



## Komponentenorientierte Webanwendungen mit wings 2.0

*Die Open Source Alternative zu JSF und Struts*



## x| **Einleitung**

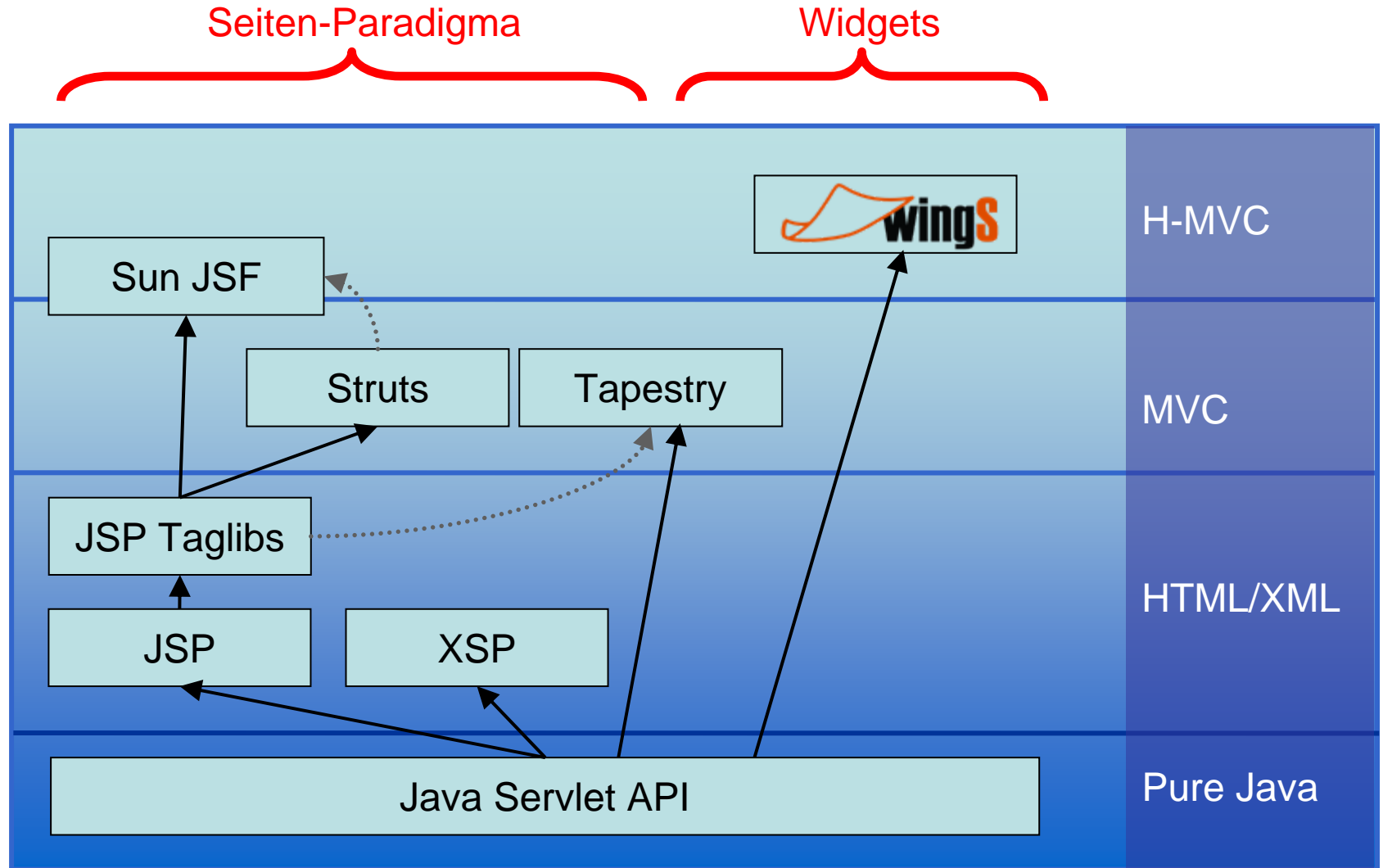
x| Was ist wingS

x| Architektur

x| Applikationserstellung

x| JSF vs. wingS am Beispiel

x| Fazit



- x| Höhere Effizienz** (Aufwand, Fehlerrate)
- x| Vereinfachung** des Entwicklungsprozesses (notwendige Skills)
- x| Kapselung** und **Behandlung technischer Beschränkungen** der Web-Plattform
- x| Gute Wiederverwendbarkeit** durch **Komponentenorientierung** und **Polymorphie**
- x| Trennung** von Logik, Darstellung und Inhalten durch **MVC** und **Event-Orientierung**
- x| Hoher Abstraktionsgrad** für modellorientierte Konzepte (MDA)
- x| Eignung** für **Refactoring** und **Unit Tests**

→ **H-MVC Architektur**

- x| Einleitung
- x| **Was ist wingS**
- x| Architektur
- x| Applikationserstellung
