JGoodies Karsten Lentzsch

JSR 296 – SWING APP FRAMEWORK

JGoodies

- Quelloffene Swing-Bibliotheken
- Beispielanwendungen
- Gestalte Oberflächen
- Berate zu Desktop und Swing

- In Expertengruppen zu JSRs 295 und 296
- Eigene 296er-Implementierungen

Ziel

Lernen, warum es die JSR 296 gibt, was sie ist, wie man sie verwendet und ob sie brauchbar ist.

It's easy to program Swing ...

It's easy to program Swing badly.

Das Problem

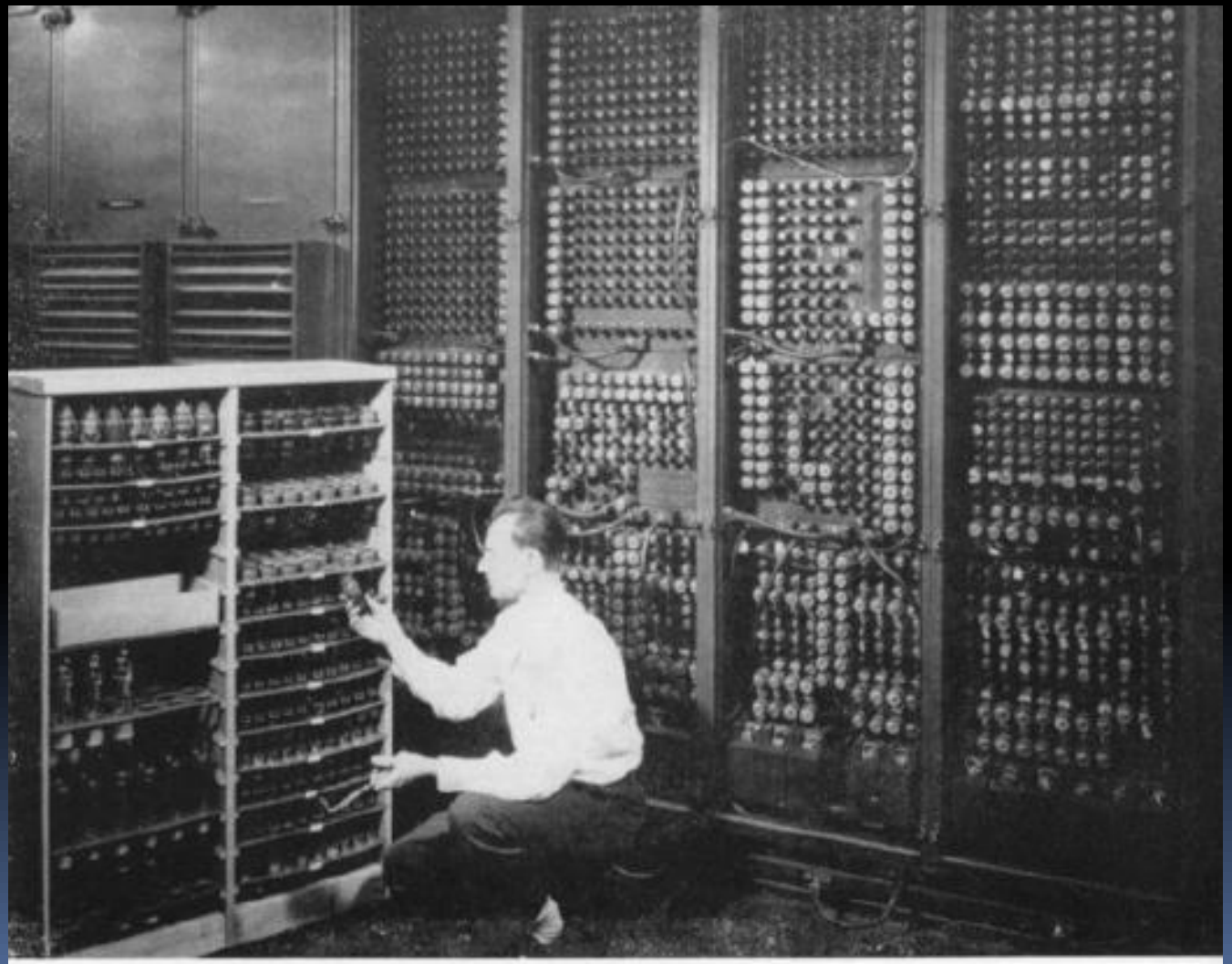
- Swing-API ist riesig
- Kaum Hilfe oberhalb des Toolkits
- Kein Standard für Desktop-Apps
- Desktop-Muster wenig verbreitet
- Schwer für Anfänger und Experten

