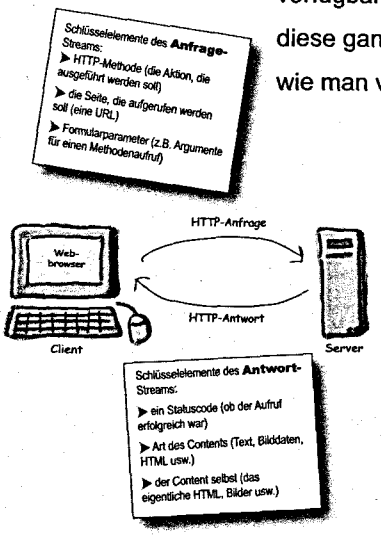
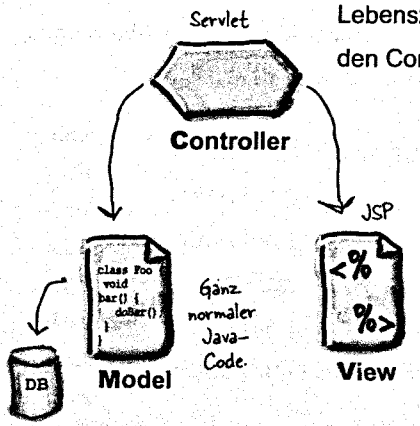
1 Warum Servlets & JSPs?

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 Web*sites* zu Web*anwendungen* kommt.



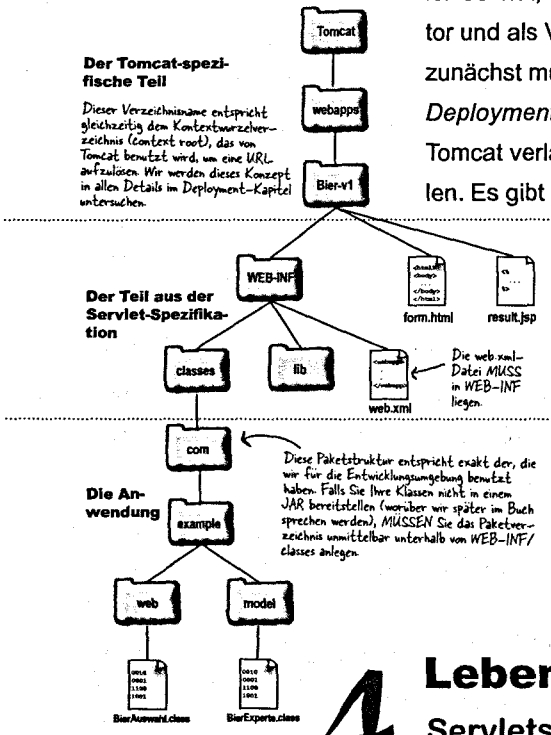
2 Architektur von Webanwendungen

Servlets brauchen Hilfe. Beim Eintreffen einer Anfrage muss jemand das Servlet instantiieren 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.



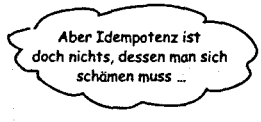
3 Mini-MVC-Kurs

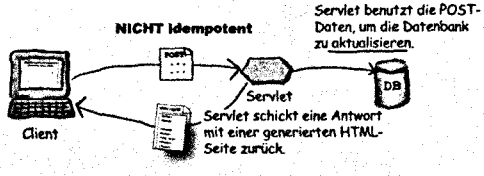
Legen Sie eine MVC-Webanwendung an und nehmen Sie sie in Betrieb. Es wird Zeit, die Ärmel hochzukrempeln und ein HTML-Formular, ein Controller-Servlet, ein Modell (eine ganz normale Java-Klasse), einen XML-Deployment Descriptor und als View eine JSP zu schreiben. Zeit zum Erstellen, Bereitstellen und 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.



4 Leben als Servlet

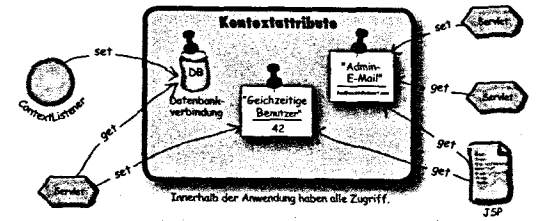
Servlets brauchen Unterstützung. Der Job eines Servlets ist es, die ***Anfrage*** eines Clients entgegenzunehmen und eine ***Antwort*** zurückzugeben. Die Anfrage kann etwas Einfaches sein: »**Zeig mir die Startseite!**«, oder etwas Komplexes: »**Mach mir die Abrechnung für meinen Einkaufskorb fertig!**«. 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* ...

