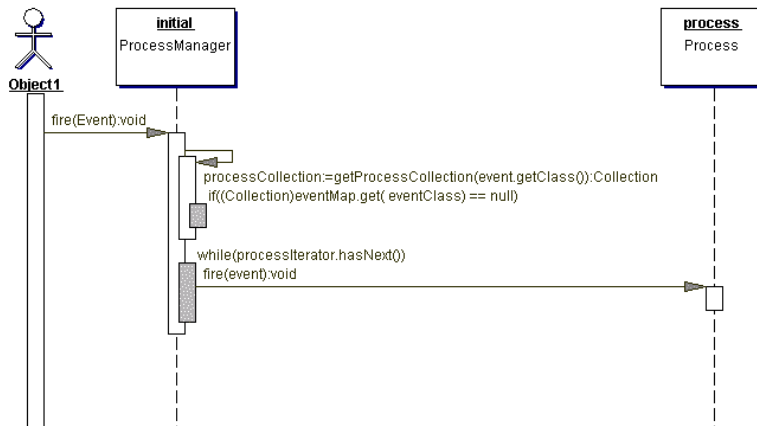
Registrierung beim ProcessManager



Feuern von Events



Orientation in Objects



47

Code Generierung mit XSLT © 2000-2002 Orientation in Objects GmbH

Client Code



Orientation in Objects

```
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();

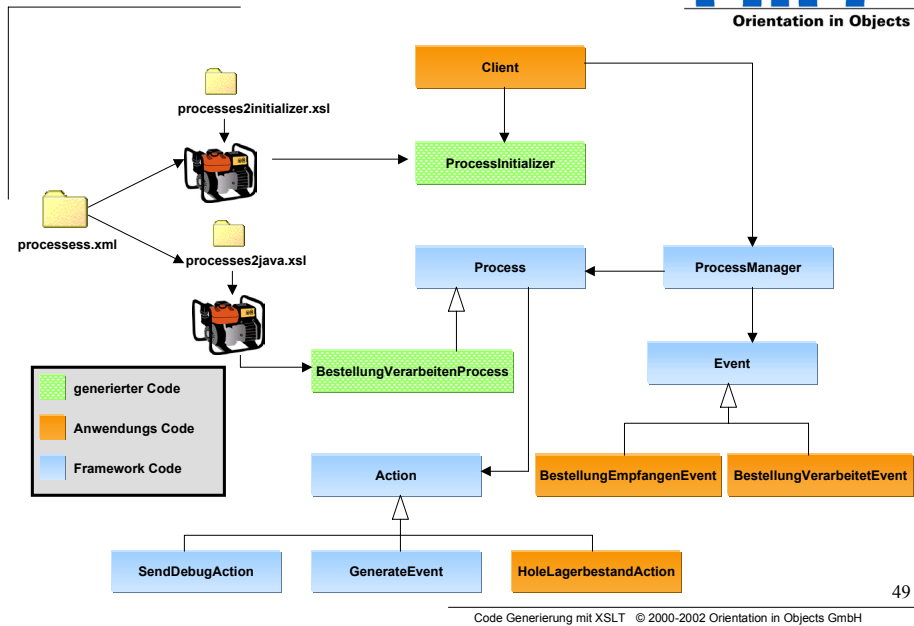
bestellungEmpfangenEvent.setData( data);

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

48

Code Generierung mit XSLT © 2000-2002 Orientation in Objects GmbH

Gesamte Anwendung



Fazit

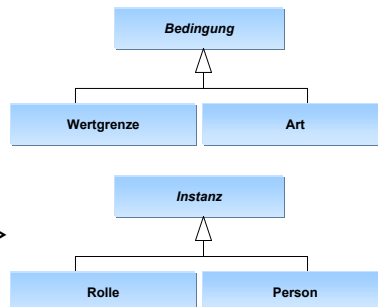
- Stylesheet kann mehrere Klassen erzeugen
- „Java“ Code vom Generator ins Framework verschieben
- Reflection durch Generator ersetzen
- Auf eine Quelle können mehrere Stylesheets angewandt werden

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

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

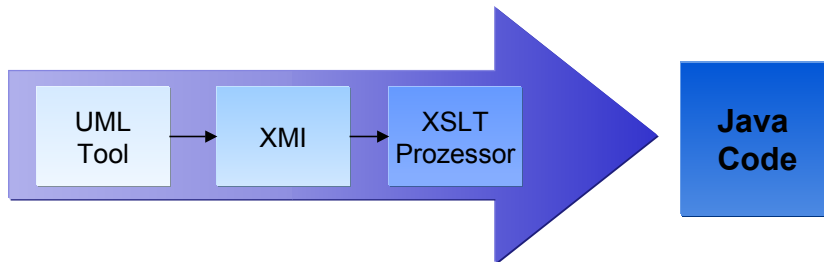
```

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



Code aus UML Diagrammen

- Generierung von Klassen ist einfach



53

Fazit

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



54

- Tobias Kieninger, Thomas Bayer, „Codegenerierung mit XML“, Java Spektrum 1/2002, S. 52 ff
- Thomas Forster, Thomas Bayer, „Dream Team: Kombination von Code-Generatoren und Frameworks“, XMLMagazin 2/2002
- Kieninger, Kipik, Bayer, „XSLT Entwurfsmuster“, XML Magazin 5/2001
- <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>

**Vielen Dank für
Ihre Aufmerksamkeit !**

Thomas Bayer
bayer@oio.de

www.oio.de

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>
```