

Komponentenorientierte Web- anwendungen mit wingsS

Eine Alternative zu JSF und Struts



x| Einleitung

x| Was ist wingS

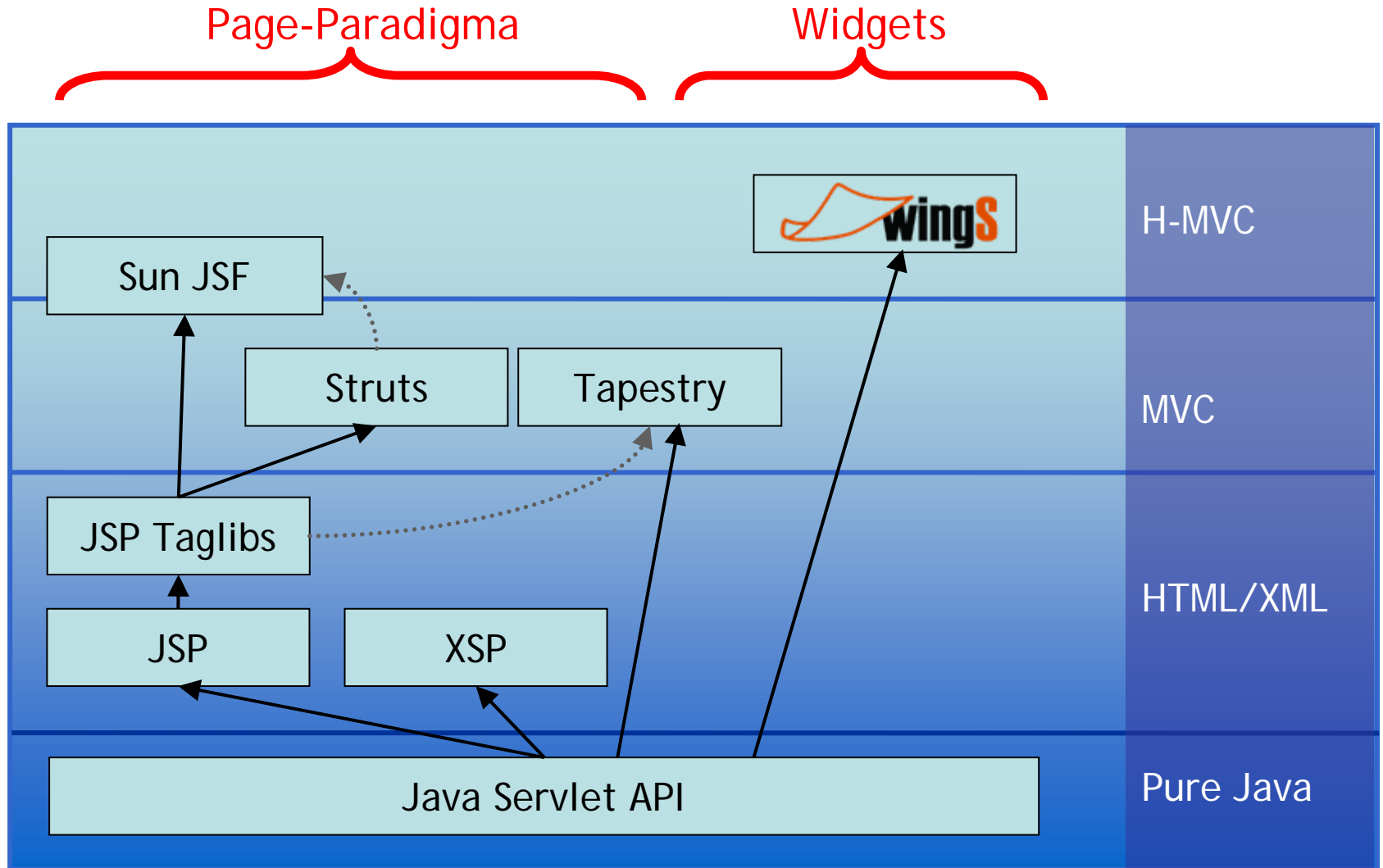
x| Architektur

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
 - x| Gute **Wiederverwendbarkeit** des Anwendungscodes durch Komponentenorientierung und Polymorphie
 - x| **Trennung** von Logik, Darstellung und Inhalten mittels MVC und Event-Orientierung
 - x| Hoher **Abstraktionsgrad** für modellorientierte Konzepte (MDA)
 - x| Eignung für Refactoring und Unit Tests
- H-MVC Architektur

