WidgetLib

Customizing von Benutzeroberflächen im Browser

W3L AG
info@W3L.de

2015

Inhalt

- Ausgangssituation
  - Customizing
  - Beispiel: Semantische Suche
- Lösung mit der WidgetLib
  - Desktops
  - Widgets
    - Konfiguration & Styles
    - Zustand
    - Kommunikation und Strukturierung
  - Besondere Widgets
    - Layouting
    - Hintergrundverarbeitung

- LIVE DEMO

Aufgabe *Customizing*

AUSGANGSSITUATION

Customizing

- **Kunde wünscht auf ihn zugeschnittene Software**
  - Verhalten (fachlich)
    - Voreinstellungen / Standardwerte
    - Besondere Logiken in Prozessschritten → ActivityLib
  - Aussehen
    - Logos, Styles, Texte
    - Layout
- **Aufwand zur Anpassung soll gering sein**
  - Kommunikation von Anforderungen minimieren
  - Optimal: Kunde kann *selbst* alle Änderungen vornehmen
- **Mehrere Anforderungsprofile bei *einem* Kunden**

Beispiel: Semantische Suche

- **Pro Kunde ca. 30 bis 70 Template-Schnipsel in HTML**
  - Enthalten HTML mit Variablen, zum Teil direkt CSS und Javascript
    - 01-articlesStart.htm

```
<div class="sa_search_results $$RESCLASS$$">
  <div class="content">
```

    - 02-articleContent.htm

```
<div class="article_row">
  <table class="articletable"><tr>
    <td class="articlecol1">
      <div class="article_hitnum">$$HITNUM$$</div>
      <img src="icons/$$ICON$$" class="icon"
           alt="$$ICON_TITLE$$" title="$$ICON_TITLE$$" />
    </td>
    <td class="articlecol2"><h1>
      <a target="_blank" href="$$RESULTHANDLER$$" id="sa_link_$$DOCID$$">
        $$DIN_SUMMARY_TITLE$$
      </a>
    </h1>$$LOCKARTICLE$$</td>
…
  </tr></table>
</div>
```

    - 03-articlesEnd.htm

```
  </div>
</div>
```

→ **Übersicht geht verloren; will man da einen Kunden dran lassen?**

Beispiel: Semantische Suche

- **Kundenspezifische Renderer-Klasse in Java**
  - Auswahl in Admin-Oberfläche
  - Liest Anfrage aus und startet Suche
  - Stellt Templates zusammen
    - AJAX-Request-Javascripts für *Fragment*-Einbindung
    - Iteration über Suchergebnisse
    - Ersetzung von Variablen, z.T. hartkodiert HTML/JS/CSS

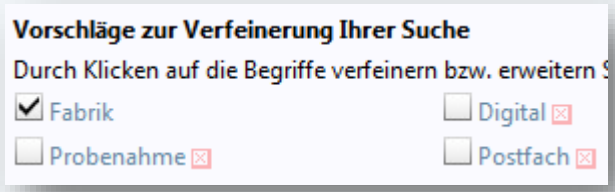
Aus Kundensicht:

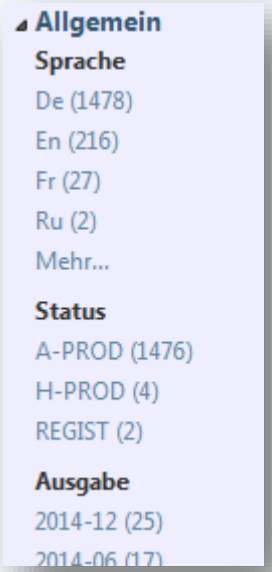
→ Query-Erzeugung nur mit den vorgegebenen Elementen

→ Neue Fragmente können nicht hinzugefügt werden

Beispiel: Semantische Suche

- **Fragmente**
  - Erzeugen HTML/JS, das Teil des einbettenden Suchformulars wird
  - Via Admin-Oberfläche global konfigurierbar
  - Instanziierungsparameter von Renderer vorgegeben
  - Interagieren nicht untereinander, einzige Möglichkeiten:
    - AJAX-Requests an sich selbst schicken
    - Inputs (ggf. hidden) mit Werten versehen
    - Suchformular abschicken





Aus Kundensicht:

→ Parametrisierung global für alle Instanzen

→ Bezug zu Suchergebnisliste muss erklärt werden

Lösung *Customizing*

WIDGETLIB

WidgetLib

- **Bibliothek für Java ab Version 7**
- **Wesentliche Elemente:**
  - Desktop — Basis für Benutzeroberfläche
  - Widget — Inhalt oder Funktionalität anbieten
    - Assoziation — Lässt Widgets interagieren
- **Einbindung in bestehende Anwendungen**
  - Nur ein Servlet muss aufgenommen werden
  - Standalone-Ansicht mit allen Features zur Bearbeitung
  - Abgespeckte Ansicht für reine Darstellung (TODO)

Desktops

- **Als Ersatz für Layout-Templates**
  - WYSIWYG-Positionierung und -Strukturierung von Komponenten
  - Einbindung in andere Komponenten
  - Parametrisierbar
- **Verschiedene Desktops**
  - Pro Benutzer
  - Pro Gruppe
  - Gespeichert
- **Versehen mit Zugriffsrechten**
  - Benutzen, Bearbeiten, Löschen, ...
- **Benutzerabhängiges Hauptmenü**

