Xpert.press

Die Reihe **Xpert.press** vermittelt Professionals in den Bereichen Softwareentwicklung, Internettechnologie und IT-Management aktuell und kompetent relevantes Fachwissen über Technologien und Produkte zur Entwicklung und Anwendung moderner Informationstechnologien.

Kai Jäger

Ajax
in der Praxis

Grundlagen, Konzepte, Lösungen

Springer

Kai Jäger
Schubartstr. 2c
70190 Stuttgart
jaeger@xwrs.net

ISBN 978-3-540-69333-8 978-3-540-69334-5

DOI 10.1007/978-3-540-69334-5

ISSN 1439-5428

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Einbandgestaltung: KünkelLopka Werbeagentur, Heidelberg

Gedruckt auf säurefreiem Papier

9 8 7 6 5 4 3 2 1

springer.com

Vorwort

Als ich mich vor einigen Jahren das erste Mal mit DHTML[1] auseinandersetzte, kam ich zu dem Schluss, dass sich diese Technologie wohl nicht durchsetzen würde. Zwar überzeugte mich die Idee, Webseiten dynamisch auf dem Client verändern zu können, doch Browserunterschiedene und schlechte JavaScript-Implementierungen führten mich schnell wieder auf den Boden der Tatsachen zurück. So führte DHTML während der drauffolgenden Jahre ein Schattendasein und nur die wenigsten Entwickler wagten es, die Technologie auch für nicht-triviale Web-Anwendungen einzusetzen. Wie groß das Misstrauen gegenüber DHTML zu dieser Zeit war, lässt ein Satz erahnen, den ich 1999 in einem Web-Forum las: „Don't use JavaScript, it just doesn't work" (Benutz kein JavaScript, es funktioniert einfach nicht).

Knappe sechs Jahre später schrieb der Autor *Paul Graham* in einem Artikel auf seiner Website folgenden Satz: „Basically, what Ajax means is ‚Javascript now works'." (Ajax bedeutet im Wesentlichen, dass JavaScript jetzt funktioniert.) JavaScript und damit auch DHTML hat sich, entgegen meiner Prognose, also doch durchgesetzt – wenn auch mit Verspätung.

Natürlich spricht inzwischen keiner mehr von DHTML, doch alle Grundkonzepte von DHTML finden sich auch in der Ajax-Technik wieder. Mit Hilfe von Ajax wird aus einer zunächst statischen HTML-Seite eine interaktive Browser-Anwendung. Dass Ajax dabei lästige Seiten-Reloads eliminieren kann, trägt nicht unerheblich zu diesem Effekt bei.

Heute, etwa zwei Jahre nach Entstehen des Begriffs Ajax ist das Interesse an dieser Programmiertechnik, entgegen mancher Vermutung, ungebrochen. Vor allem aber ist die Auseinandersetzung mit Ajax ein ganzes Stück seriöser geworden, denn die Ajax-Entwickler der ersten Stunde hatten inzwischen genug Zeit, sich „auszutoben".

[1] DHTML oder „dynamisches HTML" beschreibt den Einsatz einer Skriptsprache (üblicherweise JavaScript) zur Manipulation einer HTML-Seite.

Heute ist Ajax ein Werkzeug wie viele andere. Das heißt allerdings nicht, dass über Ajax das letzte Wort bereits gesprochen wäre. Tatsächlich gibt es in diesem Bereich ständig neue Entwicklungen – und das wird sich so schnell auch nicht ändern.

Inhaltsverzeichnis

1 Einleitung

In diesem Kapitel erfahren Sie, was Sie von diesem Buch erwarten können, wie Sie sich darin zurechtfinden und wie Sie am besten damit arbeiten.

1.1 Über dieses Buch

Vom Hype zur etablierten Technologie

Ajax ist auf dem besten Weg, sich als Web-Technologie zu etablieren. Zwar ist der anfängliche Hype um Ajax noch nicht ganz vorüber, doch es hat sich gezeigt, dass Ajax durchaus ernst zu nehmendes Potenzial besitzt und das Web als Plattform wieder interessant macht. So ist es nicht verwunderlich, dass sich inzwischen Unternehmen aus allen Bereichen und jeder Größenordnung mit Ajax befassen. Doch der Einstieg in die Ajax-Welt gestaltet sich mitunter schwierig – die meisten der Web-Technologien, auf die Ajax aufsetzt, sind zwar nicht neu, doch insbesondere das Zusammenspiel der einzelnen Komponenten und die asynchrone Kommunikation zwischen Client und Server führen in der Praxis immer wieder zu unerwarteten Problemen.

Ajax-Frameworks

Für viele Ajax-Einsteiger liegt es daher nahe, von Anfang an auf Frameworks zu setzen und damit einige der größten Probleme von Ajax zu umschiffen. Das ist sicherlich legitim, doch wer Ajax nie im Kern verstanden hat, wird auch mit Frameworks irgendwann vor unvorhergesehenen Problemen stehen. Darüber hinaus nimmt Ihnen ein Framework bestenfalls einen Teil der Arbeit ab – Anwendungslogik und Benutzerinteraktionen müssen Sie immer noch selbst programmieren und das meist in der „am häufigsten missverstandenen“[1] Programmiersprache JavaScript.

[1] Douglas Crockford, Softwarearchitekt bei Yahoo, bezeichnet JavaScript auf seiner Website als „the world's most misunderstood programming language“ (WebCode →*crockford*).

Ohne Theorie keine Praxis

In diesem Buch geht es vor allem um eines: Ajax im Kern verstehen und anwenden zu lernen. Aus diesem Grund enthält das Buch einen recht umfassenden Theorieteil. Bevor Sie nun allerdings Zweifel an der Wahl des Titels „Ajax in der Praxis“ hegen, lassen Sie mich Ihnen das diesem Buch zugrunde liegende Konzept erläutern: Theorie versteht man am besten anhand von praktischen Beispielen. Deshalb werden Sie hier kaum ein theoretisches Konzept finden, das nicht anhand von Code-Beispielen erklärt wird. Doch konkrete Beispiele allein können Ihnen bestenfalls einen Eindruck davon vermitteln, wie etwas funktioniert. Aus diesem Grund geht dieses Buch an vielen Stellen etwas weiter in die Tiefe, als es vielleicht unbedingt notwendig wäre. Mit diesem zusätzlichen Wissen haben Sie in der Praxis dann allerdings die Möglichkeit, Probleme von mehr als einer Seite anzugehen und so die für Sie beste Lösung zu finden.

