Kapitel 13

Datenbanken unter Linux im Internet

Über die stürmische Entwicklung des Internet braucht an dieser Stelle nicht ausführlich berichtet zu werden, den geneigten Lesern wird dies bekannt sein. E-Commerce-Anwendungen schießen wie Pilze aus dem Boden, Online-Buchhandlungen, Web-Auktionshäuser, Informationsbörsen usw. gehören inzwischen zum Alltag und sind oder werden so selbstverständlich wie Telefon und Fernsehen. Viele der neuen Internet-Angebote benötigen als Backend Datenbanken, und inzwischen möchten nicht nur große Konzerne, sondern auch kleinere Unternehmen und sogar private Homepages mit ihrem datenbankgestützten oder datenbankunterstützten Web-Angebot glänzen. So nimmt es nicht wunder, dass Technologien auf großes Interesse stoßen, mit denen es auf einfache Weise möglich ist, vom Web aus auf Datenbanken zuzugreifen.

Wir möchten in diesem Kapitel – in sehr unterschiedlicher Ausführlichkeit – vier hierfür geeignete Technologien vorstellen: Gründlicher besprechen wir JDBC – inzwischen schon Standard für professionelle Datenbankanbindungen im Intranet und Internet, und PHP, etwas jüngerer, aber inzwischen auch sehr weit verbreitet und wegen seiner relativ leichten Erlernbarkeit weithin geschätzt. Nur kurz angesprochen wird demgegenüber der leistungsstarke, aber komplexe Veteran PerlDBI und das noch ganz junge und noch nicht sehr weit verbreitete JSP (Java Server Pages) mit möglicherweise großen Entwicklungsmöglichkeiten.

13.1 JDBC – Java Database Connectivity

Bis vor wenigen Jahren wurde Java noch weitgehend als eine Art von »Web-Spielzeug« eingeordnet, mit dessen Hilfe man in erster Linie seine Homepage ein wenig dynamischer und interessanter gestalten konnte, wobei derartige Homepage-Auflockerungen bezeichnenderweise meistens gar in JavaScript programmiert wurden und nicht in Java. Etwas nachdenklichere Zeitgenossen sahen bereits Möglichkeiten einer effizienteren Kommunikation im WWW, etwa dadurch, dass anstelle großer Grafiken und Diagramme lediglich noch die Rohdaten zu übertragen wären nebst einem Javaprogramm, das daraus die betreffende Grafik oder das betreffende

Diagramm direkt beim Client generiert. Nur wenige allerdings konnten sich vorstellen, dass Java sich einmal zu einem ausgesprochen »seriösen« Paradigma entwickeln könnte, mit dem sich recht komplexe Systeme realisieren lassen.

Binnen kurzer Zeit hat sich die Situation grundlegend gewandelt. Auf Computermessen spielt Java inzwischen eine Hauptrolle, jede Fachzeitschrift mit etwas Anspruch lässt kaum eine Ausgabe vergehen, ohne mindestens einen Artikel zu Java und dessen Umfeld zu bringen. Java-Bücher zieren mittlerweile in großer Zahl die einschlägigen Regale der Buchhandlungen, und das Web quillt über von Dokumenten, die sich mit Java auseinandersetzen. Sun Microsystems schätzte bereits Ende 1997 die Anzahl der Desktops, auf denen mit Java gearbeitet wird, auf 100 Millionen [sun97]. Die Anzahl der Web-Pages, auf denen Java eine Rolle spielt, ist mittlerweile kaum noch zählbar. Insbesondere vor einigen Jahren war Java das Thema auf allen IT-relevanten Veranstaltungen; mittlerweile hat sich die Aufregung gelegt, Java ist in die Praxis eingezogen. Wie ist diese Entwicklung zu erklären? Was ist eigentlich dran an dieser Sprache? Wir werden diese und andere Fragen in diesem Abschnitt diskutieren. Besondere Beachtung wird dabei die JDBC-API finden, die als der initiale Entwicklungsschritt von Java zu einem universell einsetzbaren Werkzeug gesehen werden kann.

13.1.1 Java: Vom Web-Kuriosum zur seriösen Business-Lösung

Die Ursprünge von Java gehen auf das Jahr 1990 zurück. Damals begann ein Team von Sun unter Leitung von James Gosling mit der Entwicklung einer portablen und syntaktisch möglichst einfachen Programmiersprache namens *Oak*, die unabhängig von speziellen (oftmals wechselnden) Chips insbesondere die Steuerung von alltäglichen Gegenständen wie Toaster, Mikrowellengeräte oder auch PDAs (Personal Digital Assistents) übernehmen sollte. Mit der Verbreitung des Internet etwa ab 1993 erkannte das Team sehr schnell die zusätzlichen Möglichkeiten dieser neuen plattformunabhängigen Entwicklung und die neuen Perspektiven, die sich damit ergaben. Sie stellten ihre Entwicklung auf das Internet ab und nannten die auf dieser Grundlage entwickelte neue Programmiersprache *Java*. Der Name kommt von einer in den USA verbreiteten Kaffeesorte (namens Java), die auf der gleichnamigen Insel erzeugt wird. Diese Art der Namensgebung ist auf den bekanntermaßen relativ hohen Kaffeekonsum von Programmierern zurückzuführen. Die erste Version von Java wurde im Jahre 1995 freigegeben (die erste Beta-Version der Spezifikation am 30.10.1995 [jav95]), mittlerweile ist diese Version um fundamentale Erweiterungen ergänzt worden. Inzwischen hat sich Java zu einem umfassenden

technologischen Konzept entwickelt. Die Hauptintention aller von Sun Microsystems bzw. dessen Ableger JavaSoft betriebenen Neuentwicklungen besteht unverändert darin, das Netzwerk zur zentralen Komponente der Datenverarbeitung werden zu lassen, während der einzelne Rechner zu einer peripheren Angelegenheit wird.

Im Kern besteht Java aus drei Komponenten (z. B. [dal97]), und zwar:

- Die *Programmiersprache Java:* Diese beschreibt das semantische und syntaktische Konzept von Java. Hinsichtlich der Programm- und Datenstrukturen ähnelt Java weitgehend C++.
- Die *Java Virtual Machine:* Die Java Virtual Machine (JVM) ermöglicht die Ausführung von Programmen auf unterschiedlichen Plattformen. Javaprogramme werden nicht für die Ausführung auf speziellen Rechnertypen mit speziellen Prozessortypen oder speziellen Betriebssystemen entwickelt, sondern mit dem Ziel einer möglichst vollständigen Portierbarkeit (»write once – run everywhere«). Dies wird erreicht durch die Erzeugung von Bytecode (anstelle von Maschinencode) bei der Kompilierung von Javaprogrammen.
- Die Entwicklungsumgebung: Die Standardentwicklungsumgebung von Java bildet das *Java Development Kit* (*JDK*). Dieses beinhaltet im Kern neben einem Compiler (*javac*), einer virtuellen Maschine (*java*) und einem Debugger (*jdb*) einen Applet-Viewer (*appletviewer*), einen Generator für Dokumentationen (*javadoc*), einen Generator für native C-Methoden (*javah*) und einen Klassendisassembler (*javap*). Mittlerweile ist diese Standardentwicklungsumgebung durch spezielle Entwicklungs- und Laufzeitwerkzeuge ergänzt worden, etwa durch den RMI-Compiler (*rmic*) oder das Java Beans Development Kit. In einer modernen (und für die Entwicklung effizienten) Java-Entwicklungsumgebung spielt das JDK allerdings lediglich noch eine untergeordnete Rolle. Es gibt mittlerweile recht komfortable Entwicklungsumgebungen, die all die genannten Werkzeuge neben anderen beinhalten und die ein recht bequemes Entwickeln von Javaprogrammen nebst der benötigten grafischen Benutzeroberflächen ermöglichen.

Auf einschlägigen Java-Veranstaltungen mussten die Linux-Vertreter lange Zeit nachfragen, wann denn eine Portierung insbesondere des JDK nach Linux zu erwarten sei; erst im Rahmen der JAVADAYS '99 in Düsseldorf verlautete von Sun Microsystems (in Europa), dass für die Linux-Benutzer ein JDK erhältlich sein wird. Bis dahin waren die Java-Basiskomponenten von Sun lediglich für Solaris und Windows NT erhältlich, eine Linux-Variante wurde regelmäßig kurze Zeit später von der Blackdown-Initiative zur Verfügung gestellt [black].

In der nachfolgenden Tabelle sind die wichtigsten Kernkomponenten von Java und deren Aufgaben zusammengefasst.

Tabelle 13.1: Einige wichtige Bestandteile des JDK und deren Aufgaben

JDK-Komponente	Aufgabe
javac	Kompilieren von Java-Source Code in Bytecode
java	Ausführen von Java-Bytecode
jdb	Debuggen
appletviewer	Darstellung von Applets
javadoc	Erstellen von Dokumentationen
javah	Generierung nativer C-Methoden
javap	Disassemblierungen
rmic	Kompilieren spezieller RMI-Komponenten

Java wird gelegentlich immer noch auf eine Sprache für die Programmierung von Applets verkürzt. Zutreffend dagegen ist, dass Java semantisch vollständig ist. Mit Hilfe dieses Konzepts lässt sich im Prinzip jede Anwendung programmieren. Diese Eigenschaft ist durchaus nötig, denn gerade im Zusammenhang mit der JDBC-API werden sich oftmals umfangreiche Programmanteile aus den unterschiedlichsten Gründen nicht in Applets befinden, sondern z. B. auf einem Application Server oder in Servlets.

