Das Open Source Framework Struts Version 1.2

Manfred Wolff, wolff@manfred-wolff.de, www.manfred-wolff.de
Michael Albrecht, malbrecht@neusta.de, www.neusta.de

"Was erlauben Strunz?" Die wohl berühmtesten deutschen Worte des italienischen Nationaltrainers Giovanni Trapattoni sollen als Einleitung für die ernstgemeinte Fragestellung dienen: Was erlaubt Struts? Seit März 2004 ist Struts ein Apache Toplevel Projekt. Es startete im Mai 2000 auf dem Rechner von Craig McClanahan. Seit dem letzten „General Available" (GA) Release 1.1 ist über ein Jahr vergangen. Zeit genug um die aktuelle Version 1.2 unter die Lupe zu nehmen.

Struts wird Standard!?

In den letzten Jahren haben sich Open-Source Frameworks auch bei der Entwicklung kommerzieller Software mehr und mehr durchgesetzt. Besonders die Anwendungen im Bereich des Apache- und Jakarta Projekts erfreuen sich immer größerer Beliebtheit. Als Präsentationsschicht mit JSP-Seiten ist Struts unanfechtbar die Nummer Eins und zwar nicht nur in der Open Source Welt. Es gibt kaum eine kommerzielle Lösung, die so durchdacht und verbreitet ist wie Struts. Die Umfirmierung in ein Apache Top-Level-Projekt unterstreicht diese Entwicklung noch einmal.

Was ist das faszinierende an Struts?

- Struts ist vom Design her sehr einfach gehalten. Durch die Umsetzung des MVC-Paradigmas orientiert es sich am „state of the art" der patternorientierten Softwareentwicklung. Bekannte Präsentations-Patterns wie der Front-Controller, der View-Dispatcher und der Service-to-Worker sind bei Struts Teil der Implementierung.
- Struts ist sehr schlank. Der Kern besteht aus sehr wenigen Basisklassen. Um eine Anwendung zu schreiben, reicht es oft schon aus zwei Java-Klassen aus dem Framework abzuleiten und zu implementieren (die `Action`-Klasse und die `ActionForm`-Klasse). Mit Hilfe von Standard `Action`-Klassen und dynamischen Formularen ist es sogar möglich einfache Anwendungen ohne viel Programmcode zu schreiben.
- Struts besitzt alles, was heutzutage für die Entwicklung der Präsentationsschicht von Webanwendungen benötigt wird – vom Layoutmanager, über die Validierung von Eingaben, der Fehlerbehandlung, der Anzeige von Fehlern und Meldungen, der Ausnahmebehandlung bis hin zu einer Plug-In Schnittstelle.

Damit ist das Framework vollständig und muss nicht mit anderen Frameworks ergänzt werden.

- Struts ist ein richtiges Framework und nach dem „Inversion of Control“ Entwurfsmuster mehr oder weniger konsequent entwickelt worden. Dieses verspricht sehr viel Flexibilität, weil das Framework nur den Kontrollfluss vorgibt, die eigentliche Implementierung jedoch außerhalb des Frameworks liegt. Dieses Prinzip wird auch oft als Hollywood-Prinzip bezeichnet: don’t call us, we call you. Das technische Framework steuert den Workflow der fachlichen Implementierung.
- Fast alle kommerziellen J2EE Suiten unterstützen inzwischen die Entwicklung mit Struts. Insbesondere grafische Oberflächen für die Workflowmodellierung sind ein Markt geworden.

Das Struts-Framework hat in der Version 1.2 viele Features hinzubekommen, die wir hier beschreiben wollen.

Validierung

Ein wichtiger Bestandteil einer Webapplikation ist die Prüfung der Eingaben des Benutzers. Dieser Vorgang wird mit Validierung bezeichnet. Da die Validierung ein so wichtiges Thema ist, wurde sie sehr früh in ein eigenes Subprojekt von Struts, nämlich dem Validator Framework, ausgelagert. Ziel dieses Projekts ist es nicht mehr nur Validierungen aus Struts Applikationen heraus zu realisieren, sondern die Validierung allgemein zu ermöglichen.

Bei Client-Server-Applikationen gibt es architektonisch bedingt zwei Orte, an denen validiert werden könnte: beim Client oder auf dem Server.