- xl Einleitung
- xl Was ist wingS
- xl Architektur
- xl Applikationserstellung
- xl JSF vs. wingS am Beispiel
- xl Fazit





- x| „wingS is net generation Swing“ :**
 - | Komponenten- und Eventbasiert
 - | H-MVC - Architektur (schachtelbares MVC)

- x| Projektziele:**
 - | Möglichst nahe am **Swing** Programmiermodell
 - | Eigenheiten des Webs im Framework abdecken
 - | Einfacher, wiederverwendbarer Anwendungscode

- x| Open Source unter LGPL Lizenz**

- x| Version 1.0 anno 2001**
Version 2.0 seit Dezember 2005

- xl Umfangreicher Satz an visuellen Komponenten (Input fields, Buttons, Trees, Tables, Tabs, Menus)
- xl Leistungsstarke Layout Manager
- xl Verwendet Swing Modelle und Events
- xl Styling durch komponentenweise generiertes CSS
- xl Modulare Renderer (Pluggable Look & Feel)

Neue Features in wingS 2

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

x| Einleitung

x| Was ist wingS

x| **Architektur**

x| Applikationserstellung

x| JSF vs. wingS am Beispiel

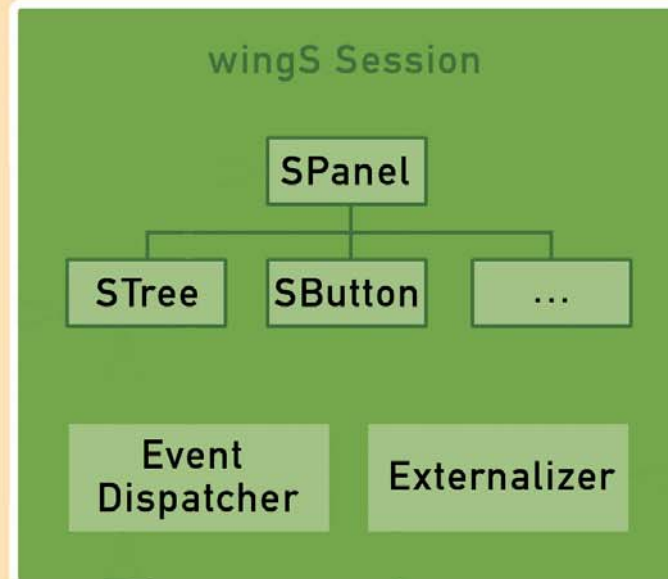
x| Fazit



FUNKTIONSWEISE VON WINGS



Geschäftslogik



Look&Feel Renderer

TreeCG

ButtonCG

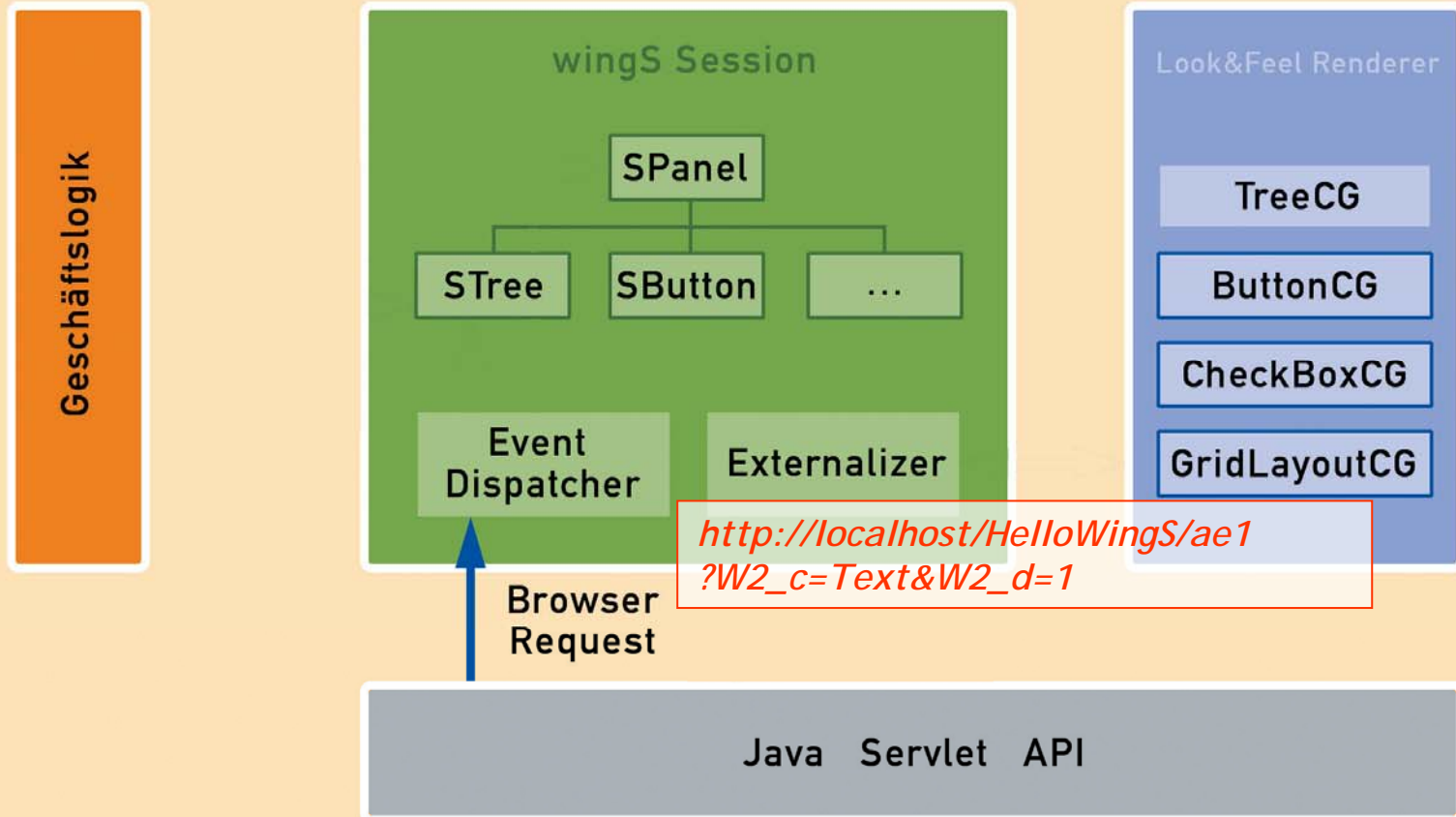
CheckBoxCG

GridLayoutCG

Java Servlet API