Die Vorzüge von Java sind in vielen Büchern und Artikeln aufgezählt worden. Hier noch einmal kurz eine Zusammenfassung [dic97]:

- Java ist *objektorientiert*: Die grundlegenden Ideen der objektorientierten Programmierung existieren bereits seit dreissig Jahren. In der Praxis hat diese Form der Softwareentwicklung allerdings erst seit Mitte bis Ende der achtziger Jahre infolge der seinerzeit oftmals zitierten »Softwarekrise« Einzug gehalten. Java verfolgt dieses Paradigma außerordentlich konsequent.
- Java ist *einfach*: Im Vergleich zu anderen objektorientierten Sprachen wie C++ oder Smalltalk ist die Syntax und Semantik von Java recht einfach und daher vergleichsweise leicht zu erlernen. So existieren beispielsweise in Java keine Zeiger, diese wurden konsequenterweise durch objektorientierte Lösungen ersetzt.
- Die Spracheigenschaften von Java sind in vielen Fällen *bekannt*: Viele der sprachlichen Möglichkeiten von Java sind an die verbreitete objektorientierte Sprache C++ angelehnt. Entwickler, die mit C oder C++ vertraut sind, haben daher kaum Schwierigkeiten beim Umstieg auf Java.

- Java ist *robust*: Java-Entwickler sind im Allgemeinen nicht genötigt, die Speicherverwaltung ihrer Systeme selber zu programmieren, dies erledigt statt dessen das Java-Laufzeitsystem. Die Entwickler sind nur noch zuständig für die Allokation von Speicherplatz durch die Instanziierung von Objekten, die Freigabe des belegten Speicherplatzes erfolgt durch eine automatische Garbage Collection. Wie andere Sprachen beinhaltet auch Java eine Syntaxüberprüfung bei der Kompilierung und einen zweiten Check zur Ausführungszeit. Die Syntaxprüfungen bei der Kompilierung sind weitgehend so ausgelegt, dass die Entwickler zu einem Stil angehalten werden, der die Zuverlässigkeit des Systems geradezu erzwingt.
- Java ist *sicher*: Ein gerade in jüngerer Zeit häufig diskutiertes Thema betrifft die Sicherheit beim Datenaustausch im Internet. Es werden immer wieder Befürchtungen laut, dass unerwünschte entfernte Zugriffe z. B. auf Systemressourcen verheerende Folgen zeitigen können. Java besitzt eine integrierte Sicherheitskomponente. Die wichtigste Auswirkung dieses Sicherheitsanspruchs besteht darin, dass ein Java-Applet extremen Zugriffsbeschränkungen unterliegt. Ein Java-Applet hat (von geregelten Ausnahmen im Zusammenhang mit TRUSTED APPLETS abgesehen) praktisch keinerlei Möglichkeiten des Zugriffs auf Dateien oder Ressourcen des Clients. Die Sicherheit von Java wurde auf Grund einiger Vorkommnisse in der Vergangenheit gelegentlich in Frage gestellt. Diese Diskussionen (bzw. die Reflexion auf Java) waren allerdings entweder die Folge mangelnder Kenntnisse (beispielsweise wurden erhebliche Sicherheitslücken von Microsoft ActiveX aufgedeckt und kurzerhand gleich mit Java in Verbindung gebracht, obwohl ActiveX von Java völlig verschieden ist) oder infolge lückenhafter Realisierungen der Java-Sicherheitsanforderungen in Browsern. Niemand dagegen hat bislang eine Sicherheitslücke in der Java-Spezifikation entdeckt.
- Java ist *portierbar*: In Java realisierte Systeme sind auf jeder Plattform lauffähig, für die ein Java-Interpreter existiert, und es gibt kaum noch ein Betriebssystem, für das dies nicht zutrifft. Die Portierbarkeit wird dadurch erreicht, dass Java-Software nicht in Maschinencode kompiliert wird, sondern in Bytecode, der anschließend plattformunabhängig von einer Java Virtual Machine interpretiert werden kann. Infolge der zunehmenden Integration von Java in diverse plattformspezifische Systeme wird die Frage der Portierbarkeit allerdings zunehmend auf die Entwickler verlagert. Mit Sicherheit (komplett) portierbar sind nur solche Systeme, die vollständig in Java implementiert werden. Da aus ökonomischen Gründen oftmals bestehende Komponenten in Java integriert werden müssen, sollte mindestens darauf geachtet werden, dass solche Komponenten

in Java realisiert werden, die von Clients heruntergeladen werden. Andernfalls wären sogar beim Client plattformspezifische Komponenten zu installieren. Ein solches Vorgehen führt jedoch neben einem nennenswerten Installations- und Administrationsaufwand auch zu einem merklichen Mangel an Flexibilität.

- Java ist *Internet- und Intranetfähig*: Die Philosophie von Java beruht zu einem großen Teil auf der Absicht, Rechnerlast vom »benutzereigenen« Desktop oder von der Workstation auf das Netzwerk zu verlagern. Nicht nur bei Sun, sondern auch bei IBM oder Apple sehen die Zukunftsperspektiven dergestalt aus, dass der gewöhnliche Anwender auf seinem Client-Rechner (wenn überhaupt) lediglich noch ein paar Standardwerkzeuge wie etwa Textverarbeitungssoftware installiert hat. Die meisten Anwendungen dagegen werden vom Netzwerk aus gestartet. Das »Netzwerk« ist in diesem Fall entweder das Intranet, auf das nur speziell autorisierte Benutzer zugreifen dürfen, oder das Internet, das einen beliebigen Zugriff erlaubt. Beispielsweise im Zuge der Entwicklung von Embedded Systems oder E-Commerce-Anwendungen spielt diese Sprache eine zentrale Rolle.
- Java ist *benutzerfreundlich*: Jeder, der im Web surfen kann, ist in der Lage, Java-Applets zu benutzen. In komplexen Systemen sind Java-Applets aber meist keine selbständigen und unabhängigen Komponenten, sondern sie sind oftmals Bestandteile von möglicherweise recht komplexen Anwendungen im Netz, die der Benutzer nicht wahrnimmt und mit denen er sich somit auch nicht beschäftigen muss.

Java hat insbesondere durch seine ausgezeichnete Netzfähigkeit und durch die Eigenschaft der Portierbarkeit Türen zu völlig neuen Anwendungsformen aufgestoßen. Die Migration von in anderen Sprachen realisierten Systemen nach Java wird durch einige Anbieter in Form von Code-Umsetzern unterstützt, JavaSoft selber bietet mit dem *Java Native Interface* (JNI) und insbesondere mit einer CORBA-Schnittstelle die Möglichkeit der Integration bestehender Software in Java. Ein JNI wird z. B. auch von Oracle für unterschiedliche Anwendungen eingesetzt (vgl. Kapitel 5).

13.1.2 Die Integration von Datenbanken mit JDBC

JDBC steht für JAVA DATABASE CONNECTIVITY. JDBC ist eine API, die entwickelt wurde, um den Bedarf nach einer standardisierten Java-Anwendungsschnittstelle zu relationalen Datenbank-Managementsystemen zu befriedigen. JDBC ermöglicht den Zugriff auf Datenbanken innerhalb von Java-Anwendungen; diese

Zugriffsmöglichkeiten sind unabhängig sowohl von der Plattform, auf der die Anwendung läuft, als auch vom zu Grunde liegenden Datenbanksystem. Die JDBC-Spezifikation ermöglicht das Erstellen von Java-Code zum Herstellen einer Datenbankverbindung, zum Verschicken von SQL-Statements an die Datenbank und zum Retrieval von Anfrageergebnissen, Metadaten und Statusmeldungen der Datenbank. Daneben beinhaltet JDBC Möglichkeiten der Konversion von SQL-Datentypen zu Javadatentypen und umgekehrt, zur Behandlung von SQL-Cursors, zum Transaktionsmanagement, zur Benutzung von Stored Procedures (das sind in der Datenbank gespeicherte vorkompilierte Prozeduren) und einiges mehr.

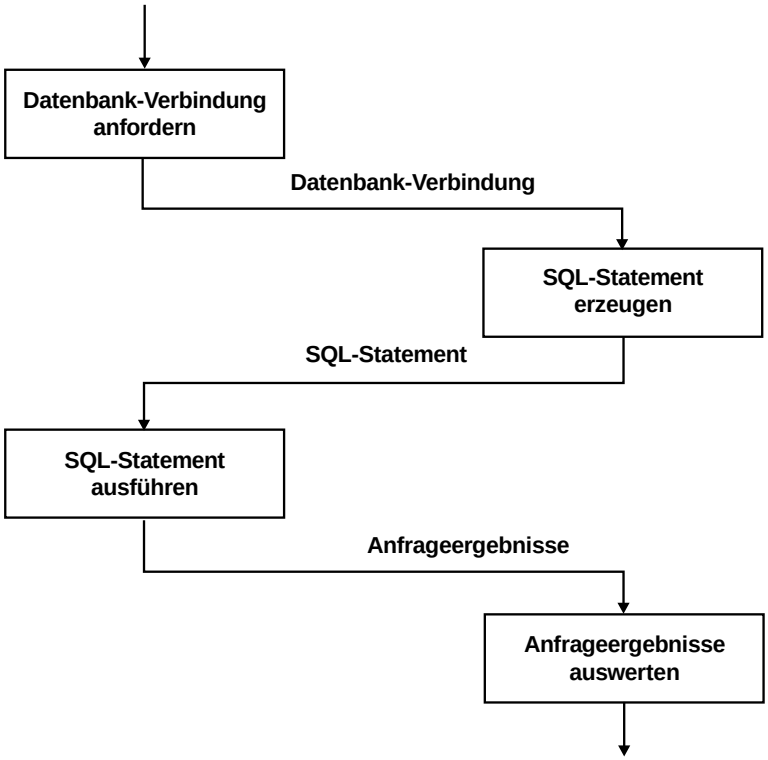


Abb. 13.1: Einige Möglichkeiten der JDBC-API in einem Ablaufszenario

»Prior to the JDBC specification, I think that a lot of people in the industry thought of Java as 'that little internet language that can make that cute little icon dance on a web site.« Diese Aussage von Karl Moss, der im Jahre 1996 bei Intersolv maßgeblich an der Entwicklung der JDBC-ODBC-Bridge beteiligt war, in einem Interview der elektronischen Zeitschrift Javology [jav97] trifft den Kern recht gut. Die Möglichkeit der

Anbindung von Datenbanken in Internet- oder Intranet-Anwendungen bestand selbstverständlich auch schon vor Java und JDBC, nämlich über CGI. Die CGI-Schnittstelle des Web besitzt allerdings im Vergleich zu Java mehrere wesentliche Nachteile. Einige davon sind:

- In CGI lassen sich nur sehr wenige Standardformen der Interaktivität benutzen.
- CGI-Programme sind plattform- und datenbankabhängig.
- Eine standardisierte oder quasi-standardisierte CGI-Schnittstelle zu Datenbanken existiert nicht. Es gibt kein einheitliches Konzept für den Datenbankzugriff über CGI.

