Entwicklung von Webapplikationen mit dem Struts-Framework

Tilmann Kuhn

Tilmann Kuhn, MatrNr. 1056679, Gildestr. 34, 76149 Karlsruhe

Abstract. Diese Seminararbeit Betrachtet die Verwirklichung einer Webapplikation, die unter Verwendung des Struts Frameworks[1] aus dem Apache-Jakarta-Projekt realisiert wurde. Dabei werden zunächst generelle Techniken und das verwendete Framework und Bibliotheken vorgestellt, auf Realisierung der Beispielapplikation eingegangen und zum Schluß die Erfahrungen ausgewertet.

1 Einführung

1.1 Warum Frameworks verwenden?

Beim Schreiben von Webapplikationen stößt man immer wieder auf verschiedene Probleme. Zum Einen sind da Dinge, die mit kurzen Programmstücken erledigt sind, wie das Einfügen einer lokalisierten Nachricht. Zum Anderen gibt es auch größere Hürden, wie beispielsweise die Einbindung von Datenbanken, die Verfolgung von Benutzersessions oder eine Transaktionsverwaltung. Allen solchen Aufgaben ist gemeinsam, dass sie bei vielen Webapplikationen benötigt werden. Frameworks bieten meist einen jeweils verschieden umfangreichen Satz solcher Grundfunktionen, der die Notwendigkeit diese neu zu schreiben und eigene Bibliotheken anzulegen in vielen Bereichen klein hält.

In Webapplikationen kommt oft eine bunte Mixtur verschiedener Technologien zum Einsatz, die oft Änderungen und Erweiterungen unterworfen sind. Auch kommt es mitunter vor, dass man in seiner Applikation Unterstützung für eine bestimmte eventuell neue Technologie hinzufügen möchte. Frameworks können die Verwendung verschiedener Technologien vereinfachen indem sie den Programmierer mit technologiespezifischen Funktionen zur Hand gehen oder sogar die Technologie komplett abkapseln und somit eine transparente Verwendung zulassen, so dass man sich um die Details der Technologie überhaupt nicht kümmern muss. Die Adaption neuer Features und Technologien wird dann größtenteils von den Entwicklern der Frameworks erledigt und verringert somit den Aufwand in der Programmierung der eigentlichen Webapplikation.

---

1 Das Struts-Framework und Informationen dazu finden Sie in [1]

Desweiteren erleichtern manche Frameworks den „guten“ Softwareentwurf und den Übergang vom Entwurf zur Implementierung, indem sie die Verwendung oft vorkommender Entwurfsmuster fördern oder zum Teil sogar vorschreiben. Damit ist eine erleichterte Wartung bestehenden Codes gegeben, aber auch dadurch, das Frameworks oft so populär werden, dass man schon von einem Quasi-Standard sprechen kann, so dass projektfremde Programmierer, die das Framework bereits kennen sich zügig einarbeiten können.

Ein wesentlicher Nachteil von Frameworks sei hier nicht verschwiegen: Es gibt inzwischen zahlreiche verschiedene von ihnen mit unterschiedlichen Fähigkeiten, sodass man zunächst erst einmal evaluieren muss, welches am besten zur angestrebten Applikation und Einsatzumgebung passt. Hat man sich dann einmal für ein Framework entschieden und seine Applikation implementiert, fällt es aufgrund der unterschiedlichen APIs schwer, die Software auf ein anderes zu portieren, falls das gewählte neuen oder geänderten Ansprüchen nicht mehr genügt.

1.2 Technologien in Webanwendungen

Im Folgenden werden kurze Beschreibungen einiger wichtiger Technologien, die häufig in Webapplikationen zur Anwendung kommen, angeboten.

Anmerkung: Hier wird nicht auf Technologien eingegangen, die ausschließlich in Microsofts .Net-Framework, das sich inzwischen auch wachsender Beliebtheit erfreut, Verwendung finden, da die hier untersuchten Anwendungen alle für eine reine Java-Umgebung entwickelt wurden.

Java Servlets: Java Servlets ist eine Technologie, die herstellerunabhängig vom JCP[2] standardisiert wird. Sie dient der dynamischen Erstellung von Webseiteninhalten und ist damit die Basis, die den meisten Webapplikationen im Java-Umfeld zu Grunde liegt. Dem Programmierer wird im Wesentlichen eine Spezifikation und ein API gegeben, nach denen er seine ‚Servlets‘ in Java erstellt. Diese werden dann in einem sogenannten Servlet-Container eingesetzt, um funktionsfähig zu sein. Kommt dann eine Anfrage an den Container, z.B. über Http, wird der entsprechende Javacode ausgeführt und das Ergebnis an den Aufrufer zurückgeliefert. Eingesetzt werden Servlets hauptsächlich im View- und Controller-Bereich von Webapplikationen. Servlet-Container-Implementierungen gibt es verschiedene; die bekanntesten sind Tomcat [5] und Jetty [6] Weitere Auskunft über Java Servlets gibt [4].

2 Java Community Process, ein internationales Standardisierungsgremium, das sich um Erarbeitung und Standardisierung von Java-Technologien kümmert. Näheres in [3]

Java Server Pages (JSP): JSP sind im Wesentlichen eine Erweiterung der Java Servlets, die eine alternative, normalen Webseiten oder auch XML sehr ähnliche Notation bietet, und damit ermöglicht, dass auch Webdesigner, die Java nicht beherrschen, auf einfache Weise übersichtlich strukturierte, dynamische Webseiten erstellen können, während die veränderlichen Inhalte nach wie vor von programmierten Servlets zur Verfügung gestellt werden. Desweiteren warten Java Server Pages noch mit dem neuen Konzept der Tag-Libraries auf, die die JSP-Syntax erweiterbar machen und es ermöglichen komplexe Vorgänge in Javacode hinter einfachen Html-ähnlichen Tags zu verbergen. Haupteinsatzgebiet ist der View-Bereich von Webanwendungen. Webcontainer in denen JSPs eingesetzt werden können sind in der Regel die selben, die auch Java Servlets unterstützen. Die JSP Spezifikation, die auch vom JCP gepflegt wird, und Weiteres finden Sie in [7].

Enterprise Java Beans (EJB): Bei dem EJB-Standard, auch vom JCP, handelt es sich um ein mächtiges Framework zur Erstellung wiederverwendbarer und verteilter Javakomponenten, deshalb wird es auch Komponentenmodell genannt. Eine Enterprise Java Bean ist eine solche Komponente, die zur Verwendung in einem EJB-Container eingesetzt wird. Damit der Programmierer der Komponente sich auf das Wesentliche konzentrieren kann und damit die Komponente unabhängig von ihrem tatsächliche Einsatzort ist, übernimmt der Container standardmäßig zahlreiche Funktionen, die der EJB und dem Client, der sie verwendet, zur Verfügung stehen. Darunter sind u.a. Transaktionsverwaltung, Persistenz von EJBs, Auffinden, Erzeugen und Vernichten von EJBs und Nachrichtenverwaltung für asynchrone Kommunikation. Die gängigen Webcontainer bieten darüber hinaus weitere Zusätze wie Verbindungs- und Bean- Pooling, Loadbalancing zwischen verschiedenen Containerinstanzen, Zugriffsschutzmechanismen und mehr. EJBs werden häufig verwendet, um den Geschäftslogikbereich von Webanwendungen abzudecken, wofür sie auch hervorragend geeignet sind. In unserer Webapplikation wurden sie jedoch nicht verwendet, da sich ihr Einsatz bei relativ kleinen Webapplikationen wenig lohnt. Genaue Informationen zu EJBs können sie [8] entnehmen.