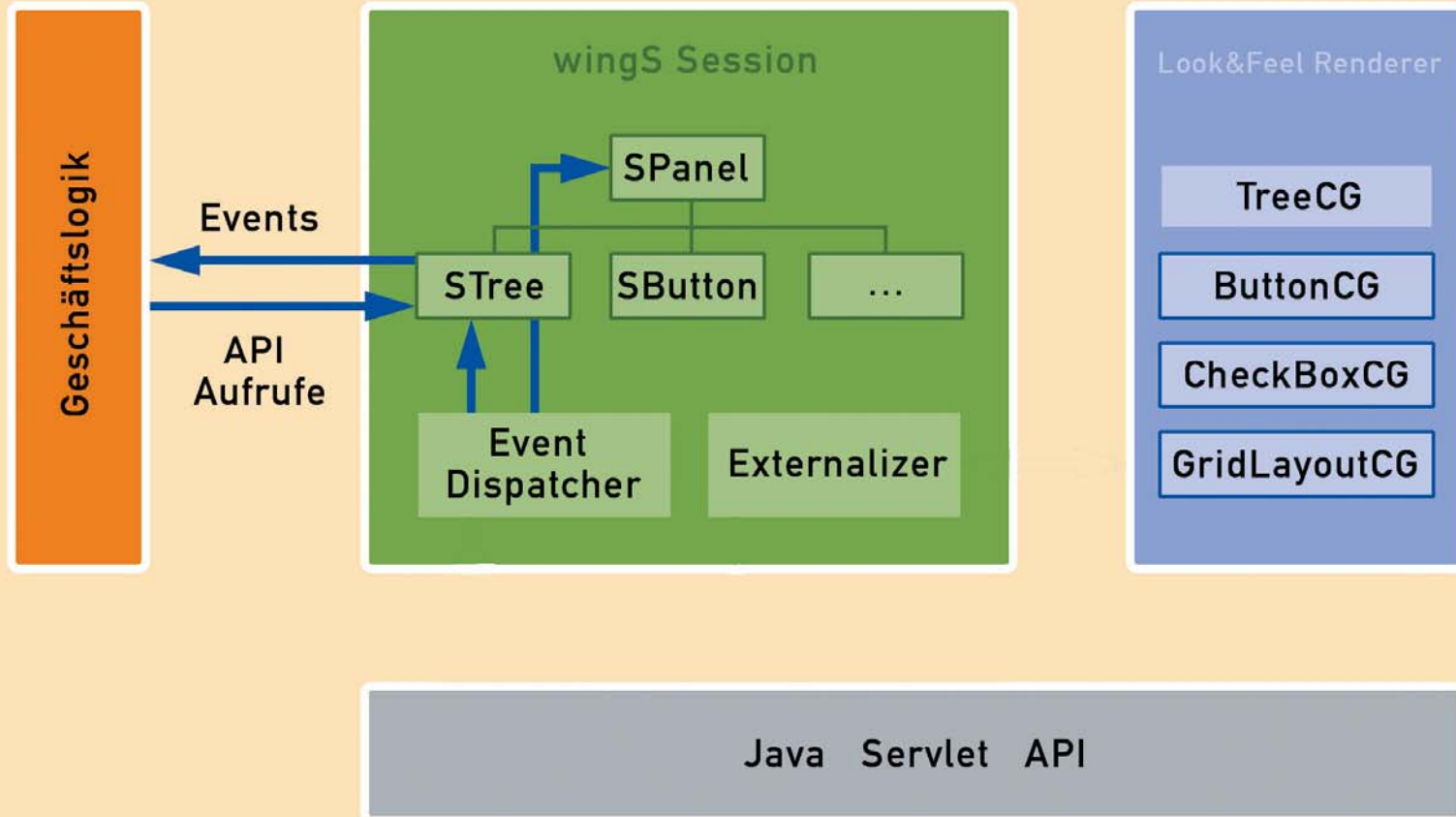
- xl Wie in Swing werden Zustandsänderungen an die interaktiven GUI Komponenten durch **Events** übermittelt und von ihren (Swing-) Modellen gehalten
- xl Die wingS **Session** hält den Komponenten-Baum und damit den gesamten Applikationszustand
- xl Interaktionen mit GUI-Elemente lösen bei Bedarf **HTTP-Requests** aus. Das wingS Servlet sucht für jeden HTTP-Request die passende wingS Session und leitet diesen an sie weiter
- xl Die Werte innerhalb eines Requests werden als Sammlung von **Low-level Events** mit jeweils ID, Epoche und Inhalt/Änderung der Komponenten verstanden