Die JDBC-API war die erste Programmierschnittstelle, die Java zu einem kommerziell nutzbaren Werkzeug werden ließ. Inzwischen gibt es etablierte zusätzliche Konstrukte wie z.B die *Remote Method Invocation* oder *Enterprise Java Beans*, welche die Palette von Anwendungen nochmals wesentlich erweitern.

Die Entwicklung der JDBC-API geschah sehr rasant: Das JDBC-Projekt wurde bei JavaSoft im Januar 1996 initiiert, eine erste Testversion wurde im März 1996 freigegeben, die erste Spezifikation bereits im Juni 1996. Dieses Tempo war deshalb möglich, weil JavaSoft sich bei der Entwicklung der JDBC-API in der ersten Version in hohem Masse an der bereits etablierten ODBC-API orientierte. Die Strategie von JavaSoft war eine möglichst rasche Markteinführung von neuen Java-bezogenen Produkten, damit sich die Hersteller ergänzender Werkzeuge entsprechend früh mit den Neuerungen auseinandersetzen konnten. Die Gefahr dabei war, dass in vielen Fällen erst die zweite oder dritte Version eines Produktes wirklich marktfähig war. Allerdings gibt es dann oftmals gleich auch schon die entsprechenden Werkzeuge anderer Anbieter. Bislang hat sich diese Strategie als sehr tragfähig erwiesen. Die JDBC-API ist mittlerweile im Datenbankumfeld ein Standard, der mit ODBC durchaus vergleichbar ist. Es gibt sehr viele umfangreiche Klassenbibliotheken und hochintegrierte Werkzeuge, die auf der JDBC-API basieren.

ODBC bildete seit langem die wahrscheinlich verbreitetste Programmierschnittstelle zu Datenbanken. Sie bietet eine Schnittstelle zu allen verbreiteten Datenbanksystemen auf fast allen Plattformen. Damit aber stellt sich die Frage, wozu JDBC eigentlich noch benötigt wird. Einige gute Argumente für JDBC finden sich in der JDBC-Dokumentation von JavaSoft:

ODBC benutzt eine C-Schnittstelle zu Datenbanken. Komponenten, die nicht in Java geschrieben sind, können aber nicht per WWW zu einem Client geladen werden.

Eine direkte Übersetzung der ODBC nach Java wäre nicht angemessen, denn ODBC benutzt relativ viele spezielle Features, die es aus guten Gründen in Java nicht gibt (wie etwa Zeiger).

ODBC ist schwer zu erlernen, denn es vermischt einfache Dinge mit komplexen und es besitzt in vielen Fällen zahlreiche komplizierte Optionen auch für sehr einfache Anfragen. JDBC dagegen wurde auch mit dem Ziel entwickelt, »einfache Dinge einfach zu halten, während erweiterte Möglichkeiten dort vorgesehen sind, wo sie benötigt werden« (JavaSoft).

Würde man ODBC benutzen, dann müssten die benötigten Treiber auf jeder Client-Maschine installiert werden. Benutzt man dagegen reine Java-Treiber, dann können sämtliche benötigten Komponenten zum Client geladen werden. Auch bei dieser Argumentation ist allerdings zu beachten, dass es Architekturlösungen gibt, bei der dieses Argument keine Rolle spielt.

JDBC ist eine Low-Level-API (SQL-Level), d.h. sie hält lediglich Basismethoden vor, die einen einheitlichen Datenbankzugriff, den Umgang mit rohen SQL-Statements und das Retrieval von Resultaten innerhalb von Javaprogrammen ermöglicht. Sie basiert auf dem X/Open *Call Level Interface* (CLI), in dem definiert ist, wie Client/Server-Aktionen für Datenbanksysteme implementiert sind. JDBC wurde bewusst auf einer niedrigen Ebene angesetzt, um einerseits die flexible Entwicklung spezifischer Systeme zu erlauben und um andererseits anderen Anbietern die Einführung von Lösungen auf höherem Level zu ermöglichen.

Die Behauptung der Plattform- und Datenbankunabhängigkeit der JDBC bedarf einiger Einschränkungen. Der Zugriff von Javaprogrammen auf Datenbanken geschieht über so genannte JDBC-Treiber. Die Treiber aber sind von Datenbank zu Datenbank verschieden. Diese Komponente der Datenbankabhängigkeit bezieht sich aber nicht auf die eigentliche Anwendung, denn man kann dem Client entweder die Möglichkeit geben, die Treiber herunterzuladen, sofern sie in Java implementiert sind, oder man kann sie durch eine entsprechende Wahl der Systemarchitektur auf einem Applikationsserver kapseln. In beiden Fällen merkt der Client nicht, mit welcher Datenbank er es zu tun hat. Durch ein solches Vorgehen wird gleichzeitig auch die Plattformunabhängigkeit der Anwendung gewahrt. Es gibt ferner einige datenbankspezifische SQL-Features, wobei Entwickler natürlich in die Lage versetzt werden sollten, diese in ihren Javaprogrammen auch zu benutzen, denn sonst wären sie oftmals dazu gezwungen, auf effiziente und einfache datenbankspezifische Möglichkeiten zu verzichten. Auch dies aber betrifft nicht die eigentliche Anwendung, es handelt sich vielmehr lediglich um eine Frage der Implementierung.

Datenbanksysteme unterstützen ein weites Feld von SQL-Syntax und Semantik-Features, die in vielen Fällen nicht zwischen einzelnen Datenbanksystemen kompatibel sind. Beispielsweise sind die Benutzung der *Outer-Join*-Operationen oder *Stored Procedures* oftmals recht datenbankspezifisch. Diese Gegebenheiten waren bei der Spezifikation der JDBC-API von Javasoft zu beachten. Die wichtigsten Folgen sind:

- JDBC lässt jede Anfrage zunächst einmal zur Datenbank durch, d.h. die JDBC-API selber überprüft die SQL-Statements nicht. Eine Anwendung kann daher zunächst einmal beliebige SQL-Queries beinhalten. Die Fehlermeldungen kommen dann vom DBMS und werden an die Anwendung übergeben. Der Nachteil dieser Philosophie besteht darin, dass Fehler oftmals erst zur Laufzeit entdeckt werden. Beim Auffüllen größerer Tabellen beispielsweise kann dies schon mal zu einem Ärgernis werden, insbesondere dann, wenn sich nach einem möglicherweise mehrstündigem Programmlauf zeigt, dass die Datenbankkonfiguration nicht den Erfordernissen entspricht. Die Datenbankunabhängigkeit wird bei diesem Vorgehen generell durch einen erhöhten Entwicklungsaufwand erkauft. Ich spreche hier aus leidvoller Erfahrung; dies ist ein Grund, weshalb der Konfiguration der Datenbank in diesem Buch ein eigenes Kapitel gewidmet ist.
- Treiber, die den Status »JDBC compliant« tragen wollen, müssen mindestens den ANSI SQL92-Standard unterstützen. Damit wird das größtmögliche Feld an unterschiedlichen Datenbanken erfasst. Java-Anwendungen sind zwischen allen Datenbanksystemen portierbar, die ebenfalls auf diesem Standard aufsetzen. Der ANSI SQL92-Standard wird derzeit von den meisten Datenbankherstellern unterstützt.

13.2 PHP: PHP Hypertext Preprocessor

PHP ist eine Programmiersprache zur Erstellung dynamischer Web-Sites. Die Abkürzung PHP steht für »PHP: Hypertext Preprocessor«. Dies ist ein selbstrekursives Akronym im Stil von GNU (»Gnu's not Unix«). PHP gibt sich damit als Open-Source-Projekt zu erkennen. PHP ist freie Software im Sinne der Debian Free Software Guidelines (DFSG). Quelltext und Binaries des PHP-Interpreters sind frei erhältlich und können für kommerzielle und nicht kommerzielle Zwecke eingesetzt werden.