Für Standard-Validierungsaufgaben gibt es bereits vordefinierte Funktionen, die auf dem Client ausgeführt werden können.

- `required` – Prüfung auf notwendige Existenz eines Werts.
- `minlength`, `maxlength` – Prüfung auf Minimal- bzw. Maximalwert.
- `intRange`, `floatRange`, `doubleRange` – Bereichsprüfung.
- `byte`, `short`, `integer`, `long`, `float`, `double`, `date` – Typprüfungen.
- `creditCard` – Prüfung des Kreditkarten – Formats.
- `email` – Prüfung des Email-Adressenformats.

Die Funktionen bzw. Validatoren, die zur Verfügung stehen, werden in der Datei `validator-rules.xml` aufgeführt. Welcher Validator welchem Feld zugewiesen wird, wird in der Datei `validation.xml` angegeben. Beide zusammen steuern die Validierung.

```
<form-validation>
 <global>
  <validator
    name="required"
    classname="org.apache.struts.util.StrutsValidator"
    method="validateRequired"
    methodparams="java.lang.Object,
                  org.apache.commons.validator.ValidatorAction,
                  org.apache.commons.validator.Field,
                  org.apache.struts.action.ActionErrors,
                  javax.servlet.http.HttpServletRequest"
    msg="errors.required"/>

 ...
 </global>
</form-validation>
```

Listing 1: validator-rules.xml

Das Formularfeld `firstField` ist abhängig vom Validator `required`, dessen Definition in der Datei `validator-rules.xml` aufgeführt ist. Dieser Validator wird mit den entsprechenden Parametern aufgerufen. Sollte ein Fehler vorliegen (in diesem Fall würde das bedeuten, dass in `firstField` kein Wert enthalten ist), so wird die entsprechende Fehlermeldung, die unter dem Key `errors.required` als Message Ressource zu finden ist, mit dem Parameterwert `label.firstField` gefüllt und ausgegeben.

Diese Einträge sind nur ein Beispiel für serverseitige Validierung.

Die clientseitige Validierung findet bei Webapplikationen ausschließlich über Javascript-Funktionen statt. Dabei werden Javascript-Funktionen bzw. Funktionalitäten durch Auslösung entsprechender Javascript-Ereignisse ausgeführt.

```
<form-validation>
 <formset>
  <form name="myForm">
    <field
      property="firstField"
      depends="required">
      <arg0 key="label.firstField"/>
    </field>
    ...
  </form>
 </formset>
</form-validation>
```

Listing 2: validation.xml

Neu in Struts 1.2 ist, dass es bei der Javascript–Validierung die Möglichkeit gibt, die Unterbrechung beim ersten aufgetretenen Fehler zu verhindern und weiter zu validieren.

Dazu dient das Attribut `stopOnFirstError` des ValidatorPlugIn. Wird es auf `false` gesetzt, werden alle Constraints geprüft, bevor Fehlermeldungen ausgegeben werden.

Bedingte Validierung

Erst mit der Struts 1.2 Version ist es beim Validator möglich, Werte in Abhängigkeit anderer Werte bzw. Felder zu prüfen. Weil in der Version 1.1 diese Möglichkeit fehlte, ist das Validator Framework oft nicht in größeren Projekten eingesetzt worden.

Die dafür notwendige Regel um bedingt zu validieren heißt `validWhen`. Diese Regel ist über einen Parameter namens `test` zu deklarieren. Dieser entspricht dabei dem zu testenden Ausdruck.

Ein Standardbeispiel für die bedingte Validierung ist die zweifache Eingabe des Passworts zur Bestätigung. Es kann die Gültigkeit eines der Passwörter nicht ohne das andere Passwort geprüft werden, also hängen diese beiden Werte voneinander ab.

Sind passwort1 und passwort2 die beiden Felder, in denen der Benutzer sein Passwort eingetragen hat, dann ist die Regel für die Prüfung des Passworts folgende: passwort1 == passwort2

Die field-Definition für die Eigenschaft passwort1 sähe wie folgt aus:

```
<field property=”passwort1” depends=”validWhen” >
   <var>
      <var-name>test</var-name>
      <var-value>
         (passwort1 == passwort2)
      </var-value>
   </var>
</field>
```

In dieser ersten Version ist der Validator `validWhen` noch sehr einfach gehalten.