- x| JSF vs. wingS am Beispiel
- x| Fazit

## x| „wingS is net generation Swing“:

- | **Komponenten-** und **Event**basiert
- | **H-MVC** – Architektur (schachtelbares MVC)

## x| **Projektziele:**

- | Möglichst nahe am Swing Programmiermodell
- | Eigenheiten des Webs im Framework abdecken
- | Schlankheit und Effizienz
- | Einfacher, wiederverwendbarer Anwendungscode

## x| **Open Source** unter **LGPL** Lizenz

## x| Version 1.0 anno **2001**

Version 2.0 im Dezember 2005

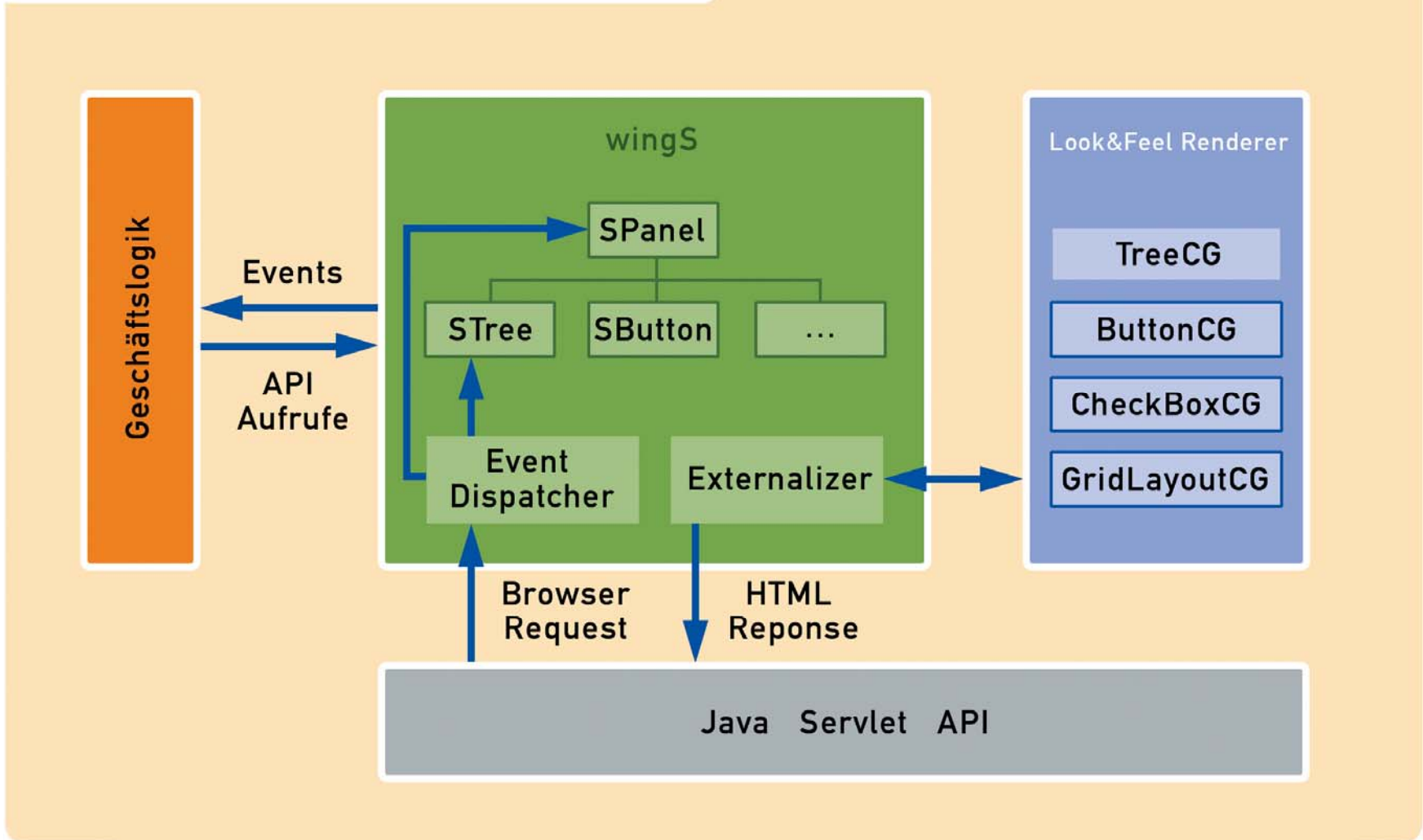
- x| Umfangreicher Satz an **visuellen Komponenten**  
(Textfields, Buttons, Trees, Tables, Tabbed Panes, Menues)
- x| Leistungsstarke **Layout Manager**
- x| Verwendet **Swing Modelle** und **Events**
- x| Styling durch komponentenweise **generiertes CSS**
- x| **Modulare Renderer** (Pluggable Look & Feel)
- x| Behandlung von **Back** Navigation im Browser