PHP ist eine Skriptsprache, deren Anweisungen in den HTML-Code einer Web-Seite eingebettet werden. Viele der syntaktischen Möglichkeiten sind den Programmiersprachen C, Java und Perl entnommen. Das Ziel der Sprache ist es, das Schreiben von Programmen zur Erzeugung von dynamisch generierten Web-Seiten zu erleichtern und zu beschleunigen. Im Gegensatz z. B. zu JavaScript werden PHP-Skripte bzw. in HTML-Seiten eingebettete PHP-Anweisungen serverseitig ausgewertet, entweder durch den Web-Server (z. B. durch den Web-Server Apache mit *mod_perl*) oder als CGI-Programm. PHP zeichnet sich durch einen sehr großen Funktionsumfang aus. Besonders bemerkenswert ist die Unterstützung sehr vieler Datenbanken über deren natives API (und nicht nur über den anerkannt schlechten Standard ODBC). Damit empfiehlt sich PHP besonders für Internet-basierte Datenbankapplikationen und damit auch für E-Commerce-Anwendungen.

Derzeit vollzieht sich bei PHP ein Entwicklungs- und Qualitätssprung, der sich auch in einer Erhöhung der Versionsnummerierung von 3 nach 4 ausdrückt. Noch wird von »offizieller Seite« empfohlen, in Produktionsumgebungen die Version 3.0.16 zu verwenden. Die Rückmeldungen aus den Mailinglisten lassen aber den Schluss zu, dass auch die neue Version 4 (PHP 4 Release Candidate 1) inzwischen sehr stabil und zuverlässig läuft. Mit der Freigabe der endgültigen PHP-Version 4 ist sicher bis zum Frühsommer 2000 zu rechnen.

Zu PHP gibt es eine hervorragende Dokumentation. Die Original-Dokumentation findet sich auf der PHP-Heimatseite. Von dem deutschsprachigen PHP-Portal `http://www.php-center.de` können darüber hinaus auch Teilübersetzungen der Originalsoftware heruntergeladen werden (vielen Dank von dieser Stelle aus an Egon Schmid, für die Übersetzung und ganz besonders für seine Hilfsbereitschaft in der PHP-Mailingliste!). Auf den Web-Seiten zu diesem Buch finden Sie Links zu zwei Versionen der HTML-Originaldokumentation: zum Standard-Manual, bei dem jedes Kapitel(chen) in einer eigenen HTML-Seite steht (1288 Dateien), und zu einer Version, in welcher der gesamte Inhalt in einer großen Datei steht (*Bigfile*, nützlich, wenn mit Hilfe der Suchen-Funktion des Web-Browsers nach Begriffen gesucht werden soll). Zum Ausdrucken eignen sich die PDF-Versionen der Dokumentation.

In seiner Aufgabenstellung konkurriert PHP mit dem freien Perl (mod_perl für Apache in Kombination mit Embperl und dem PerlDBI, vgl. hierzu Kapitel 13.3) sowie mit den kommerziellen Produkten ASP (Microsoft) und Coldfusion (Allaire). Künftig wird möglicherweise auch Java Server Pages (JSP; Apache-Jakarta, GNUJSP u. a.) eine Rolle spielen (vgl. Kapitel 13.4). Eine knappe, aber kenntnisreiche und fundierte Auseinandersetzung mit den Nachteilen und Vorzügen der der-

zeit verfügbaren Systeme findet sich in Kapitel 2 der FAQ der Newsgroup *de.comp.lang.php* [kor00]. Die dort vertretene Position wollen wir uns zu Eigen machen und hier nur kurz und platt in zwei (etwas polemischen, aber dennoch wahren) zusammenfassenden Sätzen wiedergeben: Während Cold Fusion unangenehm durch das verkrampfte Bemühen auffällt, sich Anfängern nicht als Programmiersprache zu erkennen zu geben, Datenbankzugriffe nur über ODBC ermöglicht und als kommerzielles Produkt die Möglichkeit verweigert, den Quelltext einzusehen und zu verändern, weiß Microsofts ASP durch die Unterstützung nur einer einzigen Plattform und durch die herstellertypischen Support-, Preis-, Lizenz- und Qualitätsmerkmale zu missfallen. Vorteil von Perl sind (für den *erfahrenen* Programmierer) die Ausdrucksstärke und Vielfalt der Sprache sowie die umfassende Sammlung an Bibliotheken (die mit *mod_perl* komplett für Web-Applikationen zur Verfügung stehen); nachteilig sind der hohe Speicherverbrauch (besonders bei größeren Projekten) und die für Anfänger schwer zu überschauende Vielfalt von Sprachkonstruktionen und Bibliotheken.

In diesem Kapitel soll die Möglichkeit beschrieben werden, mit PHP Internet-gestützte Datenbankapplikationen zu erstellen. Wir werden – relativ knapp – die Installation von PHP beschreiben und anschließend die Möglichkeiten der PHP Datenbank-Interfaces erörtern. Anhand einer Beispielanwendung wird gezeigt, wie über das Web auf eine Datenbank schreibend und lesend zugegriffen werden kann. Wir beziehen uns dabei in der Regel auf das für Internet-Anwendungen besonders geeignete DBMS MySQL. Wir stellen Ihnen darüber hinaus Möglichkeiten der Datenbankabstraktion vor, sodass bei der Entwicklung der Web-Applikation auf die letztlich verwendete Datenbank keine Rücksicht genommen werden braucht. Damit Sie einen Überblick bekommen, möchten wir aber eine knappe Deskription der Funktionen von PHP liefern, mit denen wir uns im Weiteren nicht befassen werden (weil sie für Datenbankanbindung nicht erforderlich sind).

13.2.1 PHP Funktionsüberblick (ohne Datenbankfunktionen)

Das Kapitel Funktionsreferenz der PHP-Dokumentation hat 61 Unterkapitel. In jedem Unterkapitel werden Funktionen zu einem Themenbereich beschrieben. So enthält z. B. das Unterkapitel LV die 62 -dokumentierten, in PHP verfügbaren String-Funktionen. Die Dokumentation hinkt den tatsächlich implementierten Funktionen meistens (wenig) hinterher. Wenn Sie eine spezifische Funktionalität vermissen, ist es durchaus möglich, dass bereits an der Implementierung gearbeitet wird. Sie können dies nur herausfinden, indem Sie entweder den Quelltext in-

spizieren oder in den (auch zu Anfängern sehr freundlichen) Mailinglisten oder der Newsgroup nachfragen. Im Folgenden referenzieren wir eine zugegebenermaßen sehr subjektive Auswahl der uns besonders wichtig oder interessant erscheinenden Funktionen. Auf selbstverständlich auch vorhandene Standardfunktionen wie z. B. String-, Array-, Verzeichnis- und Dateisystem-, Programmausführungs-, Kalender- oder arithmetische Funktionen wird hier nicht eingegangen.

1. *Rechtschreibprüfung (Aspell):* Erlaubt es, ein Wort auf korrekte Rechtschreibung zu prüfen und Alternativen anzubieten.
2. *Grafikfunktionen (Image):* Sie können die Grafikfunktionen von PHP nicht nur benutzen, um die Größe von JPEG, GIF und PNG-Bilddateien zu ermitteln, sondern auch um Grafiken bzw. Bilder dynamisch (also zur Laufzeit ihres Skripts) zu erzeugen bzw. verändern. Sie möchten die schnöden Zahlen in Ihrer Datenbank mit Hilfe von Balken-, Torten- oder sonstigen Diagrammen visualisieren? Die Image-Funktionen (bzw. frei erhältliche, auf ihnen aufsetzende Libraries) ermöglichen dies.
3. **PDF-***Funktionen (ClibPdf/Pdf/FDF):* Ermöglicht die programmgestützte Erstellung von PDF-Dateien on-the-fly. Unter anderem stehen folgende Möglichkeiten zur Verfügung: Zeichensätze auswählen und deren Größe und Attribute manipulieren, Texte verfassen, Objekte (Linien, Kreise, Rechtecke, Polygone usw.) zeichnen, Pixelgrafiken im JPG- oder GIF-Format einbinden, Dokumentinformationen, Lesezeichen und Anmerkungen setzen u.v.a. Auch auf die vielfach noch unbekannten erweiterten Möglichkeiten des PDF-Formats (Erstellung und Auswertung von Formularen/Fragebogen) kann zugegriffen werden.
4. *FTP-Funktionen (FTP):* Alle Standardfunktionen eines FTP-Clients stehen zur Verfügung.
5. **E-***Mail, Newsgroups (POP3, IMAP, NNTP):* Ermöglicht die Entwicklung von Applikationen mit Funktionen eines E-Mail-Clients (sowohl das noch bestehende Standardprotokoll POP3 als auch das neue IMAP werden unterstützt) bzw. eines Newsreaders für Usenet Newsgroups.
6. *Perl Regular Expressions (PReg):* Stellt den gesamten Leistungsumfang (aber leider auch die gesamte Komplexität) der Perl Regular Expressions zur Verfügung (ermöglicht äußerst flexible und komplexe Suchen-Ersetzen-Operationen).
7. *ASCII-Konvertierungen (GNU Recode):* Die sehr umfangreichen Möglichkeiten des GNU Recode Tools zur Zeichenkonvertierung werden zur Verfügung gestellt (Informationen liefert ein *info recode* auf der Konsole; Wechselseitige Konvertierungen zwischen Windows, Macintosh, Unix, älteren Großrechnerformaten, LaTeX, HTML usw.).

8. *XML-Dokumentformat (XML):* XML (eXtensible Markup Language) ist der kommende Standard für Web-Dokumente. Er wurde vom World Wide Web Consortium (W3C) definiert, und sogar Microsoft hat angekündigt, sich daran halten zu wollen. Die Möglichkeiten von XML (Repräsentation von Datenbankstrukturen, Definition von Formatieranweisungen für unterschiedliche Ausgabemedien wie Print, Web usw.) weisen sehr weit über HTML hinaus. Die nächste Browser-Generation wird in der Lage sein, XML-Dokumente zu lesen.
9. *Session-Unterstützung (Sessions):* Ermöglicht es, Benutzer-Sessions zu erstellen, d.h. Daten für einzelne Benutzer – über die einzelne Web-Seite hinaus – abzuspeichern und zur Verfügung zu stellen (eine essenzielle Fähigkeit für jeden Web-Shop, aber auch für viele andere Anwendungen).

