Getrenntes Spiel

Modulare Formulare mit Struts 2

Claus Kolb

Auch in Zeiten des Web 2.0 besteht in modernen Internet-Auftritten die Notwendigkeit, Daten des Benutzers mithilfe von „traditionellen" Webformularen zu erfassen und weiterzuverarbeiten. Insbesondere bei einer großen Anzahl von Formularen muss bereits in der Fachkonzeption auf eine modulare Analyse geachtet werden, die zu einheitlichen und wiederverwendbaren Formularen führt. Dieser Artikel beschreibt, wie diese Anforderungen mit Apache Struts 2.1 in einem Kundenprojekt umgesetzt wurden.

Kategorisierung von Formularen

Ein Webformular ist ein Formular auf einer Webseite zur Erfassung von Daten im Internet [Wik]. Webformulare erfüllen sehr unterschiedliche Funktionen innerhalb von Webanwendungen, z. B. Datenerfassung, Suche und Filterung. Folgende technische Kategorien von Webformularen lassen sich anhand des Funktionsumfangs definieren:

Typ 1: Einseitige Formulare mit einer festen Anzahl von Formularfeldern mit syntaktischen und semantischen Validierungen, die sich auf ein oder mehrere Formularfelder beziehen. Es existiert keine komplexe Geschäftslogik, die nach dem Absenden durch eine serverseitige Aktion ausgeführt wird. Vielmehr dient das Formular der Erfassung von Daten eines Benutzers. Ein Kontaktformular, dessen Inhalte serverseitig per E-Mail weitergeleitet werden, ist ein typisches Beispiel für diese Kategorie.

Typ 2: Dieser Typ umfasst zusätzlich eine Reihe von Feldern, die dynamisch in Abhängigkeit von einem oder mehreren Feldinhalten eingeblendet werden. Ein Anfrageformular, das in Abhängigkeit vom ausgewählten Produkt eine zusätzliche Menge von produktspezifischen Feldern einblendet, gehört zu dieser Kategorie.

Typ 3: Dieser Typ beschränkt sich nicht auf eine Formularseite. Hinsichtlich der Komplexität und des Funktionsumfangs sind keine Grenzen gesetzt. Ein Kfz-Versicherungsrechner, der eine Vielzahl von unterschiedlichen Formularseiten mit komplexen Beziehungen untereinander darstellt, fällt in diese Kategorie.

Dieser Artikel befasst sich mit den Formularkategorien der Typen 1 und 2 und zeigt auf, wie die Verwaltung dieser Formulare mit einem Content-Management-System (CMS) erfolgen kann.

Konzeption

Der Bereich der Webformulare wird in der Fachkonzeption in Bezug auf den Arbeitsumfang und die Komplexität meist unterschätzt. Beginnend mit ein „paar" Formularen wird bei näherer Analyse oft klar, dass die Formulare sehr unterschiedliche Anforderungen hinsichtlich der Zusammensetzung der Felder, der Abhängigkeiten zwischen den Feldern sowie der Validierung und der Weiterverarbeitung aufweisen. Bereits bei der Anforderungsanalyse muss angestrebt werden, wiederkehrende Formularblöcke zu identifizieren. Dies führt als positiver Nebeneffekt auch oft zu einer Vereinheitlichung der Formulare hinsichtlich der erfassten Daten, der Feldbezeichnungen und der Weiterverarbeitung und schafft klare Anforderungen für die Implementierung.

Als Basis für die Realisierung müssen für jedes Formular und jeden Formularblock eine Attributliste, der Typ des zu verwendenden Formularfelds sowie die Regeln für die syntaktische und semantische Validierung vorliegen. Oft ergeben sich bei diesem Vorgehen auch Strukturanalogien zum Persistenzmodell der Anwendung.

Getrenntes Spiel

Bereits einfache Webformulare mit syntaktischer und semantischer Validierung und unterschiedlicher serverseitiger Weiterverarbeitung erreichen hinsichtlich der Implementierung eine zu hohe Komplexität, um ausschließlich durch (CMS-)Redakteure erstellt zu werden.