Java Naming and Directory Interface (JNDI): Dieses J2EE API dient dazu, auf der einen Seite beliebige Objekte mit einem Verzeichnisnamen, ähnlich einem Verzeichnis auf der Festplatte, zu verknüpfen und auf der anderen Seite nach einem bestimmten Namen nachzuschlagen, um dann das verknüpfte Objekt zu verwenden. Benutzt wird diese Funktionalität um Webanwendungen unabhängig von ihrer Einsatzumgebung zu implementieren. Beispiel wäre eine Webanwendung, die einen Datenbankzugang unter einem bestimmten JNDI-Namen erwartet und diese von ihrem Webcontainer dort zur Verfügung gestellt bekommt.

XML und XSL: XML an sich ist eine SGML-Variante, die angibt, wie man strukturierte Daten wohlgeformt in einem Textdokument speichern kann. Auf XML-Basis lassen sich mittels Schemata Sprachen definieren, mit denen dann auch eine syntaktische Korrektheit der Daten überprüft und, somit erzwungen werden kann, was man für die Speicherung von Informationen oft benötigt. Vorteil von XML ist hauptsächlich, dass es auch von Menschen lesbar ist, und dass man keine eigenen Parser und Werkzeuge zur Handhabung schreiben muss, da diese reichlich zur Auswahl stehen.[3] XSL (Extensible Style Language) ist eine deklarative Programmiersprache zur Übersetzung von XML Dokumenten in ein nahezu beliebiges anderes Format wie z.B. wieder XML oder HTML oder das Portable Document Format (PDF). Im Bereich von Webapplikationen gibt es zahlreiche Anwendungsgebiete für XML. Es wird häufig verwendet für Ausgaben im View-Bereich, für Konfigurationsinformationen oder auch als Datenaustauschformat für Enterprise Application Integration.

Weitere Technologien, deren Einsatz lohnend sein kann, die den Rahmen dieser Ausarbeitung jedoch sprengen würden.

Java Management Extensions (JMX) zur Überwachung und Verwaltung nicht nur von Webapplikationen. [9]

Simple Object Access Protocol (SOAP) zum standardisierten und unabhängigen Fernzugriff auf Objekte auf einem Server mittels verschiedener Protokolle (smtp, http) unter der Verwendung von XML-Nachrichten. [10]

Java Mail API bietet Zugriff auf E-mail-Funktionalitäten von Java aus. [11]

1.3 Entwurfsmuster in Webanwendungen

Selbstverständlich kann man in Webanwendungen wie in jeder anderen Anwendung etliche Entwurfsmuster verwenden. Im Folgenden möchte ich einige aufführen, die sich in Webapplikationen als praktisch und häufig eingesetzt erwiesen haben. Die angegebenen Entwurfsmuster sind entweder aus [13] bzw. [17] oder GoF-Patterns.

Front Controller: Verwendung eines Controller-Objektes als zentralen Zugangspunkt zur Webapplikation. Der Front Controller kümmert sich um die Behandlung der Anfrage. Dies kann er erreichen durch Erledigen oder Delegieren von: Überwachung der Sicherheit (Authentifizierung), Aufruf der Bearbeitungslogik, Auswahl einer geeigneten View-Komponente, Fehlerbehandlung.

---

3 Keine Parser für die XML Daten, wohl aber für die selbst definierte Syntax. Dabei wird man aber durch die passenden APIs sehr gut unterstützt.

4 „GoF“ = Gang of Four. Die Patterns der vier Entwurfsmusterpioniere können in [12] nachgelesen werden.

Value Object: Ein einfache JavaBean ohne Geschäftsmethoden, die als Datentransportcontainer zwischen verschiedenen Schichten der Applikation dient und diese damit ein Stück weit entkoppelt.

Service to Worker: Eine Kombination aus einem Controller einem View-Dispatcher und weiteren Helferobjekten, bei der der Controller nach einer Anfrage zunächst veranlasst, dass die entsprechende Geschäftslogik aufgerufen wird und die Ergebnisdaten zur Verfügung stehen. Danach wird über den Dispatcher die zuständige View-Komponente ermittelt, die dann mit der Darstellung beauftragt wird. Dieses Entwurfsmuster ähnelt sehr dem „Front Controller“, betrachtet die Dinge aber aus einer anderen Sichtweise. „Front Controller“ bedeutet zunächst erst einmal nur, dass es einen zentralen Zugangspunkt zu den Funktionen der Webapplikation gibt. „Service to Worker“ legt jetzt mehr Wert auf die durchzuführenden Aktionen und auch deren Reihenfolge. „Service to Worker“ und „Dispatcher View“ (aus [13]) können beide als „Front Controller“ angesehen werden, erledigen ihre Arbeit aber auf unterschiedliche Art und Weise.

Facade: Eine Fassade bietet eine Einheitliche Schnittstelle zu einer Menge von Schnittstellen eines Subsystems. Ist in Webanwendungen oft realisiert, als eine abstrakte Schnittstelle, die die eigentliche Geschäftslogik der Anwendung für die Benutzung vereinfacht und auch austauschbar macht.

Command: Kapselt einen Befehl als ein Objekt, das von Klienten mit verschiedenen Parametern versehen werden kann. Ermöglicht auch Anfragen in eine Warteschlange zu stellen, ein Logbuch zu führen und Befehle rückgängig zu machen.

Model View Controller: MVC ist weniger ein Entwurfsmuster an sich, als ein Paradigma bei der Entwicklung von Benutzerschnittstellen. Es zielt darauf ab, die Darstellung eines Geschäftsmodells von der Anzeige (Sicht, View), in der es dargestellt wird zu entkoppeln, so dass Modell und View unabhängig voneinander ausgetauscht werden können. Dazu wird ein Controller-Objekt verwendet, das auch dafür sorgt, dass die Aktionen, die der Benutzer an der Anzeige ausführt, an die Geschäftslogik weitergegeben werden. Realisiert wird das Ganze häufig durch die Kombination geeigneter Entwurfsmuster wie Observer, Command, Adapter, Decorator, Chain of Responsibility.