1.2 Aufbau

Die Technologien

Dieses Buch besteht aus drei Teilen. In den Kapiteln 1–3 erfahren Sie, wie Sie mit diesem Buch sinnvoll arbeiten, was es mit Ajax auf sich hat und welche Werkzeuge Sie benötigen, um Ajax-Anwendungen zu entwickeln. Kapitel 4–7 beschäftigen sich mit den Technologien, die hinter Ajax stehen. Im Vordergrund steht dabei die Sprache JavaScript, die für die Entwicklung von Ajax-Anwendungen unerlässlich ist. Außerdem erfahren Sie im ersten Teil, wie Sie HTML-Seiten und XML-Dokumente dynamisch traversieren und manipulieren können, wie Sie die asynchrone Kommunikation zwischen Client und Server realisieren und welche Rolle Web-Services dabei spielen. Der erste Teil ist dabei so aufgebaut, dass er sich auch gut zum Nachschlagen einzelner Stichworte eignet.

Weiterführende Themen

Der zweite Teil (Kapitel 8–12) befasst sich mit weiterführenden Themen der Ajax-Entwicklung. Dabei wird erläutert, wie Sie Ihre Ajax-Anwendungen schnell und sicher machen können. Außerdem erhalten Sie in diesem Teil eine Einführung in die Themen Barrierefreiheit und Usability. In Kapitel 12 erfahren Sie schließlich, warum es ratsam ist, ein Ajax-Framework einzusetzen und wie Sie das richtige Framework für sich finden.

Abschließend lernen Sie in Kapitel 13 drei etwas umfangreichere Beispiele kennen, die zeigen, wie sich die in den vorherigen Kapiteln vorgestellten theoretischen Konzepte in der Praxis umsetzten lassen.

1.3 Arbeiten mit diesem Buch

Anforderungen an den Leser

Ajax umfasst so viele verschiedene Technologien, dass es unmöglich wäre, alle in angemessenem Umfang zu behandeln. Aus diesem Grund setzt dieses Buch einen klaren Schwerpunkt auf Programmierung von Web-Anwendungen. Das bedeutet für Sie zum einen, dass Auszeichnungssprachen wie HTML, XML und CSS hier nicht besprochen, sehr wohl aber eingesetzt werden. Sollten Sie noch nicht über Erfahrung mit diesen Sprachen verfügen, finden Sie im Web eine ganze Reihe guter Einführungen (WebCode →*markup*). Zum anderen setzt dieses Buch voraus, dass Sie bereits über Erfahrung mit einer objektorientierten Programmiersprache verfügen. Java, C# oder C++ bieten sich hier besonders gut an, da die Syntax aller drei Sprachen – wie auch die der hier vorgestellte Sprache JavaScript – aus der C-Familie stammt.

Die Server-Seite

Während auf der Client-Seite als Programmiersprache zwangsläufig JavaScript zum Einsatz kommt, haben Sie auf der Server-Seite deutlich mehr Möglichkeiten. Natürlich soll, obwohl dies nicht der Schwerpunkt des Buchs ist, die Server-Seite hier auch behandelt werden. Allerdings möchte ich Ihnen bei der Wahl der Programmiersprache keine Vorschriften machen. Das Buch ist daher weitestgehend „Server-agnostisch" gehalten. Für die konkreten Beispiele ließ es sich jedoch nicht vermeiden, eine Server-Technologie auszuwählen. Hier habe ich mich für *Java-Servlets* entschieden, da mir diese am neutralsten erschien und sich die meisten Konzepte der Servlets auch recht problemlos auf andere Technologien übertragen lassen. Das nötige Hintergrundwissen zur Servlet-Programmierung sowie eine kurze Anleitung zur Installation eines Servlet-Containers finden Sie im Kapitel 3.

1.4 Konventionen

Formatierung des Quelltexts

Alle Quelltextbeispiele in diesem Buch sind zur besseren Abgrenzung in einer nichtproportionalen Schrift gehalten (`etwa wie dieser Text`). Werden Befehle oder reservierte Wörter im Fließtext wiederholt, sind diese ebenfalls in einer nichtproportionalen Schrift formatiert. Bei längeren Quelltextbeispielen werden oftmals einzelne Stellen im Programmcode mit Nummern (z. B. so: ❸) versehen und an anderer Stelle wird dann auf diese verwiesen. Das soll Ihnen dabei helfen, sich im Quelltext besser zurechtzufinden und die Erläuterung leichter nachzuvollziehen.

Zeilenumbrüche und Auslassungen im Quelltext

Bei der Übertragung der Beispiele in dieses Buch, wurde versucht, die ursprüngliche Formatierung nach Möglichkeit beizubehalten. Leider ist dies aufgrund der eingeschränkten Spaltenbreite nicht immer gelungen. Aus diesem Grund mussten an manchen Stellen der Texteinzug verringert und zusätzliche Zeilenumbrüche eingefügt werden. Zeilenumbrüche, die beim Übertragen aus dem Buch nicht übernommen werden dürfen, da ansonsten mit Problemen bei der Kompilation oder Ausführung des Codes zu rechnen ist, sind mit einem Pfeil-Symbol ⇨ markiert Außerdem wurden bei manchen Beispielen einzelne weniger relevante Codezeilen ausgelassen. Solche Auslassungen sind stets mit drei Punkten (…) markiert. Um diese Beispiele selbst testen zu können, müssen die Punkte entfernt und gegebenenfalls durch eigenen Code ersetzt werden.

Ebenfalls zu beachten ist, dass bei dem in diesem Buch abgedruckten HTML-Code häufig aus Platzgründen Auslassungen vorgenommen wurden. So fehlt bei manchen Beispielen etwa ein `<head>`-Bereich oder auch die Angabe des Doctype. Obwohl der so verkürzte HTML-Code in vielen Fällen korrekt dargestellt wird, sollten Sie in Ihren eigenen HTML-Dokumenten die fehlenden Teile stets ergänzen, nicht zuletzt um die Gültigkeit Ihres HTML-Codes zu wahren.

Einheitliche Beispiele

Die meisten Quelltext-Beispiele in diesem Buch befassen sich mit dem fiktiven Softwareunternehmen *MusterSoft*, das ein Webbasiertes Customer Relationship Management System[2] (CRM) entwickeln möchte. Dieser Ansatz wurde gewählt, um die Beispiele möglichst einheitlich und vor allem auch realistisch zu halten. Selbstverständlich benötigen Sie aber keine Vorkenntnisse im Bereich Customer Relationship Management, um allen Beispielen folgen zu können.

