Quo vadis Struts?

Michael Albrecht, Manfred Wolff

Wer heute "Struts" [1] als Präsentationsframework vorschlägt, erntet in Architektenkreisen häufig nur Kopfschütteln. Längst haben Frameworks wie Spring oder Präsentationstechnologien wie Java Server Faces die Popularität von Struts ein- bzw. überholt - so erscheint es jedenfalls in den einschlägigen Fachmagazinen. Mit der Version 1.3, die im Dezember 2005 erschienen ist, geht Struts einen neuen Weg: Weg vom "monolitischen" Struts in modulare Funktionalitäten, die auch außerhalb von Struts eingesetzt werden können. Und die Zukunft ist auch schon auszumachen: Struts und Webwork, ein weiteres MVC Webframework, wollen zusammenwachsen zu Struts Ti [2]. Dennoch steht die Frage im Raum: Ist die Zeit von Struts womöglich bereits abgelaufen?

Wer in langfristig angelegten Softwareprojekten auf Struts setzt, kann laut den Struts Entwicklern auch weiterhin auf Investitionssicherheit hoffen. Die Roadmap kündigt schon die Funktionalitäten der nächsten drei Releases an. Längst haben die Struts Entwickler gemerkt, dass sich rechts und links von ihnen eine Menge tut. Deshalb gibt es "Brücken" zu anderen Frameworks wie Cocoon, Spring oder JSF. Trotz neuerer guter Ansätze in anderen Frameworks, vor allem bei Spring, merken wir momentan in der Praxis immer wieder, dass auch in neuen Projekten Struts nachgefragt wird. Struts ist also allgegenwärtig in der realen Projektwelt. Aber erfüllt es auch wirklich die Anforderungen für die nächsten Jahre, wie die Entwickler von Struts glauben machen wollen? Dazu zunächst ein Blick auf die neuen Struts Features.

Subprojekte

Die augenscheinlichste Veränderung in der Struts Version 1.3 ist die Aufteilung des monolitischen Struts in kleinere Subprojekte. Dieses hat nicht nur kosmetische Gründe. Die einzelnen Unterprojekte sind so abgegrenzt, dass sie auch autark eingesetzt werden können. Damit setzt sich ein Trend fort: Struts war schon immer ein Lieferant für Lösungen, die auch in anderen Projekten eingesetzt werden können, wie das Validator Framework (commons-validator), das Ressourcen Framework (commons-resources) oder die Implementierung des Chain-of-Responsibility Entwurfmusters (commons-chain). Die zweite große Veränderung: Struts Shale ist aus dem Beta-Stadium herausgewachsen und gleichberechtigter Teil des Struts Frameworks.

Das Projekt Struts gliedert sich jetzt in folgende Einzelbestandteile:

- Struts Actionframework: Das "klassische" Struts hat neue Funktionalitäten bekommen.
- Struts Shale: Ein Framework, welches sich zum Ziel gesetzt hat "add ons" für die Java Server Faces Programmierung anzubieten.
- Tiles: Ein Layoutframework kann jetzt auch außerhalb von Struts eingesetzt werden.
- Struts Extras: Oft genutzte Funktionalitäten, wie z.B. die Sprachumschaltung, sind jetzt in dieses Unterprojekt umgezogen.
- Struts Examples: Umfangreiche Beispielanwendungen.
- Struts Taglib: Die Strutseigenen Tag Bibliotheken.

Struts Actionframework

Die wichtigste Neuerung im Struts Actionframework ist neben der Extraktion von Tiles und anderen Komponenten der zusammenstellbare Request-Prozessor. Bislang bestand der Request-Prozessor aus einer Klasse, die bestimmte Methoden zur Verarbeitung des Requests angeboten hat.

Der Request-Prozessor wird jetzt mit dem Chain-of-Responsibility Entwurfsmuster aufgebaut (siehe dazu den Kasten). Typische Abarbeitungsschritte im Request-Prozessor waren bislang die Erzeugung benötigter Beans, die Validierung dieser Beans, das Erzeugen der richtigen Action-Klasse etc. Diese einzelnen Schritte werden jetzt in einzelne sog. Commands ausgegliedert und können mit Hilfe einer XML-Datei komponiert werden. Das Listing 1 zeigt einen Ausschnitt dieser XML-Datei.