2 Die Struts Architektur

Das Struts-Framework ist relativ klein, es beschränkt sich auf ein Minimum an Funktionalität, um die Entwicklung von Webapplikationen nach dem Model 2 Ansatz, einer Variation des Model View Controller -Paradigmas, möglichst einfach zu machen. Seine Leichtgewichtigkeit zeichnet sich auch dadurch ab, dass es sehr einfach einzusetzen ist, z.B. keine Änderungen am Webcontainer erfordert, und mit quasi allen gängigen Java-Web-Technologien zusammen eingesetzt werden kann. Das

Struts-Framework bietet in der aktuellen Version die Möglichkeit, die Webapplikation in getrennte Struts-Module einzuteilen, die jeweils eine eigene Konfiguration haben

2.1 Struts und der Modell 2 Ansatz

Modell 2[5] ist eine Übertragung des MVC-Modells auf Webanwendungen, die auf J2EE-Technologien[6] basieren. Der Model 2 Ansatz bietet vor allem für größere Webanwendungen eine sauberere Umsetzung des MVC-Paradigmas als das Model 1. Model 1 geht davon aus, dass eine Anfrage seitens des Clients direkt an die benötigte JSP-Seite gerichtet ist, die dann wiederum für das Ausführen der gewünschten Aktionen und die anschließende Darstellung der Ergebnisse zuständig ist. Im Model 2 dagegen gehen alle Anfragen, die Aktionen bewirken sollen, an ein Controller-Servlet, das dann die Geschäftslogik bereitstellt oder gar aktiviert und die zuständigen View-Komponenten selektiert. Der View-Teil, meist eine JSP-Seite, bekommt seine darzustellenden Informationen über eine JavaBean zur Verfügung gestellt. Fig. 1 veranschaulicht das Modell und die Ablaufreihenfolge.

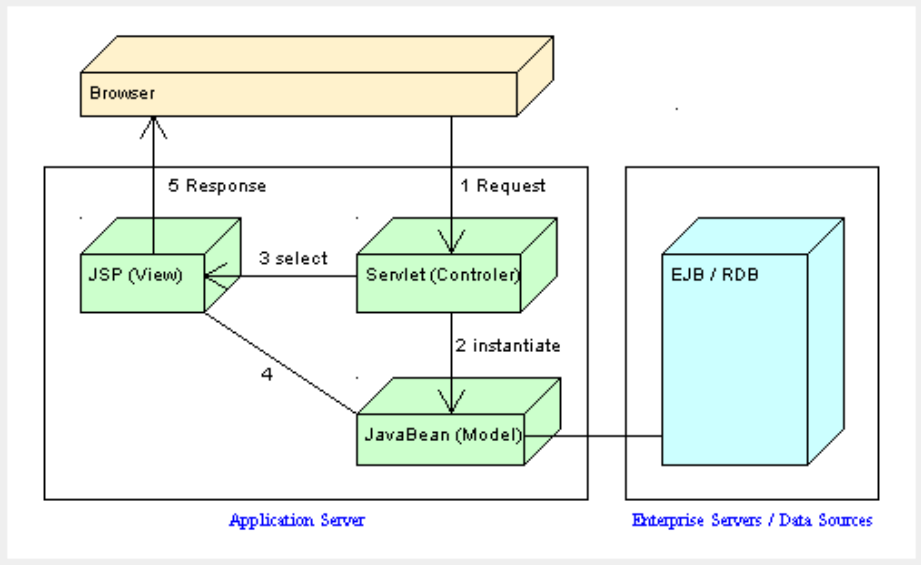

Fig. 1. Das Modell 2

Die Struts-Architektur implementiert im Wesentlichen den Controller-Teil dieses Modells bis auf die Teile, die von Anwendung zu Anwendung unterschiedlich sein müssen, um die eigentlichen Aktionen auszuführen. Struts bietet auch reichlich Unterstützung für die Implementierung der View- und Modellkomponenten. Zentraler

5 Weiteres zu Modell 2 finden Sie in [15] und [16]

6 Java 2 Enterprise Edition beinhaltet unter anderem Java Servlets, JSP und EJB Technologien.

Zugangspunkt zur Webapplikation bietet bei Struts ein Controller-Servlet das `ActionServlet` heißt.

2.2 Struts Controller-Komponenten

Das ActionServlet wird über Browserrequests angesprochen, die einem bestimmten Schema genügen. Dafür wird in der Webapplikation etwa eine bestimmte URL-Klasse auf das ActionServlet gemappt, sodass z.B. alle URLs, die auf „.do“ enden oder deren Lokalteil mit „/do/“ beginnt, durch das ActionServlet bearbeitet werden. Erhält das Servlet so einen Request, so betrachtet es den Rest der URL und sucht in seiner Konfigurationsliste ein `ActionMapping` das zu dieser URL gehört. ActionMappings sind Tupel aus einem URL-Lokalteil, einer zugehörigen `Action` und noch weiteren Informationen, auf die z.T. später eingegangen wird. Actions sind vom Entwickler zu implementierende Objekte, die Nachrichten an die Geschäftslogik schicken, Ergebnisse berechnen und endgültig entscheiden, an welche View-Komponente die Ausgabe delegiert wird. Nach der Erzeugung einer eventuell zur Action gehörenden FormBean wird die Kontrolle zunächst an die Action übergeben und anschließend an die von der Action bestimmte View-Komponente, die dann für die Bedienung der Anfrage sorgt. Das Struts ActionServlet ist somit eine geeignete Implementierung des „Service to Worker“ Entwurfmusters.

ActionMappings und ActionForwards In einem ActionMapping wird festgelegt, wie URLs auf Actions abgebildet werden. Zur Veranschaulichung betrachten wir die XML-Definition eines ActionMapping in der Strutskonfiguratinsdatei wie sie in unserer Beispielanwendung stehen könnte:

```
<action
    path="/addUrlDocument"
type="de.tkuhn.searchwebapp.action.AddUrlDocumentAction"
    name="addUrlDocumentForm"
    scope="request"
    validate="true"
    input="/index.jsp">

    <forward
        name="fail"
        path="/fail.jsp"
    />

</action>
```

Das Path-Attribut gibt an, welche URLs gemappt werden. In unserem Fall wäre das in etwa eine URL wie: `http://localhost:8080/search-webapp/addUrlDocument.do`. Mit Type wird bestimmt, welches die zu diesem Mapping gehörende Action-Klasse ist, die später die Nachfrage behandelt. Mit Name, Scope und Validate wird festgelegt, ob Struts für dieses Mapping eine ActionFormBean verwenden soll, die die Benutzereingaben aufnimmt, in welchem Kontext diese abgelegt werden soll und ob

die Eingaben auf Richtigkeit überprüft werden sollen. Unter Input ist eine JSP-Seite angegeben, von der aus auf diese URL verwiesen wird. Wird bei der Überprüfung der Benutzereingaben festgestellt, dass diese nicht in Ordnung sind, so wird sofort diese Seite bemüht, um die aufgetretenen Fehler anzuzeigen und den Benutzer zu einer Korrektur aufzufordern, ohne dass die Action selbst überhaupt aufgerufen werden muss.

