12.–15.09.2010
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Rückkehr nach Oz

Eigene Komponenten mit Wicket erstellen

Isabella Kneissl

MATHEMA Software GmbH

Agenda

- Wicket – eine kurze Einführung
  - Was ist Wicket?
  - Grundlagen
- Wicket und Komponenten
  - Was ist eine Komponente?
  - Komponenten der Distribution
  - Erstellung eigener Komponenten
    - Motivation
    - Vorgehen bei der Komponentenerstellung
    - Testen der Komponenten
- Fazit
- Quellen und weiterführende Literatur

Agenda

- Wicket – eine kurze Einführung
  - Was ist Wicket?
  - Grundlagen
- Wicket und Komponenten
  - Was ist eine Komponente?
  - Komponenten der Distribution
  - Erstellung eigener Komponenten
    - Motivation
    - Vorgehen bei der Komponentenerstellung
    - Testen der Komponenten
- Fazit
- Quellen und weiterführende Literatur

Was ist Wicket?

- Java Web Application Framework
- Beheimatet bei der Apache Software Foundation
- Aktuell Version 1.4.10, von 1.5 erstes Milestone-Release; Beginn mit Wicket 1.0 im August 2005
- Rege Community (Blogs, Wiki)
- Homepage: http://wicket.apache.org/

Grundlagen

- Die Anwendung
  - Eine Klasse als Repräsentation der Anwendung
  - Die Klasse erbt von

    `org.apache.wicket.protocol.http.WebApplication`
    - Sie enthält Konfiguration, Namen der Anwendung, Initialisierung mit Überschreiben der Methode `init()`
  - Konfiguration in Datei web.xml
- Eine Seite
  - Ein Dateipaar Java-Klasse/HTML-Seite
  - Java-Klasse abgeleitet von

    `org.apache.wicket.markup.html.WebPage`
  - Mapping zwischen HTML-Elementen und Elementen im Java-Code mittels `wicket:id`

Grundlagen

- Eine Seite: Beispiel

HTML:

```
<body>
 <form wicket:id="form">
  <span wicket:id="nameLbl">
    Default Label</span>
  <input wicket:id="nameIn"
    type="text"/>
  <input wicket:id="submit"
    type="submit" value="OK"/>
 </form>
</body>
```

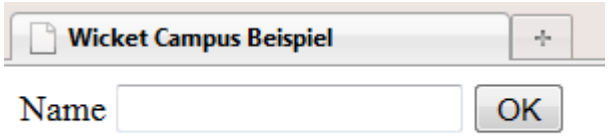
JAVA:

```
public class WelcomePage extends WebPage{
   private String name;
   public WelcomePage() {
     Form form = new Form("form");
     Label label = new Label
       ("nameLbl", "Name");
     TextField name = new TextField
       ("nameIn",
       new PropertyModel(this, "name"));
     Button btn = new Button("submit");
     add(form);
     form.add(label);
     form.add(name);
     form.add(btn);
   }
...
}
```

Grundlagen

- Eine Seite:
  Web-Design:

Im Browser:




```
1  <html
2      xmlns:wicket="http://wicket.apache.org/dtds.data/wicket-
3  <head>
4  <title>Wicket Campus Beispiel</title>
5  </head>
6  <body>
7  <form wicket:id="form">
8      <span wicket:id="nameLbl">Default Label</span>
9      <input wicket:id="nameIn" type="text" />
10     <input wicket:id="submit" type="submit" value="OK" />
11 </form>
12 </body>
13 </html>
```

Agenda

- Wicket – eine kurze Einführung
  - Was ist Wicket?
  - Grundlagen
- Wicket und Komponenten
  - Was ist eine Komponente?
  - Komponenten der Distribution
  - Erstellung eigener Komponenten
    - Motivation
    - Vorgehen bei der Komponentenerstellung
    - Testen der Komponenten
- Fazit
- Quellen und weiterführende Literatur

Was ist eine Komponente?

- Alle Bestandteile der HTML-Seite, die potentiell ihren Wicket-Gegenpart haben können, sind Komponenten
- Basisklasse `org.apache.wicket.Component`
- WebPage ist die Wurzel-Komponente einer Seite
- Container-Komponenten können auch Sub-Komponenten haben
- Zugriff auf die zugehörige Seite einer Komponente über getPage()
- Wicket Komponenten haben oft ein eigenes HTML-Markup

Was ist eine Komponente?