FUNKTIONSWEISE VON WINGS



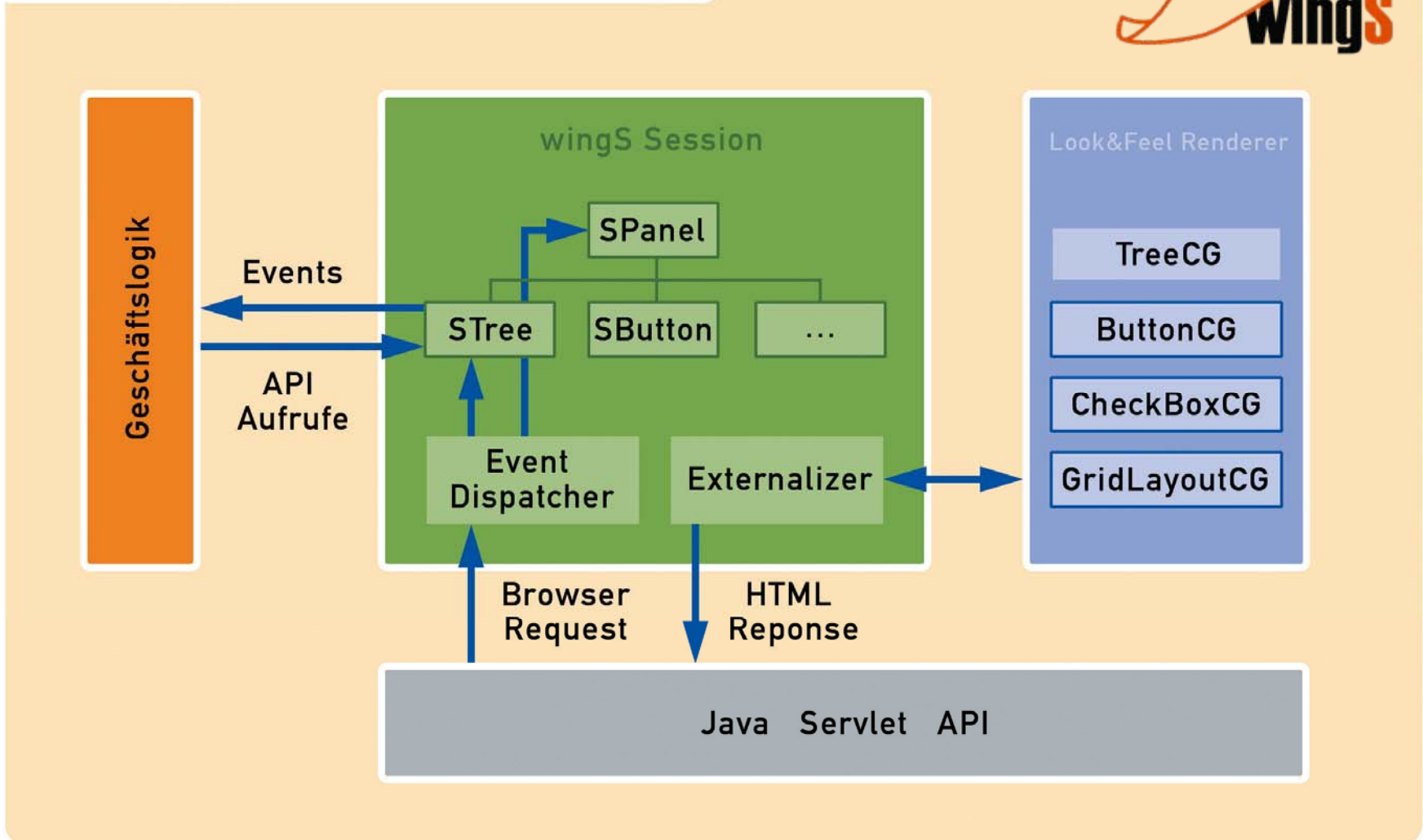
- x| Der **Dispatcher** kennt alle interaktiven GUI Komponenten einer Session und verteilt die im Request enthaltenen **Low-Level Events**
- x| Ein Request kann mehrere Komponenten auf einmal betreffen. Daher erfolgt die Abarbeitung in zwei Phasen:
 - 1) **Zustandsänderung** der betroffenen Komponenten (Daten aktualisieren)
 - 2) **Feuern** der **Komponenten-Events** (Business-Logik auslösen)
- x| Requests aus alten Sichten (**Back-Button**) tragen abgelaufene Epochennummern und führen im Standardfall zu **keinen Zustandsänderungen** und Events

FUNKTIONSWEISE VON WINGS



- x| Danach wird der Request an den zuständigen **Externalizer** weitergegeben, welcher den HTTP Request beantwortet
- x| Alle **Ressourcen** (HTML, CSS, Bilder, JavaScript, PDF) werden vom Externalizer mittels HTTP bereit gestellt
- x| Der Externalizer kennt:
 - | **Dynamische Ressourcen**
 - | **Statische Ressourcen** (Caching erlaubt)
- x| Die **HTML-Seite** (SFrame) ist eine dynamische Ressource und wird von wingS über Aufrufe der jeweiligen **Komponenten-Renderer** generiert
Analog erfolgt die Generierung des stylenden **CSS**

FUNKTIONSWEISE VON WINGS



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



Darstellung

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

Layout

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

Logik

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

Viele wingS Komponenten verwenden Swing-Modelle um die darzustellenden Daten zu halten

CheckBox

```
SCheckBox checkBox = new SCheckBox("CheckBox");  
checkBox.setSelected(true);
```

Eintrag 1

```
ComboBoxModel model = new  
    javax.swing.DefaultComboBoxModel();  
model.addElement("Eintrag 1");  
model.addElement("Eintrag 2");  
SComboBox comboBox = new SComboBox(model);
```

	A	B	C
1	1	2	3
2	5	6	

```
Object[][] data =  
    new Object[][]{{"1", "2", "3"}, {"4", "5", "6"}};  
Object[] spaltenNamen = new Object[]{"A", "B", "C"};  
TableModel tm = new javax.swing.table.  
    DefaultTableModel(data, spaltenNamen);  
STable tabelle = new STable(tm);
```

Dynamische Styles

x| Setzen der visuellen Komponenteneigenschaften

```
SComponent c = new SLabel(„Test-Komponente“);  
c.setForeground(Color.RED);  
c.setBorder(new SLineBorder(Color.GREEN, 2));  
c.setFont(new SFont(„Arial“));  
c.setVisible(true);
```

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

Automatische, regelbasierte Anordnung der Komponenten einer Container-Komponente (Tabellenweise, Zeilenweise, etc.).