Er kann nur zwei Ausdrücke miteinander logisch verknüpfen. Die schlauen Mathematiker unter den Lesern könnten nun anmerken, dass dies reichen wird. Dazu müssten dann aber alle boolschen Ausdrücken erlaubt sein. Dies ist mitnichten der Fall.

Die Syntax für die Übergabe an den Parameter `test` ist wie folgt:

`( (Ausdruck_1) [&& || (Ausdruck_2)])`

Die booleschen Ausdrücke dürfen neben den relationalen Operatoren (`<`, `>`, `<=`, `>=`, `==`, `!=`) nur aus folgenden Bausteinen bestehen:

- `null`
- Zeichenkettenkonstanten wie `"Hallo Welt"` oder `'Hallo Welt'`
- Ganzzahlwerte in Dezimal-, Hexadezimal- und Oktalformat
- Einfache `field`-Elemente der Form
- Indizierte `field`-Elemente der Form mit Angabe des konkreten Index
- Indizierte `field`-Elemente der Form mit gleichem Index wie das aktuell zu validierende Feld
- Eigenschaften eines Array-Elements mit gleichem Index wie das aktuell zu validierende Feld
- `*this*` (das aktuell zu validierende Feld)

Im Wesentlichen sind also nur Vergleiche mit den Eigenschaften, Attributen bzw. Feldwerten eines Objekts mit entsprechenden typisierten Konstanten zugelassen.

Alles in allem ist das Validator Framework ein sehr ausgefeiltes, fortschrittliches Projekt, dessen Einsatz den Autoren auch schon außerhalb des Struts Umfelds sehr viel Freude gemacht hat und vor allen sehr viel Arbeit gespart hat. Ohne deklarative Validatoren müsste nämlich jedes Eingabeformular mit Hilfe von selbstverfasstem Quellcode validiert werden. Die einfache Konfigurierbarkeit der Regeln, Validatoren und der zu validierenden Felder

macht es zu einem nützlichen Werkzeug innerhalb der Software Entwicklung.

Wildcards nicht nur beim Tennis

Ein Problem bei der Anwendung von Struts in großen Projekten ist immer die explodierende Struts-Konfiguration, die `struts-config.xml`. Diese Datei wächst schnell über 1000 Zeilen hinaus und ist dann kaum mehr zu warten. Struts Module sind eine Antwort auf diese Explosion. Eine weitere komfortable Möglichkeit die Konfiguration einzudämmen ist die Benutzung des Wildcard Mappings. In Listing 1 zeigen wir ein normales Mapping in einer `struts-config.xml`.

```
<action path="editCustomer"
        type="de.struts.action.EditCustomerAction"
        name="CustomerForm"
        scope="request"
        validate="false">
        <forward name="success"
                 path="/pages/Customer.jsp"/>
</action>
```

Listing 3: Standardmapping

Standard-Aktionen bei Webanwendungen sind immer das Erzeugen, Modifizieren und Löschen von Daten. Neben der *editCustomer* Aktivität wird es auch eine *createCustomer* und eine *deleteCustomer* Aktivität geben. Außerdem wird es diese drei Aktivitäten nicht nur bei der Eingabe von Kundendaten geben, sondern auch an noch sehr vielen anderen Punkten in der Anwendung.

An diesem Punkt setzt das Wildcard Mapping ein. Dieses neue Struts Feature bietet eine generische Lösung, um mit wiederkehrenden Aktivitäten umzugehen. Das Wildcard Mapping ermöglicht die Angabe von Platzhaltern in der Definition des Mappings. Wie in Listing 4 zu sehen, benötigen wir nur noch einen Eintrag pro Aktivitätstyp.

```
<action path="edit*"
        type="de.struts.action.Edit{1}Action"
        name="{1}Form"
        scope="request"
        validate="false">
        <forward name="success"
                 path="/pages/{1}.jsp"/>
</action>
```

Listing 4: Wildcardmapping

Der Rest ist Disziplin bei der Einhaltung von Namenskonventionen. Der Asterix (*) dient hier als Platzhalter und kann mit der Variablen {n} adressiert werden, wobei n für das n-te vereinbarte Wildcard steht. Es ist also auch möglich mehrere Wildcards einzusetzen.