1.5 Code-Bibliothek

Ausgleich von Browser-Unterschieden

Die meisten Code-Beispiele in diesem Buch sind darauf ausgelegt, Ihnen bestimmte Konzepte der Entwicklung von Ajax-Anwendungen näher zu bringen. Das ist allerdings nicht immer ganz einfach, denn zum Ausgleich von Inkompatibilitäten zwischen Browsern und aufgrund der Unzulänglichkeiten bestimmter APIs muss häufig sehr viel zusätzlicher Code geschrieben werden. Um diesen Code nicht von Beispiel zu Beispiel wiederholen zu müssen, werden im Verlauf

[2] Vereinfacht gesagt erlaubt ein Customer Relationship Management System einem Unternehmen, seine Kundenbeziehungen abteilungsübergreifend zu verwalten und auszuwerten.

dieses Buchs an einigen Stellen Lösungen für solche Probleme vorgestellt, die Sie immer wieder verwenden können. Wenn Sie alle diese Lösungen zusammentragen, erhalten Sie eine Art Code-Bibliothek, die viele der häufigsten Probleme bei der Entwicklung von Ajax-Anwendungen löst und Ihnen den Einstieg in die Ajax-Programmierung erleichtert. Die Code-Bibliothek besteht aus einer einzelnen JavaScript-Datei, die Sie sich von der Website zu diesem Buch herunterladen können (siehe Abschnitt 1.7). Sie können sich die Code-Bibliothek auch selbst anlegen. Die Code-Stücke, die Sie in die Bibliothek aufnehmen sollten, sind mit dem Buch-Symbol (📖) gekennzeichnet. Unabhängig davon, ob Sie die Code-Bibliothek nun selbst anlegen oder herunterladen möchten, sollten Sie beachten, dass in vielen Code-Beispielen auf Funktionen aus der Bibliothek zurückgegriffen, die Bibliothek selbst aber nicht eingebunden wird. Solche Beispiele müssen Sie zuerst entsprechend ergänzen, bevor Sie sie testen können.

1.6 Danksagungen

Mein allergrößter Dank gebührt Herrn Prof. Walter Kriha, der maßgeblich daran beteiligt war, dass ich dieses Buch schreiben durfte und der mich in vielerlei Hinsicht unterstützt hat. Besonderen Dank auch an Rainer Jäger, meinen Vater, der sich freundlicherweise dazu bereit erklärt hat, das Buch für mich Korrektur zu lesen. Herzlich für die gute Zusammenarbeit bedanken möchte ich mich auch beim Springer Verlag, insbesondere bei Frau Glaunsinger, Frau Fleschutz und Herrn Engesser.

Nicht zuletzt möchte ich mich bei meiner Familie und meinen Freunden bedanken – für ihre Geduld und Unterstützung, aber auch für manchen guten Rat.

1.7 URL zum Buch

Das WebCode-System

In den verschiedenen Kapiteln finden sich immer wieder Verweise auf Websites und Ressourcen im Internet. Da Web-Adressen in der Regel eine weit kürzere Lebensdauer haben als Fachbücher und da das Abtippen von mit unter sehr langen URLs etwas mühselig ist, werden Sie in diesem Buch keine solchen Adressen finden. Stattdessen verwendet dieses Buch ein WebCode-System. WebCodes sind mit einem voranstehenden Pfeilsymbol gekennzeichnet und kursiv

gedruckt (etwa so: →*einleitung*). Diese können als eine Art Alias verstanden werden, welche Sie unter der Adresse

```
http://www.ajax-in-der-praxis.de
```

eingeben können. Sie werden dann unmittelbar auf die entsprechende Seite weitergeleitet. Sollte sich eine URL einmal ändern, kann die Weiterleitung des entsprechenden Alias angepasst werden, ohne dass das Buch damit seine Gültigkeit verliert.

Den vollständigen Quelltext zu den Code-Beispielen finden Sie ebenfalls unter der oben genannten Web-Adresse als Download.

2 Einführung

Worum geht es bei Ajax eigentlich? Mit dieser scheinbar trivialen Frage könnten Sie so manchen Web-Entwickler in Verlegenheit bringen. In diesem Kapitel erfahren Sie, worin der eigentliche Vorteil von Ajax besteht, wie Ajax entstanden ist und was Ajax für das Web 2.0 bedeutet.

2.1 Was ist Ajax?

Komfortable Web-Anwendungen

Kein „Buzzword" hat in letzter Zeit so sehr Furore gemacht wie Ajax. Sucht man über eine der gängigen Suchmaschinen nach Ajax, findet man in der Regel Seiten über „Asynchronous JavaScript and XML" noch vor dem Fußballverein und dem Reinigungsmittel gleichen Namens. Umso verwunderlicher ist es, dass viele Leute zwar den „Markennamen" Ajax kennen, jedoch keine konkrete Vorstellung haben, was technologisch dahintersteckt: Ajax wird inzwischen quasi synonym für alle Web-Anwendungen verwendet, die einen gegenüber herkömmlichen Webseiten erhöhten Bedienkomfort bieten.

Ursprüngliche Bedeutung

Ursprünglich jedoch stand Ajax für „Asynchronous JavaScript and XML". Betrachtet man jeden dieser Begriffe einzeln, erhält man eine Liste von Anforderungen, die eine Ajax-Anwendung erfüllen muss, um als solche bezeichnet werden zu dürfen. Zunächst muss jede Ajax-Anwendung asynchron Daten mit dem Server austauschen. Desweiteren muss auf dem Client (also dem Browser) die Programmiersprache JavaScript eingesetzt werden und zu guter Letzt müssen die Daten strukturiert in Form eines XML-Dokuments zwischen Client und Server ausgetauscht werden.

Heutzutage ist man sich allerdings uneinig, ob die Wahl der Abkürzung AJAX so sinnvoll war und weicht zunehmend auf die Schreibweise „Ajax" aus, um die ursprüngliche Bedeutung zu verschleiern. Grund dafür ist, dass viele der häufig angeführten Beispiele für typische Ajax-Anwendungen kein XML verwenden und

manche noch nicht einmal asynchron arbeiten. Der einzige Punkt, über den man sich einig zu sein scheint, ist, dass Ajax-Anwendungen JavaScript verwenden müssen – vielleicht aber auch nur, weil es momentan auf der Browser-Plattform praktisch keine Alternative gibt.