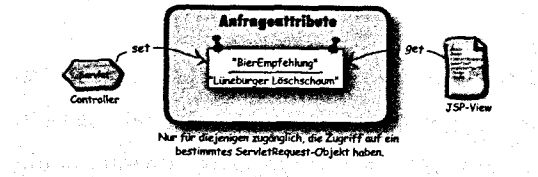




5 Leben als Webanwendung

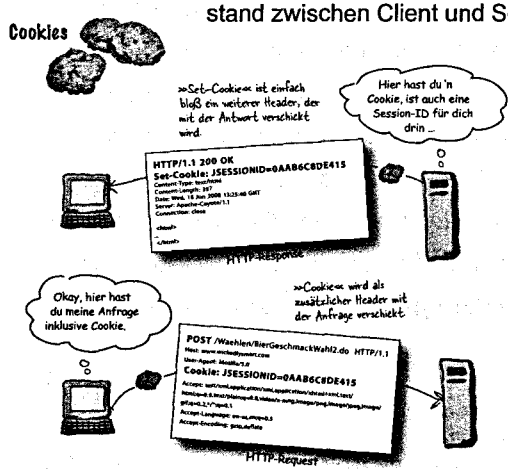
Kein Servlet ist unabhängig. In heutigen, modernen Webanwendungen arbeiten viele Komponenten zusammen, um ein gemeinsames Ziel zu erreichen. Man hat Models, Controller und Views. Man hat Parameter und Attribute. Und Hilfsklassen. Aber wie verbinden Sie diese Teile zu einem Ganzen? Wie bringen Sie Ihre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





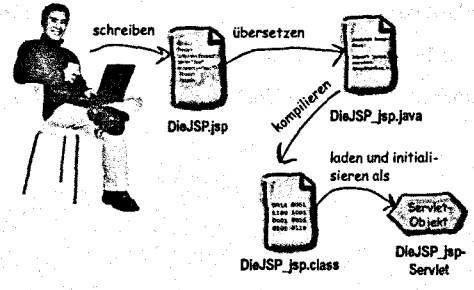
6 Kommunikation mit Gedächtnis

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.



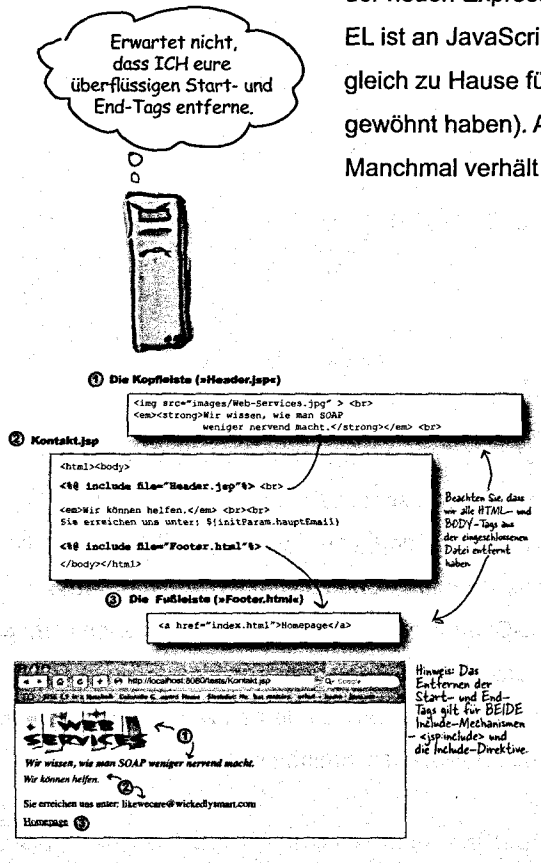
7 Wie es ist, eine JSP zu sein

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.