Entwicklerstudie ->

Die Lösung

- Wiederverwendbares, erweiterbares Rahmenwerk für typische Swing-Aufgaben
- Öffentlicher Prototyp auf java.net
- Entwickelt unter einer JSR
- [War] vorgesehen für Java 7

Ein Monster?



Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.

Monster

- Eclipse RCP
- Netbeans RCP
- Spring RCP / [Spring Desktop]

Harmlos

- So klein wie möglich
- Viel kleiner als Eclipse oder Netbeans RCP
- Etwa 20 Klassen
- Erklärbar in 1 Stunde
- Zielt auf kleine bis mittlere Anwendungen

- Keine Module, Docking, Scripting, Event Bus, GUI- Markup, generisches Datenmodell

Entwurf

- Noch nichtmal ein early-draft.
- Alle Klassen und Methoden sind in Arbeit.
- Folien zu Features, nicht Implementierung

Gliederung

Lebenszyklus

Ressourcen

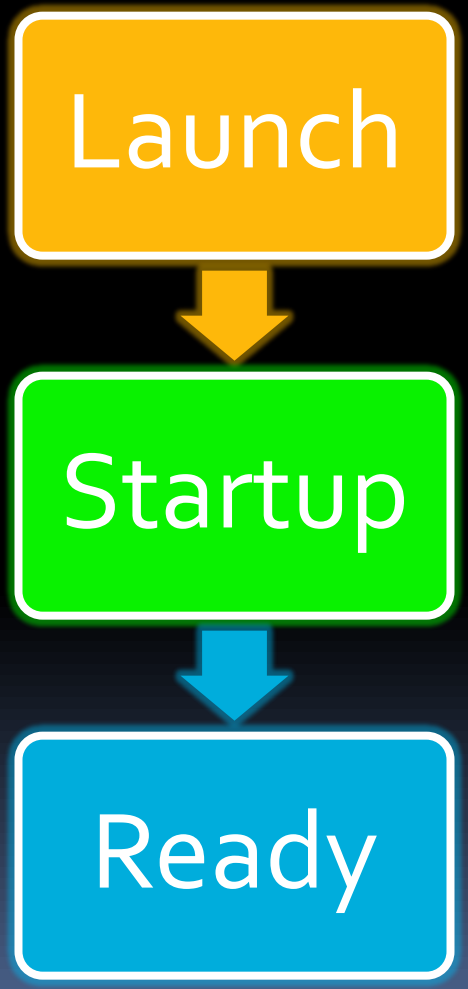
Actions

Tasks

Sonstiges

Stand der JSR

Anwendungslebenszyklus I



Ruft startup() und ready() im EDT auf.
Zumeist aufgerufen von main().

Erzeugt, konfiguriert und zeigt die GUI.
Pflicht.

Dinge, die erledigt werden sollen, wenn die GUI sichtbar ist und eingabebereit.