Nutzen für den Anwender

Um nach dieser ernüchternden Zwischenbilanz nun dennoch verstehen zu können, worum es bei Ajax eigentlich geht, muss man sich vor Augen führen, welchen Nutzen der Anwender letztendlich aus Ajax zieht. Der englische Begriff „responsiveness“, der im Deutschen gern mit „Reaktionsbereitschaft“ übersetzt wird, trifft den Punkt: Ajax-Anwendungen reagieren scheinbar schneller auf Benutzereingaben. Aber warum ist das so?

Probleme der Client-Server-Architektur

Anwendungen arbeiten in der Regel mit zwei verschiedenen Arten von Daten – mit transienten und persistenten Daten. Die transienten Daten liegen im flüchtigen Arbeitsspeicher, während die persistenten Daten nicht flüchtig auf Festplatte gelagert sind. Wenn Sie beispielsweise mit einem Textverarbeitungsprogramm arbeiten, sind ihre Eingaben transient, bis Sie Ihr Dokument abspeichern und damit persistent machen. Sie überführen also Daten vom transienten in den persistenten Zustand. Gleichermaßen überführen Sie beim Öffnen eines Dokuments die persistenten Daten in einen transienten Zustand. Dieses Prinzip liegt praktisch allen Anwendungen zugrunde und auch die meisten Web-Anwendungen arbeiten so. Bei Web-Anwendungen ist die Überführung der Daten aus einem Zustand in den anderen jedoch weitaus komplizierter als bei Desktop-Anwendungen, weil hier die transienten und persistenten Daten auf unterschiedlichen Rechnern (Client und Server) liegen. Ein Browserbasiertes Textverarbeitungsprogramm beispielsweise müsste zum Speichern eines Dokuments eine Anfrage an den Server stellen. Diese Anfrage hat eine gewisse Laufzeit, die sich aus Faktoren wie der Netzwerk-Latenz und verschiedenen Kodierungs- und Interpretierungsvorgängen zusammensetzt.

Ajax ist nicht schneller

Zwar kann Ajax Server-Anfragen nicht schneller machen, aber es kann zumindest ihre Auswirkungen auf die „Reaktionsbereitschaft“ einer Anwendung minimieren. Vor dem Einsatz von Ajax musste, um eine neue Anfrage an den Server zu stellen, die Seite im Browser gewechselt werden. Dies bedeutete für den Anwender, dass er in seiner Arbeit unterbrochen und so lange zur Passivität gezwungen war, bis die neue Seite vom Server abgerufen wurde. Ajax löst dieses Problem durch Nebenläufigkeit, indem es dem Entwickler erlaubt, neu Anfragen an den Server zu stellen, ohne die aktuelle Seite zu verlassen und ohne den Ablauf der Anwendung zu unterbrechen. Für unser Beispiel mit dem Browser-basierten Textverarbeitungsprogramm bedeutet das, dass der Benutzer nicht darauf warten muss,

bis sein Dokument gespeichert ist, sondern bereits weiterarbeiten kann, während die Server-Anfrage noch übermittelt wird.

Ajax löst ein Usability-Problem

Auf den Punkt gebracht, löst Ajax also ein Usability-Problem. Zwar gibt es viele Anwendungen, die erst seit der Einführung von Ajax sinnvoll als Web-Anwendungen zu realisieren sind – letztendlich kann Ajax jedoch nichts, was nicht auch zuvor schon möglich gewesen wäre. Dass erst seit der Einführung von Ajax immer mehr Anwendungen ihren Weg ins Web finden, verdeutlicht dennoch, welchen Stellenwert die Usability eigentlich einnimmt. Das Web wird heute von einem beachtlichen Teil der Bevölkerung genutzt und dank sinkender Verbindungspreise und erschwinglicher Pauschaltarife verbringen Web-User heute deutlich mehr Zeit im Netz. Daneben gewinnen Online-Anwendungen wie Webmail, Kartendienste und in inzwischen sogar Office-Anwendungen immer mehr an Bedeutung. Dass man bei der zunehmenden Verlagerung von Anwendungsprogrammen ins Web nicht auf den gewohnten Bedienkomfort verzichten möchte, leuchtet ein.

2.2 Hintergründe

Namensgebung

Anders als bei den meisten Technologien, die in der Regel einem einzigen oder einigen wenigen Urhebern zugeordnet werden können, fällt dies bei Ajax deutlich schwerer. Der Begriff Ajax wurde im Jahr 2005 durch Jesse James Garret eingeführt, der in seinem Artikel „Ajax: A new approach to web applications“ (WebCode →*garret*) die Vorzüge des asynchronen Modells beschrieb. Garret hat Ajax jedoch nicht erfunden, vielmehr hat er etwas längst Bekanntem einen klangvollen Namen verliehen.

Die XMLHttp-Request-API

Der fundamentale Unterschied zwischen herkömmlichen Web-Anwendungen und solchen, die Ajax einsetzen, besteht in der asynchronen Datenübertragung. Greift man diesen Bestandteil von Ajax einmal heraus, stellt man schnell fest, dass die `XMLHttpRequest`-API dabei eine entscheidende Rolle zu spielen scheint. Die `XMLHttpRequest`-API (oder kurz XHR) dient, wie der Name schon erahnen lässt, der asynchronen Übertragung von XML, aber auch anders formatierter Daten über das HTTP-Protokoll. Eingeführt wurde XHR durch Microsoft, ursprünglich als Bestandteil von Outlook Web Access, einem Web-Mail-Client für den Microsoft Exchange Server. Als Microsoft schließlich damit begann, XHR mit dem Internet Explorer auszuliefern, wurden auch andere Browser-Hersteller darauf aufmerksam und entwickelten eigene Implementierungen der bis heute nicht abschließend standardisierten API.

Später Erfolg

Dass XHR nicht unmittelbar zum Erfolg wurde, lässt sich zum einen mit dem Ende des Internetbooms und zum anderen mit dem nur langsamen Aussterben veralteter Browser-Versionen erklären. Als XHR jedoch eine ausreichend große Verbreitung gefunden hatte, sorgte Google für eine kleine Revolution. Mit GMail und später Google Maps veröffentlichte der Suchmaschinenhersteller zwei Web-Anwendungen, die für Furore sorgten. Auf einmal schien im Browser alles möglich zu sein – ganz ohne schwergewichtige Plugins und unüberwindliche Cross-Browser-Inkompatibilitäten.

