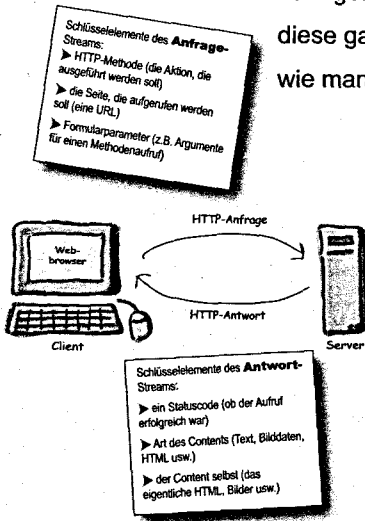


# 1 Warum Servlets & JSPs?

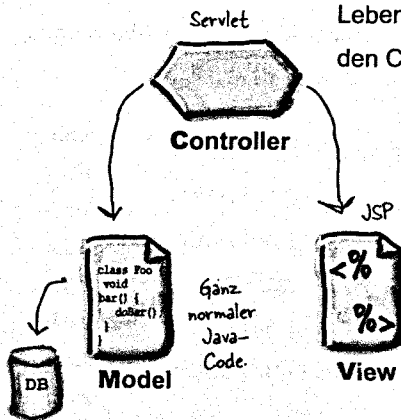
**Webanwendungen sind toll.** Wie viele GUI-Programme kennen Sie, die weltweit von Millionen von Anwendern benutzt werden? Als Entwickler von Webanwendungen können Sie sich aus dem Schwitzkasten der Deployment-Probleme befreien, mit denen alle Desktop-Programme zu tun haben, und Ihr Programm jedem verfügbar machen, der einen Browser hat. Aber Sie brauchen Servlets und JSPs. Weil diese ganz normalen statischen HTML-Seiten so, na ja, etwa von 1999 sind. Erfahren Sie, wie man von *Websites* zu *Webanwendungen* kommt.



Prüfungsanforderungen	2
Was Webserver und Clients machen und wie sie sich unterhalten	4
HTML-Schnellkurs	7
Was ist das HTTP-Protokoll?	10
Anatomie von GET- und POST-Anfragen und HTTP-Antworten	16
Webseiten über URLs auffinden	20
Webserver, statische Webseiten und CGI	24
Servlets entmystifiziert: Ein Servlet schreiben, verteilen und ausführen	30
JSPs: Als HTML mit Java bekannt gemacht wurde	34

# 2 Architektur von Webanwendungen

**Servlets brauchen Hilfe.** Beim Eintreffen einer Anfrage muss jemand das Servlet instantiiieren oder zumindest einen neuen Thread anlegen, um die Anfrage zu bearbeiten. Jemand muss die *doPost()*- oder *doGet()*-Methode des Servlets aufrufen. Jemand muss Anfrage und Antwort zum Servlet durchreichen. Jemand muss sich um den Lebenszyklus des Servlets kümmern und es versorgen. In diesem Kapitel werden wir uns den Container ansehen und einen ersten Blick auf das MVC-Muster werfen.



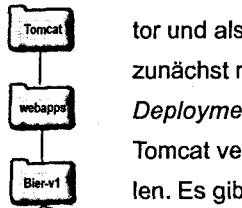
Prüfungsanforderungen	38
Was ein Container ist und was er Ihnen bietet	39
Wie der Code aussieht (und was ein Servlet macht)	44
Servlets benennen und über den DD URL zuordnen	46
Geschichte: Björns Dating-Site (und MVC-Einführung)	50
Das Model-View-Controller-Muster (MVC)	54
Ein »laufender« Deployment Descriptor (DD)	64
Wie sich J2EE in all das fügt	65