Listing 1

```
<chain name="process-action">
    <command className="org.apache.struts.chain.commands.servlet.SelectLocale"/>
    <command className="org.apache.struts.chain.commands.servlet.SetOriginalURI"/>
    <command className="org.apache.struts.chain.commands.servlet.RequestNoCache"/>
    <command className="org.apache.struts.chain.commands.servlet.SetContentType"/>
    <command className="org.apache.struts.chain.commands.servlet.SelectAction"/>
    <command className="org.apache.struts.chain.commands.servlet.AuthorizeAction"/>
    <command className="org.apache.struts.chain.commands.servlet.CreateAction"/>
    <command className="org.apache.struts.chain.commands.servlet.ExecuteAction"/>
</chain>
```

/Listing 1

Die Ereigniskette wird genau in der Reihenfolge abgearbeitet, in der sie konfiguriert wird. Dabei muss jeder einzelne Command entscheiden, ob die Verarbeitung abgebrochen werden soll, oder ob der nächste Command zur Ausführung kommt. Während früher die Abarbeitung sehr starr war, kann jetzt ohne Änderungen im Quelltext Funktionalität hinzugefügt werden oder entfernt. Des Weiteren ist die Abarbeitungsreihenfolge flexibel. es können selbst implementierte Commands einfach eingefügt werden.

Tiles

Tiles ist schon länger Bestandteil von Struts und eines der interessantesten Features des Frameworks. Tiles hat zwei sehr große Vorteile, die auch außerhalb von Struts einsetzbar sind.

In größeren JSP-Projekten machen sog. JSP-Includes große Probleme, weil diese Includes nie kontextfrei sind. Sie erwarten bestimmte Objekte und das führt dazu, dass diese "Teile" nicht wiederverwendbar sind. Im Grunde genommen liegt dies an der ständigen Vermischung von Inhalt und Layout. Es soll im Allgemeinen das gleiche Layout verwendet werden, und nur ähnlicher Content.

Beim Tiles Projekt wird dies wie folgt umgangen. Erst wird ein Layout definiert, in dem sich Platzhalter für Content befinden. Dann werden Tiles Definition deklariert, in denen die Platzhalter des jeweiligen Layouts mit Werten gefüllt werden. Diese Werte sind Tiles. Dabei stellt ein Tile (zu dtsch: Fliese) einen rechteckigen Bereich einer Webseite dar. Tiles-Definitionen selbst können auch aus weiteren Tiles-Definitionen bestehen, so dass sich ein rekursiver Baum aufspannt. Im Listing 2 besteht die Definition "standardLayout" wieder aus Tiles-Definitionen (z.B. standardMenu, standardFooter).

Listing 2

```
<tiles-definitions>
  <!-- Allgemeines StandardLayout -->
    <definition name="standardLayout" path="/layouts/StandardLayout.jsp">
       <put name="title" type="string">E.N.T.E. - Interwall</put>
       <put name="header" type="page">/pages/header.jsp</put>
       <put name="menu" type="definition">standardMenu</put>
       <put name="footer" type="definition">standardFooter</put>
    </definition>
</tiles-definitions>
```

/Listing 2

Ein JSP-Include kann nur andere JSP-Seiten inkludieren. Hier kommen wir zum zweiten großen Vorteil von Tiles: Die Quelle der View kann alles sein: Eine JSP Seite, eine Velocity Seite, eine XSLT ge"render"te Seite - was auch immer in letzter Konsequenz ASCII-Inhalt hat. Tiles ist also der Türöffner für weitere Präsentationslogiken unter Struts. Auch ein gemischter Betrieb z.B. Seiten mit

Struts-spezifischen Tags und Seiten, die mit Velocity entwickelt wurden, sind mit Tiles überhaupt kein Problem mehr. Fazit: Wenn Struts, dann bitte nur mit Tiles.

Struts Shale

Java Server Faces (JSF) bilden womöglich die Zukunft für webbasierte Programme. Im Unterschied zum Struts Actionframework bietet JSF eine eher an Swing angelehnte Präsentationstechnik. Im Mittelpunkt des Layout steht nicht das Tagging (gewrappte HTML Tags) sondern die Layoutkomponente mit ihren Bestandteilen Model, View und Controller. Der Controller besteht in diesem Fall aus Eventhandler, die auf verschiedene Ereignisse der Layoutkomponente reagiert. JSF ist also in erster Linie ein GUI-Framework, das sich auf die Erstellung und Verarbeitung von Benutzerschnittstellen (User Interfaces) spezialisiert hat. Es wird zusätzlich eine Art Werkzeugkasten mitausgeliefert, mit dem Oberflächen nach dem Baukastenprinzip zusammengestellt und mit entsprechenden Validatoren und Eventmechanismen erstellt werden können.