Wildcards sind die sichtbarste Veränderung von der Struts Version 1.1 zur Version 1.2.

Die Action für alle Fälle

Struts hat ein klassisches Event-Action Konzept. Events von der Oberfläche werden mit Hilfe des Struts-Controllers und des Request-Prozessors auf eine Action-Klasse weitergeleitet, welche als Schnittstelle zur Geschäftslogik dient.

Struts liefert einige Standard-Actions mit dessen Hilfe die Lösung immer wiederkehrender Aufgaben erleichtert wird. Diese Standard-Actions sind in der Version 1.2 erweitert worden.

Locale Action Mit Hilfe der `LocaleAction` kann die Sprach- und Ländereinstellungen für einen Benutzer geändert werden. Diese Unterstützung ist wünschenswert, weil immer mehr Anwendungen mehrsprachig ausgeliefert werden müssen. Die `LocaleAction` hat jedoch den Nachteil, dass nach Sprachumschaltung auf eine „bestimmte" Seite geschaltet werden muss. Dieses wird in der Struts-Konfiguration (`struts-config.xml`) beschrieben. Sinnvoller wäre eine Lösung, die nach Sprachumschaltung auf die Seite verzweigt, von der die Sprachumschaltung angestoßen wurde. Die hier implementierte Lösung hat in der Praxis leider keinen Bestand.

MappingDispatchAction – Struts besitzt bereits in der Version 1.1 die `DispatchAction` und eine Konkretisierung, die `LookupDispatchAction`. Grundkonzept dieser Implementierung ist es eine Überfrachtung der `execute()`-Methoden zu verhindern, die vom Controller aufgerufen wird. So können verschiedene Aktionen auf der gleichen View (klassisches Bespiel erzeugen, löschen und verändern von Daten) auf die gleiche Action geleitet werden, werden hier aber auf verschiedene Methoden der Action-Klasse verteilt – daher der Name DispatchAction.

Während die `DispatchAction` aufgrund von übergebenden Parametern entscheidet, welche Methode aufgerufen wird, wird dieser Mechanismus in der `MappingDispatchAction` bereits in der Konfiguration festgelegt. Dadurch sind die Informationen über den Screenworkflow einheitlich in der Struts Konfiguration enthalten und nicht in den Views „verstreut".

Die Maven Revolution macht vor Struts nicht halt

Mit der Version 1.2 wurde Struts wie viele anderen Apache- und Jakarta-Frameworks auf Maven umgestellt. Maven unterstützt das Konfigurationsmanagement von Softwareprojekten und erleichtert vor allem die Kompilierung derselben. Maven lädt die abhängigen Bibliotheken selbstständig aus einem zentralen Repository, so dass sich die Entwickler nicht umständlich alle Bibliotheken selber aus dem Internet herunter laden müssen.

Die Umstellung auf Maven, in der Struts Entwicklergemeinde nicht unumstritten, wird in der Version 1.3 vollständig abgeschlossen werden. So sind zwar zur Zeit mehrere Maven-Unterprojekte entstanden, wie *struts-faces* oder *struts-chain*, eine Einbindung in eine Maven-Multiproject-Umgebung wurde aber noch nicht durchgeführt. Die Website von Struts ist auch noch in der gewohnten Form und nicht im Maven Look, wie sehr viele andere Jakarta Projekte.

Die „Mavenisierung" von Struts kommt gerade zur richtigen Zeit. Die Vision der Struts Entwickler ist es den Struts-Controller weiterzuentwickeln. Außerdem soll eine breite Basis von Standard-View-Komponenten unterstützt werden, wie XSLT, JSTL oder JSF. Dieser Ansatz führte in der Vergangenheit dazu, dass Struts als Gesamtpaket immer mehr in Abhängigkeiten zu anderen Bibliotheken und Frameworks kam. Dadurch war es kaum noch einem normalen Entwickler möglich das Struts-Gesamtpaket zu kompilieren.