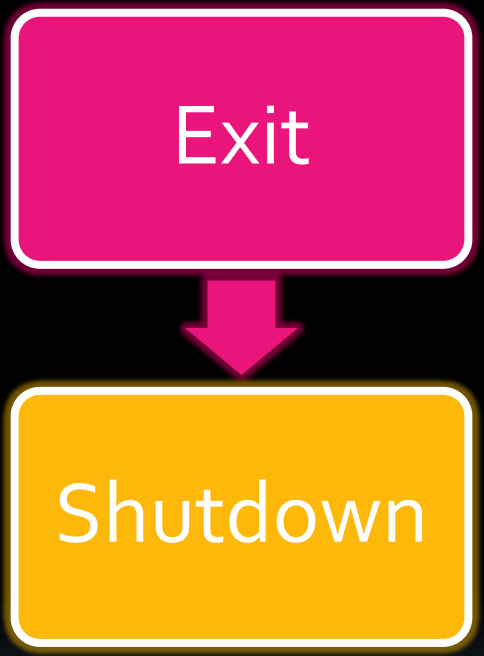
Anwendungsstart

```
public final class Starter {

   public static void main(String[] args) {
      Application.launch(MyApp.class, args);
   }

}
```

Anwendung hoch fahren

```
public class MyApp extends Application {

   protected void startup(String[] args) {
      // Baue, konfiguriere, zeige die GUI
   }

   protected void ready() {
      // Lade Bilder, hole Daten, etc.
   }

}
```

Anwendungslebenszyklus II



Ruft shutdown(), wenn kein ExitListener widerspricht. Meldet den ExitListenern das Runterfahren.

Fährt die Anwendung runter. Aufräumen.

Anwendung runter fahren

```
public void windowClosing(WindowEvent e) {
  Application.getInstance().exit(e);
}

public interface ExitListener {
  // Erlaubst Du das Runterfahren?
  boolean exitAllowed(EventObject e);

  // Tu was vor dem Runterfahren
  void exiting();
}
```

Gliederung

Lebenszyklus

Ressourcen

Actions

Tasks

Sonstiges

Stand der JSR

TuneQ: Anlagen > Antrieb HW61, AVV_ZVV

TuneQ | Hilfe | Anlagen | Instandhaltung | Analyse | Wartung

Navigation

- **Strecken**
  - Strecke 1
    - EW 7 / 1445
      - EW 7
        - Antrieb HW61, AVV_ZVV
        - Heizung HWH
      - Steuerung HN-P 7 / 1445
      - Außenanlagen
    - EW 12 / 1449
  - Strecke 2
  - Strecke 3
  - Strecke 4
  - Strecke 5
- **Datensammler**
  - GUW Albertplatz
  - Btf. Gorbitz
  - Btf. Reich (wird angepasst)
  - Btf. Trachenberge
  - GUW Coswig
  - GUW Jahnstraße
  - GUW Meschwitzstrasse
  - GUW Postplatz
- **Anlagen**

Übersicht | Dokumente | Inspektionen | Meldungen | Historie

Antrieb EW 7 - Strecke 1
Friedrichstr. - Maxstr./Könneritzstr.

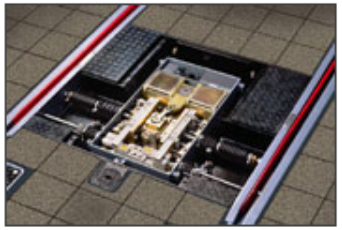
Gerätenr.:	320 513
EDV-Nr.:	109 244 318
Einbau:	12.03.2005
Inspektion:	21.02.2007

Status:	In Betrieb
Montage:	Gleismitte
Ausführung:	Flachbettweiche

Eigenschaften bearbeiten

Produkt

Baureihe:	HW 61	Stellkraft:	5000 N
Antriebsform:	Elektromagnetisch	Verschluß:	Ja
Variante:	AVV-ZVV	Feder:	Ja
Art:	Umstellweiche	Dämpfung:	Ja
Zungenaufschlag:	30 - 70 mm	Richtungsschalter:	nein
Höhe mit Kasten:	300 mm	Zungenprüfer:	Ja
Höhe ohne Kasten:	205 mm	Verschlossen:	Ja
Betriebsspannung:	600/750 V DC	Auffahrbar:	Ja