Zunächst: Struts Shale ist keine JSF Implementierung. Craig R. McClanahan, der Hauptentwickler von Struts Shale, der unter anderem auch federführend bei der Entwicklung der JSR 127 (JSF) war, betont immer wieder in den Struts Mailinglisten, dass wer sich mit JSF beschäftigen will, zunächst Struts Shale außen vor lassen soll. Struts Shale stützt sich auf vorhandene JSF-Implementierungen, wie die SUN Referenzimplementierung oder das MyFaces Framework. In wieweit die Extensions von Struts Shale wie der ViewController, der Dialog- oder der Application Manager tatsächlich von der JSF Entwicklergemeinde angenommen werden, wird sich erst in der Praxis zeigen. Zurzeit kennen wir persönlich noch keine großen Projekte, die sowohl JSF als auch Struts Shale einsetzen.

Struts Zukunft

Bestimmte Dinge im Struts-Framework führen immer wieder zu Ärgernissen so die Tatsache, dass die Struts Action kein POJ O ist, sondern Servlet-Spezifische Signaturen mitbringt (siehe dazu auch [3]). Dadurch ist die Logik der Action schwer zu testen und nicht wiederverwendbar für andere Präsentationsformen. Struts ist also in erster Linie ein Webframework und nicht in andere Umgebungen nutzbar (wie z.B. in einem Swing Client).

Die Actionsignature sieht zurzeit wie folgt aus:

```
public ActionForward execute(ActionMapping mapping,
                             ActionForm form,
                             HttpServletRequest request,
                             HttpServletResponse response)
        throws Exception {
   // some code
}
```

Abhilfe soll hier ein Redesign der Anwendung führen. Zukünftig soll es ein Action-Interface geben, welches mit einem Context gefüttert wird. Der Actionaufruf wird dadurch schlanker:

```
public void execute(Context context);
```

Der Context muss dann nicht ungetypt sein, sondern kann servlet- oder auch portletspezifische Aspekte als Methoden zur Verfügung stellen. Die Struts-Entwickler wollen hier aus einer Sackgasse raus, denn die ursprüngliche Bindung an Servlets und an JSP-Technologie ist schon lange zu eng geworden. In jeder Version wurde ein Schritt in Richtung "Öffnung" gemacht, die nächsten beiden Versionen sollen dann den Durchbruch bringen. Ob diese Version dann noch "Struts" ist bleibt offen. Auf der Seite von opensymphony, der Vertreiber von Webwork, wird bereits angekündigt, dass Struts 2.0 und Webwork 2.2 zu einem neuen Framework zusammenwachsen werden. "Released and ready for Struts" ist die Überschrift des letzten Announcements am 11. Januar 2006. Vielleicht ist

das tatsächlich das Beste, was Struts passieren kann, weil hier neue Kräfte freiwerden können. Denkbar ist aber auch das parallele Vorhandensein von Struts 1.x und Struts 2.x.

Der Zugang zur Businesslogik

Jede Struts-Schulung, die wir durchführen, fängt mit den magischen Worten an: Struts ist ein Präsentationsframework! Das ist die Stärke von Struts aber im gleichen Maße auch die Schwäche. Nehmen wir Spring: Spring glänzt mit einem leichtgewichteten Container, einem MVC Präsentationsframwork und bietet gleichzeitig Möglichkeiten "quasi Standards" wie iBatis oder Hibernate anzubinden. Auch kommerzielle Produkte, wie z.B. den O/R-Mapper Toplink, kann mit Spring eingebunden werden. Damit ist Spring komplett, was auch die Stärke dieses Frameworks ausmacht.

Seit Jahren gibt es Diskussionen in der Struts-Entwicklergemeinde, wie der Zugang zur Businesslogik gestaltet werden kann. In der Diskussion ist die Erweiterung Overdrive, welches commons-chain einsetzt, um eine generische Schnittstelle zur Businesslogik zur Verfügung zu stellen. Die Entwicklung wird von einem Struts-Frontman - Ted Husted - vorangetrieben. Hier muss jedoch aus unserer Sicht deutlich gesagt werden: Schuster bleib bei Deinen Leisten bzw. Struts bleib bei Deinen Verstrebungen.