Auch existiert hinsichtlich der Neuerstellung und Pflege von Formularen auf Kundenseite oft die Vorstellung, dass sie komplett von Redakteuren erstellt werden. Bei einer näheren Analyse dieser Anforderung wird klar, dass der Kunde einen generischen, für Nicht-Techniker geeigneten Formulareditor will. Leider sprengen die Kosten für die Entwicklung meist das veranschlagte Budget. Ferner vermischen sich bei diesen generischen Formulareditoren zwangsläufig redaktionelle und technische Aspekte. Beispielsweise verbleibt auch bei grafischer Modellierung die Erstellung von komplexen Validierungsregeln eine Entwicklertätigkeit.

Eine klare Rollentrennung hilft, sowohl das Budget einzuhalten als auch jeder Rolle den passenden Aufgabenbereich bei der Erstellung zuzuweisen. Ferner erlaubt die Trennung den Einsatz von effizienten Werkzeugen für jede Rolle.

Die Rolle (CMS-)Entwickler erstellt die Formulare, implementiert die syntaktische und semantische Validierung und die Weiterverarbeitung der Daten auf dem Server. Ferner legt der Entwickler die Formulare im verwendeten CMS an und stellt sicher, dass die Formulare korrekt konfiguriert werden.

Der Redakteur ist für die inhaltliche Gestaltung der Formularseite zuständig. Dazu pflegt er die Bezeichnungen der Formularfelder, die Fehlermeldung und gegebenenfalls die E-Mail-Adressen der Empfänger. Die Anzahl der Formularfelder, die Validierung und die Weiterverarbeitung sowie das Layout der Seite können durch den Redakteur nicht beeinflusst werden.

Auch systemseitig benutzen die beiden Rollen getrennte Systeme. Während die Arbeit des Redakteurs ausschließlich mit einem CMS durchgeführt wird, arbeitet der Entwickler auch mit seiner Entwicklungsumgebung, z. B. Eclipse.

Realisierung mit Apache Struts 2.1

Die Anforderungen hinsichtlich einer modularen Implementierung der Webformulare lassen sich sehr gut mithilfe des Frameworks Apache Struts 2.1 umsetzen. Dazu wird in den folgenden Abschnitten für jede Schicht des Model-View-Controller-Architekturmusters anhand von Code-Beispielen beschrieben, wie die Implementierung mithilfe des Struts-Frameworks durchzuführen ist. Grundlegende Kenntnisse des Struts-Frameworks werden im Folgenden vorausgesetzt.

Das Beispielformular (s. Abb. 1) besteht aus zwei Blöcken, die auf der einen Seite persönliche Daten und auf der andere Seite eine Bankverbindung erfassen. Zusätzlich wird in einem separaten Block eine Einwilligung abgefragt, die Daten zur weiteren Verarbeitung zu speichern.

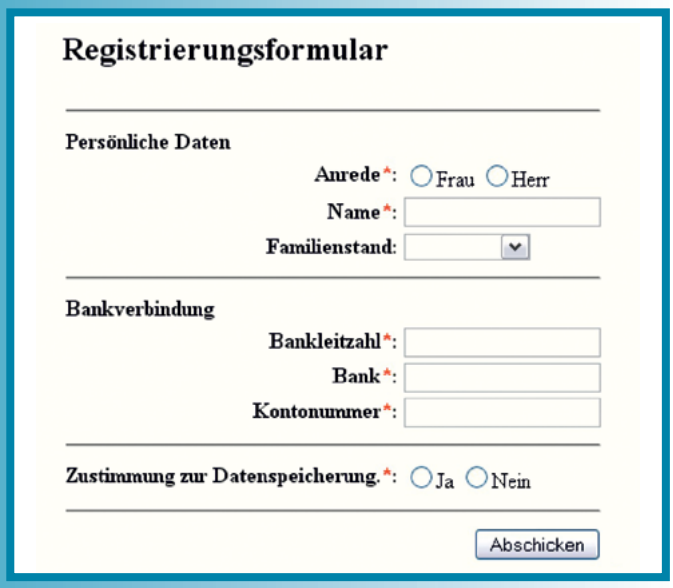

Abb. 1: Das Beispielformular

Controller

Der Controller nimmt die Formular-Requests vom Browser entgegen, ruft die Validierung der Eingabedaten auf, überträgt die Daten ins Model und zeigt eine Folgeseite in Abhängigkeit vom Ergebnis der Validierung auf. Die Ablaufsteuerung wird in Struts deklarativ in der struts.xml-Datei hinterlegt. Für unser Beispiel ist in Listing 1 die Action-Konfiguration zu sehen.

```
<action name="RegistrierungAction"
        class="org.ck.struts2.registrierung.RegistrierungAction">
  <result name="success">registrierung_ok.jsp</result>
  <result name="input">registrierung.jsp</result>
</action>
```