Web-Standards

Doch es war eine weitere Entwicklung am Entstehen von Ajax beteiligt. Der Browser-War zwischen Microsoft und Netscape wurde vor allem über die Entwicklung neuer, meist keinem Standard folgender Browser-Funktionen ausgetragen. So entwickelten sich nach und nach zwei Browser, die bis auf eine kleine Teilmenge des HTML-Standards praktisch in keinem Punkt kompatibel waren. Wollte man damals dynamische Web-Anwendungen programmieren, so gelang dies meist nur über Browser-Weichen und mit viel Trickserei. Einige Web-Entwickler ließen sich selbst davon nicht abschrecken und trotz aller Kompatibilitätsprobleme entstanden in dieser Zeit einige beeindruckende Web-Anwendungen. Doch für die meisten Programmierer war diese Form der Softwareentwicklung schlichtweg zu wenig effektiv.

Obwohl auch heute noch zahlreiche Inkompatibilitäten zwischen den verschiedenen Browsern bestehen, hat sich die Situation doch ganz erheblich entspannt. Viele der während des Browser-Wars entstandenen neuen Schnittstellen sind inzwischen offiziell standardisiert und die meisten Browser halten sich zumindest weitgehend an diese Standards. Somit ist es nunmehr möglich, wenn auch mit einigem Aufwand, Web-Applikationen zu entwickeln, die auf den meisten aktuellen Browsern korrekt funktionieren.

2.3 Ajax im Kontext von Web 2.0

Ajax als Schlüsselkomponente des Web 2.0

In seinem Artikel „What is Web 2.0?“ beschreibt Tim O’Reilly die Entwicklung des Web nach dem Platzen der Dotcom-Blase. Neben den sozialen Phänomenen die in diesem Zusammenhang zu beobachten sind, erwähnt er auch Ajax als eine „Schlüsselkomponente von Web 2.0 Anwendungen“ und die durch Ajax entstandenen neuen Möglichkeiten im Bereich der grafischen Benutzeroberflächen.

Vom Konsumenten zum Produzenten

Die Notwendigkeit neuer Ansätze im Bereich der Benutzeroberflächen ergibt sich zum Teil aus einem veränderten Anwenderverhalten. War die Hauptaktivität eines Web-Users bis vor Kurzem

noch das Lesen von Hypertext-Seiten, so beteiligen sich heute immer mehr User aktiv am Geschehen, indem sie selbst Texte verfassen, kommentieren und verlinken. Der Web-User wird dadurch vom Konsumenten zum Produzenten und das Web von einem Medium zu einer Plattform. Diese Entwicklung hat weitreichende Folgen: Der Web-User von heute nutzt Web-Anwendungen in zunehmendem Maß ähnlich intensiv wie vergleichbare Desktop-Anwendungen. Mit intensiverer Nutzung steigen aber auch die Anforderungen an den Bedienkomfort. Die Möglichkeiten statischen HTMLs sind hierfür oft unzulänglich.

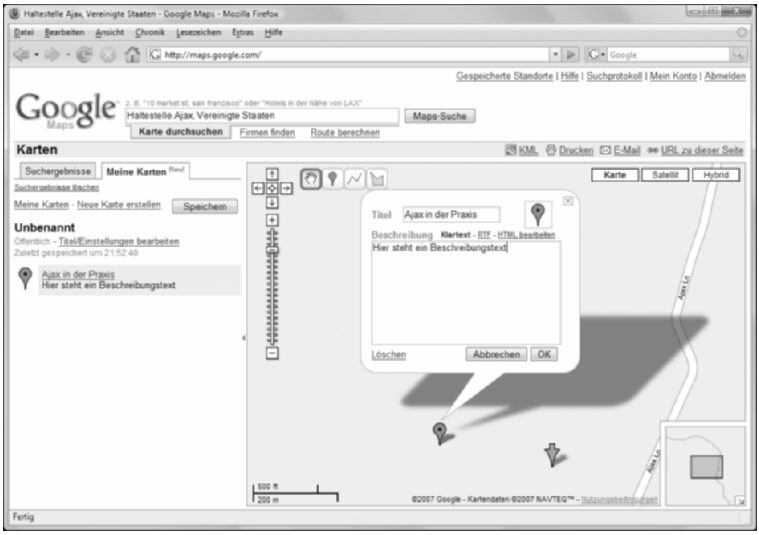

Abb. 2.1 Ortsmarke hinzufügen in Google Maps

Gleichzeitig sind viele Web 2.0-Anwendungen geradezu prädestiniert für den Einsatz von Ajax. In Google Maps haben Sie beispielsweise die Möglichkeit, besondere Orte mit eigenen Hinweisen zu versehen, die dann von allen anderen Google Maps Benutzern gesehen werden können. Ohne Ajax würde das Anlegen einer solchen „Ortsmarke" zwangsläufig zu einem Reload der Seite führen. Damit müsste auch der aktuelle Kartenausschnitt neu geladen werden, was längere Ladezeiten zur Folge hätte. Auf die gleiche Weise profitieren auch andere Web 2.0-Anwendungen von Ajax, die beispielsweise eine Kommentar- oder Bewertungsfunktionalität anbieten.

Bedeutung für Unternehmen

Gleichzeitig hat Ajax auch eine große Bedeutung für Unternehmen, die für das Web 2.0 entwickeln möchten. Da Ajax-Anwendungen plattformunabhängig sind, können auf diesem Weg weitaus mehr Anwender erreicht werden als über vergleichbare Desktop-Applikationen. Darüberhinaus ist die Entwicklung von Web-An-

wendungen in den meisten Fällen einfacher und damit kostengünstiger als vergleichbare Offline-Anwendungen. Dies ermöglicht letztlich auch kleinen Start-up-Unternehmen, Web-Projekte umzusetzen, ohne dabei von Investoren abhängig zu sein.

2.4 Der Ajax-Entwickler

On- und Offline-Entwickler

Der große Erfolg von Ajax sorgt dafür, dass sich zunehmend auch Anwendungsentwickler aus dem Offline-Bereich mit dem Web auseinandersetzen. Obwohl sich diese Entwickler zunächst eine Vielzahl von Script- und Auszeichnungssprachen aneignen müssen, um überhaupt Inhalte für das Web produzieren zu können, haben sie doch einen entscheidenden Vorteil gegenüber Web-Entwicklern der „alten Schule". Während es ein Web-Entwickler gewöhnt ist, Anwendungen in einzelne, hintereinander ablaufende Vorgänge zu unterteilen, denkt ein Entwickler von Desktop-Anwendungen in Ereignissen und nebenläufigen Prozessen. Aus diesem Grund empfinden es Web-Entwickler häufig als große Umstellung, wenn sie plötzlich Ajax-Anwendungen entwickeln sollen, die schon per Definition ereignisbasiert und nebenläufig arbeiten. Ein Entwickler von Desktop-Anwendungen hingegen muss sich hier kaum umgewöhnen.