Struts besteht jetzt aus mehreren Maven-Projekten, welche jeweils durch ihre eigene `project.xml` repräsentiert sind. So ist es jedem möglich, auch die zukünftigen Struts Erweiterungen auszuprobieren und vor allen eine ablauffähige Versionen dieser Erweiterungen auf einfache Weise zu compilieren. Für „Maven Muffel" wurden die

build.xml Dateien überarbeitet und aufgetrennt, sodass auch mit der Ameise „ant“ Struts aus den Quelltexten zu einer Bibliothek kompiliert werden kann.

Modularisierung

Die Modulunterstützung, mit der in der Version 1.1 bereits begonnen wurde, ist mit der Version 1.2 weitgehend abgeschlossen. Durch die Einführung des Modulskonzepts wurde es möglich Struts Anwendungen besser zu strukturieren. Die Überfrachtung der zentralen Konfigurationsdatei, der `struts-config.xml`, konnte bereits in der Version 1.0 durch Split in mehrere kleinere Dateien entgegengewirkt werden. Eine gute Lösung war dies jedoch nicht, weil darauf geachtet werden musste, dass Bezeichner disjunkt gewählt werden. Was in der Version 1.0 noch fehlte, waren Namensräume. Diese gibt es jetzt in Form von Modulen. Kompliziert war schon immer der Modulwechsel. Für diesen Wechsel wurde eine Standard Action zur Verfügung gestellt, die `SwitchAction`.

Erst kurz vor dem Rollout der 1.2 Version wurde noch ein Patch eines Contributers in die Version hineingenommen. Jetzt ist es auch möglich direkt, z. B. durch einen Link, in eine anderes Modul zu verzweigen. Zu diesem Zweck haben einige Struts-Tags wie `frame`, `link`, `forward` und `rewrite` das zusätzliche Attribut `module` bekommen, mit dessen Hilfe ein direkter Einsprung in ein anderes Modul möglich ist.

Bildet fest Ketten

Herzstück des Struts Frameworks ist der Request-Prozessor. Dieser hat verschiedene Aufgaben, von denen nur einige hier kurz aufgeführt werden:

- Ermittlung der Ländereinstellungen des Users (Locale).
- Aufruf eines Hooks für anwendungsspezifische Aktionen.
- Bearbeitung von Zugriffsrechten.
- Übertragung der Werte des Requests in das Struts-eigene Model.
- Ermittlung der richtigen View oder der nächsten aufzurufenden Action-Klasse.

Durch die Bildung von Sub-Klassen lassen sich die Implementierungen der einzelnen Schritte verändern. Soll auch die Ablaufreihenfolge verändert werden, muss großer Teil des Codes kopiert werden. Auch lassen sich nur schwerlich neue Schritte in den Request-Prozessor hinzufügen.

Bei genauer Betrachtung der Implementierung ist jeder der Schritte, die der Request-Prozessor abarbeitet, dafür verantwortlich zu entscheiden, ob die Verarbeitung weitergeführt wird, oder ob abgebrochen werden. soll. Listing 5 zeigt einen kleinen Ausschnitt aus der *process()*-Methode des Request-Prozessors, der dieses Verhalten verdeutlicht.

Dieses Verhalten entspricht dem „Chain of Responsibility“ Pattern, zu Deutsch Zuständigkeitsketten Muster. Bei diesem Entwurfsmuster wird jeder Schritt durch ein Command-Objekt repräsentiert, welches nach Abarbeitung der eigenen Logik selbstständig entscheidet, ob der Prozess weitergeführt oder abgebrochen werden soll. Genau dieses tut der Request-Prozessor: Jeder Schritt entscheidet, ob die Methode verlassen werden soll oder ob die nächste Funktion aufgerufen werden soll.

Die Erweiterung Struts-Chain, realisiert ein Request-Prozessor mit dem „Chain of

Responsibility“ Entwurfsmuster. Dabei stützt sich diese Erweiterunng auf das commons-chain Framework, welches Mitte des Jahres aus dem Jakarta-Sandbox Stadium in die Jakarta-Commons übernommen wurde.