Listing 1: Action-Konfiguration

Die Struts-Validator-Komponente erlaubt es, syntaktische und semantische Regeln für die Überprüfung von Eingabedaten deklarativ zu definieren. Wenn der Struts-Controller aufgerufen wird, ruft er unter bestimmten Bedingungen automatisch die Validierung der Eingabedaten auf. Dazu müssen die Dateien mit Validierungsregeln nach bestimmten Namenskonventionen benannt werden. Falls die Validierung ohne Fehler durchgeführt wird, wird die in Listing 1 konfigurierte „success“-Seite aufgerufen. Andernfalls wird der Benutzer zur Eingabeseite (Result „input“) zurückgeleitet.

Nach der erfolgreichen Validierung ruft die Action-Klasse gegebenenfalls die Weiterverarbeitung der Daten auf.

Model

Das Model im MVC-Architekturmuster speichert die Formulareingaben des Benutzers und enthält die Geschäftslogik, die in unserem Beispiel aus der Validierung der Benutzereingaben und der serverseitigen Weiterverarbeitung der Daten besteht. Die Daten können bspw. per E-Mail versandt, in einer Datenbank gespeichert oder in XML umgewandelt werden. Für den Versand von E-Mails bietet sich die Apache Velocity Engine [Vel] an, um die in die Formulare eingegebenen Werte in ein E-Mail-Template zu übertragen.

Für jeden wiederverwendbaren Formularblock muss eine eigene Java-Bean implementiert werden, die für jedes Formularfeld eine Instanzvariable besitzt. Gegebenenfalls sind die Java-Beans aus der Persistenzschicht auch für die Formulare geeignet. So definiert die Klasse **`Bankverbindung`** bspw. die Attribute Bank, Bankleitzahl und Kontonummer.

```
...
public class RegistrierungAction extends DefaultActionSupport {
  private Bankverbindung bankverbindung;
  private PersoenlicheDaten persoenlicheDaten;
  private Boolean einwilligung;

  /* get/set-Methoden */
  ...
```

Listing 2: Controller-Action-Klasse

Für jedes Webformular muss eine eigene Action-Klasse erstellt werden, die die spezifische Zusammenstellung von Formularblöcken und weiteren Eingabefeldern auf Klassenebene leistet. Im Beispiel definiert die in Listing 2 dargestellte Action-Klasse **`RegistrierungAction`** die oben genannten Java-Beans als Instanzvariablen **`bankverbindung`** und **`persoenlicheDaten`**. Des Weiteren kann sie für das jeweilige Formular spezifische Eingabefelder als Instanzvariablen definieren (Attribut **`einwilligung`**).

Um auch die Validierungsregeln wiederverwenden zu können, werden für die beiden Beans **`Bankverbindung`** und **`PersoenlicheDaten`** jeweils separate XML-Dateien erstellt. Sie beinhalten nur die Validierungsregeln für einen Formularblock. In unserem Beispiel heißen diese Dateien **`Bankverbindung-validation.xml`** (s. Listing 3) und **`PersoenlicheDaten-validation.xml`**. Indem die Validierungsdateien nach dem Schema **`<Klassenname>-validation.xml`** benannt und im selben Verzeichnis wie die Klassen abgelegt werden, findet das Struts-Framework sie automatisch.

```
...
<validators>
  <field name="bankverbindung.kontoinhaber">
    <field-validator type="requiredstring">
      <message key="errors.required" />
    </field-validator>
  </field>
</validators>
...
```

Listing 3: Bankverbindung-validation.xml

Struts liefert eine Reihe von Validatoren bereits fertig aus, z. B. Validatoren für E-Mail-Adressen und Integer, aber auch komplexe Validatoren mit regulären oder OGNL-Ausdrücken [Ognl]. Für jedes Eingabefeld `<field>` können ein oder mehrere `<field-validator>`-Validatoren definiert (s. Listing 3) werden.