Zum Schluss des Mappings wird noch ein sogenanntes `ActionForward` definiert. Diese Forwards sind ein wichtiger Bestandteil des Struts-Frameworks, um Actions unabhängig von den tatsächlichen View-Elementen zu halten. Ein ActionForward verknüpft einen logischen Namen in diesem Beispiel „fail“ mit einer View-Instanz. Das Forward hat, so definiert, nur in Verbindung mit diesem ActionMapping Gültigkeit. Schlägt man in der Action `AddUrlDocumentAction` also nach einem Forward namens „fail“ nach, erhält man als Ergebnis „/fail.jsp“, falls die Action über dieses Mapping aktiviert wurde. Damit wird deutlich, dass man durchaus über verschiedene URLs die gleiche Action aktivieren kann, aber mit verschiedenen Forwards dann unterschiedliche Darstellungen der Ergebnisdaten erhält. Eine Anwendung dafür wäre z.B., Informationen aus einer Anwendung einmal für einen Browser und einmal für ein Wap-Gerät darzustellen. Über die mappinggebundene Definition hinaus lassen sich auch noch globale ActionForwards definieren, die in allen Actions immer sichtbar sind, wenn sie nicht von einem lokalen Forward überlagert werden.

Actions Eine Action ist der Teil von Struts, durch den letztendlich die Arbeit am Geschäftsmodell der Webanwendung veranlasst wird. Wie der Name der Klasse schon sagt, handelt es sich bei Actions also um aktive Vorgänge, sodass sie ideal geeignet sind, um die Usecases der Applikation abzubilden. Struts Actions entsprechen nur teilweise dem „Command“ Entwurfsmuster, da sie weder in Warteschlangen organisiert werden können, noch die Fähigkeiten besitzen, direkt rückgängig gemacht werden zu können. Actions werden vom Programmierer dadurch erstellt, dass er von `org.apache.struts.action.Action` eine Klasse ableitet und die Methode

```
ActionForward execute(ActionMapping mapping,
                      ActionForm form,
                      HttpServletRequest request,
                      HttpServletResponse response)
```

implementiert. Neben dem Mapping, das für die Weiterleitung verantwortlich war, und den typischen Http-Parametern erhält man auch noch die `ActionForm`, die die Benutzereingabedaten enthält. Innerhalb der execute-Methode kann man dann die notwendigen Geschäftslogikaufgaben direkt berechnen oder alternativ, was auch von den Struts-Entwicklern empfohlen wird, nochmals getrennte Geschäftslogikbeans dazu bemühen. Dies hat mehrere Vorteile: Zum Einen kann man dann auf einfache Weise auf die gleiche Geschäftslogik aus unterschiedlichen Actions zugreifen, zum Anderen hat man dadurch die Möglichkeit die Geschäftslogik mittels des Entwurfmusters „Facade“ zu kapseln und damit ohne großen Aufwand austauschbar zu machen. Als Rückgabe erwartet das ActionServlet ein ActionForward, welches man in der Regel vom ActionMapping bekommt, über die Methode `ActionForward findForward (String name)`, der man den in der Konfiguration festgelegten logischen Namen gibt. In seltenen Fällen, wird die Anfrage auch direkt von der Action direkt bedient, so

dass man hier auch `null` zurückgeben kann. Zur Kommunikation mit JSPs gibt es von einer Action aus im Wesentlichen zwei Möglichkeiten: Man kann Daten in Form von Javaobjekten, die man der Sicht zur Darstellung geben möchte, unter einem Schlüssel in den Kontext des Requests einhängen, wo die JSP-Seite sie dann später abrufen kann. Dies würde man mit dem Aufruf von `request.setAttribute ("BeispielSchlüssel", datenObjekt)` erreichen. Zur Datenübergabe gibt es noch zwei weitere Kontexte, die in kurz beschrieben sind. Darüber hinaus bietet Struts noch eine Funktion an, um sogenannte `ActionMessages` oder auch für noch schlimmere Fälle, die davon abgeleiteten `ActionErrors`, an eine JSP-Seite zu übergeben. Dabei handelt es sich um internationalisierte Textnachrichten, die über spezielle Tags in der JSP sichtbar gemacht werden können.

Um beim Schreiben von Actions Arbeit zu sparen gibt es von Struts einige vordefinierte Actions, die manche Standardfälle abdecken. Das sind u.a.

DispatchAction und LookupDispatchAction ermöglichen es, mehrere verwandte Actions in nur einer Klasse zusammenzufassen. Man wählt dann im ActionMapping aus, welche Methode aufgerufen werden soll. Damit kann man dann eine Art „Facade“ auf Actionebene realisieren und die Anzahl der Klassen weiter reduzieren.

ForwardAction und IncludeAction erleichtern durch Weiterleitung oder Inklusion von URLs die Verwendung von Struts zusammen mit anderen Frameworks.

SwitchAction schaltet von einem Modul der Webanwendung in ein anderes um. D.h. ab jetzt gilt die ActionServlet-Konfiguration des neuen Moduls mit eigenen ActionMappings und Forwards etc.

ActionForms sind JavaBeans, die mit ihren verschiedenen Properties die Benutzereingaben in die Webapplikation aufnehmen. Um dies zu erreichen, erzeugt das ActionServlet die ActionForm im entsprechenden Kontext, falls noch nicht vorhanden, und füllt ihre Properties mit den Werten der gleichnamigen Requestparameter der Http-Anfrage. Darüber hinaus besitzen ActionForms noch Methoden zum Zurücksetzen und zum Validieren der Werte. Bei letzterem überprüft die FormBean, ob die Inhalte ihrer Properties den Erwartungen entspricht, und liefert, falls nicht, als Ergebnis der Operation eine Reihe ActionErrors zurück. Diese sagen dem ActionServlet wiederum, dass etwas nicht in Ordnung war, so dass es dem Benutzer die Eingabeseite erneut anbieten kann. Dabei ist es über eine Struts Tag-Bibliothek sehr einfach möglich, in einer JSP-Seite die FormBean mit einem Html-Formular zu verknüpfen und die alten Eingabewerte und die Fehlermeldungen in die neue Eingabeaufforderung zu übernehmen. ActionForms sind also eine Art erweitertes „Value Object“, das zum Datenaustausch zwischen den View- und Controller-Komponenten eingesetzt wird.

Struts bietet noch weitere Unterstützung beim Schreiben solcher ActionForms. Schreibt man etliche FormBeans für viele Formulare, so fällt auf, dass man oft nur ganz wenige Properties pro Bean benötigt und diese auch noch den selben Typ besitzen. Dennoch muss man für jeden Fall, indem die Semantik der Bean nicht die selbe ist, eine neue Java-Klasse anlegen. Struts bietet hier einen wirkungsvollen Mechanismus, die massenweise Produktion solcher Klein-Beans einzudämmen. Es

bietet die Klasse `DynaActionForm`, die eine ActionForm definiert, deren Objekte dynamisch konfigurierbar sind. D.h. man kann ihnen zur Laufzeit mitteilen, welche Properties sie besitzen sollen und ob diese indexed oder mapped Properties sein sollen oder nicht. In der Regel wird dies jedoch in der Strutskonfiguration getan. Kann man auf die Formvalidierung verzichten hat man mit der `DynaActionForm` also eine leicht handhabbare universelle FormBean.

2.3 Sichtkomponenten

Am häufigsten mit Struts verwendete View-Komponenten sind Java Server Pages, wenn auch es mit Struts möglich ist, auch XML eventuell zusammen mit XSL zu verwenden. Selbstverständlich können auch beide kombiniert werden. Struts bietet verschiedene Hilfen bei der Erstellung von JSP-Seiten.

