# Struts 1.3 – Alle Neuerungen

*Michael Albrecht,m.albrecht@interwall.net, www.interwall.net*
*Manfred Wolff, wolff@manfred-wolff.de, www.manfred-wolff.de*

**Struts war von Anfang an treibende Kraft für weitere Frameworks im Jakarta Umfeld. So wurden der Digester, das Validator Framework, das Ressourcen Framework oder das Chain Framework im Wesentlichen im Kontext von Struts entwickelt, um dann als Commons Frameworks extrahiert zu werden. Mit der Version 1.3 wird nun (endlich) der Schritt vollzogen, Struts selbst nicht als starres Framework zu betrachten, sondern einzelne Subprojekte anzubieten. Die wichtigsten Subprojekte sind das Struts Action Framework (das traditionelle Struts), das Struts Shale Framework sowie einige Erweiterungen wie das Tiles Framework oder die Struts Tag Bibliothek. Das erstgenannte ist ein mehr "requestorientiertes" Framework, während Struts Shale ein eher komponentenbasiertes Framework auf der Basis von JSF-Technik darstellt. Mit diesem Artikel wollen wir die Neuerungen in der Version 1.3 darstellen**

### Subprojekte

Auf den ersten Blick sieht es so aus, als ob die Umbenennung von Struts in Struts Action Framework die einzige Änderung ist. Dann aber wird nach dem ersten Einbinden dieses neuen Jars sofort deutlich, dass auch so spezielle Actions wie `DispatchAction` durch das Refactoring in ein eigenes Subprojekt, nämlich struts-extras, verlagert wurden. Genauso ist es mit den Struts eigenen Tag Bibliotheken passiert. Im Prinzip besteht Struts jetzt aus zwei Core Modulen (Struts Action Framework, Struts Shale Framework) sowie optionalen Erweiterungen. Damit ist auch die Diskussion beendet, die über zwei Jahre in den Mailing Listen geführt wird, nämlich nach der Versionierung von Struts. Bisher konnte davon ausgegangen werden, dass das traditionelle Struts Framework in der Version 1.x weitergeführt wird und das neue JSF-basierte Framework die Version 2.x bekommt. Mit der jetzigen Lösung hat das Framework Zukunftssicherheit bewiesen.

### Struts-Chain

Die bedeutendste Änderung in der Struts Version 1.3 ist die Einführung von Struts-Chain. Das Herzstück des Struts-Controllers ist in der Version 1.1 von der `ActionServlet` Klasse zum `RequestProzessor` umgelagert worden. Dieses war mit der Einführung von Modulen notwendig geworden. Der Request Prozessor ist dafür verantwortlich einen Request abzuarbeiten. Dafür wurde eine Reihe von Methoden aufgerufen, die jeweils einen Aspekt der Requestverarbeitung beinhaltete. Damit auch anwendungsspezifische Verarbeitung stattfinden konnte, stellte der Requestprozessor einen Hook zur Verfügung. Dieses Vorgehen war jedoch sehr inflexibel, weil der Hook an einer ganz bestimmten Stelle der Requestverarbeitung abgearbeitet wurde.

Mit Struts-Chain ist jetzt der Request-Prozessor zu einem Baukasten geworden, der je nach Anforderung variabel gestaltet werden kann. Grundlage hierfür ist das Chain-Of-Resposibility Entwurfsmuster (2), welches im commons-chain Framework (3) von einigen Struts Entwicklern implementiert wurde.

Auch von den Autoren des Struts Frameworks wird der "composable request processor" als die Hauptänderung in der Version 1.3 angepriesen. Die verschiedenen Aspekte der Requestverarbeitung können nun in einer XML-Datei deklarativ beschrieben werden. Die notwendigen Verarbeitungsschritte, die früher vom Request-Processor durchlaufen wurden, werden jetzt vom Standard-Chain durchgeführt. Die folgende Liste stellt die Methoden des alten Request-Prozessors mit den neuen "Kettengliedern" gegeneinander. Weiter unten ist ein Ausschnitt der neuen Chain-Konfiguration, welche die fundamentale Änderung noch einmal dokumentiert. Hier eine Gegenüberstellung der Methoden des "alten" Requestprozessors zu den neuen Kettengliedern.