- Eine Komponente hat oft ein **Modell**, um ihre (fachlichen) Daten zu halten
- Die Komponente zeigt die Daten an (`getObject()`) und sie können auch durch sie verändert werden
- Das Modell dient als Abstraktion zwischen der Komponente und den Daten
- Eine Komponente hat kein Modell, z.B. wenn sie...
  - Keine eigenen Daten repräsentiert
  - Das Modell der Elternkomponente benutzt
    (→CompoundPropertyModel)

```
setDefaultModel(new CompoundPropertyModel<Adresse>(adresse));
TextField<String> name = new TextField<String>("name");
add(name);
```

Was ist eine Komponente?

- Daten von Komponenten werden durch **Konverter** für das Senden mittels Request/Response und die Darstellung auf der Webseite vorbereitet
- Konverter werden erstellt durch Implementierung von org.apche.wicket.util.convert.IConverter convertToObject und convertToString
- Konverter können global registriert werden oder direkt an eine Komponente geknüpft werden
  - Global: Überschreiben von Application.newConverterLocator()
  - An der Komponente: Component.getConverter(Class type)

Was ist eine Komponente?

- Daten einer Komponente werden durch **Validatoren** geprüft
- Pflichtfeld-Validierung mittels .setRequired
- Erstellen von eigenen Validatoren durch Implementierung von IValidator

```
RequiredTextField<String> nicknameInput =
    new RequiredTextField<String>("nickname",
    new PropertyModel<String>(this, "nickname"));
nicknameInput.setRequired(true);
nicknameInput.add(StringValidator.lengthBetween(3, 30));
nicknameInput.add(new PatternValidator("[a-zA-Z0-9 ]*"));
```

Was ist eine Komponente?

- **Validierung** von Einzelfeldern und mehrerer Felder kombiniert möglich mit IFormValidator → getDependentFormComponents
- Formvalidierungen werden gerufen, sobald keines der jeweils beteiligten Felder einen Fehler in der Einzelfeld-Validierung hat
- Anzeige der Validierungsmeldungen in FeedbackPanel

Komponenten der Distribution

- Ausgabe-Komponenten: Label, <wicket:message>, Image
- Formular-Komponenten: Form, TextField, PasswordTextField, ListChoice, DropDownChoice, RadioChoice, CheckBoxMultipleChoice, ListMultipleChoice
- Komponenten für Aktionen: Link, Button
- Ajax-Komponenten: AjaxSubmitLink, AjaxSubmitButton, AjaxCheckBox
- Listen: ListView, DataView, RepeatingView
- Einige Converter und Validatoren
- ...

Erstellung eigener Komponenten

- Motivation: Templating
  - Durch Gruppieren von Layoutbereichen z.B. Navigation
    - HTML-seitig durch Eingliederung des Inhalts in `<wicket:panel>`
    - Codeseitig Ableiten von Panel und „Bestücken“ mit den nötigen Komponenten im Konstruktor
    - Flexibleres Hinzufügen als Variante 2
  - Durch Vererbungshierarchie der Komponenten
    - HTML-seitiger Platzhalter mit `<wicket:child>`
    - Vererbungshierarchie, Grundbestandteile in der Basisklasse, Spezialisierungen in Ableitungen

Erstellung eigener Komponenten

- Motivation: Fachlich
  - Wiederverwendung von fachlicher Anwendungslogik innerhalb einer Anwendung(-sgruppe)
  - Verhindert Code-Duplizierung
  - Einmalige Entwicklung und Test
  - Beispiel: Adresseingabe

Erstellung eigener Komponenten

- Motivation: Technisch
  - Bestimmte Interaktionsmöglichkeiten mit dem Benutzer
  - Wiederverwendung in verschiedenen Anwendungen
  - Werden analog zu den Komponenten der Distribution nach Fertigstellung im Code verwendet
  - Beispiel: Pflichteingabefeld, abhängige Auswahlliste