Die Idee des leichtgewichteten Containers (Spring, Hivemind, PicoContainer) ist einleuchtend einfach und damit einfach genial:

- Die Businesskomponenten sind POJOs und somit auch außerhalb des Containers testbar.
- Die Komponenten werden zur Laufzeit komponiert (dependency injection) und haben somit eine hohe Abstraktion für den Anwender.
- Die Komponenten haben eine typisierte Schnittstelle, die der Präsentationsschicht zur Verfügung gestellt wird.

Der Overdrive Ansatz, die Businesslogik mit commons-chain einzubinden hat dagegen den großen Nachteil, bzw. bürgen zumindest die Gefahr der nicht-typisierten Schnittstelle. Die Komponenten erben ein einfaches Interface gleich des Struts Action-Interfaces `(public boolean execute (Context context))` und übergeben ihre Werte in einem Context. Der Ansatz ist sicherlich universell, kann aber aus unserer Sicht mittelfristig gegen Spring und andere leichtgewichtete Container nicht bestehen. Dazu ist z.B. Spring schon viel zu weit voraus. Struts bietet auch eine Anbindung an Spring, das scheint der bessere Weg zu sein.

Fazit

Wenn nur die Entwicklung innerhalb der JSP-Technologie betrachtet und der Servlet-Container als Ablaufumgebung gesehen wird, dann ist Struts immer noch auf dem Stand der Technik webbasierter Softwareentwicklung. Das Struts Actionframework erneuert sich womöglich durch das Verschmelzen mit Webwork und bekommt dadurch bestimmt neuen Schub. Die Struts Entwicklergemeinde hat noch Lust auf Neues und ist in der Lage mit neuen Versionen Impulse zu setzen. Der Baukasten - Request-Prozessor ist ein großer Schritt in Richtung mehr Flexibilität des Frameworks; Tiles öffnet die Türen für andere Präsentationsframeworks. Es ist immer wieder schön die Neuerungen von Release zu Release zu betrachten, sie reichen aber wahrscheinlich nicht aus. Ein Kollege meinte neulich: "Struts kommt langsam da an, wo Cocoon schon vor Jahren war". Die Releasezyklen von Struts sind tatsächlich sehr groß (mehr als ein Jahr pro Release) und dafür der Funktionszuwachs nicht adäquat. Die funktionalen Änderungen, welche für die Version 1.4 und 1.5 angekündigt sind, kommen sehr spät.

Ob Struts Shale eine solche Popularität wie das Struts-Actionframework erfahren wird, ist fraglich - sicherlich wird Struts Shale aber erst dann zu beurteilen sein, wenn sich JSF auch in größeren Projekten durchsetzt.

Struts ist aus unserer Sicht auch deshalb ein Auslaufmodell, weil es nicht geschafft hat, die anderen Layer webbasierter Enwicklung rechtzeitig zu bedienen um damit ein "komplettes" Framework zu werden. Die Struts-Ansätze, die es für den Zugang zur Businessschicht gibt, überzeugen ebenfalls nicht. Hier hat Spring beispielsweise als vollständiger leichtgewichteter Container deutlich mehr zu bieten. In der nächsten Zukunft werden Frameworks wie Spring den Markt erobern, weil sie auch Lösungen in der Anbindung von "quasi Standards" wie Hibernate bieten und damit ein komplettes Framework anbieten - da sind wir uns ziemlich sicher. Aber wer kann sich in der IT-Welt schon wirklich sicher sein?

Links & Literatur

[1] http://apache.struts.org

[2] http://struts.apache.org/struts-sandbox/struts-ti/index.html

[3] http://manfred-wolff.de/data/Designschwaechen.pdf

Zu den Autoren:

Michael Albrecht ist Chefarchitekt der NEUSTA GmbH. Er ist außerdem für die Qualifizierung der Mitarbeiterinnen und Mitarbeiter verantwortlich.

Manfred Wolff ist freiberuflich tätig und NEUSTA Partner. Sein Schwerpunkt ist die Entwicklung und Realisierung großer webbasierter Java-Anwendungen.