Internationalisierung (i18n) ist ein wichtiger Aspekt in den heutigen Applikationen, vor allem in einem universell verfügbaren Medium wie dem Internet. Struts trägt diesem Trend Rechnung, indem es die gängigen Java-i18n-Methoden von Java nutzt und diese für Webanwendungen verfügbar macht. Es bietet die Möglichkeiten, über `ResourceBundles` verfügbare internationalisierte Zeichenketten auszugeben und sogar Platzhalter in der Zeichenkette durch aktuelle Parameter zu ersetzen. Dies geschieht aus den Actions über ein `MessageResources` Objekt. Aus den FormBeans über die `ActionErrors` und für die JSP-Seiten stellt Struts ein eigenes „message“-Tag zur Verfügung. Die verwendeten `ResourceBundles` lassen sich über die Strutskonfiguration genau einstellen.

Html-Form-Interaktion. Die Eingabe von Informationen durch den Benutzer über Html-Formulare wird mit Struts zum Kinderspiel, verwendet man die Struts-Html-Tag-Bibliothek. In ihr findet man spezielle Tags für Html-Formulare und für alle gängigen Html-Input-Felder, die man, einfach versehen mit den Property-Namen einer `ActionForm`, in die JSP-Seite einfügt. Jetzt muss man nur noch im Submit-Tag angeben, welches `ActionMapping` verwendet werden soll und schon hat man in der ausgeführten Action alle Eingabewerte in der entsprechenden FormBean. Zur Überprüfung der Benutzereingaben kann man zum einen den bereits vorgestellten Mechanismus der ActionForms verwenden, zum anderen kann man auch zusätzlich einen clientseitigen auf JavaScript basierenden Validierer benutzen, der in seiner Überprüfungsfähigkeit für die einzelnen Properties über XML-Dateien konfiguriert werden kann.

Zusammengesetzte Sichten sind Sichten, die über Schablonen aus verschiedenen anderen Sichten gemäß dem GoF-Pattern „Composite“ oder spezieller „Composite View“ aus [13] zusammengesetzt sind. Hierfür bieten sich im Struts-Framework gleich zwei Bibliotheken an, eine komplexere und eine einfachere.

Die Template Taglib ist recht einfach gehalten und erlaubt das einfache Verschachteln von Templates und Html-Fragmenten. Sie ist aber zu Gunsten der anderen Bibliothek in Struts 1.1 bereits deprecated.

Die Tiles Bibliothek ist ein relativ komplexes, aber dennoch leicht einzusetzendes Framework zur Erzeugung bis ins Detail konfigurierbarer Sichten. Ein Sichtteil, der hier Tile heißt, kann auf unterschiedliche Art und Weise definiert werden. Tiles können auch voneinander erben und ergänzen. Man kann auch erreichen, dass die Webapplikation nur die Grobstruktur und die zur Verfügung stehenden Tiles festlegt und der Benutzer selbst festlegen kann, was er genau sehen will. Die Auswahl dargestellter Tiles kann man auch vom jeweiligen Http-Request abhängig machen, wenn man auf Benutzersessions, Benutzerrechte, Benutzer-Locale oder den Browsertyp eingehen will. Insgesamt bietet Tiles also genug Flexibilität, um auch anspruchsvolle Webanwendungen zu realisieren. Tiles ist außerdem voll kompatibel zur Template Taglib.

Tagbibliotheken, die in Struts enthalten sind, bieten Lösungen für viele JSP-Alltagsprobleme. Diese werden aber inzwischen auch schon zu einem großen Teil von der standardisierten JSTL[7] abgedeckt. Hier nicht aufgeführt sind die Template- und Tiles-Bibliotheken.

Bean Tags: Erzeugen von Beans aus Cookies, Ressourcen, Requestparametern, Headerwerten oder aus Struts-Konfigurationsobjekten. Ausgabe von Bean-Properties und internationalisierten Nachrichten.

Html Tags: Viele Tags zur Interaktion mit Struts ActionForms. Anzeigen von (Fehler) Meldungen aus Struts Actions. Link und URL Tags für einfaches Benutzersessionmanagement. Die Tags können sowohl für Html-, als auch für XHtml-Generierung verwendet werden.

Logic Tags: Tags zum Vergleichen, Prüfen auf (Un-)Gleichheit, Vorhandensein etc. verschiedener Werte und davon abhängiges Generieren von Ausgaben. Iterieren über eine Menge von Elementen.

Nested Tags: Weiten die anderen Struts Tags auf verschachtelte Beaninstanzen aus.

2.4 Modellkomponenten

Im Modellbereich des MVC-Paradigmas bietet Struts wenig über die von Java Servlets hinausgehende Unterstützung für bestimmte Techniken und Technologien. Es macht aber auch keine Einschränkungen. Empfohlen wird von den Struts-Entwicklern jedoch, für jeden Bereich von MVC eigene JavaBeans zu verwenden, um eine

7 Die JSP Standard Tag Library standardisiert viele JSP-Tags für häufige Anwendungen. Mehr in [18]

möglichst gute Wiederverwendbarkeit der einzelnen Komponenten zu gewährleisten, so auch für den Modellbereich.

JavaBeans und ihr Kontext: In JSP-Webanwendungen und damit auch mit dem Struts-Framework können JavaBeans in mehreren verschiedenen Kontexten zur späteren Verwendung abgelegt werden.

- **Page** Beans im Pagekontext sind nur für die aktuelle JSP-Seite sichtbar.
- **Request** Requestbeans haben Gültigkeit für die Dauer des aktuellen Http-Requests und sind damit zur Kommunikation zwischen View- und Controller-Bereich geeignet.
- **Session** Beans im Sessionkontext gelten für die aktuelle Benutzersession über mehrere Requests hinweg.
- **Application** Applicationbeans bestehen, bis die Webanwendung als ganze beendet wird.

ActionForm-Beans Struts-ActionForm-Beans können für den Request- und den Sessionkontext definiert werden, sollten aber immer als Controller-Objekte verstanden werden und sie sollten keine Geschäftslogik enthalten.

Systemzustandbeans sind Beans, die den aktuellen Zustand der Anwendung widerspiegeln. Das wären z.B. Informationen über einen angemeldeten Benutzer und dessen Einkaufswageninhalt. Diese Beans befinden sich oft im Sessionkontext und werden auch oft in Datenbanken o.ä. persistent gemacht.

Geschäftslogikbeans kapseln die Geschäftslogik und auch oft Datenpersistenz einer Anwendung von den Controller- und View-Komponenten ab und werden meist im Application-Kontext der Anwendung abgelegt. Die geschäftslogischen Vorgänge in solchen Beans können einfach sein oder sich auch über komplexe Vorgänge auf EJBs und Datenbanken erstrecken.

Plugins Bei Struts ist ein Plugin einfach eine Klasse, die dem `PlugIn` Interface genügt, welches lediglich eine Methode zur Initialisierung und eine zum Beenden des Plugins definiert. Pluginklassen können verwendet werden, um leicht Geschäftslogikbausteine beim Start der Webanwendung verfügbar zu machen und beim Beenden wieder abzubauen. Bei der Initialisierung bekommt man eine Referenz auf das ActionServlet und etwaige Konfigurationsparameter für das Plugin mitgeliefert, die in der Strutskonfiguration eingetragen werden können, sodass man Plugins, wenn sie richtig geschrieben sind, auf einfache Weise mit verschiedenen Einstellungen für unterschiedliche Webapplikationen verwenden kann.