Analog zu den Swing Layout- Managern:

Border-, Grid-, GridBag-, FlowLayout, ...

Beispiele:

```
// Horizontales Flow-Layout
SPanel flowPanel = new SPanel(new SFlowLayout());
for (int i = 1; i < 15; i++) {
    SLabel label = new SLabel("Komponente " + i);
    label.setBorder(new SLineBorder(i%2==0 ? Color.red : Color.green));
    flowPanel.add(label);
}

// Tabellenbasiertes GridBag-Layout
SPanel gridBagPanel = new SPanel(new SGridBagLayout());
GridBagConstraints c = new GridBagConstraints();
gridBagPanel.setPreferredSize(new SDimension(600, SDimension.AUTO_INX));
for (int i = 1; i < 5; i++) {
    c.weightx = i*2;
    gridBagPanel.add(new SLabel("Komponente "+i), c);
}
```

Statischer Layout-Manager

Positionierung von Komponenten innerhalb eines HTML-Fragments

Ideal zur Realisierung komplexerer, statischer Rahmenlayouts

Beispiel:

Deklaration des Layout-Managers:

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

HTML-Vorlage:

```
<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 Swing 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 JComboBox auswahl = new JComboBox(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);
    }
});
```

Das **WingSet-Demo** ist Teil der wingS Distribution und demonstriert den verfügbaren Satz an Komponenten

The screenshot shows a Mozilla Firefox browser window displaying the 'WingSet Example' page. The page title is 'WingSet Demo - Mozilla Firefox' and the address bar shows 'http://localhost:8080/wingset/WingSet/nh17W28_examples=14'. The page content includes a header with the 'WingSet' logo and a description: 'An example demonstrating some wingS properties, similar to the Java SwingSet example.' Below the description is a horizontal menu of component names: wingS!, Label, Border, Text Component, Tree, OptionPane, Table, List, Button, ToggleButton, CheckBox, RadioButton, Faces, FileChooser, ScrollPane, PageScroller, Menu, Tabbed Pane, Template Layout, Interactive Template, ProgressBar, Memory Usage, Script Listener, Popup Menu, Keyboard Bindings, Dynamic Layouts, Browser Back, DesktopPane, Drag and Drop, Raw Text Component, Error Page, and Limited table nesting (DEVEL). Below the menu are input fields for 'width' and 'height', and checkboxes for 'Show as Form Component', 'Paged Scrolling', 'table', and 'tree'. The main content is a table with 10 columns (col 3 to col 12) and 10 rows (5 to 14). The table cells contain text like 'cell 4:6' and 'cell 4:7'. At the bottom of the browser window, there is a 'view java source code' link and a status bar showing 'Fertig' and '0.594s Adblock'.

wingS Example

An example demonstrating some wingS properties, similar to the Java SwingSet example.

wingS! | Label | Border | Text Component | Tree | OptionPane | Table | List | Button | ToggleButton | CheckBox | RadioButton | Faces | FileChooser | ScrollPane | PageScroller | Menu | Tabbed Pane | Template Layout | Interactive Template | ProgressBar | Memory Usage | Script Listener | Popup Menu | Keyboard Bindings | Dynamic Layouts | Browser Back | DesktopPane | Drag and Drop | Raw Text Component | Error Page | Limited table nesting (DEVEL)

apply width height Show as Form Component Paged Scrolling table tree

	col 3	col 4	col 5	col 6	col 7	col 8	col 9	col 10	col 11	col 12
5		<input checked="" type="checkbox"/>	4	cell 4:6	cell 4:7	cell 4:8	cell 4:9	cell 4:10	cell 4:11	cell 4:12
6		<input type="checkbox"/>	5	cell 5:6	cell 5:7	cell 5:8	cell 5:9	cell 5:10	cell 5:11	cell 5:12
7		<input checked="" type="checkbox"/>	6	cell 6:6	cell 6:7	cell 6:8	cell 6:9	cell 6:10	cell 6:11	cell 6:12
8		<input type="checkbox"/>	7	cell 7:6	cell 7:7	cell 7:8	cell 7:9	cell 7:10	cell 7:11	cell 7:12
9		<input checked="" type="checkbox"/>	8	cell 8:6	cell 8:7	cell 8:8	cell 8:9	cell 8:10	cell 8:11	cell 8:12
10		<input type="checkbox"/>	9	cell 9:6	cell 9:7	cell 9:8	cell 9:9	cell 9:10	cell 9:11	cell 9:12
11		<input checked="" type="checkbox"/>	10	cell 10:6	cell 10:7	cell 10:8	cell 10:9	cell 10:10	cell 10:11	cell 10:12
12		<input type="checkbox"/>	11	cell 11:6	cell 11:7	cell 11:8	cell 11:9	cell 11:10	cell 11:11	cell 11:12
13		<input checked="" type="checkbox"/>	12	cell 12:6	cell 12:7	cell 12:8	cell 12:9	cell 12:10	cell 12:11	cell 12:12
14		<input type="checkbox"/>	13	cell 13:6	cell 13:7	cell 13:8	cell 13:9	cell 13:10	cell 13:11	cell 13:12