Es ist leider häufiger zu beobachten, dass Web-Entwickler die alten Denkmuster nur schwer ablegen können. Immer wieder sieht man Ajax-Anwendungen, die dasselbe tun wie herkömmliche Web-Anwendungen, bei denen nur die Datenübertragung zwischen Client und Server asynchron abläuft. Diese Herangehensweise ist problematisch, da eine solche Anwendung den höheren Entwicklungsaufwand kaum rechtfertigt. Wer wirklich von Ajax profitieren möchte, muss andere Wege einschlagen.

Eine neue Herangehensweise

Wie bereits erwähnt, liegt der Hauptunterschied zwischen klassischen Web-Anwendungen und Ajax-Anwendungen im Ablauf der jeweiligen Prozesse. Es ist daher unerlässlich, sich gründlich mit der Ereignis-basierten Programmierung auseinanderzusetzen. Ebenso wichtig ist es jedoch, zu wissen, welche Möglichkeiten Ajax bietet. Viele Web-Entwickler haben sich mit den eingeschränkten Möglichkeiten von statischem HTML abgefunden. Ajax hebt einige dieser Einschränkungen auf, doch es erfordert einen gewissen Pioniergeist, sich dies auch zu Nutze zu machen. Eine gute Herangehensweise ist es daher, zunächst anzunehmen, alles sei möglich. Wenn Sie sich gut gestaltete Desktopanwendungen einmal unter dem Gesichtspunkt der Benutzerführung anschauen, werden Sie feststellen, dass man hier selten den technisch einfachsten, sondern

meist den für den Anwender sinnvollsten Weg geht. Dieses Prinzip lässt sich auf Web-Anwendungen übertragen und tatsächlich lassen sich mit Ajax grafische Benutzeroberflächen realisieren, die in puncto Bedienkomfort denen aktueller Desktop-Anwendungen kaum nachstehen.

Ajax ist nicht der Weisheit letzter Schluss

Allerdings muss im Einzelfall abgewogen werden, ob der Mehraufwand, der mit Ajax im Vergleich zu einer gewöhnlichen Web-Anwendung praktisch immer entsteht, auch wirklich gerechtfertigt ist. Es mag zunächst verlockend sein, Ajax in jeder nur erdenklichen Situation einzusetzen, doch Ajax ist nie die einzige und oftmals auch nicht die beste Wahl. Sie sollten sich daher immer die Frage stellen, ob Anwender Ihrer Software durch den Einsatz von Ajax tatsächlich einen Mehrwert erhalten. Ist dies nicht der Fall, kann es besser sein, auf Ajax zu verzichten.

3 Die richtigen Werkzeuge

Vor einigen Jahren meinte so mancher noch, es sei ein Zeichen besonderen Könnens, seine Webseiten mit einem herkömmlichen Texteditor zu erstellen. Doch mit dem Aufkommen von Ajax wachsen nicht nur die Möglichkeiten, sondern auch Umfang und Komplexität von Web-Anwendungen. Grund genug, sich mit Werkzeugen auszurüsten, die einem das Entwicklerleben erleichtern. In diesem Kapitel finden Sie eine kleine Auswahl von Anwendungen und Browser-Erweiterungen, die Sie sich auf jeden Fall ansehen sollten.

3.1 Web-Browser

Der Browser als Entwicklungstool

Für die meisten Web-User ist der Browser einfach nur ein Programm zum Betrachten von Web-Seiten. Für den Web-Entwickler nimmt der Browser jedoch einen ganz andere Stelle ein: Er ist Laufzeit-, Debugging- und Testumgebung in einem und damit ein wichtiger Bestandteil des Entwicklungsprozesses.

Die wichtigsten Browser

Seit dem Ende des Browser-Wars, aus dem der *Internet Explorer* als Sieger hervorging, hat sich einiges getan. Der damalige Konkurrent Netscape hat sich inzwischen zwar aus dem Browsergeschäft zurückgezogen, doch nicht ohne vorher noch die Mozilla Organisation zu gründen, die inzwischen als Mozilla Foundation bekannt ist und unter deren Leitung der *Firefox* Browser entwickelt wird. Firefox ist bislang der einzige Browser, der gegenüber dem Internet Explorer größere Marktanteile gewinnen konnte. Immerhin gibt es drei weitere Mitspieler im Browser-Geschäft, die hier noch erwähnt werden sollten. Der wichtigste Browser auf dem Mac ist zurzeit *Safari*. Safari nutzt die HTML-Rendering-Engine des *Konqueror* Browsers, der vor allem auf Unix- bzw. Linux-Betriebssystemen anzutreffen ist. Außerdem zu erwähnen ist der aus Norwegen stammende *Opera* Browser. Während Opera auf dem PC und Mac bisher nur eine relativ kleine Anhängerschaft besitzt, konnte der Browser

im Bereich der mobilen Endgeräte inzwischen größere Erfolge verzeichnen. Außerdem hat Opera inzwischen im Konsolenmarkt Fuß gefasst und bietet nun eine Version des Browsers für die Nintendo Wii Spielkonsole an.

Web-Standards

Allen Konkurrenten des Internet Explorer gemein ist die gute Unterstützung von Web-Standards. Dies hat letztlich auch bei Microsoft zu einem Umdenken geführt und so erschien Ende 2006 nach Jahren des Stillstands eine neue Version des Internet Explorers, die nun ebenfalls eine gute Unterstützung von Web-Standards bietet. Bis allerdings ein Großteil der Web-User auf die neue Version umgestellt hat, wird erfahrungsgemäß noch einige Zeit vergehen. So ist beispielsweise selbst heute noch eine beachtliche Zahl an Web-Usern zu verzeichnen, die mit dem zur Drucklegung dieses Buchs acht Jahre alten Internet Explorer 5 im Web unterwegs sind. Es sind jedoch nicht nur die veralteten Browser, die dem Web-Entwickler das Leben schwer machen – auch aktuelle Browser verhalten sich nicht alle gleich.