- processMultipart : Wurde im Action-Servlet integriert.
- processPath: **SelectAction,** dieses Command ersetzt auch die alte processMapping-Methode.
- processException: **ExceptionCatcher** / **ExceptionHandler**
- processLocale: **SelectLocale**
- processContent: **SetContentType**

- processNoCache: **RequestNoCache**
- processPreprocess: Entfällt. Es ist nun möglich an allen Stellen der Prozesskette eigene Lookup-Commands einzubetten.
- processMapping: **SelectAction**, dieses Command ersetzt auch die alte processPath-Methode.
- processRoles **AuthorizeAction**
- processActionForm: **CreateActionForm**
- processPopulate: **PopulateActionForm**
- processValidate: **ValidateActionForm** / **SelectInput**
- processForward: **SelectForward**
- processInclude: **SelectInclude** / **PerformInclude**
- processActionCreate: **CreateAction**
- processActionPerform: **ExecuteAction**

```
<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"/>
   etc.
   <command
    className="org.apache.struts.chain.commands.servlet.ExecuteAction"/>
</chain>
```

Listing 4: Der Requestprozessor mit Struts Chain

Um auf das Chaining überhaupt zugreifen zu können, muss es über das ActionServlet (immer noch der Frontmann unter den Struts Servlets) konfiguriert werden. Diese Konfiguration geschieht ja bekanntermaßen über den Eintrag in der `web.xml`:

```
<servlet>
   <servlet-name>action</servlet-name>
   <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
   <init-param>
   <param-name>config</param-name>
   <param-value>/WEB-INF/struts-config.xml</param-value>
   </init-param>
    evtl. weitere Module
   <init-param>
     <param-name>chainConfig</param-name>
     <param-value>org/apache/struts/tiles/chain-config.xml</param-value>
   </init-param>
   weitere Optionen
</servlet>
```

Listing 5: Konfiguration der Struts Chain

Wie die Pfadangabe vermuten lässt, führt dieser Parameter zu einer Struts Standardkonfiguration der Chain. Die Standardkonfiguration ist so gewählt, dass sie der Struts 1.* - Kompatibilität genügt. Wird die Standardkonfiguration genutzt (wie in Listing 5 beschreiben), so kann die Angabe auch entfallen, weil dies die Voreinstellung ist.

### Flexiblere Konfiguration

Es ist seit Struts 1.3 möglich, die Struts Konfiguration um eigene, proprietäre Schlüssel-Werte-Paare zu erweitern, die dann den entsprechenden Konstrukten, beispielsweise den Action-Klassen, zur Verfügung stehen. Damit ist einerseits eine deutlich flexiblere Konfiguration machbar; andererseits ist eine Wiederverwendung solcher Konstrukte oftmals nur bedingt möglich. Dennoch soll das folgende Beispiel einen solchen Einsatz zeigen.

Stellen wir uns mal eine Situation vor, in der eine Action request-invariante Information aus der Session lesen soll. Dies könnte beispielsweise die User-Information aus der `HttpSession` sein. Dann wäre es schön, wenn die Übergabe des Namens des Session Attributs, unter dem diese Information zu finden ist, als Wert des Schlüssels übergeben wird.

```
<action-mapping ...>
   <action ...>
      <set-property key="sessionUser" value="userInSession" />
   </action>
 </action-mapping>
```

Listing 5: Erweiterung des Action Mappings durch beliebige Key-Value-Paare

Dieser Konfigurationsparameter kann dann in der Action wie folgt ausgelesen werden:

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request,
        HttpServletResponse response)
        throws Exception {
      // ...
      String sessionAttributeName = mapping.getProperty("sessionUser");
      //Benutzer auslesen
      User user =
              request.getSession().getAttribute(sessionAttributeName);
      //...
}
```

Listing 6: Zugriff der Action auf erweiterte Konfiguration

Diese erweiterten Konfigurationsparameter können in folgenden Tags verwendet werden:

- `<form-bean .../>`
- `<form-property .../>`
- `<exception .../>`
- `<forward .../>`
- `<action .../>`
- `<controller .../>`
- `<message-resources .../>`

Diese Form der Erweiterung ist auch beim Wildcardmapping möglich:

```
<action path="/Save*"
   name="Save{1}"
  extends="BaseSave">
  <set-properties key="Save" value="{1}"/>
</action>
```

Listing 7: Erweiterte Konfiguration beim Wildcard Mapping

### Erweiterte Fehlerbehandlung

Die Behandlung von Fehlern, nachdem der Response bereits ausgeliefert wurde ist durch die Einführung globaler Parameter bei den Exception-Handler ausgeweitet worden. Bei der Benutzung von Tiles ist es häufiger vorgekommen, dass eine Ausnahme aufgetreten ist nachdem der Response bereits aufgebaut und an den Browser abgeschickt wurde. In diesem Fall soll die Exception vielleicht gar nicht so behandelt werden, dass Sie eine Benutzerausgabe provoziert. Exception-Handler können jetzt so implementiert werden, dass Sie diese Exceptions ignorieren.