[view java source code](#)

Fertig 0.594s Adblock

Live-Coding einer einfachen Mini-Demoapplikation

```
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);  
    }  
}
```



In den Session Materialien enthaltene **Quickstart-Applikation**

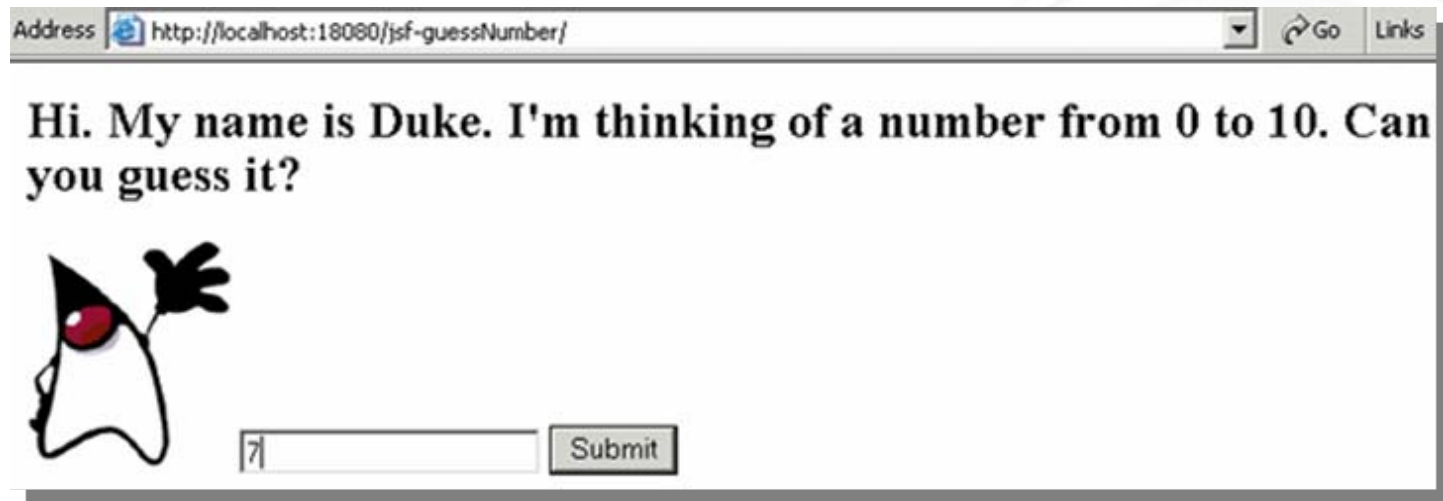


- x|** Einleitung
- x|** Was ist wingS
- x|** Architektur
- x|** Applikationserstellung
- x|** JSF vs. wingS am Beispiel
- x|** Fazit



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>
```

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>
```