DataSources Zur Unterstützung von Geschäftslogik, die auf externe Datenquellen, wie z.B. relationale Datenbanken angewiesen ist, kann man durch einfache Einträge in der Strutskonfiguration Datenquellen zur Verwendung in den Actions einblenden, die dem standardisierten Interface `javax.sql.DataSource` entsprechen. Dabei wird für diese Datenbank sogar über eine Bibliothek aus dem Jakarta-Commons-Projekt ein Verbindungspool aktiviert. Die Struts Entwickleranleitung empfiehlt allerdings für den Fall, dass ein mächtigeres Werkzeug für die Datenbankabstrahierung und das Verbindungsmanagement, wie z.B. eine JNDI-Implementierung, zur Verfügung steht, lieber diese zu verwenden. Benutzt man also den Tomcat Webcontainer, so sind die Struts-DataSources hinfällig.

2.5 Struts zusammen mit anderen Technologien

Natürlich gibt es einige Technologien, die untrennbar mit Struts verbunden sind, weil sie Struts erst möglich machen, wie z.B. Java Servlets, aber wie sieht es mit Struts-fremden Technologien aus? Natürlich gibt es einige wenige, die sich nicht oder zumindest im Moment nicht sinnvoll zusammen mit Struts verwenden lassen; dazu zählen z.B. die typischen GUI-Toolkits wie AWT, JFC/Swing und SWT. Das liegt daran, dass eine Webanwendung kein eigenes GUI benötigt, sondern die Darstellung teils dem Browser des Benutzers überlässt. Ansonsten gibt es nur wenige Grenzen für eine gemeinsame Verwendung mit Struts: Man kann alles mit Struts kombinieren, was Java entweder direkt unterstützt oder aber Informationen über standardisierte Kommunikationsprotokolle austauscht, für die eine Implementierung in Java existiert. Falls nicht, bleibt wohl keine andere Möglichkeit als selbst eine Javaanbindung zu erstellen. Struts selbst macht jedoch keine Prämissen, was die Wahl betrifft, es unterstützt keine Technologie besonders, behindert aber auch keine.

Struts-Erweiterungen Trotz des relativ geringen Alters von Struts gibt es bereits etliche Erweiterungen, die Arbeit abnehmen können. Sie reichen von Tools zur Benutzerautorisierung über Workflowmanager, die Abläufe über mehrere Actions koordinieren, bis hinzu kompletten Frameworks, wie dem hier ebenfalls vorgestellten Expresso.

Struts-Entwicklung Auch zur Entwicklung von Anwendungen, die auf Struts basieren, gibt es bereits einige Hilfskomponenten. So zum Beispiel ein Plugin für die Entwicklungsplattform Eclipse, erhältlich über [19], ein StrutsTestCase, das beim Testen von Anwendungen mit dem JUnit-Framework aus [20] hilft, oder eine Anbindung an das Rational Rose UML-Modell.

3 Die Lucene Bibliothek

Lucene ist eine unter [14] frei erhältliche Volltextsuchmaschine, die über ein Java API verfügt und damit leicht in jede Java-Applikation integriert werden kann. Lucene

unterstützt das Einfügen und Löschen von „Dokumenten“ in einen Index und das Ausführen komplexer Suchausdrücke auf diesem Index.

3.1 Die Lucene „Dokumente“

Ein „Dokument” in Lucene entspricht nicht ganz der natürlichen Vorstellung von Dokument, wie etwa ein Word-Dokument oder eine www-Seite. Lucene definiert Dokumente vielmehr als eine Ansammlung von Feldern, wobei ein Feld mehrere Eigenschaften haben kann. Es hat zunächst einen Namen zur Identifizierung des Feldes und einen Text, der die eigentlichen Daten enthält. Darüber hinaus kann man noch angeben, ob die Daten eines Feldes indiziert werden sollen, um auf ihnen suchen zu können, ob sie vor dem Indizieren in Token zerlegt werden sollen und auch, ob sie bei der Suche als Ergebnis zurückgeliefert werden sollen. Diese Abstraktion ermöglicht es, nahezu beliebige Kombinationen von Character-Daten mit dem Lucene-Index zu verwenden.

In unserer Beispielanwendung ergeben sich damit für die Dokumente nur zwei Felder. Zum einen das Feld, das die eigentlichen Character-Daten aufnimmt und mit einem `java.io.Reader`-Objekt verknüpft wird, es wird sowohl indiziert als auch in Tokens zerlegt, aber nicht zur Rückgabe gespeichert, zum anderen ein Feld, das einen eindeutigen Dokumentnamen als `java.lang.String`-Objekt aufnimmt, das ausschließlich für die Rückgabe beim Suchen bestimmt ist und nicht indiziert werden soll.

3.2 Das Indizieren

Um dem Lucene-Index neue Daten hinzuzufügen, erzeugt man zunächst ein Lucene-Dokument, öffnet den Index im Schreibmodus und übergibt dann das Dokument zusammen mit einem `org.apache.lucene.analysis.Analyzer` dem Index. Der Analyzer ist eine Möglichkeit, die Lucene bietet um universell einsetzbar zu sein. Er besteht im Wesentlichen aus einem `Tokenizer` und beliebig vielen `TokenFilter`. Der Tokenizer ist dafür zuständig, Text aus einem `Reader` in Einzelworte, sogenannte Token, zu zerlegen. Ein TokenFilter bekommt einen Token-Strom als Eingabe und gibt wieder einen neuen Token-Strom aus. Dadurch können verschiedene Umwandlungseffekte realisiert werden, wie z.B. das Auslassen von Stopp-Worten oder das Reduzieren von Wörtern auf ihren Wortstamm. Analyzer, Tokenizer und TokenFilter werden für Deutsch, Russisch und Englisch mitgeliefert, können aber auch für die meisten mitteleuropäischen Sprachen verwendet werden. Darüber hinaus steht einem frei, eigene Versionen zu implementieren.

3.3 Suchen im und Bearbeiten des Index

Öffnet man den Index im Lesemodus, kann man Dokumente im Index abfragen. Das geht zwar, ist aber nicht allzu benutzerfreundlich, da man nur einzelne Dokumente für ihre Id-Nummer bekommt, die man in der Regel nicht vorher weiß. Es eignet sich also

allenfalls, um eine Liste aller Dokumente im Index zu erstellen. Ebenfalls für die Id-Nummer kann man dann auch Dokumente aus dem Index löschen. Was auf diesem Wege nicht geht, ist Dokumente mit einem bestimmten Feldwert zu bekommen. Aber dafür gibt es die komfortable Suchfunktion. Öffnet man den Index im Suchmodus kann man mit sehr komplexen Anfragen, die sich auf mehrere Felder erstrecken können und neben einfachen booleschen Verknüpfungen auch Wildcards und Suchheuristiken zulassen, nach Treffern suchen. Dabei gibt es u.a. das Feature, mit maximaler Editierdistanz zu suchen oder auch alle Werte in einem abgegrenzten Bereich zuzulassen. Außerdem liefert Lucene zusammen mit den Treffern jeweils einen Relevanzwert, wie gut das Dokument auf die Anfrage zutrifft und sortiert die Ergebnisse gleich noch entsprechend. Die Relevanzbestimmung kann sogar durch die Anfrage beeinflusst werden, indem man für jeden Anfragenteil nochmals einen „Boostfactor“ angibt, der beim Suchen verrechnet wird.