## Neue Features in wingS 2

- x| **leichter anpassbares** Default Look & Feel
- x| **Drag & Drop** Support
- x| **Remote Scripting** (AJAX)
- x| Kontextmenüs, Tooltips und Tastatur-Shortcuts

- x| Einleitung
- x| Was ist wingS
- x| **Architektur**
- x| Applikationserstellung
- x| JSF vs. wingS am Beispiel
- x| Fazit

## FUNKTIONSWEISE VON WINGS



- x| Jede wingS Komponente hält ihren Zustand innerhalb ihres **Modells**
- x| Die **wingS Session** hält den Komponenten-Baum und damit den gesamten Applikationszustand
- x| HTTP Anfragen werden als Menge von **Low-level Events** verstanden und vom **Dispatcher** an die betroffenen Komponenten verteilt
- x| Dort führen Sie zu **Modelländerungen** und lösen ggf. **Applikations-Events** aus
- x| Danach übernimmt der **Externalizer** die Beantwortung der HTTP Anfrage und stellt dem Browser alle Ressourcen zur Verfügung (HTML, CSS, Graphiken, Javascript)
- x| Die HTML-Darstellung wird dabei über die zuständigen **Renderer** durch Durchlaufen des Komponenten-Baums generiert

- x| Einleitung
- x| Was ist wingS
- x| Architektur
- x| **Applikationserstellung**
- x| JSF vs. wingS am Beispiel
- x| Fazit

## Darstellung

- x| Die visuelle Darstellung der Komponenten bestimmt sich vorrangig über die **Inhalte ihres Modells** aber auch über ihre gesetzten **Styling Eigenschaften** (Größe, Farbe, etc.)

## Layout

- x| Über Container-Komponenten lassen sich Komponenten zu komplexeren Komponenten kombinieren
- x| Verantwortlich für die Anordnung innerhalb eines Containers sind dabei **Layout-Manager**

## Logik

- x| **Event Listener** erlauben die Anbindung der Geschäftslogik an die einzelnen Komponenten und damit die Reaktion auf Benutzerinteraktionen

## Inhalte

### x| Verändern der Modellinhalte einer Komponente

```
label.setText( „Text“ )  
radioButton.getModel().setSelected(true);
```