# 3 Mini-MVC-Kurs

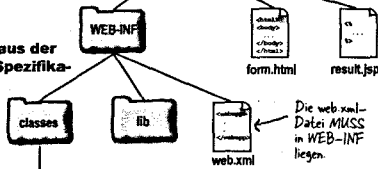
Legen Sie eine MVC-Webanwendung an und nehmen Sie sie in Betrieb. Es wird Zeit, die Ärmel hochzukrempeln und ein HTML-Formular, ein Controller-Servlet, ein Modell (eine ganz normale Java-Klasse), einen XML-Deployment Descriptor und als View eine JSP zu schreiben. Zeit zum Erstellen, Bereitstellen und Testen. Aber zunächst müssen Sie Ihre *Entwicklungsumgebung* einrichten. Und dann müssen Sie Ihre *Deployment-Umgebung* so gestalten, wie es die Servlet- und JSP-Spezifikationen sowie Tomcat verlangen. Es stimmt schon, es ist eine sehr kleine Anwendung, die wir hier erstellen. Es gibt aber so gut wie KEINE Anwendung, die zu klein für das MVC ist.

**Der Tomcat-spezifische Teil**

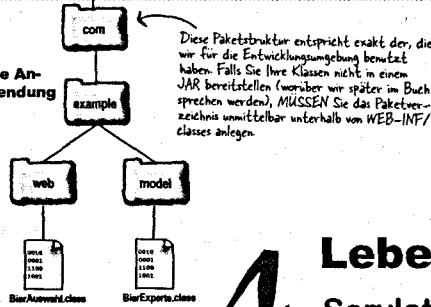
Dieser Verzeichnisname entspricht gleichzeitig dem Kontextwurzelverzeichnis (context root), das von Tomcat bewahrt wird, um eine URL aufzulösen. Wir werden dieses Konzept in allen Details im Deployment-Kapitel untersuchen.



**Der Teil aus der Servlet-Spezifikation**



**Die Anwendung**



Prüfungsanforderungen	68
Erstellen wir eine MVC-Anwendung – der erste Entwurf	69
Die Entwicklungs- und Deployment-Umgebungen aufbauen	72
Das HTML für die anfängliche Formulareseite erstellen und testen	75
Den Deployment Descriptor (DD) erstellen	77
Das Controller-Servlet erstellen, kompilieren, verteilen und testen	80
Die Model-Komponenten erstellen, kompilieren und testen	82
Den Controller das Model aufrufen lassen	83
Die View-Komponente (eine JSP) erstellen und verteilen	87
Das Controller-Servlet die JSP aufrufen lassen	88

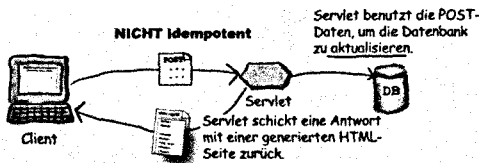
# 4 Leben als Servlet

Servlets brauchen Unterstützung. Der Job eines Servlets ist es, die *Anfrage* eines Clients entgegenzunehmen und eine *Antwort* zurückzugeben. Die Anfrage kann etwas Einfaches sein: »Zeig mir die Startseite!«, oder etwas Komplexes: »Mach mir die Abrechnung für meinen Einkaufskorb fertig!«. Die Anfrage transportiert wichtige Daten, und der Code Ihres Servlets muss wissen, wo er sie *suchen* und was er damit *machen* soll. Und Ihr Servlet-Code muss wissen, wie man eine *Antwort* versendet. Oder nicht ...

Aber Idempotenz ist doch nichts, dessen man sich schämen muss ...

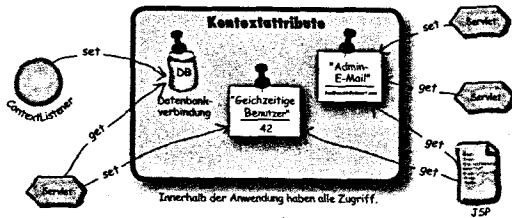


Prüfungsanforderungen	94
Das Leben eines Servlets im Container	95
Servlet-Initialisierung und Threads	101
Die WAHRE Aufgabe eines Servlets: GET- und POST-Anfragen	105
Die Geschichte der Anfrage, die nicht idempotent war	112
Was festlegt, ob Sie eine GET- oder eine POST-Anfrage erhalten	117
Parameter senden und verwenden	119
Das war die Anfrage ... und wie sieht die Antwort aus?	126
Sie können Antwort-Header setzen und hinzufügen	133
Servlet-Umleitung vs. Anfrageweiterleitung	136
Zusammenfassung: HttpServletResponse	140