Produktdetails zeigen

Baugruppen:

Bezeichnung	EDV-Nummer	Gerätenr.	Einbau	Bemerkung
Erdkasten		320 514	12.03.2005	
Doppelzugmagnet	30032002		12.03.2005	
Antriebswelle	31051008		30.11.2006	

Neu... | Bearbeiten... | Löschen | Hoch | Runter

! 3 Ungelesene Nachrichten

ResourceMap

- Definiert mittels ResourceBundles
- Abgelegt in **`resources`**-Unterpaketen
- Setzt Eigenschaften, die variieren mit:
  - Locale, Plattform, Look&feel, Kunde

- "Besseres" ResourceBundle
  - Konvertiert Strings zu Werten
  - Expandiert Variablen
  - Bietet eine Hierachie (Vaterkette)

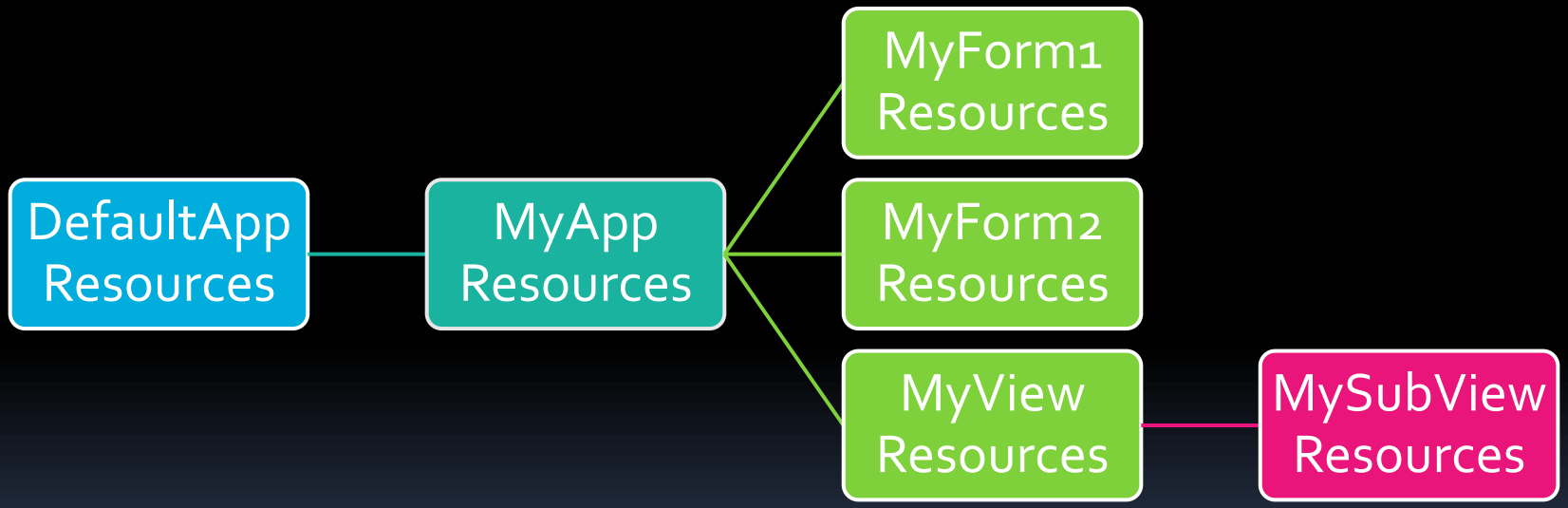
Properties-Datei

```
search.enabled=true
background.color=#A0A0A0

open.icon=open.png
open.icon=/myapp/resources/open.png

properties.title=%s Properties

editCustomer.title=${edit.title}
```