13.2.2 PHP-Installation

Es gibt zwei prinzipiell verschiedene Möglichkeiten, PHP zu verwenden: Einerseits können PHP-Skripten in alter CGI-Manier aufgerufen werden, indem bei jedem Aufruf einer PHP-Seite der PHP-Interpreter gestartet wird, der die PHP-spezifischen Teile der Web-Seite abarbeitet. Linux-seitig muss bei jedem Seitenzugriff zunächst ein Prozess erzeugt und der PHP-Interpreter in diesen Prozess geladen werden. Speicher muss alloziert und wieder freigegeben, Filehandels und gegebenenfalls Datenbankverbindungen müssen geöffnet und ebenfalls wieder geschlossen werden. Dies alles verbraucht eine ganze Menge Systemressourcen. CGI-PHP hat zwei Vorteile: Ein Massen-Web-Hoster kann nur mit einem CGI-PHP Sicherheit für jeden seiner Kunden (vor den anderen Kunden) gewährleisten. Außerdem kann ein CGI-PHP-Interpreter genau wie ein Perl- oder Python-Interpreter zur Ausführung von PHP-»Shell«-Skripten auf der Konsole genutzt werden.

Andererseits kann PHP auch als Modul, z.B. in einem Apache-Web-Server installiert werden. Dann ist das PHP Bestandteil des Web-Servers und ständig geladen. Datenbanklinks können über die Lebensdauer einer PHP-Seite hinaus offen gehalten werden (*datenbank_pconnect()*), was insbesondere dann wichtig ist, wenn Sie als Datenbank-Backend Oracle verwenden. (Der Aufbau von Netzwerkverbindungen ist bei Oracle sehr aufwendig.)

Wenn Sie nicht gerade ein Massen-ISP sind, sollten Sie sich beide Verfahren ermöglichen, indem Sie sowohl ein PHP-Modul als auch ein CGI-PHP kompilieren und installieren. Wenn Sie mit PHP erst einmal ein wenig vertraut sind, werden Sie auch bei der Shellprogrammierung oder bei der Erstellung von Quick-and-Dirty-Skripten zur Problemchenlösung gerne auf PHP zurückkommen (und dann brau-

chen Sie PHP als separates Binary). Wir werden im Folgenden die Installation von PHP als Web-Servermodul beschreiben. Wenn diese Installation gelingt, gelingt auch die CGI-PHP-Installation problemlos. Um PHP als Modul zu installieren, benötigen Sie einen Web-Server, mit dem dies möglich ist. Der Web-Server Apache harmonisiert hervorragend mit PHP, weswegen wir ihn in einem kurzen Exkurs vorstellen möchten.

Exkurs: Apache Web-Server

Der Apache Web-Server ist das Paradebeispiel für ein sehr erfolgreiches Open-Source-Projekt. Dieser Freeware-Server surft allen davon: Um den Faktor drei ist Apache im Bereich der Server-Software jetzt dem Konkurrenten Microsoft überlegen. Im April 2000 sind laut Untersuchung von Netcraft (`http://www.netcraft.com/survey/`) 62 Prozent der Rechner mit dem Apache Web-Server ausgerüstet. Microsoft IIS läuft demnach gerade noch auf 21 Prozent der Server. Netcraft hat nach eigenen Angaben 14,3 Millionen Sites für die Untersuchung herangezogen. Den dritten Platz belegt Netscape Enterprise mit 6,9 Prozent.

Der Web-Server Apache zeichnet sich aber nicht nur durch seine Verbreitung aus (verbreitet ist so manche Software), sondern auch durch ausgezeichnete Qualität und insbesondere durch eine sehr hohe Stabilität (diese beiden Eigenschaften können nicht alle weit verbreiteten Softwareprodukte für sich in Anspruch nehmen). Im April 2000 (Ausgabe 8) testete die Computer-Zeitschrift c't die Erreichbarkeit von Web-Servern nach Software. Ihr Ergebnis: Der Apache Web-Server unter Unix (Linux und Solaris) erwies sich im Test als fünfmal besser erreichbar als der Microsoft Internet Information Server (IIS) unter Windows NT.

Die c't überprüfte in einem breit angelegten Test 32 Tage lang alle zehn Minuten die Erreichbarkeit von über 100 der meist besuchten deutschen Web-Server und zählte Häufigkeit und Dauer der Ausfälle. Bei der Auswertung der Ausfallzeiten pro Betriebssystem schnitten die Microsoft Windows-NT-Server deutlich schlechter ab als ihre Konkurrenten aus dem Unix-Lager. Sie waren im Schnitt während der Dauer des Tests knapp 15 Stunden nicht erreichbar, also etwa 1,9 Prozent der Zeit. Vor allem das Wochenende schlägt sich bei NT in deutlich erhöhten Ausfallzeiten nieder. Offensichtlich erfordern Web-Server unter NT eine intensivere Betreuung durch Administratoren. Systeme mit dem frei erhältlichen Betriebssystem Linux waren im Durchschnitt vier Stunden und damit 0,5 Prozent der Zeit nicht zu erreichen. Nur 0,2 Prozentpunkte besser schnitten die Server mit Suns kommerziellem Unix Solaris ab, die mit lediglich 2,5 Stunden nur etwa 0,3 Prozent der Testzeit nicht online waren.

Die neue Version 2.0 des Apache Web-Server bringt wesentliche Performance-Verbesserungen (Unterstützung von Unix Threading), eine vereinfachte Installations- und Konfigurationsroutine, eine bessere Unterstützung von Nicht-Unixplattformen sowie ein deutlich verbessertes API. Die Version 2.0 ist noch im Alphastadium, die Verwendung in Produktionsumgebungen wird derzeit ausdrücklich nicht empfohlen (und wir empfehlen, sich an diese Empfehlung zu halten). Source- und Binärdistributionen des Apache Web-Servers (sowie inzwischen auch eine Dokumentation im PDF-Format) finden Sie auf der Homepage des Apacheprojekts (`http://www.apache.org`) und auf dessen deutschen Spiegelserver (`http://www.apache.de`).

PHP Installationsdetails

Eine ganze Reihe von Funktionen fügen Sie Ihrem PHP-Modul nur dann hinzu, wenn Sie diese tatsächlich benötigen. Wollen Sie beispielsweise mit der Datenbank MySQL arbeiten, benötigen Sie nicht die Funktionen für die 15 anderen Datenbanken, die PHP unterstützt. (Natürlich können Sie aber die Funktionen für mehrere Datenbanken installieren, wenn Sie das möchten.) Wir wollen im Folgenden eine *einfache* Installation durchführen und als Spezialitäten lediglich eine MySQL- und PostgreSQL-Unterstützung installieren. Sie werden schneller Erfolgserlebnisse haben, wenn Sie ebenso verfahren. In einer Produktionsumgebung wird man später weitere Funktionalitäten hinzu kompilieren wollen. Dies erfordert aber in der Regel zunächst die Installation weiterer Software und Bibliotheken: Häufig wird für den Web-Server beispielsweise SSL-Support benötigt, damit zwischen Clients und Web-Server eine abhörsichere Verbindung etabliert werden kann. (Wenn Sie in Kapitel 9 unseren Installationsvorschlägen zu Webmin gefolgt sind, haben Sie das OpenSSL-Library bereits installiert, mod_ssl finden Sie unter `http://www.modssl.org`.) Für die Manipulation von GIF-, JPG- oder PNG-Grafiken braucht es ebenso jeweils eigene Bibliotheken wie z. B. für die Installation der PDF-Funktionen (PDFlib von Thomas Merz; `http://www.pdflib.com`). Hinweise zur Installation finden sich in den PHP-Handbüchern (sowohl in der englischen als auch in der deutschen Übersetzung), aber auch auf der PHP Portal-Seite `http://www.php-center.de`. Ein empfehlenswertes Tutorial zur LAMP-Installation (Linux, Apache, MySQL und PHP einschließlich SSL und Grafikfunktionen) hat Jörg Baach zur Verfügung gestellt (`http://www.baach.de/lamp-tutorial.html`). Sie merken wahrscheinlich schon: Wollen Sie ein eierlegendes, wollegebendes und melkbares PHP-Modul mit allen Funktionen erstellen, müssen Sie sich etwas Zeit nehmen.