# 5 Leben als Webanwendung

**Kein Servlet ist unabhängig.** In heutigen, modernen Webanwendungen arbeiten viele Komponenten zusammen, um ein gemeinsames Ziel zu erreichen. Man hat Models, Controller und Views. Man hat Parameter und Attribute. Und Hilfsklassen. Aber wie verbinden Sie diese Teile zu einem Ganzen? Wie bringen Sie Ihre Komponenten dazu, Informationen *auszutauschen*? Wie *verbergen* Sie Informationen? *Wie machen Sie Informationen Thread-sicher*? Von Ihren Antworten könnte Ihr Leben abhängen.

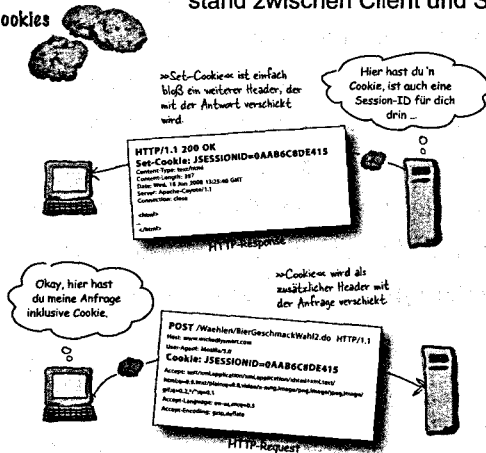


Prüfungsanforderungen	148
Initialisierungsparameter und ServletConfig greifen ein	149
Wie eine JSP Servlet-Initialisierungsparameter erhält	155
Kontextinitialisierungsparameter retten uns	157
ServletConfig und ServletContext im Vergleich	159
Sie möchte einen ServletContextListener	166
Tutorial: Ein einfacher ServletContextListener	168
Listener kompilieren, verteilen und testen	176
Die ganze Geschichte: ServletContextListener im Überblick	178
Acht Listener: Sie sind nicht nur für Kontext-Events gedacht ...	180
Was genau ist ein Attribut?	185
Die Attribute-API und die dunkle Seite von Attributen	189
Der Kontextgültigkeitsbereich ist nicht Thread-sicher!	192
Das Problem in Zeitlupe ...	193
Probieren Sie es mit Synchronisierung	195
Sind Sitzungsattribute Thread-sicher?	198
Das SingleThreadModel	201
Nur Anfrageattribute und lokale Variablen sind Thread-sicher!	204
Anfrageattribute und Anfrageweiterleitung	205

# 6 Kommunikation mit Gedächtnis

Webserver besitzen kein Langzeitgedächtnis. Sobald sie Ihnen eine Antwort geschickt haben, vergessen sie, wer Sie sind. Wenn Sie das nächste Mal eine Anfrage schicken, werden Sie schon nicht mehr wiedererkannt. Mit anderen Worten, sie erinnern sich nicht daran, was Sie in der Vergangenheit angefordert haben, und sie erinnern sich nicht daran, was sie Ihnen als Antwort zurückgeschickt haben. Manchmal muss aber über *mehrmalige Anfragen hinweg* der Kommunikationszustand zwischen Client und Server festgehalten werden.