Mehrere Browser zum Testen

Bis vor einigen Jahren war es noch durchaus üblich, Web-Anwendungen auf einen einzelnen Browser zuzuschneiden. Benutzer anderer Browser schauten dann in die Röhre. Heute jedoch wird von Web-Anwendungen *Cross-Browser-Kompatibilität* verlangt. Ein Web-Entwickler muss sich dafür eine Sammlung der gängigsten Browser anlegen und ständige Tests durchführen. Das ist zwar mühselig, zahlt sich in der Regel aber aus: Wer viele Browser unterstützt, hat auch eine große potenzielle Anwendergemeinde.

3.1.1 Browser-Versionen

Welche Browser soll ich installieren?

In einem gedruckten Buch über Browser-Versionen zu sprechen ist angesichts des rapiden Fortschritts in diesem Bereich immer etwas problematisch. Allgemein kann man jedoch sagen, dass Sie sich auf jeden Fall alle für Ihr Betriebssystem verfügbaren „mainstream"-Browser installieren sollten. Als Windows- oder Mac-User haben Sie dabei die größte Auswahl. Unter beiden Betriebssystemen laufen neben dem Internet Explorer auch Firefox, Opera und Safari. Die MacOS-Version des Internet Explorers ist allerdings etwas in die Jahre gekommen und wird von Microsoft nicht mehr weiterentwickelt. Leider ist es dadurch auch praktisch unmöglich, anhand der Mac-Version verlässliche Aussagen über das Verhalten der Windows-Version zu treffen.

Um auch mit Browsern testen zu können, die für Ihr Betriebssystem nicht zur Verfügung stehen, können Sie sich häufig mit virtuel-

len Maschinen behelfen. Dort installieren Sie dann einfach ein zweites Betriebssystem und können so bequem und ohne einen zweiten Rechner anschaffen zu müssen auf weitere Browser zurückgreifen.

Browser-Downloads und weitere Informationen finden Sie unter WebCode →*browser*. Wo Sie Virtualisierungs-Software herunterladen können, erfahren Sie unter WebCode →*virtualisierung*.

3.1.2 Browser-Erweiterungen für Web-Entwickler

Praktisch alle modernen Browser lassen sich über Plugins mit neuer Funktionalität versehen. Waren diese Plugins bislang vor allem auf normale Anwender zugeschnitten, so gibt es inzwischen auch einige Plugins speziell für Web-Entwickler.

Web-Developer-Toolbars

Ein gutes Beispiel hierfür sind die sogenannten Web-Developer-Toolbars. Sie richten sich im Browser als Symbolleiste ein und bieten Web-Entwicklern interessante Möglichkeiten. So können Sie auf Knopfdruck JavaScript oder CSS deaktivieren, Bilder ein- und ausblenden, die Gültigkeit Ihres HTML-Quelltextes validieren oder sich die HTTP-Antwort-Header Ihrer Seite anzeigen lassen. Web-Developer-Toolbars gibt es inzwischen für Firefox, Internet Explorer und Opera.

DOM-Inspector für den Mozilla Firefox

Eine praktische Erweiterung für Firefox ist der *DOM-Inspector*. Mit ihm lässt sich die Struktur einer HTML-Seite inspizieren. Das funktioniert sogar dann, wenn Seitenelemente dynamisch aus JavaScript heraus erzeugt wurden. Das Add-on wird mit dem Firefox Browser ab Version 1.5 ausgeliefert, standardmäßig jedoch nicht sofort installiert. Wenn Sie Firefox installieren, wählen Sie „benutzerdefinierte“ als Installationsart und vergewissern Sie sich, dass die Option „DOM-Inspector“ ausgewählt ist. Sollten Sie Firefox bereits installiert haben, klicken Sie auf „Extras“ und suchen Sie dort den Menüpunkt „DOM Inspector“. Sollte er fehlen, bleibt Ihnen nichts anderes übrig als Firefox neu zu installieren.

Download-Links zu den Erweiterungen finden Sie unter WebCode →*browsererweiterung*.

3.1.3 JavaScript-Debugger

Für die Sprache JavaScript existieren inzwischen verschiedene Debugger, die entweder als Browser-Plugin oder als eigenständige Anwendung helfen, Scriptfehler aufzuspüren. Im Folgenden werden die vier wichtigsten JavaScript-Debugger vorgestellt.

3.1.3.1 Microsoft Internet Explorer

Script Debugger vs. Script Editor vs. Visual Web Developer

Wenn Sie den Microsoft Internet Explorer verwenden, haben Sie die Wahl zwischen drei verschiedenen Debuggern. Der *Microsoft Script Debugger* kann von der Microsoft Website kostenlos heruntergeladen werden (WebCode →*scriptdebugger*) und bietet eine rudimentäre Debugging-Funktionalität. Leider ist das Programm inzwischen stark in die Jahre gekommen, neigt zu Abstürzen und wird von Microsoft nicht mehr unterstützt. Stattdessen empfiehlt Microsoft den *Microsoft Script Editor*, der in der Tat eine hervorragende Debugging-Funktionalität bietet, oder *Visual Web Developer Express* (WebCode →*visualwebdeveloper*), dessen Debugging-Funktionen weitestgehend mit denen des Script Editors übereinstimmen.

Aus unerfindlichen Gründen kann der Microsoft Script Editor nicht bei Microsoft heruntergeladen werden. Stattdessen wird das Programm ausschließlich mit Microsoft Office ab Version 10 und Microsoft Frontpage ab Version 4 ausgeliefert, dort allerdings bei einer Standardinstallation nicht gleich installiert. Um den Microsoft Script Editor zu installieren, müssen Sie sich bei einer Neuinstallation von Office bzw. Frontpage also vergewissern, dass für das Paket „Microsoft-Skript-Editor“ einschließlich aller Unterpakete die Option „Vom ‚Arbeitsplatz‘ ausführen“ gewählt ist. Das Paket finden Sie in der Kategorie „Office-Tools“.