Wir wollen indes nun eine einfache Version erstellen. Dabei gehen wir davon aus, dass beispielsweise die Datenbanken MySQL und PostgreSQL bereits installiert wurden. Dies ist erforderlich! Ist auf Ihrem System nicht mindestens eine der beiden DBMS installiert, sollten Sie das jetzt nachholen. Wir benötigen die Source für den Web-Server Apache und natürlich für PHP:

```
> cd /usr/local/src
> wget http://www.apache.de/dist/apache_1.3.12.tar.gz
> tar zxvf apache_1.3.12.tar.gz
> wget http://www.php.net/version4/do_download.php?download_file=\
   php-4.0RC1.tar.gz
> tar zxvf php-4.0RC1.tar.gz
> ln -s apache_1.3.12 apache
> ln -s php-4.0RC1/ php
```

Die letzten beiden Befehle (Anlegen von symbolischen Links) sind nicht zwingend erforderlich, ersparen aber Schreibarbeit. Im Folgenden ist es wichtig, die Reihenfolge präzise einzuhalten (sonst werden benötigte Libraries und Dateien nicht gefunden bzw. sind noch nicht installiert, wenn sie benötigt werden). Wenn zwischendurch etwas schief läuft und Sie es nach Änderungen (z. B. Hinzufügen von Libraries, Setzen von Links) erneut versuchen wollen, denken Sie daran, in beiden Verzeichnissen (*apache, php*) zuvor aufzuräumen (*rm config.cache; make clean*).

```
> # Apache vorbereiten
> cd apache
> ./configure --prefix=/usr/local/apache
> cd ..
> # PHP konfigurieren
> cd php
> ./configure  --with-mysql=/usr/local/mysql \
>            --with-pgsql=/usr/local/pgsql \
>            --with-apache=../apache \
>            --with-config-file-path=/etc \
>            --enable-track-vars \
>            --enable-sysvsem \
>            --enable-sysvshm
> make
> make install
```

Die Optionen *--with-mysql* und *--with-pgsql* sorgen für die Einbindung der MySQL bzw. PostgreSQL-Unterstützung in PHP. Sollten Sie zusätzlich oder alternativ z. B. Oracle-Unterstützung benötigen, geben Sie entsprechend *--with-oracle* an (und lassen die Optionen für nicht benötigte DBMS weg). Die bei den Optionen angegebenen Pfade zeigen auf die Installationsverzeichnisse. Die Option *--with-apache* sorgt dafür, dass ein PHP-Modul für den Web-Server Apache erstellt wird und kein CGI-PHP. Der Pfad zeigt auf das Source-Verzeichnis, nicht auf das Installationsverzeichnis! Die Option *--with-config-file-path=/etc* ändert den Suchpfad für die PHP-Konfigurationsdatei *php.ini* von */usr/local/lib* nach */etc*. *--enable-track-vars* schließlich sorgt dafür, dass PHP verfolgen kann, von wo GET/POST/COOKIES-Variablen kommen. Die Optionen *--enable-sysvsem* und *--enable-sysvshm* aktivieren die Unterstützung von Sys V Semaphores bzw. von Sys V Shared Memory. Es folgt das übliche *make, make install*, bei dem sich normalerweise keine Probleme ergeben.

```
> cd ..
> cd apache
> # Konfiguration des Apache Web-Servers
> ./configure --prefix=/usr/local/apache \
            --enable-module=most \
            --enable-shared=max  \
            --activate-module=src/modules/php4/libphp4.a
> make
> # Vorsichtshalber sichern wir das alte Apache-Binary
> mv /usr/local/apache/bin/httpd \
    /usr/local/apache/bin/httpd-<Erstellungsdatum>
> # Installation des Apache
> make install
>
```

Die Konfiguration und Kompilierung des Apache bereitet ebenfalls keine Probleme. Mit *--prefix* wird das Installationsverzeichnis bestimmt. Die beiden Configure-Optionen *--enable-module=most* und *--enable-shared=max* sorgen in dieser Kombination einerseits dafür, dass die DSO-Funktionalität (Dynamic Shared Objects) des Apache aktiviert wird. Damit ist es möglich, Zusatzmodule durch einfache Konfigurationsänderung in der Datei */usr/local/apache/conf/httpd.conf* (und anschließendem */usr/local/apache/bin/apachectl graceful*) dem Apache hinzuzufügen bzw. wieder aus ihm zu entfernen. Dies ist gerade dann nützlich, wenn die abschließende Konfiguration noch nicht fest steht und mit verschiedenen Modulen experimentiert werden soll. Zweitens werden die »meisten« Module aktiviert (alle

mit Ausnahme einiger problematischer). Nicht benötigte Module können über das Auskommentieren der entsprechenden *LoadModule/AddModule*-Direktiven in der *httpd.conf* deaktiviert werden. Die abschließende Option *--activate-module=src/modules/php4/libphp4.a* bindet das PHP-Modul ein. Danach werden mit *make* der Apache und alle angeforderten Module kompiliert und mit *make install* alles unter */usr/local/apache* installiert.

Schließlich sollte eine PHP-Konfigurationsdatei erstellt werden. Es kann zunächst die der Distribution beiliegende Beispieldatei verwendet werden. Diese ist kommentiert, und es ist sinnvoll, sich gelegentlich mit den Möglichkeiten vertraut zu machen (eine umfassende Besprechung der Konfiguration der Datei *configuration.html* findet sich auch in der deutschen Übersetzung des PHP-Handbuchs).

```
> cp /usr/local/src/php/php.ini-dist /etc/php.ini
```

Falls Sie Ihren Web-Server nicht selbst administrieren und Ihr Web-Angebot bei einem Massen-ISP gehostet ist, können Sie einen Teil der Konfiguration auch lokal (d.h. gültig nur für Ihre Verzeichnisse) vornehmen, indem Sie in das Wurzelverzeichnis Ihrer Web-Präsenz eine Datei *.htaccess* (der führende Punkt ist Bestandteil des Dateinamens) anlegen, in die Sie neben Direktiven für den Apache *Web-Server* auch Anweisungen für das PHP-Modul ablegen können. (Genauere Informationen bezüglich der möglichen Direktiven für den Apache entnehmen Sie der Apache-Dokumentation bzw. der genannten PHP-Dokumentation.) Das funktioniert allerdings nur, wenn in der *httpd.conf* diese Möglichkeit nicht vom Provider deaktiviert worden ist und – soweit es die PHP-Direktiven angeht – wenn PHP 4 (und nicht PHP 3) installiert ist.

Als Letztes muss der Apache Web-Server so konfiguriert werden, dass er mit PHP-Dateien etwas anfangen kann. Dies erreicht man mit folgenden Eintragungen in der Datei */usr/local/apache/httpd.conf:*

```
AddType application/x-httpd-php .phtml .php3 .php4 .php
AddType application/x-httpd-php-source .phps
DirectoryIndex index.html index.php3 index.php index.php4
```

Zunächst sorgen wir dafür, dass der Apache Dateien mit bestimmten Extendern als PHP-Dateien erkennt und diese zur Vorbehandlung an das PHP-Modul übergibt. Hierfür sorgt die erste AddType-Anweisung. Im Prinzip können Sie hier jeden beliebigen Extender angeben. Die von uns genannten sind (bzw. waren oder werden

es zukünftig sein) gebräuchlich. Es ist aber auch möglich, alle HTML-Dateien vom PHP-Modul parsen zu lassen (dazu müssen lediglich die Extender *.htm* und *.html* ergänzt werden). Das ist aber nur sinnvoll, wenn Ihre Dateien ganz überwiegend von PHP Gebrauch machen (z. B. weil Standardköpfe und -füße für HTML-Dateien aus einer Datenbank via PHP generiert werden sollen oder weil die Verwendung von PHP z. B. gegenüber Suchmaschinen verschleiert werden soll, weil manche Suchmaschinen PHP-Dateien nicht indizieren). Die zweite *AddType*-Anweisung sorgt dafür, dass für Dateien mit dem Extender *.phps* der Quelltext angezeigt wird (und zwar mit Syntax-Highlighting und hübsch koloriert). Weiter ist die Ergänzung der *DirectoryIndex*-Anweisung in der angegebenen Weise sinnvoll: Sie sorgt dafür, dass beim Aufruf eines Verzeichnis-Url zunächst nach einer Datei mit den aufgeführten Namen gesucht wird (und da sollen zusätzlich Dateien mit den Namen *index.php, index.php3* und *index.php4* gesucht werden) und diese statt eines Verzeichnislistings angezeigt wird. Vergessen Sie nicht, mit

```
/usr/local/apache/bin/apachectl graceful
```

dafür zu sorgen, dass der Apache seine Konfigurationsdateien neu einliest (sonst nimmt der Apache alle Ihre gut gemeinten Änderungen einfach nicht zur Kenntnis).

Wenn Sie den Apache Web-Server zum ersten Mal verwenden, müssen gegebenenfalls noch weitere Einstellungen in der *httpd.conf* vorgenommen werden. Schauen Sie diese Datei in Ruhe durch, sie ist ausreichend kommentiert. Auf jeden Fall sollte die *DocumentRoot*-Anweisung sinnvoll eingestellt sein (standardmäßig ist dieser Pfad auf */usr/local/apache/htdocs* gesetzt). Alle Dateien innerhalb dieses Verzeichnisses und aller seiner Unterverzeichnisse sind – wenn der Apache gestartet ist und Sie eine Verbindung zum Internet hergestellt und nicht irgendwelche besonderen Maßnahmen ergriffen haben – weltweit lesbar. Die Standardkonfiguration des Apache führt eigentlich immer zu einem lauffähigen System. Wenn Sie die Möglichkeiten dieses Web-Server voll ausreizen wollen, werden Sie allerdings etwas Zeit investieren müssen. Wir möchten Ihnen hierzu sehr Lars Eilebrechts Werk »Apache Web-Server« ans Herz legen [eil00].