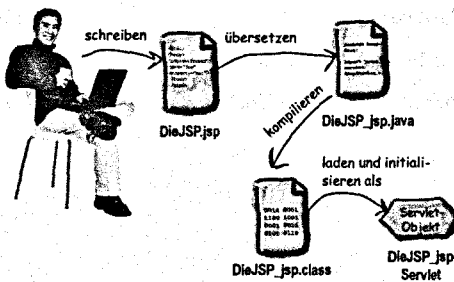
Cookies



Prüfungsanforderungen	224
Es sollte eine Unterhaltung sein (darüber, wie Sitzungen funktionieren)	226
Sitzungs-IDs, Cookies und weitere Sitzungsgrundlagen	231
URL-Rewriting: etwas, auf das man zurückgreifen kann	237
Wenn Sitzungen veralten; faule Sitzungen loswerden	241
Kann ich Cookies auch für andere Zwecke einsetzen?	250
Wichtige Meilensteine einer HttpSession	254
Vergessen Sie HttpSessionBindingListener nicht	256
Sitzungsmigration	257
Listener-Beispiele	261

# 7 Wie es ist, eine JSP zu sein

Eine JSP wird zu einem Servlet. Ein Servlet, das Sie nicht erstellen. Der Container sieht sich Ihre JSP an, übersetzt sie in Java-Quellcode und kompiliert diesen zu einer vollständigen Java-Servlet-Klasse. Aber Sie müssen wissen, was passiert, wenn der Code, den Sie in der JSP geschrieben haben, in Java-Code umgewandelt wird. Sie *können* in Ihre JSP Java-Code schreiben, aber sollten Sie das auch? Und was *schreiben* Sie, wenn Sie keinen Java-Code verwenden? Wie erfolgt die *Übersetzung* in Java-Code? In diesem Kapitel werden wir uns sechs verschiedene Arten von JSP-Elementen ansehen – die alle einen eigenen Zweck und, auch das, eine *eigene Syntax* haben. Sie lernen wie, warum und was Sie in Ihre JSP schreiben. Und Sie erfahren, was Sie in Ihre JSP *nicht* schreiben sollten.



Prüfungsanforderungen	282
Mit »out« und einer Page-Direktive eine einfache JSP erstellen	283
JSP-Ausdrücke, Variablen und Deklarationen	288
Das zu einer JSP generierte Servlet	296
Die Variable out ist nicht das einzige implizite Objekt ...	298
Der Lebenszyklus und die Initialisierung einer JSP	306
Etwas mehr zu den drei Direktiven	314
Scriptlets sollen schädlich sein? – Hier ist EL	317
Moment ... etwas fehlt noch: Aktionen	323

# 8 Skriptfreie Seiten

**Befreien Sie sich vom Scripting.** Können Ihre Webdesigner tatsächlich Java? Erwarten sie von den für den Server verantwortlichen Java-Programmierern, dass sie, hmm, sagen wir, Grafiker sind? Und selbst wenn Sie allein das gesamte Team bilden ... wollen Sie in Ihren JSPs wirklich einen Haufen kleiner Fetzen Java-Code haben? Schon mal etwas von »Wartungsabtraum« gehört? Skriptfreie Seiten zu schreiben, ist nicht bloß *möglich*. Mit der neuen JSP-2.0-Spezifikation ist es, dank der neuen Expression Language (EL), auch viel *einfacher* und flexibler geworden. EL ist an JavaScript und XPath angelehnt und sorgt dafür, dass Webdesigner sich gleich zu Hause fühlen – und auch Sie werden EL mögen (wenn Sie sich einmal daran gewöhnt haben). Aber es gibt einige Fallen ... EL *sieht* aus wie Java, ist es aber nicht. Manchmal verhält sich EL anders als die gleiche Syntax in Java. Geben Sie also acht!

Erwartet nicht, dass ICH eure überflüssigen Start- und End-Tags entferne.