8 Skriptfreie Seiten

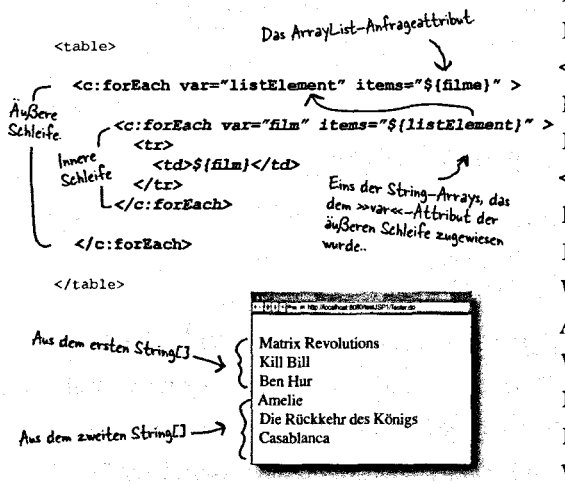
Befreien Sie sich vom Scripting. Können Ihre Webdesigner tatsächlich Java? Erwarten sie von den für den Server verantwortlichen Java-Programmierern, dass sie, hmm, sagen wir, Grafiker sind? Und selbst wenn *Sie* allein das gesamte Team bilden ... wollen Sie in Ihren JSPs wirklich einen Haufen kleiner Fetzen Java-Code haben? Schon mal etwas von »Wartungsalbtraum« 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!



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.



10 Wenn selbst die JSTL nicht ausreicht ...

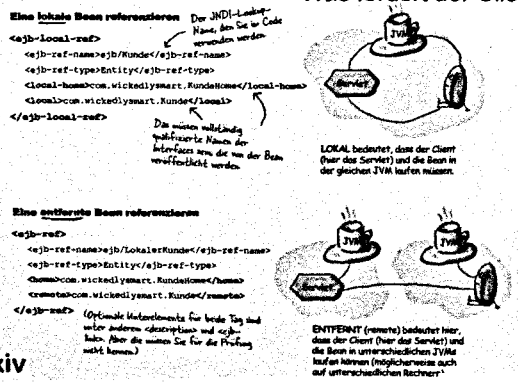
Manchmal reichen die JSTL und Standardaktionen nicht aus.

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).

11 Ihre Webanwendung verteilen

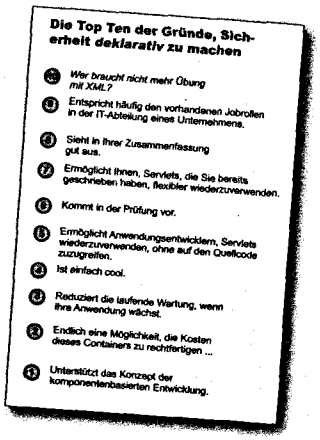
Endlich ist Ihre Webanwendung bereit für das wahre Leben.

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?



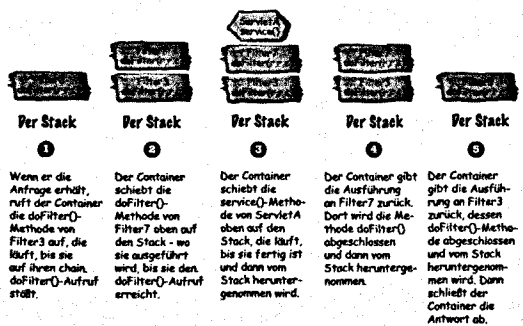
12 Geschützt und sicher

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?



13 Die Macht der Filter

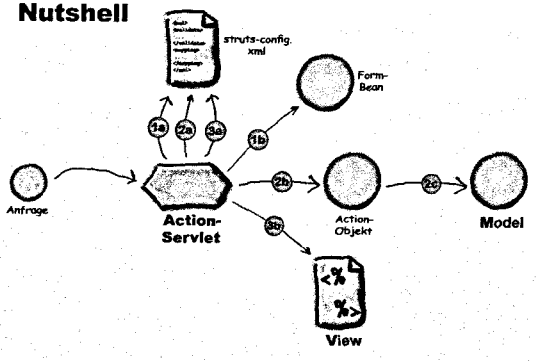
Mit Filtern können Sie Anfragen abfangen. Und wenn Sie die *Anfrage* abfangen können, können Sie auch die *Antwort* steuern. Und das Beste ist: **Das Servlet hat keine Ahnung.** Es erfährt nie, dass jemand zwischen die Anfrage vom Client und den Container-Aufruf der service()-Methode des Servlets getreten 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*.



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



A

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

i Index