Damit ist die Installation und Konfiguration von PHP als Apache-Modul abgeschlossen, und es steht dem Einsatz von PHP nichts mehr im Wege. Einen ersten Test können Sie durchführen, wenn Sie im *DocumentRoot*-Verzeichnis des Apache Web-Server eine Datei *php_info.php* mit folgendem Inhalt einstellen (heben Sie sich

diese Datei auf, Sie werden sie immer wieder einmal benötigen, wenn Sie wissen möchten, wie Ihr Web-Server bzw. Ihre PHP-Installation konfiguriert ist):

```
<html>
<head><title>PHP-Info</title></head>
<body>
<?php echo phpinfo() ?>
</body>
</html>
```

Rufen Sie nun in Ihrem Web-Browser diese Datei auf, indem Sie als Url *http://localhost/php_info.php (bzw. http://www.ihre-domain.de/php_info.php*) angeben. Wenn alles geklappt hat (und wir sind sicher, dass die Installation mit Hilfe dieser Anleitung keine größeren Probleme bereitet), werden Ihnen neun Tabellen mit unzähligen Informationen zu PHP, dem Web-Server, dem Serverbetriebssystem und seiner Environment-Variablen und zu vielen anderen Dingen ausgegeben. Wenn Sie der Auffassung sind, dass sich hierunter auch Informationen befinden, die nicht jedermann etwas angehen, dann sollte das Verzeichnis, in dem künftig diese und andere sensible Dokumente abgespeichert werden (Vorschlag: *DocumentRoot/admin*), durch ein Kennwort geschützt werden. Wie das geht, ist in Kapitel 9.2.8 im Zusammenhang mit der Installation des Datenbank-Client phpMyAdmin unter dem Stichwort Standardauthentifikation beschrieben.

13.2.3 Datenbankfunktionen

Eine besondere Stärke von PHP ist die Unterstützung sehr vieler Datenbanken. Derzeit werden von PHP unterstützt: Adabas D, dBase, Empress, FilePro, Informix, InterBase, Microsoft SQL Server, mSQL, MySQL, Oracle, PostgreSQL, Solid, Sybase, Unix dbm und Velocis. Über die ODBC- Schnittstelle lassen sich eine ganze Reihe weiterer Datenbanken ansprechen, sogar Microsoft Access, für alle, die meinen, dies zu brauchen. Auf alle diese Datenbanken können Sie mit PHP zugreifen. Das bedeutet auch, dass Sie alle diese Datenbanken als Backend für eine Web-Site verwenden können. Das ändert natürlich nichts daran, dass einige Datenbanken sich für eine solche Anwendung gut und andere weniger gut eignen.

Der Zugriff auf eine Datenbank vollzieht sich regelmäßig in sechs Schritten:

1. Herstellen einer Verbindung zum DBMS
2. Auswählen der Datenbank, auf die zugegriffen werden soll
3. SQL-Befehle abschicken
4. Anzahl der zurückgegebenen Datensätze ermitteln

5. Das Ergebnis der Datenbankabfrage holen
6. Schließen der Verbindung zum DBMS
7. Diese Funktionen sollen im Folgenden für die beiden in diesem Buch besprochenen Open-Source-Datenbanken überblicksartig aufgelistet werden (PostgreSQL und MySQL). Bei der Zusammenstellung der Funktionen handelt es sich nur um eine Auswahl und keinesfalls um eine komplette Liste. Es sind aber alle Funktionen besprochen, die benötigt werden, um gezielt Daten aus einer Datenbank zu holen und auf einer HTML-Seite darzustellen, oder mit denen Eingaben aus einem HTML-Formular in eine Datenbank abgespeichert werden können. Im nächsten Unterkapitel wird dann eine konkrete Anwendung unter Nutzung der nativen Datenbankfunktionen für das DBMS MySQL erstellt, die Sie sich ggf. für die von Ihnen bevorzugte Datenbank anpassen können. Sie können aber auch gleich eine der Möglichkeiten zur Datenbankabstraktion nutzen, die im übernächsten Unterkapitel vorgestellt werden. Sie brauchen dann bei einem Wechsel der Datenbank keine Veränderungen in Ihren PHP-Programmen vorzunehmen.

Tabelle 13.2: Ausgewählte PHP-PostgreSQL-Funktionen

DBMS Öffnen	*int pg_connect(string Hostname, string Port, string Optionen, string Tty, string Datenbankname);* *int pg_pconnect(string Hostname, string Port, string Optionen, string Tty, string Datenbankname);* *pg_pconnect* stellt eine persistente Verbindung her. Beispiel: *$link = pg_pconnect(»«,"","","","db-name«);*
DB auswählen	entfällt; wird direkt beim Öffnen erledigt
SQL absetzen	*int pg_exec(int Verbindungskennung, string Anfrage);* Beispiel: *$result = pg_exec($link,"SELECT * FROM tab-name«);*
Ergebnisse holen	*int pg_numrows(int Ergebniskennung);* *array pg_fetch_row(int Ergebniskennung, int Zeilennr);* *array pg_fetch_array(int Ergebniskennung, int Zeilennr, int [Ergebnistyp]);* *pg_numrows* ermittelt die Anzahl zurückgegebener Datensätze; *pg_fetch_row bzw. pg_fetch_array* holt die Datensätze. Beispiel: *$anzahl = pg_numrows($result);* *for ($i=1; $i <= $anzahl; $i++){ pg_fetch_result($result,$i-1)}*
Fehlerinformation	*string pg_errormessage(int Verbindungskennung);* Beispiel: *if(!$result){* *echo »Folgender Fehler ist aufgetreten: » . $pg_errormessage($link);}*
DB schließen	*bool pg_close(int Verbindungskennung);* Beispiel: *pg_close($link);*

Tabelle 13.3: Ausgewählte PHP-MySQL-Funktionen

DBMS Öffnen	*int mysql_connect(string [hostname [:port] [:/Socketpfad]], string [Benutzername] , string [Kennwort]);* *int mysql_pconnect(string [hostname [:port] [:/Socketpfad]], string [Benutzername], string [Benutzerkennwort]);* *mysql_pconnect stellt eine persistente Verbindung her.* Beispiel: *$link = mysql_pconnect(»localhost«,»nobody«,»nothing«);*
DB auswählen	*int mysql_select_db(string Datenbankname, int [Verbindungskennung]);* *int mysql_db_query(string Datenbankname, string Anfrage, int [Verbindungskennung]);* mit *mysql_select_db* wählen Sie die Datenbank aus, auf die spätere Befehle angewendet werden sollen. *mysql_db_query* schickt gleichzeitig einen SQL-Befehl ab. Beispiel: *$result=mysql_db_query»db-name","SELECT * from tab-name");*
SQL absetzen	*int mysql_query(string Anfrage, int [Verbindungskennung]);* *Beispiel: $result=mysql_ query(»SELECT * from tab-name«,$link);*
Ergebnisse holen	*int mysql_num_rows(int Ergebniskennung);* *array mysql_fetch_row(int Ergebniskennung);* *array mysql_fetch_array(int Ergebniskennung, int [Ergebnistyp]);* *mysql_num_rows* ermittelt die Anzahl zurückgegebener Datensätze; *mysql_fetch_row bzw. mysql_fetch_array* holt die Datensätze. Beispiel: *$anzahl = mysql_num_rows($result);* *for ($i=1; $i <= $anzahl; $i++){ mysql_fetch_result($result,$i-1)}*
Fehlerinformation	*int mysql_errno(int [Verbindungskennung]);* *string mysql_error(int [Verbindungskennung]);* Beispiel: *if(!$result){* *echo »Folgender Fehler mit der Nummer« . mysql_errno($link) .* *»ist aufgetreten: » . $mysql_errormessage($link);}*
DB schließen	*int mysql_free_result(int Ergebniskennung);* *int mysql_close(int [Verbindungskennung]);* *Beispiel: mysql_free_result($result); mysql_close($link);* Diese beiden Befehle können bei MySQL in aller Regel ohne Nachteile weggelassen werden.
Spezialitäten	*int mysql_insert_id(int [Verbindungskennung]);* Mit dieser Funktion ermittelt man die ID, die zuvor in einer *auto_increment*-Spalte mit *INSERT* erzeugt wurde. Beispiel: *mysql_query(»INSERT INTO tab-name SET id='NULL'«);* *$id=mysql_insert_id($link);* *echo »Es wurde die ID« . $id . » erzeugt«;*

13.2.4 Web-Interface zur Beispieldatenbank »Software«

Im Folgenden soll anhand eines sehr einfachen Beispiels gezeigt werden, wie auf eine Datenbank über eine Web-Seite lesend, wie schreibend zugegriffen werden kann. Wir verwenden die schon in Kapitel 9 eingeführte Beispieldatenbank »Software« (bei der Diskussion des grafischen Clients KSql im Unterkapitel 9.2.9, einen SQL-Dump dieser Beispieldatenbank finden Sie auf der Web-Seite zum Buch). Im Beispiel benutzen wir das DBMS MySQL. Der PHP-Code ist aber mit geringem Aufwand auch auf andere DBMS übertragbar. Die erforderlichen Befehle für PostgreSQL sind in der Tabelle 13.2 wiedergegeben, für alle anderen DBMS verweisen wir auf die Original-Dokumentation (Download-Adressen finden Sie auf der Web-Seite zum Buch).

Wenn Sie das Beispiel nachvollziehen wollen, müssen Sie zunächst das DBMS MySQL installieren (vergleiche gegebenenfalls Kapitel 9.2.2 bzw. 9.2.3) und dann die Datenbank »Software« einrichten (sofern Sie das nicht bereits getan haben). Diese Datenbank könnte dazu dienen, Informationen über Softwareprodukte im Intranet oder Internet zur Verfügung zu stellen. (Sie enthält derzeit Hinweise auf einige nützliche frei verfügbare Programme für diejenigen, die auch mit Windows NT arbeiten.) Mit einem entsprechenden Web-Interface (das im Folgenden erstellt werden soll) können Informationen wie »Art der Software«, »Name«, »genauere Beschreibung«, »WWW-Heimatadresse« und »lokale Download-Adresse« abgerufen bzw. eingegeben werden.