Widgets

- **Wiederverwendbare Komponenten**
  - Die Elemente, die auf dem Desktop verwaltet werden
  - Stellen bestimmte Funktionen zur Verfügung
    - Text, Bild, ...
    - Steuerelemente
    - Layouts
    - ?
- **Verwaltung in einer erweiterbaren Palette**

- **Hierarchische und nicht-hierarchische Strukturen möglich**
  - Bei Löschung werden Kind-Widgets auch gelöscht
- **Desktop hat Root-Widget**
  - Alle weiteren Widgets sind diesem untergeordnet

Widgets

Konfiguration

- **Alle Einstellungen, die im Normalbetrieb konstant sind**
- **An konkrete Instanz im Desktop gebunden**
  - Im Bearbeitungsmodus direkt am Widget via Doppelklick/Button
- **Einfache Programmierung**
  - Annotiertes POJO
  - GUI zur Bearbeitung wird automatisch erzeugt

```
@ConfigurableProperty(ergName="Beschriftung")
public String getLabel()
{
    return this.label;
}
public void setLabel(String label)
{
    this.label = label;
}
```

Widgets

Konfiguration: Stile

- **Das Aussehen eines Widgets kann Teil seiner Konfiguration sein**
- **Auswahl & Einstellung vordefinierter Stile**
  - Hintergrund
  - Farben
  - Schrift
  - Abstände
- **Benutzer-CSS**
- **Widgets können weitere Stile für Unterelemente vorsehen**
- **Eigene Stil-Definitionen möglich**
  - Beispiel Bild-Widget: Skalierungsmodus

Widgets

Zustand

- **Daten, die im Normalbetrieb verändert werden**
  - Steuerelementzustände
    - Eingegebener Text
    - Ausgewählter Eintrag
    - ...
  - Letztes Rendering
    - Bei Updates werden nur geänderte Widgets neu gerendert
    - Kann invalidiert werden
- **Kann separat gespeichert und wiederhergestellt werden**
- **Standardimplementierung: Key-Value-Store**

Widgets

Kommunikation und Strukturierung

- **Kommunikation mit anderen Widgets nur über sogenannte *Pins***

- **Daten-Pins für Modellierung des *Datenflusses***
  - Daten via Ausgabe-Pin **push**en
  - Daten via Eingabe-Pin **pull**en

  - Nutzdaten wie z.B. Texte
  - Ereignisse (z.B. Button-Klick)

- **Struktur-Pins für Modellierung von *Kenne-Ich-Relation***
  - Andere Widgets via Struktur-Pins erreichen, die ein Interface implementieren und Operationen darauf ausführen

  - Zum Beispiel Zusammenfassung verstreuter Eingabe-Widgets zu einem Formular

Besondere Widgets

Layouting

- **Liefern selbst keine eigentlichen Inhalte**
- **Machen flexible Gestaltung des Desktops erst möglich**
- **Anordnung und Größe der direkten Kind-Widgets**
  - Können Widget-inherente Layout-Angaben respektieren
    - Minimale/Maximale/Bevorzugte Größe

- **Änderung des Layouts durch Endnutzer oder nur Autor**

- **Beispiele:**
  - Horizontale / vertikale Aneinanderreihung
    - Feste oder prozentuale Breite / Höhe
    - Größe je nach Inhalt
  - Tabs, Aufklappbox
    - Wenn unsichtbar → mehrseitige Anwendung
  - Flexibles Grid

Besondere Widgets

Hintergrundverarbeitung

- **Im Normalbetrieb unsichtbar, im Bearbeitungsmodus sichtbar**
- **Ermöglichen visuelle Gestaltung von Prozessen, die zwischengeschaltet sind**
  - Beispiele:
    - Pushen von Daten via Pins bei Eintritt von Ereignissen
    - Verknüpfung von Eingabefeldern mit virtuellem Suchformular
- **Interface-Widgets: Inter-Desktop-Kommunikation**
  - Ähnlich wie in ActivityLib
    - Eingabe-Widget stellt Daten für Desktop zur Verfügung
    - Ausgabe-Widget stellt Daten für andere Desktops bereit
  - Ermöglicht Einbettung von Desktops in andere Desktops
    - Einbettendes Widget pusht/erhält Werte via Interface-Widgets

LIVE DEMO

Vielen Dank
für die Aufmerksamkeit

Inhouse-Schulungen

Wir bieten Inhouse-Schulungen und Beratung durch unsere IT-Experten und –Berater.

Schulungsthemen

- Softwarearchitektur (OOD)
- Requirements Engineering (OOA)
- Nebenläufige & verteilte Programmierung

Gerne konzipieren wir auch eine individuelle Schulung zu Ihren Fragestellungen.

Sprechen Sie uns an!
Tel. 0231/61 804-0, info@W3L.de

W3L-Akademie

In Zusammenarbeit mit der Fachhochschule Dortmund bieten wir

zwei Online-Studiengänge

- B.Sc. Web- und Medieninformatik
- B.Sc. Wirtschaftsinformatik

und 7 Weiterbildungen im IT-Bereich an.

Besuchen Sie unsere Akademie!
http://Akademie.W3L.de