## Dynamische Styles

### x| Setzen der visuellen Komponenteneigenschaften

```
c.setForeground(Color.RED);  
c.setBorder(new SEmptyBorder(5,5,5,5));  
c.setFont(new SFont(„Arial“));
```

### x| Explizite Zuweisung von CSS-Eigenschaften und -Klassen

```
c.setAttribute(CSSProperty.FONT-STYLE, „bold“);  
c.setClass(„fehlermeldung“)
```

## Statische Eigenschaften

### x| Einbinden **globaler CSS** Deklarationen

### x| Schreiben **eigener Renderer**

## Dynamische Layout-Manager

Regelbasierte Anordnung der Komponenten eines Containers (Tabellenweise, Zeilenweise, etc.). Analog zu den Swing Layout-Managern

### Beispiele:

SBorderLayout, SFrameSetLayout, SGridLayout, GridBagLayout, ...

---

## Statischer Manager (STemplateLayout)

Platzierung von Komponenten innerhalb eines HTML-Fragments

Eignet sich gut zur Realisierung des statischen Rahmenlayouts

```
panel.setLayout(new STemplateLayout(url));  
panel.add(new SLabel("Hello World!"), "hello");  
panel.add(new SButton("Ok"), "button");
```

```
<h1><object name=„hello“></object></h1>  
<table>  
    <tr><td>...</td></tr>  
    <tr><td><object name=„button“></object>  
    </td></tr>  
</table>
```

## Analog zu Swing:

- x| Anbindung der Logik an die wingS Komponenten via **Events**
- x| Typische Events:
  - „Button gedrückt“, „Tabellenzeile gewählt“, „Text eingegeben“
- x| Werte und Zustände von Eingabe-Komponenten werden in ihren **Modellen** gehalten und können dort abgefragt werden

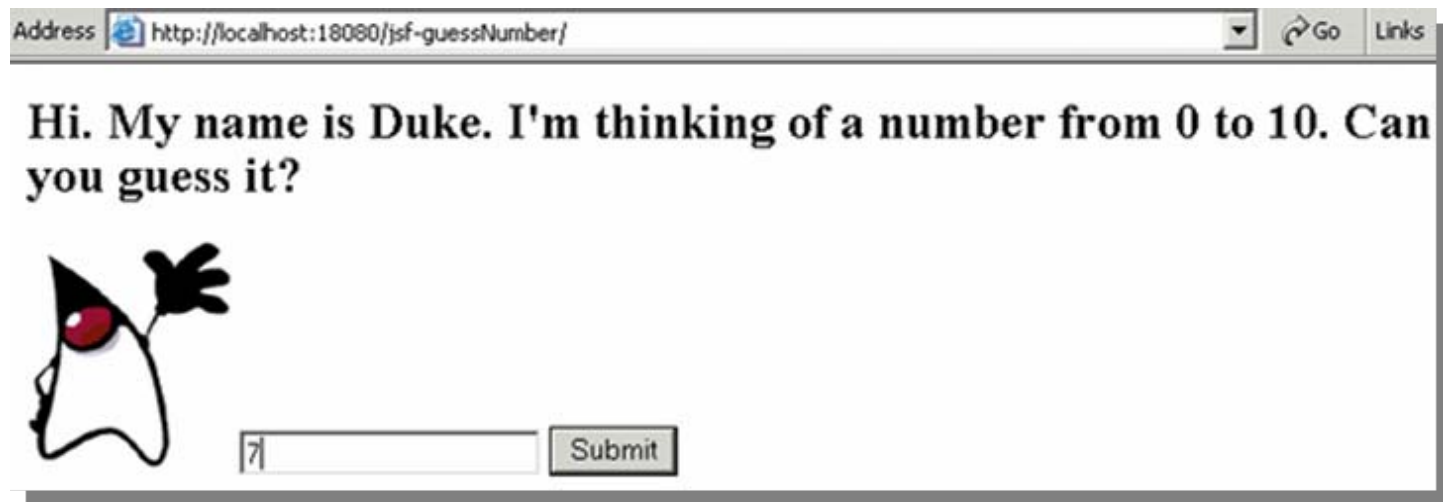
---

```
final SComboBox auswahl = new SComboBox(new String[] {"Ja", "Nein"});
SButton absenden = new SButton("Wählen");
absenden.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent actionEvent) {
        Object wahl = auswahl.getSelectedItem();
        // Daten via Geschäftslogik verarbeiten
        verarbeiteAuswahl(wahl);
    }
});
```