① Die Kopfzeile (=Header.jsp)

```
 <br>
<em><strong>Wir wissen, wie man SOAP
weniger nervend macht.</strong></em> <br>
```

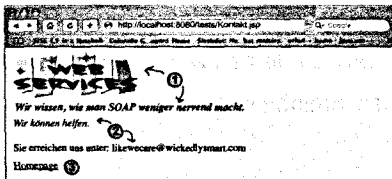
② Kontakt.jsp

```
<html><body>
<% include file="Header.jsp"% > <br>
<em>Wir können helfen.</em> <br><br>
Sie erreichen uns unter: <%=initParam.hauptEmail%>
<% include file="Footer.html"% >
</body></html>
```

Beachten Sie, dass von alle HTML- und BODY-Tags aus der eingeschlossenen Datei entfernt haben

③ Die Fußzeile (=Footer.html)

```
<a href="index.html">Homepage</a>
```



Hinweis: Das Entfernen der Start- und End-Tags gilt für BEIDE Include-Mechanismen =>jsp-include und die include-Direktive

Prüfungsanforderungen	344
Wenn Attribute <i>Beans</i> sind	345
Standardaktionen: useBean, getProperty, setProperty	349
Kann man polymorphe Bean-Referenzen machen?	354
Das <i>param</i> -Attribut schafft Abhilfe	360
Properties konvertieren	363
Expression Language (EL) rettet uns!	368
Mit dem Punktoperator (.) auf Properties und Map-Werte zugreifen	370
[ ] bietet mehr Möglichkeiten (Listen, Arrays ...)	372
Weitere Einzelheiten zum Punktoperator und [ ]	376
Die impliziten EL-Objekte	385
EL-Funktion und der Umgang mit »null«	392
Wiederverwendbare Teile – zwei Arten von »Include«	402
Die <jsp:forward />-Standardaktion	416
Sie kennt JSTL-Tags noch nicht (eine Vorschau)	417
Zusammenfassung von Standardaktionen und Includes	417

# 9 Benutzerdefinierte Tags sind mächtig

## Manchmal brauchen Sie mehr als EL und Standardaktionen.

Was ist, wenn Sie die Daten in einem Array durchlaufen und zeilenweise in einer HTML-Tabelle anzeigen wollen? Sie wissen, dass Sie das mit einer for-Schleife in einem Scriptlet in zwei Sekunden schreiben könnten. Aber Sie versuchen ja, das Scripting loszuwerden. Kein Problem. Wenn EL und Standardaktionen nicht ausreichen, können Sie *benutzerdefinierte Tags* oder *Custom-Tags* verwenden. Sie sind in JSPs genauso einfach zu verwenden wie Standardaktionen. Noch besser ist, dass schon ein ganzer Haufen für Sie geschrieben und zur JSP Standard Tag Library (JSTL) gebündelt wurde. In *diesem* Kapitel werden Sie lernen, wie man benutzerdefinierte Tags *verwendet*, im nächsten werden wir unsere eigenen erstellen.

```

<table>
  <c:forEach var="listElement" items="${filme}" >
    <c:forEach var="film" items="${listElement}" >
      <tr>
        <td>${film}</td>
      </tr>
    </c:forEach>
  </c:forEach>
</table>

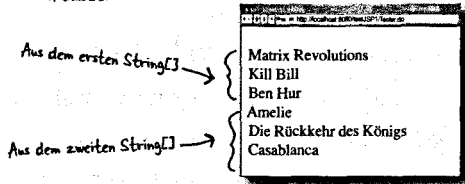
```

Das ArrayList-Anfrageattribut

Äußere Schleife

Innere Schleife

Eins der String-Arrays, das dem var-Attribut der äußeren Schleife zugewiesen wurde.