ResourceMap verwenden

```
public class MyForm1 {

    static final ResourceMap RESOURCES =
        Application.getInstance().
            getResourceMap(MyForm1.class);
    ...
    RESOURCES.getColor("background.color");
    RESOURCES.getIcon("open.icon");
    RESOURCES.getString("properties.title",
        objectName);
```

ResourceMap-Kette



Gliederung

Lebenszyklus

Ressourcen

Actions

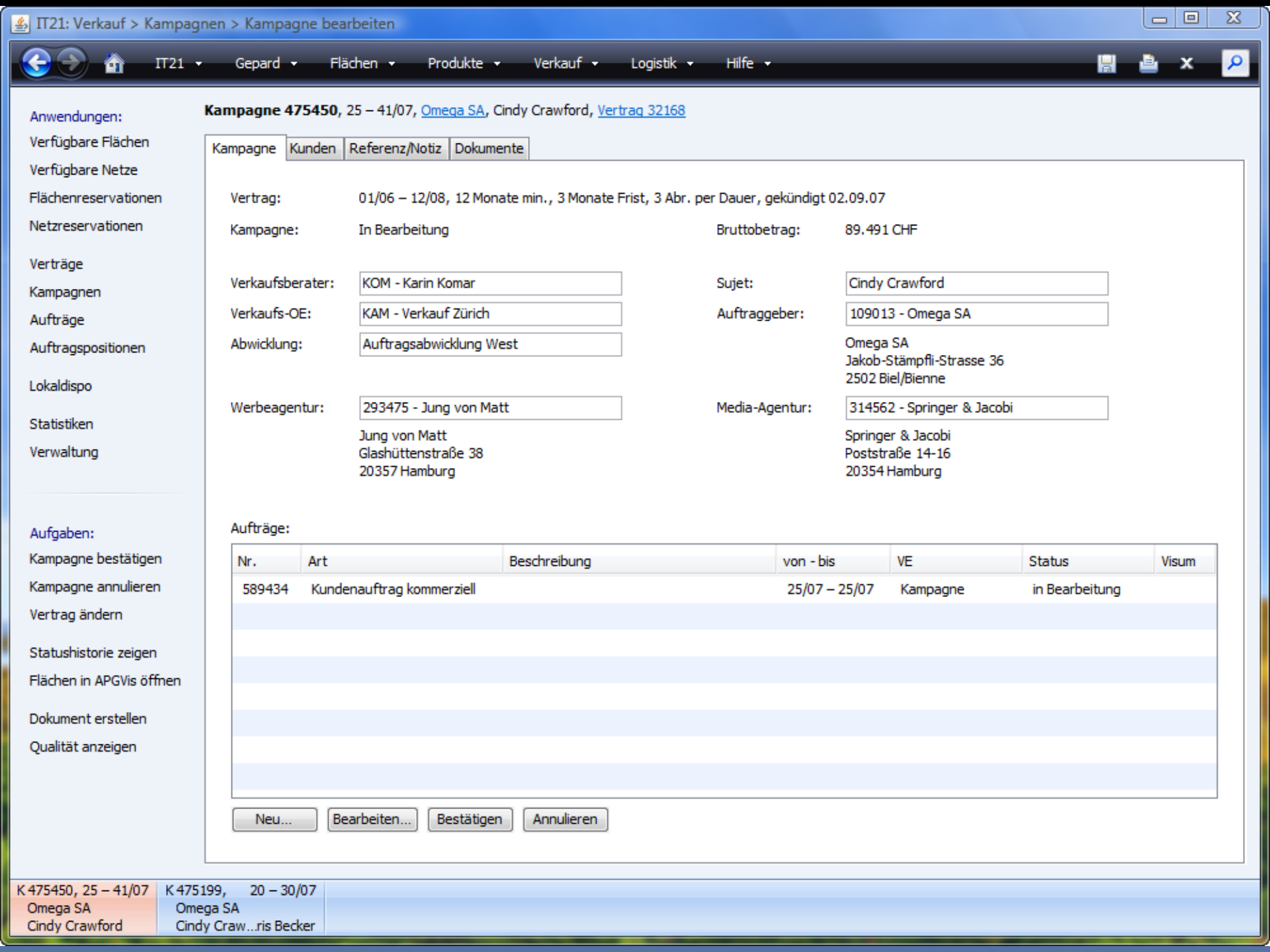
Tasks

Sonstiges

Stand der JSR

Swing-Actions

- ActionListener plus visuelle Eigenschaften:
  - Text, Tastaturkürzel, Zugriffstasten (Mnemonics), Tooltip, Hilfetext
  - enabled-Zustand


IT21: Verkauf > Kampagnen > Kampagne bearbeiten
IT21
Gepard
Flächen
Produkte
Verkauf
Logistik
Hilfe
Anwendungen:
Verfügbare Flächen
Verfügbare Netze
Flächenreservationen
Netzreservationen
Verträge
Kampagnen
Aufträge
Auftragspositionen
Lokaldispo
Statistiken
Verwaltung
Aufgaben:
Kampagne bestätigen
Kampagne annulieren
Vertrag ändern
Statushistorie zeigen
Flächen in APGVis öffnen
Dokument erstellen
Qualität anzeigen
Kampagne 475450, 25 – 41/07, Omega SA, Cindy Crawford, Vertrag 32168
Kampagne
Kunden
Referenz/Notiz
Dokumente
Vertrag:
01/06 – 12/08, 12 Monate min., 3 Monate Frist, 3 Abr. per Dauer, gekündigt 02.09.07
Kampagne:
In Bearbeitung
Bruttobetrag:
89.491 CHF
Verkaufsberater:
KOM - Karin Komar
Sujet:
Cindy Crawford
Verkaufs-OE:
KAM - Verkauf Zürich
Auftraggeber:
109013 - Omega SA
Abwicklung:
Auftragsabwicklung West
Omega SA
Jakob-Stämpfli-Strasse 36
2502 Biel/Bienne
Werbeagentur:
293475 - Jung von Matt
Jung von Matt
Glashüttenstraße 38
20357 Hamburg
Media-Agentur:
314562 - Springer & Jacobi
Springer & Jacobi
Poststraße 14-16
20354 Hamburg
Aufträge:
Nr.
Art
Beschreibung
von - bis
VE
Status
Visum
589434
Kundenauftrag kommerziell
25/07 – 25/07
Kampagne
in Bearbeitung
Neu...
Bearbeiten...
Bestätigen
Annulieren
K 475450, 25 – 41/07
Omega SA
Cindy Crawford
K 475199, 20 – 30/07
Omega SA
Cindy Craw...ris Becker

Alte Action-Definition

```
Action action = new AbstractAction("New…"){
  public void actionPerformed(ActionEvt e){
    // perform the new operation here
  }
};

aTextField.setAction(action);
aButton.setAction(action);
```

Alte Action-Definition

```
public class MyModel {

  private Action newAction;

  public Action getNewAction() {
    if (newAction == null) {
       newAction = new AbstractAction("New…") {
         public void actionPerformed(ActionEvent e){
           // perform the new operation here
         }
       };
       newAction.putValue(Action.MNEMONIC, …);
       newAction.putValue(Action.SHORTCUT, …);
    }
    return newAction;
  }
}
```

Alte Action-Definition

```
public class MyModel {

  private Action newAction;

  public Action getNewAction() {
    if (newAction == null) {
       newAction = new AbstractAction("New…") {
         public void actionPerformed(ActionEvent e){
           // perform the new operation here
         }
       };
       newAction.putValue(Action.MNEMONIC, …);
       newAction.putValue(Action.SHORTCUT, …);
    }
    return newAction;
  }
}
```