- x| Einleitung
- x| Was ist wingS
- x| Architektur
- x| Applikationserstellung
- x| JSF vs. wingS am Beispiel**
- x| Fazit

## Eindrücke aus den JavaServer Faces (JSF):

Das *Guess-A-Number* Beispiel von Sun



Quelle: Sun

## Einbinden der Logik via JSF-Tags in die JSP Seite

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<body bgcolor="white">
<f:view>
  <h:form id="helloForm" >
    <h2>Hi. My name is Duke. I'm thinking of a number from
      <h:outputText value="#{UserNumberBean.minimum}"/> to
      <h:outputText value="#{UserNumberBean.maximum}"/>.
      Can you guess it?</h2>
    <h:graphicImage id="waveImg" url="/wave.med.gif" />
    <h:inputText id="userNo" value="#{UserNumberBean.userNumber}"
      validator="#{UserNumberBean.validate}"/>
    <h:commandButton id="submit" action="success" value="Submit"/>
    <p><h:message style="color: red; serif; font-style: oblique;
      text-decoration: overline,, id="errors1" for="userNo"/>
    </h:form>
</f:view>
```

Quelle: Sun

## Ablaufsteuerung in JSF via faces-config.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE faces-config PUBLIC
"-//Sun Microsystems, Inc.//DTD JSF Config 1.1//EN"
"http://java.sun.com/dtd/web-facesconfig_1_1.dtd">
<faces-config>

<application>
  <locale-config>
    <default-locale>en</default-locale>
    <supported-locale>de</supported-locale>
    <supported-locale>fr</supported-locale>
    <supported-locale>es</supported-locale>
  </locale-config>
</application>

<navigation-rule>
  <description>
    The decision rule used by the NavigationHandler
  </description>
  <from-view-id>greeting.jsp</from-view-id>
  <navigation-case>
    <description>
      Indicates to the NavigationHandler [...]
    </description>
    <from-outcome>success</from-outcome>
    <to-view-id>response.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

```
<navigation-rule>
  <description>
    The decision rules used by [...]
  </description>
  <from-view-id>response.jsp</from-view-id>
  <navigation-case>
    <description>
      Indicates to the NavigationHandler [...]
    </description>
    <from-outcome>success</from-outcome>
    <to-view-id>greeting.jsp</to-view-id>
  </navigation-case>
</navigation-rule>

<managed-bean>
  <description>
    The "backing file" bean that backs up [...]
  </description>
  <managed-bean-name>UserNumberBean</managed-bean-name>
  <managed-bean-class>guessNumber.UserNumberBean</managed-
bean-class>
  <managed-bean-scope>session</managed-bean-scope>
  <managed-property>
    <property-name>minimum</property-name>
    <property-class>int</property-class>
    <value>0</value>
  </managed-property>
  <managed-property>
    <property-name>maximum</property-name>
    <property-class>int</property-class>
    <value>10</value>
  </managed-property>
</managed-bean>