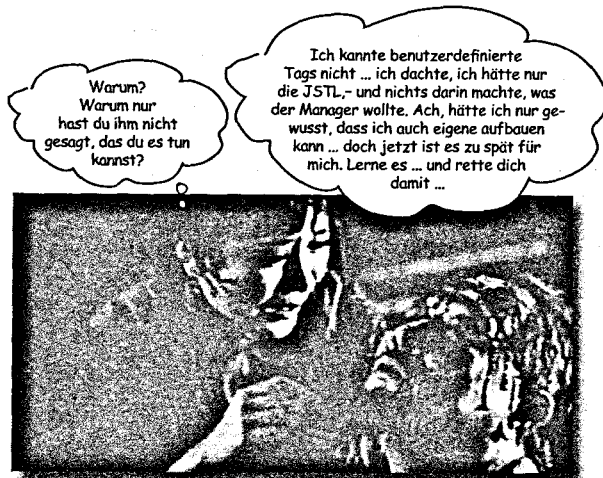
Prüfungsanforderungen	440
Schleifen ohne Scripting: <c:forEach>	446
Bedingter Ablauf mit <c:if> und <c:choose>	451
Die Tags <c:set> und <c:remove> einsetzen	455
<c:import> ist das <i>dritte</i> Verfahren zum Einschließen von Inhalten	460
Das Einzuschließende anpassen	462
Das Gleiche mit <c:param> machen	463
<c:url> erfüllt all Ihre Hyperlink-Bedürfnisse	465
Erstellen Sie Ihre eigenen Fehlerseiten	468
Das Tag <c:catch>. Wie try/catch ... <i>so ungefähr</i>	472
Was ist, wenn Sie ein Tag benötigen, das die JSTL nicht bietet?	475
Achten Sie auf <rtexprvalue>	480
Was im Body eines Tags stehen kann	482
Der Tag-Handler, der TLD und die JSP	483
Der Taglib-<uri> ist bloß ein Name, kein Ort	484
Wenn eine JSP mehrere Tag-Libraries verwendet	487

# 10

## Wenn selbst die JSTL nicht ausreicht ...

Manchmal reichen die JSTL und Standardaktionen nicht aus.

Wenn Sie etwas Angepasstes benötigen und nicht zum Scripting zurückwollen, können Sie Ihre *eigenen* Tag-Handler schreiben. Dann können Ihre Seitendesigner Ihr *Tag* in ihren Seiten verwenden, während die ganze *harte* Arbeit im Hintergrund in Ihrer Tag-Handler-Klasse abgewickelt wird. Aber es gilt drei verschiedene Wege, eigene Tag-Handler aufzubauen ... es gibt also, eine Menge zu lernen. Von den drei Wegen wurden zwei mit JSP 2.0 eingeführt, um Ihnen das Leben zu erleichtern (einfache Tags und Tag-Dateien).



Prüfungsanforderungen	500
Tag-Dateien: Wie Include, nur besser	502
Wo der Container nach Tag-Dateien sucht	509
Einfache Tag-Handler	513
Ein einfaches Tag mit Body	514
Was ist, wenn der Tag-Body einen Ausdruck verwendet?	519
Auch klassische Tag-Handler müssen Sie noch beherrschen	529
Ein sehr kleiner klassischer Tag-Handler	531
Rückgabewerte steuern den Lebenszyklus eines klassischen Tags	536
Mit IterationTag können Sie den Body wiederholen	537
Default-Rückgabewerte von TagSupport	539
Das Interface DynamicAttributes	556
BodyTag liefert Ihnen zwei neue Methoden	563
Was ist, wenn Tags kooperieren müssen?	567
Die PageContext-API bei Tag-Handlern einsetzen	577

# 11

## Ihre Webanwendung verteilen

Endlich ist Ihre Webanwendung bereit für das wahre Leben. Ihr Seiten sind ausgefeilt, Ihr Code getestet und abgestimmt und der Abgabetermin vor zwei Wochen verstrichen. Aber wo kommt der ganze Kram hin? So viele Verzeichnisse, so viele Regeln. Welche Namen geben Sie Ihren Verzeichnissen? Welche Namen erwartet der *Client*? Was fordert der Client tatsächlich an, und wie weiß der Container, wo er suchen muss?