```
<exception key="GlobalExceptionHandler.default"
           type="java.lang.Exception"
           path="/ErrorPage.jsp">
           <set-property key="SILENT_IF_COMMITTED"
           value="true" />
</exception>
```

Listing 8: Ignorierung von Fehlern nach erfolgtem Response

### Catalog und command

Mit der Einführung von Katalogen und Ereignisketten (chain of responsility) wurde die Starrheit des Request-Prozessors aufgelöst. Die Abarbeitungskette des Requests ist nun mit einer XML-Datei konfigurierbar. Konsequente Weiterentwicklung dieses Prozesses ist, dass (theoretisch) jedes Action-Mapping mit einem eigenen Katalog bzw. mit einer eigenen Ereigniskette abgearbeitet werden kann.

Catalog und command sind Attribute, die Action Mappings und Controller Tags hinzugefügt werden können und welche die Ansteuerung eines speziellen Commands in einem angegebenen Katalog ermöglichen. Normalerweise wird der Standard Katalog abgearbeitet. Dieses ist der Katalog mit dem Namen "`struts`" und der Command mit dem Namen "`servlet-standard`" der wiederum unter anderem die Prozesskette mit dem Namen "`process-action`" aufruft. Diese ist in Listing 4 in Ansätzen dokumentiert. Die XML-Datei für den Standard-Katalog ist in der Struts Distribution unter dem Namen `chain-config.xml` abgelegt. Wollen Sie einen eigenen Katalog abarbeiten, so können Sie in der web.xml kommasepariert weitere XML-Dateien hinzufügen. Pro XML-Datei kann nur ein Katalog konfiguriert werden.

```
<init-param>
   <param-name>chainConfig</param-name>
   <param-value>org/apache/struts/tiles/chain-config.xml,
                de/neusta(projekt/self/meine-chain-config.xml
    </param-value>
</init-param>
```

Listing 9: Hinzufügen von eigenen Katalogen

Dann kann in einem Action-Mapping über die Attribute `catalog` und `command` auf die erweiterte Konfiguration zugegriffen werden. Ob dies tatsächlich in der Praxis benötigt wird, ist fraglich, aber hier ist das Struts Framework jetzt zukunftssicher auch für die Anbindung anderer Technologien entwickelt worden.

### Vererbung von Struts Konfigurationen

Struts Konfigurationen können nun erweitert werden, indem mit dem Attribut

"`extends`" auf eine bestehende Konfiguration Bezug genommen wird. Damit folgt man dem Muster wie es bereits bei Tiles Konfigurationen möglich und sehr hilfreich war.

- `<form-bean .../>`
- `<exception .../>`
- `<forward .../>`
- `<action .../>`

Auch hier soll ein Beispiel das Vorgehen und die Vorteile erläutern.

```
<form-beans>
   <form-bean name="Person"
              type="org.apache.struts.action.DynaValidatorForm">
        <form-property name="vorName" type="java.lang.String" />
        <form-property name="nachName" type="java.lang.String"/>
   </form-bean>

   <form-bean name="Kunde"
              type="org.apache.struts.action.DynaValidatorForm"
              extends="Person">
        <form-property name="kundenNummer" type="java.lang.String" />
    </form-bean>
</form-beans>
```

Listing 10: Vererbung von Struts Konfigurationen

Im vorliegenden Beispiel wird mit dynamischen Formularen gearbeitet. Dynamische Formulare werden vollständig in der Struts-Konfigurationsdatei vereinbart und es muss keine separate Klasse angelegt werden. Das Formular "Kunde" erbt von dem Formular Person die Attribute "`vorName`" und "`nachName`".

### Mavenizing projects

Wer es mal versucht hat, alle Struts Features zu bauen, hat früher sein blaues Wunder erlebt. Struts besteht ja nicht nur aus dem `core`, sondern hat mit jeder Version die zukünftigen Erweiterungen im `contrib` package anzubieten. Im `legacy` Ordner sind außerdem alle früheren Versionen zu finden. Die Abhängigkeiten sind so enorm, dass es kaum möglich war, alle abhängigen Bibliotheken im Netz zu finden.

Maven macht es möglich (4): Jetzt macht es richtig Spaß, Struts "from the scratch" zu bauen, weil alle notwendigen Bibliotheken in der Maven `project.xml` zusammengefasst sind und automatisch in das lokale Repository geladen werden. Die Umstellung auf Maven, in der Struts Entwicklergemeinde nicht unumstritten, führt nun dazu, dass jede und jeder Struts aus den Quellen bauen kann und ist somit ein Gewinn.

Mit dem Befehl `maven jar` werden die Abhängigkeiten von Struts aufgelöst, die Quellen werden kompiliert, die Tests werden ausgeführt und das jar-Archiv wird gebaut. Bei Erstellung des Artikels lagen nur die Nightly-Builds vor, die Tests liefen leider nicht durch. Soll dennoch ein jar-Archiv gebaut werden (in der Binary Distribution ist das jar immer enthalten) so hilft es eine eigene `build.properties` Datei zu erstellen, hier ein `maven.test.skip=true` einzutragen und sie in das Verzeichnis des Subprojekts zu legen, welches gebaut werden soll. Sicherlich ist es nicht im Sinne des Erfinders die Tests einfach zu überspringen, aber als Notlösung hilft es.

Der Gesamtprozess sieht dann wie folgt aus:

```
> maven jar
 __  __
|  \/  |__ _Apache__ ___
| |\/| / _` \ V / -_) ' \  ~ intelligent projects ~
|_|  |_\__,_|\_/\___|_||_|  v. 1.0

build:start:
java:prepare-filesystem:
java:compile:
    [echo] Compiling to /struts-action-1.3.0-dev/target/classes
java:jar-resources:
test:prepare-filesystem:
test:test-resources:
test:compile:
test:test:
jar:jar:
    [jar] Building jar: /struts-action-1.3.0-dev/target/struts-action-
1.3.0-dev.jar
BUILD SUCCESSFUL
Total time: 4 seconds
Finished at: Thu Jan 05 09:47:37 CET 2006
```

### Migration

Die Migration von Struts 1.2 zu Struts 1.3 ist denkbar einfach.

Die Bibliothek `struts.jar` ist mindestens durch die Bibliotheken `struts-action`, `struts-extras`, `struts-taglib` und `struts-tiles` (auch wenn Tiles nicht zum Einsatz komme)zu ersetzen

Als zusätzlich Abhängigkeit muss die `commons-chain` Bibliothek hinzugefügt werden.

Da sich häufiger etwas an den Tag-Bibliotheken ändert und sich teilweise Attribute in der Version 1.3 auch geändert haben, sollten die Definitionen direkt aus der `struts-taglib` Bibliothek genommen werden. Hierzu müssen die Definitionen direkt aus dem jarfile geladen werden. Die JSP-Direktive

```
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
```

(als Beispiel für die logic Bibliothek) tut genau das. Dann können auch Verweise in der `web.xml` entfallen und die Bereitstellung von tld Dateien im `WEB-INF` Verzeichnis.

Nach dieser Minimal-Migration wird immer noch der "alte" Requestprozessor von Struts 1.2 bedient und wir sind im Kompatibilitätsmodus. Weitere Migrationen können dann step-by-step vollzogen werden.

Noch ein Tip zur Migration: Wenn Sie die Fehlermeldung

```
org.apache.jasper.JasperException: Module 'null' not found.
```

sehen, dann liegt das immer daran, dass Sie die `commons-chain` Library nicht eingebunden haben. Die Fehlermeldung führt einen bei der Fehlersuche völlig in die Irre.

### Resumee

Zwei Fragen stehen bei der Auswahl von Frameworks aus Managementsicht immer im Mittelpunkt:

- Wie sieht es mit der Investitionssicherheit aus?
- Ist das Framework nicht bereits veraltet?

Die erste Frage hat Struts selbst eindrucksvoll beantwortet. Das Framework hat sich in

den letzten 5 Jahren zu einem Standard entwickelt. Die Releasezyklen von ca. einem Jahr sind solide; das Framework entwickelt sich kontinuierlich weiter. Mit der Aufnahme von Struts-Shale werden moderne Techniken wie Java Server Faces in das Framework integriert, so dass Struts auch in den kommenden Jahren "state of the art" bleiben wird, auch wenn andere sehr gute Präsentationsframeworks, wie z.B. das MVC Spring Framework, im Kommen sind und eine gute Alternative zu Struts bieten.

(2) Siehe GoF: Entwurfsmuster, Elemente Wiederverwendbarer objektorientierter Software, Addison Wesley, S. 410

(3) Siehe http://jakarta.apache.org/commons/commons-chain

(4) Eine Kurzanleitung zur Installation von Maven ist hier erhältlich: http://manfred-wolff.de/data/Maven-short.pdf