Neben der Action-Klasse muss für jedes Webformular zusätzlich eine Datei mit Validierungsregeln für das gesamte Formular erstellt werden. Für die Action-Klasse `RegistrierungAction` wird die Validierung durch die Datei `RegistrierungAction-validation.xml` (s. Listing 4) realisiert. Sie integriert die oben genannten Validierungsdateien mithilfe des von Struts mitgelieferten Visitor-Validators. Dieser delegiert die Validierung der Attribute an die oben genannten Validierungsregeln der Attribute. Zusätzlich können in der Datei eigene Prüfungen ergänzt werden, um Besonderheiten der Formularzusammenstellung zu berücksichtigen (z. B. Abhängigkeiten zwischen den Blöcken).

```
...
<validators>
  <field name="bankverbindung">
    <field-validator type="visitor">
      <param name="appendPrefix">false</param>
      <message />
    </field-validator>
  </field>

  <field name="persoenlicheDaten">
    <field-validator type="visitor">
      <param name="appendPrefix">false</param>
      <message />
    </field-validator>
  </field>

  <field name="einwilligung">
    <field-validator type="fieldexpression">
      <param name="expression"><![CDATA[einwilligung == "ja"]]>
      </param>
      <message key="einwilligung.required" />
    </field-validator>
  </field>
</validators>
...
```

Listing 4: RegistrierungAction-validation.xml

Auf diese Weise können die Webformulare inkl. der Validierung der Eingabedaten modular implementiert werden. Die genannten Artefakte müssen für jeden standardisierten Formularblock nur einmal implementiert und können mehrfach wiederverwendet werden. Neue Formulare können schnell erstellt werden, indem vorhandene Komponenten zusammengestellt werden.

View

Der View des MVC-Architekturmusters präsentiert die Model-Daten im Browser des Benutzers. Als View-Technologie können JSP-, Freemarker- oder Velocity-Templates sowie XSLT-Transformationen verwendet werden. In unserem Beispiel implementieren Struts-JSP-Tags den View. Struts 2 bringt eine Vielzahl vorgefertigter Benutzungsoberflächen-Tags mit, aus denen die Webformulare zusammengebaut werden können. Alle Eingabekomponenten der Webformulare, z. B. Textfelder und Checkboxen, stehen zur Verfügung. Listing 5 zeigt exemplarisch das Seitenfragment des Formularblocks für die Bankverbindung.

Mithilfe sogenannter Themes kann das Layout an die Bedürfnisse eines Internet-Auftrittes angepasst werden. Die Themes bestehen aus FreeMarker-Templates, wobei für jeden Tag mindestens ein Template existiert. Struts liefert die Themes simple, xhtml und css_xhtml bereits mit aus. Diese Themes müssen sowohl optisch als auch funktional an die eigenen Erfordernisse angepasst werden. Dazu wird das template-Verzeichnis aus der Bibliothek struts2-core.jar extrahiert und in den Classpath der Webanwendung gelegt.

```
...
<!--  Block: Bankverbindung -->
...
<s:textfield key="bankverbindung.blz" theme="xhtml"
             template="text.ftl" required="true" />
<s:textfield key="bankverbindung.bank" theme="xhtml"
             template="text.ftl"
   required="true" title="1900" />
<s:textfield key="bankverbindung.kontonummer" theme="xhtml"
             template="text.ftl" required="true" title="1900" />
...
```

Listing 5: Ausschnitt des Views

Das Attribut „key“ der oben genannten Struts-Tags verweist auf einen Schlüssel einer Propertys-Datei, die die Beschriftung des Eingabefelds beinhaltet. Darüber hinaus werden in dieser Datei die Fehlermeldungen hinterlegt, die bei fehlgeschlagener Validierung der Eingaben dem Benutzer angezeigt werden.

Abschließend müssen die Formularblöcke noch zu einem Gesamtformular zusammengesetzt werden. Verschiedene Templating-Frameworks bieten die flexible Kombination der genannten wiederverwendbaren Formularblöcke. Das Struts-Tiles-Plug-in integriert das Apache Tiles Framework [Til]. Mithilfe dieses Frameworks können Seitenfragmente flexibel zu kompletten Seiten zusammengestellt werden. Jeder Formularblock würde bei der Implementierung mit Tiles in einem eigenen Seitenfragment hinterlegt werden.