Eine lokale Bean referenzieren

```
<ejb-local-ref>
  <ejb-ref-name>ejb/Kunde</ejb-ref-name>
  <ejb-ref-type>Entity</ejb-ref-type>
  <local-home>com.wickedlysmart.KundenHome</local-home>
  <local>com.wickedlysmart.Kunde</local>
</ejb-local-ref>
```

Der JNDI-Lookup-Name, den Sie in Code verwenden werden

Die meisten vollständig qualifizierte Namen der Lieferanten werden der Bean veröffentlicht werden

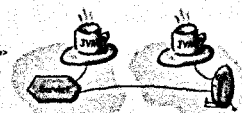


LOCAL bedeutet, dass der Client (hier das Servlet) und die Bean in der gleichen JVM laufen müssen.

Eine entfernte Bean referenzieren

```
<ejb-ref>
  <ejb-ref-name>ejb/LokalerKunde</ejb-ref-name>
  <ejb-ref-type>Entity</ejb-ref-type>
  <remote-home>com.wickedlysmart.KundenHome</remote-home>
  <remote>com.wickedlysmart.Kunde</remote>
</ejb-ref>
```

Optimale Metadaten für beide Tag und unter anderem <description> und <ejb-link>. Aber die meisten Sie für die Prüfung nicht kennen.



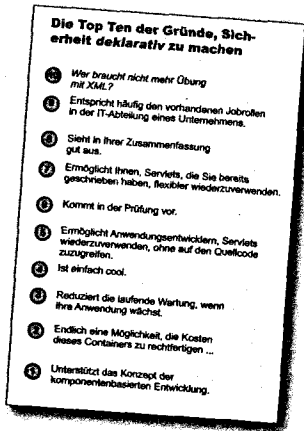
ENTFERNT (remote) bedeutet hier, dass der Client (hier das Servlet) und die Bean in unterschiedlichen JVMs laufen können (möglicherweise auch auf unterschiedlichen Rechnern)

Prüfungsanforderungen	602
Die Deployment-Schlüsselanforderung: Was gehört wohin?	603
WAR-Dateien	612
Wie Servlet-Mapping TATSÄCHLICH funktioniert	616
Im DD Willkommen-Dateien konfigurieren	622
Im DD Fehlerseiten konfigurieren	626
Im DD die Servlet-Initialisierung konfigurieren	628
Eine XML-konforme JSP: JSP Document	629

# 12

## Geschützt und sicher

Ihre Webanwendung ist in **Gefahr**. Ärger lungert in jedem Winkel des Netzwerks. Sie wollen nicht, dass die Bösen Buben die Transaktionen in Ihrem Onlinestore belauschen und Kreditkartennummern abfischen. Sie wollen nicht, dass die Bösen Buben Ihren Server überzeugen, dass sie »Besonders Gute Kunden Mit Dicken Rabatten« sind. Und Sie wollen nicht, dass *irgendwer* (gut ODER böse) sensible Angestelltendaten einsieht. Muss Tim aus dem Marketing wirklich wissen, dass Lisa aus der Entwicklungsabteilung dreimal so viel verdient wie er?



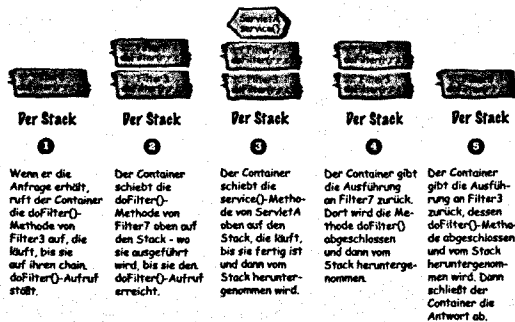
Prüfungsanforderungen	650
Die Großen 4 der Servlets-Sicherheit	653
Authentifizierung in der HTTP-Welt	656
Die zehn wichtigsten Argumente für deklarative Sicherheit	659
Wer in einer Webanwendung Sicherheit implementiert	660
Autorisierungsrollen und -einschränkungen	662
Die VIER Authentifizierungsarten	677
Daten bei der Übertragung sichern: HTTPS macht es möglich	682
Datenvertraulichkeit und -integrität spärlich und deklarativ	684

# 13

## Die Macht der Filter

Mit Filtern können Sie **Anfragen abfangen**. Und wenn Sie die *Anfrage* abfangen können, können Sie auch die *Antwort* steuern. Und das Beste ist: **Das Servlet hat keine Ahnung**.