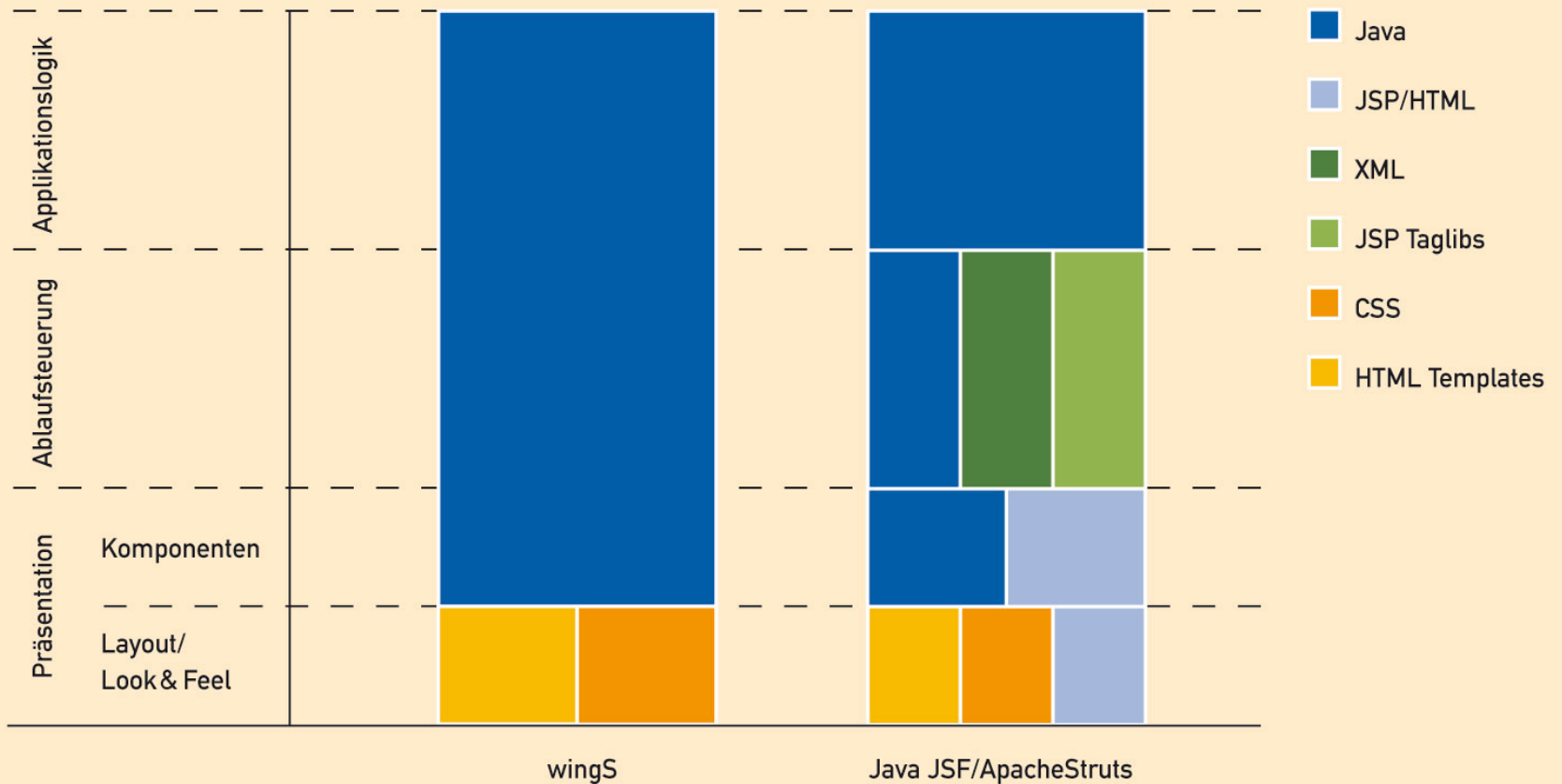
</faces-config>
```

## Prüflogik in JSF als Validator

```
public void validate(FacesContext context,
    UIComponent component,
    Object value) throws
ValidatorException {
    if ((context == null) || (component == null)) {
        throw new NullPointerException();
    }
    if (value != null) {
        try {
            int converted = intValue(value);
            if (maximumSet &&
                (converted > maximum)) {
                if (minimumSet) {
                    throw new ValidatorException(
                        MessageFactory.getMessage(
                            context, component,
                            Validator.NOT_IN_RANGE_MESSAGE_ID,
                            new Object[]{
                                new Integer(minimum),
                                new Integer(maximum)
                            }
                        ));
                } else {
                    throw new ValidatorException(
                        MessageFactory.getMessage(
                            context, component,
                            LongRangeValidator.MAXIMUM_MESSAGE_ID,
                            new Object[]{
                                new Integer(maximum)
                            }
                        ));
                }
            }
            if (minimumSet &&
                (converted < minimum)) {
                if (maximumSet) {
                    throw new
validatorexception(messagefactory.getMessage(
                        context, component,
                        validator.not_in_range_message_id,
                        new object[]{
                            new double(minimum),
                            new double(maximum)
                        }
                    ));
                } else {
                    throw new validatorexception(
                        messagefactory.getMessage(
                            context, component,
                            longrangevalidator.minimum_message_id,
                            new object[]{
                                new integer(minimum)
                            }
                        ));
                }
            }
        } catch (numberformatexception e) {
            throw new validatorexception(
                messagefactory.getMessage(
                    context, component,
                    longrangevalidator.type_message_id));
        }
    }
}
```

Quelle: Sun

## Weniger Medienbrüche und deutlich reduzierte Komplexität in der Entwicklung durch wingS