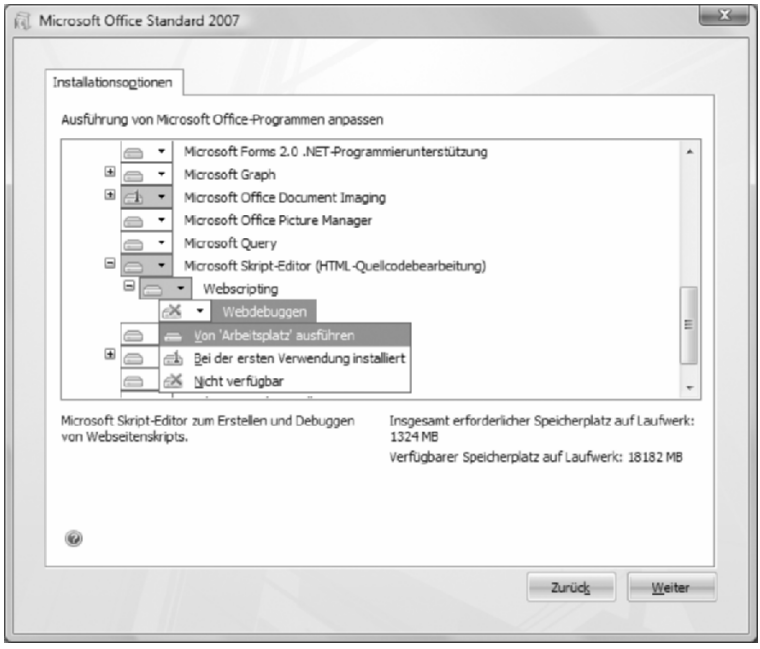

Abb. 3.1 Microsoft Script Editor bei der Office Installation hinzuwählen

Nachträgliche Installation

Wenn Sie Office oder Frontpage bereits installiert haben, können Sie den Microsoft Script Editor auch nachträglich hinzuinstallieren. Hierzu müssen Sie in der Systemsteuerung von Windows den Punkt „Software“ (unter Windows Vista „Programme“ und dann „Programme und Funktionen“) wählen, dort entsprechend Office oder Frontpage selektieren und dann auf „Ändern/Entfernen“ (unter Windows Vista heißt die Schaltfläche nur „Ändern“) klicken. Es öffnet sich das entsprechende Installationsprogramm, über das Sie den Microsoft Script Editor dann analog zur Neuinstallation hinzufügen können.

Die Debugging-Unterstützung im Internet Explorer aktivieren

Ist das Programm einmal installiert, müssen Sie zunächst noch die Debugging-Unterstützung des Internet Explorers aktivieren. Dazu klicken Sie auf den Menüpunkt „Extras“ und wählen dort den Punkt „Internetoptionen“ aus. Klicken Sie in dem nun erscheinenden Dialogfenster dann auf den Reiter „Erweitert“ und entfernen Sie dort unterhalb von „Browsing“ die Häkchen vor „Skripdebugging deaktivieren (Andere)“ und „Skriptdebugging deaktivieren (Internet Explorer)“, falls diese gesetzt sind.

Wenn Sie mit Windows arbeiten und eine Kopie von Office oder Frontpage besitzen, sollten Sie sich den Microsoft Script Editor auf jeden Fall einmal ansehen, selbst wenn Sie normalerweise nicht den Internet Explorer verwenden. Zwar sind die im Folgenden vorgestellten Debugger für Firefox durchaus empfehlenswert, sie reichen jedoch nicht an den Microsoft Script Editor heran.

3.1.3.2 Mozilla Firefox

Venkman aus dem Hause Mozilla

Für den Firefox gibt es momentan zwei JavaScript-Debugger zur Auswahl. Der ältere der beiden heißt *Venkman*. Venkman bietet alles, was man von einem Debugger erwartet und integriert sich sehr gut in den Browser. Das Firefox Add-On wird von Mozilla selbst entwickelt und kann von der Website kostenlos heruntergeladen werden (WebCode →*venkman*).

FireBug ist Debugger und Toolsammlung in einem

Eine populäre Alternative zu Venkman ist *Firebug*. Diese Firefox-Extension ist leichtgewichtig, einfach zu bedienen und bietet neben einem guten JavaScript-Debugger außerdem noch eine Vielzahl weiterer Entwickler-Tools. Darunter befinden sich eine Konsole, ein DOM-Inspektor, ein Netzwerk-Monitor und vieles mehr. Firebug ist wie Venkman kostenlos und quelloffen (WebCode →*firebug*).

3.2 Web-Server

Wahl des Web-Servers

Auch wenn der Web-Server bei einer Ajax-Anwendung je nach deren Aufbau eine unterschiedlich große Rolle spielt, bleibt er doch in jedem Fall unverzichtbar. Welchen Web-Server Sie jedoch wählen, hängt letztlich nur von Ihren Vorlieben und den Anforderungen Ihrer Applikation ab. JavaScript- und HTML-Dokumente ausliefern können praktisch alle Web-Server. Interessant wird es, wenn der Web-Server Anfragen des Clients empfangen und verarbeiten soll. Hierzu muss die Möglichkeit bestehen, auf dem Server Programmcode auszuführen. Ob dies nun in Form eines PHP-Skripts oder einer mehrschichtigen JEE (Java Enterprise Edition) Anwendung erfolgt, spielt dabei, zumindest für den Client, keine Rolle.

Da es sehr viele verschiedene Web-Server gibt und das Thema Web-Server nur ein Aspekt von Ajax ist, kann an dieser Stelle verständlicherweise nicht umfassend darauf eingegangen werden. Stattdessen befassen wir uns, quasi exemplarisch, mit dem Apache Tomcat, der sowohl HTTP-Server als auch Servlet-Container ist. Woher Sie den Apache Tomcat beziehen können und wie Sie ihn einrichten, erfahren Sie im nächsten Abschnitt.

3.2.1 Apache Tomcat

Bezug zum Apache Web-Server

Wie der Name vielleicht erahnen lässt, ist der Apache Tomcat ein Abkömmling des Apache-Webservers, der als Open-Source-Projekt von der Apache Software Foundation entwickelt wird. Tatsächlich wird der Apache Tomcat, obwohl er auch „stand-alone“ eingesetzt werden kann, in Produktivsystemen häufig mit einem vorgeschalteten Apache-Webserver betrieben. Für unsere Zwecke reicht jedoch ein einfacher Tomcat bereits vollkommen aus. Die Installation und Konfiguration gestaltet sich dementsprechend einfach:

Installation von Apache Tomcat

Stellen Sie zunächst sicher, dass Sie ein JDK (Java Development Kit) ab Version 1.5 installiert haben. Sollte dies nicht der Fall sein, finden Sie unter WebCode →*jdk* die nötigen Dateien. Die herkömmliche Java-Runtime ist hier nicht ausreichend.

Laden Sie sich dann den Apache Tomcat für Ihr Betriebssystem herunter (WebCode →*tomcat*) und entpacken Sie das Tomcat-Archiv in einen Ordner auf Ihrer Festplatte. Tomcat erwartet beim Start zwei Umgebungsvariablen, die zum einen den Pfad zu Tomcat selbst und zum anderen zum JDK angeben. Um das Anlegen der Umgebungsvariablen zu automatisieren, können Sie sich unter Win-