CMS-Integration

Im konkreten Projekt, in dem das beschriebene Verfahren angewendet wird, ist das CMS FirstSpirit der Firma e-Spirit in der Version 4.2 im Einsatz. Der beschriebene Ansatz kann prinzipiell in jedes CMS-System integriert werden, das über Seiten- und Absatzvorlagen verfügt.

Prinzipiell definiert eine Vorlage eine Reihe von fest vorgegebenen Content-Elementen, die vom Redakteur nicht editiert werden können. Darüber hinaus können durch den CMS-Entwickler in einer Vorlage eine Reihe von Eingabekomponenten definiert werden. Diese erlauben dem Redakteur, den für den Absatz oder die Seite spezifischen Content zu pflegen. So kann bspw. eine Seitenüberschrift oder eine Verweisliste durch den Redakteur pro Seite bzw. Absatz Formular-basiert gepflegt werden. Das CMS setzt bei der Erzeugung der Seite den Content an die entsprechenden Stellen im Absatz bzw. auf der Seite ein. Diese Content-Pflege mithilfe von Formularen ist nicht mit den Webformularen zu verwechseln, die in diesem Artikel thematisiert sind.

Eine Seitenvorlage definiert das Grundgerüst einer Seite und enthält z. B. die Navigation, den Header, den Footer und mindestens einen Content-Bereich. Die Absatzvorlagen werden verwendet, um einen oder mehrere Absätze mit redaktionell gepflegtem Inhalt in den Content-Bereich einzufügen.

Die folgenden CMS-Artefakte werden durch die Rolle (CMS-)Entwickler implementiert. Eine spezielle Seitenvorlage „Formular“ enthält neben den grundlegenden oben genannten Seitenelementen das Grundgerüst des Webformulars (z. B. `form`-Tag) und einen Content-Bereich „content“. Eine Eingabekomponente der Seitenvorlage erlaubt die Angabe der Struts-Action, die in das `form`-Tag der Seite durch das CMS eingefügt wird. Eine weitere Eingabekomponente der Seitenvorlage er-

laubt die Angabe der E-Mail-Adresse des Empfängers. Für jeden Formularblock wird eine eigene Absatzvorlage entwickelt, die den Content einschließlich der Struts-Tags enthält.

Um eine neues Webformular anzulegen, wird eine Content-Seite auf Basis der Seitenvorlage „Formular“ im CMS angelegt. In den Content-Bereich „content“ können die einzelnen Absätze eingefügt werden, die die Formularblöcke enthalten.

Die Pflege unter anderem der Beschriftungen und Fehlermeldungen wird durch die Rolle Redakteur durchgeführt. Dazu wird entweder die oben genannte Struts-Propertys-Datei im CMS hinterlegt und direkt durch den Redakteur pflegbar gemacht. Alternativ kann für jede Fehlermeldung oder Beschriftung ein eigenes Eingabefeld in den Absatzvorlagen der Formularblöcke vorgesehen werden, die in die Propertys-Datei des CMS eingefügt wird.

Fazit

Das Struts-Framework bietet in Verbindung mit einer Templating-Engine (CMS oder Tiles) eine einfache Möglichkeit, wiederverwendbare Formularblöcke zu Formularen zusammenzusetzen. Durch eine strikte Rollentrennung bei der Erstellung der Formulare wird vermieden, dass programmiertechnische und redaktionelle Aspekte vermischt werden.

Links

[eSp] Homepage der Firma e-Spirit, `http://www.e-spirit.de`
[Ognl] Object-Graph Navigation Language, `http://www.opensymphony.com/ognl/`
[Til] Apache Tiles Project, `http://tiles.apache.org/index.html`
[Vel] Apache Velocity Project, `http://velocity.apache.org/engine/index.html`
[Wik] Wikipedia, `http://de.wikipedia.org/wiki/Webformular`

Claus Kolb ist Senior Consultant bei der Unternehmensberatung Acando in Hamburg.
Seine technologischen Schwerpunkte liegen in den Bereichen Softwareentwicklung, Portale und Content-Management-Systeme. Claus Kolb arbeitet seit 2000 als Berater in der IT-Branche und ist aktuell als Projektmanager sowie als Requirements Engineer tätig.
E-Mail: claus.kolb@acando.de