Es erfährt nie, dass jemand zwischen die Anfrage vom Client und den Container-Aufruf der service()-Methode des Servlets getreten ist. Was das für Sie bedeutet? Mehr Urlaub. Weil Sie die Zeit, die Sie ansonsten damit verbringen würden, nur *eins* Ihrer Servlets anzupassen, stattdessen dem Schreiben und Konfigurieren eines Filters widmen können, der dazu in der Lage ist, *all* Ihre Servlets zu beeinflussen. Sie möchten *jedes* Servlet in Ihrer Anwendung mit einer Benutzerverfolgung ausstatten? Kein Problem. Sie möchten die Ausgabe *aller* Servlets in Ihrer Anwendung anpassen? Kein Problem. Und Sie müssen den Servlet-Code nicht einmal *anrühren*.



Prüfungsanforderungen	702
Den Filter zur Anfrageverfolgung schreiben	707
Der Lebenszyklus eines Filters	708
Filter deklarieren und anordnen	710
Ausgaben mit einem Antwortfilter komprimieren	713
Wrapper bringen es	719
Der wahre Kompressionsfilter-Code	722
Kompressions-Wrapper-Code	724

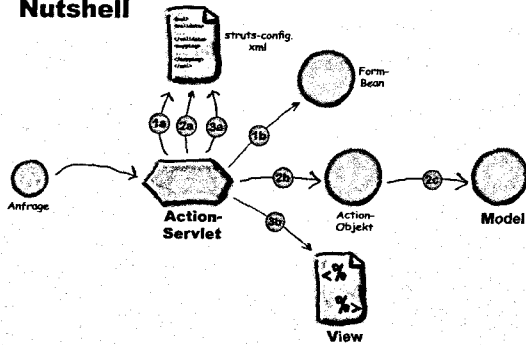


# 14

## Enterprise-Entwurfsmuster

**Irgendwer hat es bereits gemacht.** Wenn Sie gerade mit der Programmierung von Webanwendungen beginnen, haben Sie Glück. Sie können die angesammelte Weisheit Zehntausender Entwickler nutzen, die diesen Weg schon hinter sich haben und das T-Shirt besitzen. Mit J2EE-spezifischen und *anderen* Entwurfsmustern können Sie Ihren Code *und* sich selbst das Leben erleichtern. Und für das wichtigste Entwurfsmuster für Webanwendungen, MVC, gibt es sogar ein sehr beliebtes Framework, Struts, das Ihnen hilft, flexible, wartbare Servlet-Front Controller zu schnitzen. Sie schulden es sich selbst, die Arbeit aller *anderen* zu nutzen, damit Sie mehr Zeit mit den wichtigen Dingen des Lebens verbringen können ...

### Struts in a Nutshell



Prüfungsanforderungen	738
Hardware- und Softwarekräfte hinter Mustern	739
Zusammenfassung der Software-Entwurfsprinzipien ...	744
Muster zur Unterstützung entfernter Model-Komponenten	745
Überblick über JNDI und RMI	747
Das Business Delegate ist ein Vermittler	753
Zeit für ein Transfer Object?	759
Geschäftsschichtmuster: Kurzzusammenfassung	761
Rückkehr zu unserem allerersten Muster ... MVC	762
Ja! Das ist Struts (und Front Controller) in a Nutshell	767
Die Bier-Anwendung für Struts umgestalten	770
Zusammenfassung der Muster	778

## A KEIN SCHON-KAFFEE

**Der Abschlusstest.** Das ist es. 69 Fragen. Sprache, Themen und Schwierigkeitsgrad sind mit der *tatsächlichen* Prüfung fast identisch. Wir kennen uns damit aus.

Abschlusstest	791
Lösungen	828

## i Index

865