Swing-Actions

- Action-Objekterzeugung ineffizient
- Text, Mnemonic, Kürzel, etc. sollten internationalisiert werden und können von der Plattform abhängen
- Asynchrone Actions sind knifflig

- Viele innere Action-Klassen
- Dispatching Action (eine Klasse für viele Aktionen) hilft ein bisschen

Neue Action-Definition

```
public class MyModel {

   @Action
   public void newItem(ActionEvent e) {
       // Definiere hier die new-Operation
   }


   @Action(enabled=false)
   public void editItem() {  // Kein ActionEvent
       // Definiere hier die edit-Operation
   }
```

Action-Eigenschaften

```
newItem.Action.text=&New…
newItem.Action.accelerator=Ctrl N
newItem.Action.shortDescription=New item
newItem.Action.icon=images/new.png
```

Actions verwenden

```
public class MyView {

   private MyModel model;
   ...

   ActionMap map =
         Application.createActionMap(model);
   Action action = map.get("newItem");
   JButton button = new JButton(action);
```

Gliederung

Lebenszyklus

Ressourcen

Actions

Tasks

Sonstiges

Stand der JSR

Blockiere nicht den EDT!

- Verwende Hintergrund-Threads für:
  - blockierende Operationen wie Datei- oder Netzwerk-IO
  - Dinge, die länger als einen Augenblick brauchen
- Ansätze
  - SwingWorker
  - Spin
  - Foxtrot
- Was wir noch wollen:
  - Fortschritt und Nachrichten
  - einfache Definition
  - Abhängigkeiten zwischen Hintergrundaufgaben

Task und BlockingScope

- Task erbt die SwingWorker-Fähigkeiten
- Plus bequeme Fortschritts-Operationen
- Plus Nachrichten
- Konfiguriert mittels ResourceMap
- Sicheres Abschlussverhalten
- Blockiert: nichts, Action, Komponente, Fenster, Anwendung

Task-Definition

```
public class SaveTask extends Task {

    public SaveTask() {
        super(BlockingScope.APPLICATION,
              SaveTask.class);
    }

    protected Object doInBackground() {
        setMessage("A message");
        setProgress(30);
    }
    protected void succeeded() { … }
```

Tasks verwenden

```
public class MyModel {

   @Action
   public Task save(ActionEvent e) {
      if (!valid()) {
         // Show notifier
         return null;
      }
      return new SaveTask();
   }
```

TaskService und TaskMonitor

- TaskService: Wie werden Tasks ausgeführt?
  - seriell
  - über einen Thread-Pool
  - usw.
- TaskMonitor
  - Fasst mehrere Tasks zusammen
  - bound properties für einen *Vordergrund*-Task
  - hilft beim Bau von Statuszeilen

Gliederung

Lebenszyklus

Ressourcen

Actions

Tasks

Sonstiges

Stand der JSR

Resource Injection

Setze Eigenschaften aus Ressourcen per Name

```
resrcMap.injectComponents(aComponent)
myPanel.setBackground(Color c)
myLabel.setIcon(Icon i)
```

Setze markierte Felder aus Ressourcen per Name

```
resrcMap.injectFields(anObject)
@Resource Color foreground;
@Resource Icon icon;
```

Resource Injection II

- Pro:
  - sofort lokalisierbar
  - visuelle Eigenschaften sind leicht zu ändern
  - auch von Nicht-Programmierern
  - auch zur Laufzeit
- Kontra:
  - Verliert Übersetzungszeit-Sicherheit
  - Mehrere Lesequellen
  - Kaum Unterstützung durch IDEs

Anwendungszustand sichern

- Anwendungen sollten Zustand retten:
  - Fensterpositionen
  - Tabellenspaltenbreiten und -reihenfolge
  - Split Bar-Positionen
  - usw.
- Die JSR 296 soll das automatieren
- Siehe auch die UIState-Bibliothek