4 Erfahrungen mit Struts

4.1 Die erste Webapplikation

Zu den Hintergründen Die in diesem Seminar von mir erstellte Webapplikation war für mich nicht nur die erste Webapplikation mit Struts, sondern auch die erste Webapplikation überhaupt. Also spiegeln meine Erfahrungen vor allem auch wider, ob und wie Struts für Anfänger in Webapplikationen geeignet ist. Allerdings muss ich eingestehen, dass ich mich schon länger mit der Entwicklung von Anwendungen außerhalb des Webkontextes beschäftigt habe, ebenso wie mit der Erstellung von Internetseiten, mit XML, XSL-T und der EJB-Technologie.

Die Anfänge (Struts kennenlernen) Da Struts im Wesentlichen auf Java Servlets und JSPs basiert, musste ich mich zunächst einmal darüber informieren, beschloss aber nach der Lektüre von eher grundlegenden Informationen über die J2EE-Techniken für Webanwendungen, mich zunächst wieder der Struts-Dokumentation zu widmen. Mit diesem Vorgehen bin ich dann auch zu einem Ergebnis gekommen. Von Java Servlets braucht man für einfache Applikationen mit Struts keine umfassende Kenntnis. Es genügt, wenn man über den Lebenslauf eines Servlets und die „Scopes“, in denen JavaBeans abgelegt werden können, Bescheid weiß. Mit XML- und HTML-Kenntnissen im Hinterkopf genügte es für den JSP-Teil, dass ich mich kurz mit dem JSP-Referenzblatt beschäftigte und nachschlug, was eine Tag Library ist. Anleitung und auch Beispiele für den Einsatz zusammen mit Struts findet man reichlich in der Struts-Distribution.

4.2 Die Entwicklung am Beispiel

Entwurfsentscheidungen Jetzt da ich mich mit Struts nun ein wenig auskannte, kam die Zeit zu entscheiden, welche Möglichkeiten von Struts ich ausprobieren wollte. Ich entschloss mich dazu, JSP in XML-Repräsentation, normale selbst erstellte Actions zu verwenden und auch Plugins und DataSources auszuprobieren. Es gab auch einige Dinge, die ich der Einfachheit halber zunächst nicht verwenden wollte. Das waren dynamische FormBeans, zusammengesetzte Sichten, den JavaScript-Validator und die ActionMessages. Am Anfang entschloss ich mich, auch bei dieser einfachen Webanwendung die Geschäftslogik teilweise in den Actions unterzubringen, ein Schritt, den ich später noch bereute.

Umsetzung des Entwurfs Zum Programmieren verwendete ich die Eclipse-Plattform zusammen mit einem Tomcat-Plugin und einem Plugin zur Struts-Entwicklung, das einen grafischen Editor für die Strutskonfiguration bot und Wizards zur Erzeugung von Templates für die gängigen Struts-Objekte.

Die Modellkomponenten Zunächst schrieb ich ganz unabhängig von der Webapplikation ein Modul zur Ansteuerung des Lucene Index mit einer für unsere Anwendung passende Schnittstelle. Dazu ein passendes `PlugIn`, um den Index in die Anwendung einzubinden. Dabei wollte ich auch die Indexposition auf der Festplatte über die Pluginkonfiguration vorgeben lassen, was aber nicht funktionierte, weil ich in den Konfigurationsdaten den Schlüssel nicht finden konnte. Dies konnte vielleicht daran gelegen haben, dass ich mit einer Entwicklungsversion von Struts arbeitete und die Plugins erst gerade in Struts eingefügt werden.

Die erste JSP und Action Zunächst wollte ich, dass man Dokumente in den Index einfügen kann und begann parallel eine Action, ActionForm und auch eine JSP für die Eingabe zu schreiben. Wie zu erwarten, war es ein Leichtes, eine Benutzerinteraktion zu erstellen; es waren nur wenige Zeilen Code zu schreiben. Man braucht in der JSP ein Formular, das man mit der Struts Html-Taglib erstellt:

```
<?xml version="1.0" encoding="UTF-8"?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
          xmlns:html="/WEB-INF/struts-html.tld"
          version="1.2">
  <html:html xhtml="true">
    <head>
      <title>Search WebApp</title>
    </head>
    <body>
      <h1>Search WebApp</h1
      <h2>Add remote document:</h2>
      <html:form action="/addUrlDocument">
        <p>Url : <html:text property="url"/></p>
        <html:submit value="Add document"/>
      </html:form>
```

```
      <p><html:errors property="url"/></p>
    </body>
  </html:html>
</jsp:root>
```

Dazu benötigt man noch eine FormBean, die die entsprechenden Properties, hier lediglich „url“ besitzt, und kann noch gleich die automatische Überprüfung für Eingabedaten mit einbeziehen. Sie überprüft hier, ob der eingegebene String tatsächlich einer gültigen URL entspricht:

```
package de.tkuhn.searchwebapp.form;
import java.net.URL;
import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.*;
public class AddUrlDocumentForm extends ActionForm {

    private String url;

    public ActionErrors validate(ActionMapping mapping,
                         HttpServletRequest request) {

        ActionErrors errors = new ActionErrors();
        try {
            new URL(url);
        } catch (MalformedURLException e) {
            errors.add("url", new ActionError
                       ("AddUrlDocumentForm.badUrl"));
        }
        return errors;
    }

    public String getUrl() { return url; }

    public void setUrl(String url) { this.url = url; }
}
```

Der im catch-Block erzeugte `ActionError` erhält als Parameter einen Schlüssel, der auf eine Nachricht in den internationalisierten Application-Ressourcen zeigt. Diese Nachricht wird dann in der Eingabe-JSP-Seite anstelle von `<html:errors property="url"/>` ausgegeben.