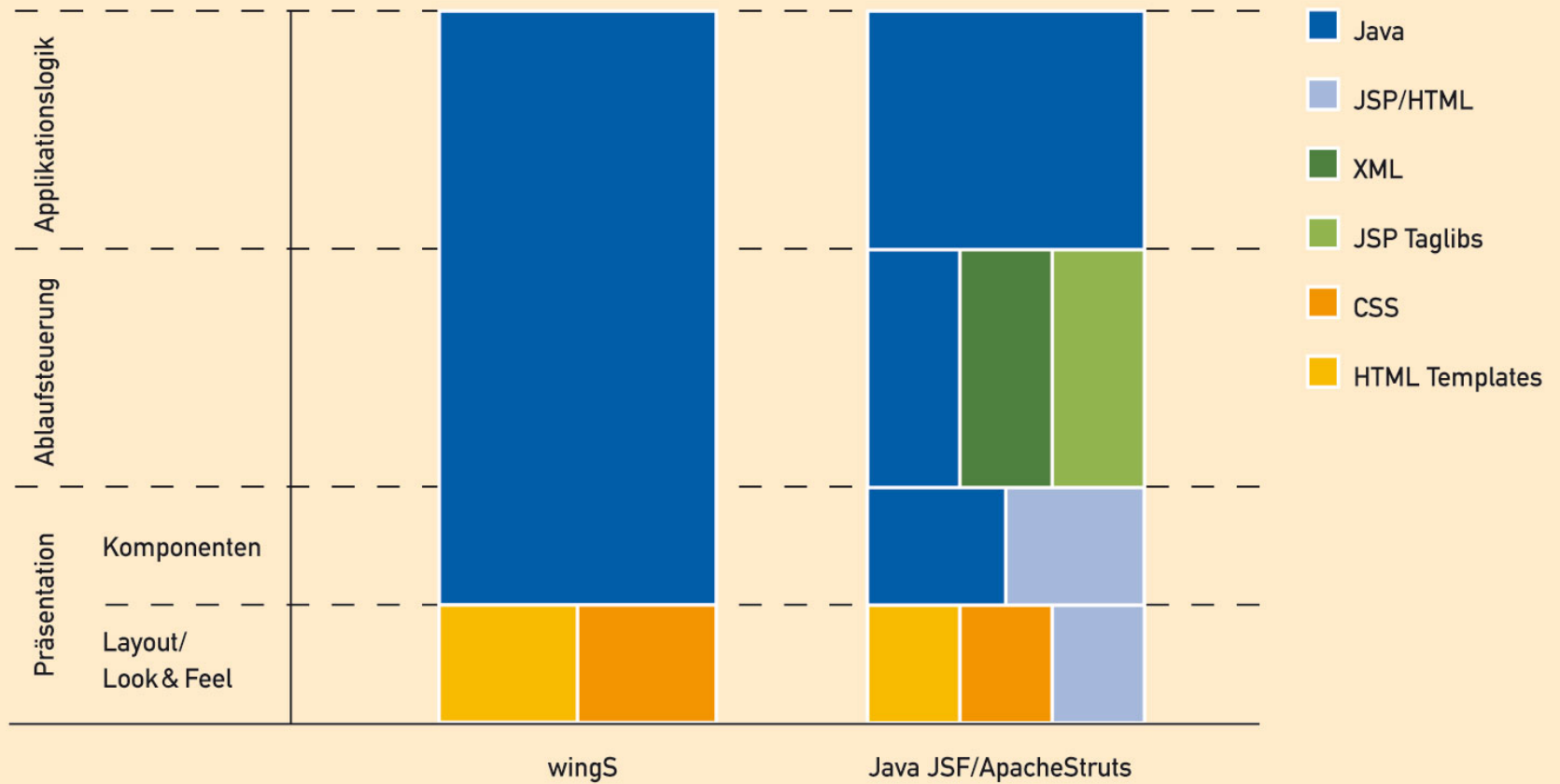
Prüflogik in JSF als Validator

```
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**



- xl **J2EE Standard**
- xl Hohe Öffentlichkeitswirkung
- xl Konzeptionell stark **JSP/Struts**

- xl Effizienz über Tools
- xl Pflege von **drei Quellen** notwendig
 - | Dialogseiten (HTML/JSP/Taglibs)
 - | Dialogbeans (Java)
 - | Ablaufsteuerung (XML)

- xl **JSP/Seiten-Paradigma** impliziert:
 - | Geringerer **Abstraktionsgrad**
 - | Problematische **Wiederverwendung / Polymorphie**
 - | Starke Koppelung der **Darstellung und Logik**
 - | Schwieriges **Refactoring und Debugging**

- xl **Java-/OO-/Komponenten**-zentriertes Design von Webapplikationen durch H-MVC
- xl Hohe **Effizienz** durch hohen Abstraktionsgrad
 - | Reduzierter und übersichtlicher Code
 - | Ermöglicht MDA orientierte Konzepte (vgl. pleXX)
 - | Viele Web-Aspekte für Entwickler transparent
- xl Klare Trennung von Darstellung und Logik
- xl Java-Fokussierung bedeutet
 - | **Einfach**: Java-Skills genügen, Swing Know-how hilft
 - | Bessere Fehlervermeidung (Compiler/Syntax)
 - | reduzierter Technologie-Mix und einfachere Wartung
 - | **Polymorphie** und Refactoring
 - | Ermöglicht HotSpot Deployment und **Debugging**



www.j-wings.org



www.excellent.de/wings/
j-wings-users@lists.sourceforge.net