Auch wenn Struts-Chain erst in Release 1.3 vollständig in die Distribution integriert werden soll ist sie schon jetzt vollständig nutzbar. Außerdem ist diese Erweiterung ein wirklicher Meilenstein seit dem 1.1 Release. Struts-Chain ist einer von mehreren Schätzen, die seit der Version 1.2 im Contribution Ordner angeboten werden. Neben der Struts-Faces Unterstützung und dem Workflow-Paket ist diese Erweiterung bereits sehr weit fortgeschritten.

```
if (!processPreprocess(request, response)) {
    return;
}
ActionMapping mapping = processMapping(request, response,
path);
if (mapping == null) {
    return;
}
if (!processRoles(request, response, mapping)) {
    return;
}
ActionForm form = processActionForm(request, response,
mapping);
processPopulate(request, response, form, mapping);
if (!processValidate(request, response, form, mapping)) {
    return;
}
```

Listing 5: Abarbeitung durch den Struts Request-Prozessor

Ausblick auf 2.0

Seit langem befindet sich im Contribution-Ordner von Struts die Vision von Struts 2.0 mit Codename Jericho. Dabei ist die Richtung schon grob vorgegeben. Die beiden Dateien overview.txt und readme.txt verraten viel über die Intentionen der Struts-Entwickler.

Vor allem wollen sie sich mehr dem Controller-Part von Struts widmen. Es sollen sowohl mehrere Request-Medien unterstützt werden wie Servlets, Portlets, SOAP und Mocks als auch verschiedene Darstellungsformen wie JSTL, JSF oder Velocity unterstützt werden. Insgesamt sollen die eher proprietären Teile des Frameworks, wie die Struts Taglib-Bibliothek zugunsten von Standards aus dem Framework herausgenommen werden.

Die Struts Entwickler sind vor allem von dem Konzept „Inversion of Control“ inspiriert worden und wollen Struts konsiquenter in diese Richtung weiterentwickeln, welches vom Design her auf Abstraktionen setzt (Interfaces) und für die verschiedenen Kanäle Basisimplementierungen zur Verfügung stellt. Dazu gehört vor allem die sehr Servlet-Technik nahen Bestandteile aus dem Framework herauszulösen. Klassen wie HttpRequest etc. sollen dann nicht mehr an den Schnittstellen vorhanden sein, sondern allgemeine Kontexte.

An Ideen ist bereits viel zusammengesammelt worden, jedoch datieren sie bereits auf Ende 2003. Von daher ist es zumindest anzuzweifeln, ob es je ein Release 2.0 geben wird. Dafür sind die jetzigen Hauptentwickler des Frameworks zu sehr in andere

Aktivitäten eingespannt.

Fazit

Soll die Zeit berücksichtigt werden, die seit dem Release 1.1 vergangen ist, so ist der Schritt von Struts 1.1 zu 1.2 sehr klein. Im Wesentlichen hat das Framework eine Konsolidierungsphase hinter sich, die vor allem der Übersichtlichkeit gut getan hat. Lust auf Struts macht das, was noch kommt. Hier sei sowohl die struts-chain Erweiterung als auch die Unterstützung von JSF (struts-faces) genannt. Seit der Beta Freigabe von 1.2.1 sind bereits wieder zwei Monate vergangen, aber Ende August ist es dann doch noch gelungen ein GA-Realease auszurollen. Beim nächsten Release, 1.3, wird wieder mehr auf die Erweiterung der Funktionalität geachtet. Angedacht sind neben den erwähnten Erweiterungen die Einbindung eines SSL-Frameworks, Struts TestCase eine JUnit Erweiterung, StrutsCX zur Erstellung der View mit XSLT, einem Workflow Framework, einem Cocoon Plugin und ein Bean-Scripting Framework. Bleibt zu hoffen, dass das nächste Release nicht wieder ein Jahr bis zur Fertigstellung benötigt.

Wir haben fertig.

Michael Albrecht ist mathematisch-technischer Assistent und arbeitet seit seiner Ausbildung als Trainer, Coach und Architekt für J2EE Softwaresysteme zunächst für ein Trainingscenter und jetzt bei der NEUSTA GmbH in Bremen.

Manfred Wolff ist Diplom Informatiker und arbeitet seit fast 10 Jahren freiberuflich als IT-Dienstleister. Er konzipiert seit mehreren Jahren J2EE Projekte. Dabei setzt er bevorzugt Struts als Präsentations-Framework ein. Zusammen haben Sie das Buch Struts ge-packt, geschrieben.