Mit einer Action konnte ich dann, nachdem ich das `ActionMapping` aus in die Konfiguration eingetragen und Tomcat neu gestartet hatte, ohne weiteres auf die Eingaben zugreifen:

```
package de.tkuhn.searchwebapp.action;
import java.net.MalformedURLException;
import javax.servlet.http.*;
import org.apache.struts.action.*;
import de.tkuhn.searchwebapp.form.AddUrlDocumentForm;

public class AddUrlDocumentAction extends Action {
```

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

      try {
        FileIndex index = (FileIndex)getServlet()
              .getServletContext().getAttribute("index");
        if (index == null) {
          request.setAttribute("message", "No Index! ");
          return mapping.findForward("fail");
        }
        AddUrlDocumentForm addForm =
                (AddUrlDocumentForm)form;
        index.addUrlDocument(addForm.getUrl());

        // Datei hier in die Datenbank eintragen!

        request.setAttribute("message","File added
                                        successfully.");
      } catch (Exception e) {
        request.setAttribute("message","File could not be
                                          added: \n" + e);
        getServlet().log("File could not be added.",e);
      }
      return mapping.findForward("start");
    }
  }
```

Als dies soweit funktionierte, fügte ich zur Action den Code für den Zugriff auf die Datenbank hinzu. Ist die `DataSource` in der Strutskonfiguration eingetragen, so bekommt man sie in der Action mittels `getDataSource(request)` und kann dann Verbindungen öffnen. Aber bereits beim Schreiben der zweiten Action `addFileDocumentAction` musste ich mich umentscheiden und auch den Datenbankcode in eine eigene BusinessLogicBean auslagern, da die Actions schon sehr unübersichtlich wurden und ich den Code in den Actions nur schlecht wiederverwenden konnte. Ich erstellte also das Objekt `FileDatabase` für die Datenbankhandhabung und noch ein drittes Objekt, das den FileIndex und die FileDatabase verwendet und von nun an als einziges von den Actions verwendet werden sollte. Dabei warf ich auch die Struts DataSources über Bord, die ja von der Action aus an die Geschäftslogik hätten übergeben werden müssen, und konfigurierte die Datenbank direkt über JNDI. Das hätte man auch besser machen können, indem man zumindest für den JNDI-Namen den Umweg über einen Parameter des Struts-Plugins geht, so sollte es aber zunächst einmal reichen.

Ein schleierhaftes Problem war das nächste, womit ich nach dieser Refaktorisierung zu kämpfen hatte. Beim Aufruf einer Action bekam ich von Tomcat nur die Fehlermeldung „Servlet action unavailable“ zu sehen, keine weiteren Informationen.

Nach einer längeren Suche stieß ich auf das Logfile der Applikation, darin fand ich dann auch den Grund: Mein Code hatte während der Initialisierung des Plugins eine `SQLException` geworfen und das ActionServlet konnte deshalb nicht vollständig geladen werden.

Zügiges Vorankommen Ich fuhr anschließend fort, die weiteren Usecases unseres Beispielentwurfes auf weitere Actions abzubilden und kam so recht flott zum Ziel und damit zu einer gut funktionierenden Webanwendung. Die Actions selbst prüfen jetzt immer als erstes, ob der Index vorhanden ist und geben eine Fehlermeldung zurück, falls nicht. So etwas könnte man in Zukunft vielleicht noch in Struts integrieren, dass man in der Konfiguration eine Art Zustandsprüfobjekt angeben kann, das dann vor jedem Aktionsaufruf aktiviert wird. Als zweiten Schritt fragen fast alle meine Actions ihre Formproperties ab und delegieren die Verarbeitung an das BusinessLogic-Objekt. Hier wäre ein geeigneter Punkt um die `DispatchAction` zum Einsatz kommen zu lassen.

4.3 Vorteile von Struts

Leichtgewicht Struts bietet seine Funktionen durch eine kompakte Implementierung mit wenigen Komponenten an. Dadurch bleibt es überschaubar, ist in kurzer Zeit zu erlernen und bietet genug Freiraum für experimentierfreudige Entwickler.

Erweiterbar Man verliert durch Struts keinerlei Funktionalität gegenüber Java Servlets und kann quasi beliebige Bibliotheken mit Struts kombinieren. Diese Tatsache, hat auch dazu geführt, das es schon zahlreiche Erweiterungen gibt für Struts gibt.

Forminteraktion Das Hin- und Herschieben von Information von Objekten zu Html-Formularen ist wunderbar gelöst, auch über mehrere Html-Formulare hinweg. Hinzu kommt das solide Konzept für die Formvalidierung und die Rückmeldung durch internationalisierte (Fehler-)Nachrichten.

Zusammengesetzte Sichten Mit Tiles steht ein hervorragendes Framework für anspruchsvolle View-Komponenten zur Verfügung.

Konfiguration Die Einstellungen von Struts sind alle über eine zentrale XML-Datei auszuführen. Diese kann zwar mitunter recht lang werden, ist aber trotz umfangreicher Konfigurationsmöglichkeiten gut verständlich und leicht aufzufinden. Die Konfigurationsvielfalt ermöglicht es, den eigentlichen Javacode so zu schreiben, dass er leicht wiederverwendbar ist.

Interoperabilität Struts kann auch zusammen mit anderen Frameworks, die auf Java Servlets beruhen, eingesetzt werden, zum Beispiel mit Content-Management-Frameworks wie „Cocoon“.

4.4 Nachteile von Struts

Keine automatischen Sichten Struts selbst bietet keine Möglichkeit, Formulare, z.B. aus FormBeans, selbst zu generieren (wohl aber Struts-Entwicklungstools).

Wenig Unterstützung in Modellkomponenten Bei der Entwicklung der Geschäftslogik wünscht man sich manchmal mehr Unterstützung. Eine sinnvolle Anbindung von EJB oder JDO könnte helfen. Natürlich kann man diese dennoch gut mit Struts einsetzen. Auch Möglichkeiten für Transaktionsverwaltung und Java-Messaging gibt es ohne Erweiterungen nicht.

Entwicklung auf relativ niedriger Ebene Struts bietet keine Möglichkeit Entwurfsmodelle auf einer höheren Ebene, z.B. endliche Automaten, direkt umzusetzen. Diese müssten über Systemzustandbeans und entsprechende Actions per Hand realisiert werden.

Referenzen

1. Das Struts Framework http://jakarta.apache.org/struts/
2. Das Expresso Framework http://www.jcorporate.com/econtent/Content.do?state=template&template=2&resource=636&db=default
3. Der Java Community Process http://jcp.sun.com
4. Java Servlets http://java.sun.com/products/servlet/
5. Der Tomcat Servlet/JSP Container http://jakarta.apache.org/tomcat/
6. Der Jetty Servlet/JSP Contaienr http://jetty.mortbay.org/jetty/
7. Java Server Pages http://java.sun.com/products/jsp/
8. Enterprise Java Beans http://java.sun.com/products/ejb/
9. Java Management Extension http://java.sun.com/products/JavaManagement/
10. Simple Object Access Protocol http://www.w3.org/2000/xp/Group/
11. Java Mail API http://java.sun.com/products/javamail/
12. Entwurfsmuster, E.Gamma R.Helm R.Johnson J.Vlissides, Addison-Wesley, ISBN 3-8273-1862-9
13. J2EE Patterns, A.Bien, Addison-Wesley, ISBN 382731903X
14. Jakarta Lucene http://jakarta.apache.org/lucene/
15. Understanding JavaServer Pages Model 2 architecture http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html
16. Web-Tier Application Framework Design http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/web-tier/web-tier5.html

17. Core J2EE Pattern Catalog
http://java.sun.com/blueprints/corej2eepatterns/Patterns/index.html
18. Die JSP Standard Tag Library http://java.sun.com/products/jsp/jstl
19. Die Eclipse Entwicklungsplattform http://www.eclipse.org/
20. Das JUnit Test-Framework http://www.junit.org/