Erstellung eigener Komponenten

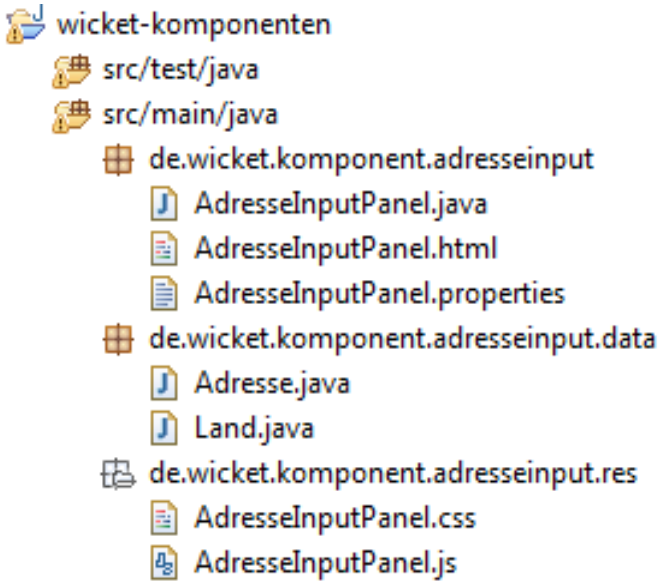
- Benötigte Bestandteile:
  - Komponenten-Klasse
  - Eigenes HTML-Markup *
  - Komponenten-Ressourcen (CSS, Properties...) *
  - Unittest
  - Oberflächentest

* optional

Erstellung eigener Komponenten

- Vorgehen:
  - Oft Refakturierung von existierendem Code für Wiederverwendung

    HTML-seitig
  - Erstellung des Designs der HTML-Seite
  - Auslagern von JavaScript-Code in eine JS-Datei
  - Verwendung von CSS Klassen und Auslagern der Styles in eine CSS-Datei



Erstellung eigener Komponenten

- Vorgehen:

Java-seitig

- Implementierung der Komponentenklasse
- Anbindung von Konvertern und Validatoren wenn nötig, innerhalb oder aus extra Implementierungen
- Registrieren der externen Ressourcen an der Komponente

```
add(CSSPackageResource.getHeaderContribution(
        AdresseInputPanel.class, "res/AdresseInputPanel.css"));
add(JavascriptPackageResource.getHeaderContribution(
        AdresseInputPanel.class, "res/AdresseInputPanel.js"));
```

Erstellung eigener Komponenten

- Paketieren der Komponente als Jar
  - Mit Eclipse Paketierung
  - Mit Maven: mvn package
- Einbinden des Jars in die Anwendung
  - Im Eclipse Aufnehmen als Library in den Java Build-Path
  - Bei Verwendung von Maven Aufname in pom.xml

```
<dependency>
    <groupId>wicket</groupId>
    <artifactId>wicket-komponenten</artifactId>
    <version>1.0-SNAPSHOT</version>
</dependency>
```

Erstellung eigener Komponenten

- Vorgehen:

  Test
  - Implementieren der Unittests ohne ServletContainer für die Komponente
    - Testen mit JUnit
    - Mockimplementierung in `org.apache.wicket.util.tester.WicketTester` integriert
    - Testen von Formularkomponenten mit `org.apache.wicket.util.tester.FormTester`
  - Erstellen eines Testprojektes für Oberflächentests

Agenda

- Wicket – eine kurze Einführung
  - Was ist Wicket?
  - Grundlagen
- Wicket und Komponenten
  - Was ist eine Komponente?
  - Komponenten der Distribution
  - Erstellung eigener Komponenten
    - Motivation
    - Vorgehen bei der Komponentenerstellung
    - Testen der Komponenten
- Fazit
- Quellen und weiterführende Literatur

Fazit

- Angenehmes Designen der Komponenten durch saubere Trennung des Layouts von der UI-Logik
- Gute Testbarkeit durch Unittests und sparsame Testprojekte
- Kein aufwändiges Mocken der Webumgebung nötig

- Komponentenerstellung in Wicket macht Spass!

Agenda

- Wicket – eine kurze Einführung
  - Was ist Wicket?
  - Grundlagen
- Wicket und Komponenten
  - Was ist eine Komponente?
  - Komponenten der Distribution
  - Erstellung eigener Komponenten
    - Motivation
    - Vorgehen bei der Komponentenerstellung
    - Testen der Komponenten
- Fazit
- Quellen und weiterführende Literatur

Quellen und weiterführende Literatur

- http://wicket.apache.org/
- https://cwiki.apache.org/WICKET/
- Wicket – Komponentenbasierte Webanwendungen in Java; R. Förther, C.-E. Menzel, O. Siefart; dpunkt.verlag; 1. Auflage 2010
- http://www.theserverside.com/news/1363828/Introducing-Apache-Wicket
-

Vielen Dank!

Isabella Kneissl

MATHEMA Software GmbH