```
public class HelloWingS {
    public HelloWingS() {
        SGridLayout gridLayout = new SGridLayout(1);
        SForm panel = new SForm(gridLayout);
        SLabel titel = new SLabel("Hello World - this is wings!");
        SButton okButton = new SButton("Guess!");
        titel.setFont(new SFont(null, SFont.BOLD, 18));
        gridLayout.setVgap(10);

        final SLabel message = new SLabel();
        final STextField textField = new STextField();
        final int randomNr = new Random().nextInt(11);

        // check our guesses and respond with according message
        okButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                if (Integer.toString(randomNr).equals(textField.getText()))
                    message.setText("Congratulations! You guessed my number!");
                else
                    message.setText("No - '"+textField.getText()+
                        "' is not the right number. Try again!");
            }
        });

        // arrange components using a grid layout
        panel.add(titel);
        panel.add(new SLabel("We want fun, so let's play a game!\n" +
            "Try to guess a number between 1 and 10."));
        panel.add(textField);
        panel.add(okButton);
        panel.add(message);

        SFrame rootFrame = new SFrame();
        rootFrame.getContentPane().add(panel);
        rootFrame.setVisible(true);
    }
}
```



- x| Einleitung
- x| Was ist wingS
- x| Architektur
- x| Applikationserstellung
- x| JSF vs. wingS am Beispiel
- x| **Fazit**

## JavaServer Faces

- x| **J2EE Standard**
- x| Hohe **Öffentlichkeitswirkung**
- x| Konzeptionell immer noch **JSP/Struts**
  
- x| **Effizienz** nur über Tools möglich
- x| Fehlerträchtige Pflege von **drei Quellen** notwendig
  - | Dialogseiten (HTML/JSP/Taglibs)
  - | Dialogbeans (Java)
  - | Ablaufsteuerung (XML)
  
- x| JSP/HTML Basis impliziert:
  - | Mangelnder **Abstraktionsgrad**
  - | Effizienz Dynamische/**MDA** Konzepte schwierig
  - | Schlechte **Wiederverwendbarkeit / Polymorphie**
  - | **Vermischung Darstellung/Logik**
  - | Schwieriges **Refactoring** und **Debugging**



- x| „**Echtes**“ **OO-/** **Komponenten**basiertes Design von Webapplikationen durch **H-MVC**
- x| Hohe **Effizienz** durch hohen Abstraktionsgrad
  - | Reduzierter und übersichtlicher Code
  - | Ermöglicht **MDA** orientierte Konzepte (vgl. pleXX)
  - | Viele Web-Aspekte für Entwickler komplett transparent
  
- x| Klare **Trennung** von **Darstellung** und **Logik**
  
- x| Java-Basiertheit bedeutet
  - | **Einfach**: Java-Skills genügen, Swing Know-how hilft
  - | weniger **Fehler** möglich (Compiler/Syntax)
  - | kein Technologie-Mix und einfache **Wartung**
  - | **Polymorphie** und **Refactoring** kein Problem
  - | Ermöglicht HotSpot Deployment und **Debugging**

## Ausstellungsbereich Stand 11.3

Benjamin Schmid (B.Schmid@excellent.de)

www.j-wings.org

www.excellent.de/wings