SessionStorage, LocalStorage

- SessionStorage
  - save(rootComponent, filename)
  - restore(rootComponent, filename)
- LocalStorage
  - abstrahiert Nutzer-bezogene Dateien
  - Geht auch für unsignierte Anwendungen
- Preferences?
  - sind bereits im Java-Kern
  - beschränkte Datengröße

Ressourcen-Varianten

```
prefs.Action.text=${prefs.Action.text.[$os]}
prefs.Action.text.default=Preferences
prefs.Action.text.win=Optio&ns
prefs.Action.accelerator=
                    ${prefs.Action.accelerator.[$os]}
prefs.Action.accelerator.default=${null}
prefs.Action.accelerator.mac=meta COMMA
```

(eigener Vorschlag)

Gliederung

- Lebenszyklus
- Ressourcen
- Actions
- Tasks
- Sonstiges
- Stand der JSR

Stand der JSR

- Inaktiv
- Spec lead und EG haben in mehr als 18 Monaten keinen Milestone Draft geliefert.

- Im übrigen: Zombie

Kurze Geschichte: Vor 2006

- Bedarf an Desktop Blueprints
- Mangel an Desktopmustern
- Kaum Sun-Entwickler auf Anwendungsebene
- Hintergrund-Aufgaben:
  - Alter SwingWorker
  - Spin
  - Foxtrot (synchronous)

2006

- Projektstart
- Projektleiter und Spec lead: Hans Muller
- Sun reicht die JSR ein
- EG gebildet
- EG diskutiert den Umfang

- November: Durchbruch für effizientes Swing

2007

- Erster öffentlicher pre-draft-Prototyp
- Unfug beseitigt
- Feb – Aug: Versionen 0.1 – 0.4
- September: Version 1.0
- November: tot

2008

- Jan – Mai: tot
- Mai: Hans Muller verlässt Sun
  - Siehe auch "Hans's swan song"
- Juli: Neuer Spec lead Alexander Potochkin
- Aug: Beta-Versionen
- Sep: tot

2009

- März: Spec lead meldet sich kurz
- Einige Änderungen ohne EG-Mitwirkung

JavaFX

Diskussionen

- EG: Schweigen im Walde
- appframework: Hühnerhaufen

- API muss überarbeitet werden, wird aber kaum diskutiert

appframework-Implementierung

- Showstopper erfordern API-Änderungen
- Etliche Problem nicht einmal erkannt
- API kann sich stark ändern
- Nicht produktionsreif

Andere Implementierungen

- Kommerzieller öffentlicher JGoodies-Code
  - Lebenszyklus, Ressourcen, Actions
  - sicherer SwingWorker
  - Preferences
  - Einfache local storage
- Kommerzieller vertraulicher JGoodies-Code
  - Bietet Tasks, Blocking
  - Keine Resource Injection
- Dein Rahmenwerk angepasst an JSR 296

Fazit

- JSR beinhaltet, was Entwickler brauchen
- Einige Features sind ausreichend stabil:
  - Lebenszyklus, Ressourcen, Actions
- Implementierung[en] noch fehlerhaft
- Schwacher Entwicklungsprozess

- Trotzdem: Ein Bringer, ein Erfolgsfaktor.
- Auch Dir kann diese JSR nützen

Weitere Informationen

- Google "JSR 296"
- appframework.dev.java.net
- appframework-User-Mailing-Liste

- www.jgoodies.com/articles

Swing-Überlebenshilfe

Desktop-Muster

Datenbinding

JSR 296

Erste Hilfe für Swing

Layout-Management

Meta-Design

Zur Anwendungsgliederung

- Scott Delap: *Desktop Java Live*
  (etwas veraltet)
- JGoodies: *Desktop-Muster und Datenbindung*

FRAGEN UND ANTWORTEN

JGoodies Karsten Lentzsch

JSR 296 – SWING APP FRAMEWORK