Die Datenbank »Software« können Sie z. B. auf dem folgenden Weg erzeugen (genauso kann natürlich auch einer der grafischen Clients, z. B. phpMyAdmin, verwendet werden):

```
> cd /usr/local/mysql/bin
> # Erzeugen einer leeren Datenbank "software"
> ./mysqladmin -u root -p create software
  Enter password: top_secret
  Database "software" created.
> mount /cdrom
> # Aufspielen des SQL-Dump
> mysql -u root -p software \
  < /wo/auch/immer/MySQL-software.sql
  Enter password: top_secret
>
```

Die Pfade müssen natürlich gegebenenfalls angepasst werden. Damit ist die Datenbank eingerichtet und kann mit einem beliebigen Client bearbeitet werden. Wir verwenden im Folgenden immer den MySQL-Standard-Client *mysql*. Wie das nachfolgende Listing zeigt, enthält die Datenbank lediglich sechs Tabellen, von denen zwei administrative Funktionen haben und von KSql zum Zwecke der Formulargenerierung eingerichtet worden sind (die *__Kmysql*-Tabellen und ihre Funktion wird in Kapitel 9.2.9 diskutiert):

```
shell> /usr/local/mysql/bin/mysql -u root -p
  Password: top_secret
mysql> show tables;
  +--------------------+
  | Tables_in_software2 |
  +--------------------+
  | __KmysqlFields      |
  | __KmysqlForms       |
  | rubrik              |
  | soft                |
  | stichw              |
  | stichwoerter        |
  +--------------------+
  6 rows in set (0.00 sec)
mysql> exit
```

Ein *mysqldump*-Aufruf verschafft uns einen Überblick über die relevanten vier Tabellen:

```
> cd /usr/local/mysql/bin
> ./mysqldump -u root -p --no-data \
  software soft rubrik stichwoerter stichw
  Enter Password: top_secret

# MySQL dump 8.2
#
# Host: localhost    Database: software2
#--------------------------------------------------------
# Server version       3.23.14-alpha-log
#

# Table structure for table 'soft'
#
```

```
CREATE TABLE soft (
  id int(10) NOT NULL auto_increment,
  name varchar(120) DEFAULT '' NOT NULL,
  beschreibung text DEFAULT '' NOT NULL,
  url varchar(120) DEFAULT '' NOT NULL,
  homepage varchar(120),
  rubrik int(10) DEFAULT '' NOT NULL,
  zeit varchar(14),
  PRIMARY KEY (id)
);
```

Die Tabelle *soft* hält die zentralen Informationen der Datenbank. Die Spalte id ist ein Primärschlüssel mit dem Attribut *auto_increment*: Immer, wenn in diese Spalte der Wert *NULL* eingefügt wird, bekommt diese den Wert x+1 zugewiesen, wobei x der größte derzeit in der Spalte enthaltene Wert ist. *name* enthält den Namen und *beschreibung* eine nähere Beschreibung. Die Spalte ist vom Typ *text*, dieser Typ entspricht *blob*, mit dem Unterschied, dass Suchoperationen standardmäßig Groß- und Kleinschreibungen nicht unterscheiden. *url* enthält die lokale Download-Adresse, *rubrik* ist ein Fremdschlüssel auf den Primärschlüssel *schluessel* in der Tabelle *rubrik*. *zeit* soll den Zeitpunkt erhalten, zu dem ein Datensatz zuletzt geändert wurde.

```
#
# Table structure for table 'rubrik'
#
CREATE TABLE rubrik (
  schluessel int(10) NOT NULL auto_increment,'
  name varchar(120) DEFAULT '' NOT NULL,
  PRIMARY KEY (schluessel)
);
```

Die Tabelle rubrik enthält Rubriken wie »Archivierung«, »Datenbank«, »Editoren« (zur Beschreibung der Art der erfassten Software).

```
#
# Table structure for table 'stichwoerter'
#
CREATE TABLE stichwoerter (
  id int(10) NOT NULL auto_increment,
  soft_id int(10) DEFAULT '0' NOT NULL,
```

```
  stichwort int(10) DEFAULT '' NOT NULL,
  PRIMARY KEY (id)
);
```

In der Tabelle *stichwoerter* sollen den besprochenen Softwareprodukten Stichwörter zugeordnet werden (*soft_id* ist dabei ein Fremdschlüssel auf die Spalte *id* der Tabelle *soft, stichwort* ein Fremdschlüssel auf die Spalte *schlüssel* der Tabelle *stichw*.

```
#
# Table structure for table 'stichw'
#
CREATE TABLE stichw (
  schluessel int(10) NOT NULL auto_increment,
  name varchar(120) DEFAULT '' NOT NULL,
  PRIMARY KEY (schluessel)
);
```

Die Schlüsseltabelle *stichw* enthält die Klartexte der Stichwörter. Auf die so definierte Datenbank soll nun mit PHP zugegriffen werden. Wir beginnen mit einem einfachen Auslesen der Datenbankinhalte und deren Ausgabe auf einer Web-Seite.

Wir gehen im Folgenden davon aus, dass Sie über grundlegende HTML-Kenntnisse verfügen. Diese sollen hier nicht vermittelt werden. Im Zweifelsfall hilft die wirklich umfassende und hervorragend strukturierte Dokumentation SelfHTML weiter (`http://www.teamone.de/selfaktuell/extras/download.htm`).

Die prinzipielle Arbeitsweise ist folgende: Sie erstellen wie gewohnt ein HTML-Dokument (in einem Editor, der Ihnen den Zugriff auf den Quelltext erlaubt, empfehlenswert ist z. B. der Editor »Emacs« mit dem PHP-Modus *php-mode.el*, `http://guru.nu/turadg/software/php-mode.el`). An beliebigen Stellen in diesem HTML-Dokument werden *<?php php_befehle; ?>*- bzw. kürzer *<? php_befehle; ?>*-Sequenzen eingefügt. Wenn Sie das HTML-Dokument mit einem Extender abspeichern, der dieses Dokument gegenüber dem Web-Server als PHP-Dokument ausweist (üblich sind .php bzw. .php3), dann arbeitet das PHP-Modul zunächst die PHP-Befehle ab und entfernt die sie umgebenden, kennzeichnenden Tags. Das klassische Hallo-Welt-Programm sähe so aus:

```
<html>
<head>
  <title>Hallo Welt</title>
```

```
</head>
<body>
  <?php echo "Hallo Welt"; ?>
</body>
</html>
```

Wenn Sie diese Datei im Browser aufrufen, z. B. durch *http://localhost/hallo-welt.php*, dann wird in Ihrem Browser »Hallo Welt« ausgegeben. (Das funktioniert *nicht* mit der Preview-Funktion Ihres Editors, weil ja die PHP-Befehle zunächst vom PHP-Preprozessor bearbeitet werden müssen; so etwas wie *file:/home/ihrname/hallo-welt.php* kann prinzipiell niemals funktionieren.) Wenn Sie die Funktion Quelltext anzeigen Ihres Web-Browsers nutzen, werden Sie feststellen, dass die öffnenden und schließenden PHP-Tags verschwunden sind. Solange das PHP-Modul einwandfrei funktioniert, bekommt jemand, der über das HTTP-Protokoll auf die PHP-Dateien zugreift, niemals PHP-Quelltext zu Gesicht, sondern immer nur das Ergebnis der Bearbeitung durch den PHP-Preprozessor. Dies kann auch eine Fehlermeldung sein: Wenn Sie die Anweisung *echo* z. B. in *echot* ändern, erhalten Sie die folgende Ausgabe, mit der PHP deutlich macht, dass es mit der Anweisung *echot* nichts anfangen kann:

```
Parse error: parse error in
  /usr/local/www/htdocs/test/hallo-welt.php on line 6
```

Als Nächstes soll ein Datenbankzugriff durchgeführt werden. Um das Beispiel einfach zu halten, werden die Authentifikationsinformationen für den Zugriff auf die Datenbank in der Beispieldatei vorgehalten. Dies ist jedoch ein Sicherheitsrisiko (weil dann ein Datenbankkennwort weltweit lesbar wird, falls einmal der PHP-Interpreter eine Fehlfunktion hat). Besser ist es, solche Informationen in eine Datei auszulagern (z. B. *db_setup.inc*), die außerhalb des Web-Server-DocumentRoot abgespeichert wird. Anschließend kann sie dann mit *<?php require '/pfad/zur/datei/db_setup.inc'* ?> eingelesen werden. Zunächst das Listing im Zusammenhang (die Listings können Sic sich auch von der Web-Seite zum Buch herunterladen):

```
1 <?php
2 # Datei: MySQL-Software01.php
3 # Konfigurationsvariablen fuer den Zugang zur Datenbank
4 $db_benutzer = 'nobody';
5 $db_passwort = 'niemand';
6 $db_name = 'software';
```

```
7  $db_server_host = 'localhost';
8  $admin_mail= "ihr-name@ihr-domaine.de";
9  ?>
10 <html>
11 <head>
12 <title>N&uuml;tzliche Software</title>
13 </head>
14 <body>
15 <h1>N&uuml;tzliche Software</h1>
16
17 <?php
18 mysql_connect("$db_server_host",
19             "$db_benutzer",
20             "$db_passwort")
21             or die("Kann keine Verbindung
22             zur Datenbank herstellen");
23
24 $query = "SELECT name, homepage FROM
25          soft ORDER BY name";
26 $result = mysql_db_query("$db_name", $query);
27
28 if(!$result){
29   $meldung="Es gibt Probleme!
30         \nFehlerhaftes SQL:\n\n$query
31         \nFehler-Nr:    " . mysql_errno() .
32        "\nFehlermeldung: " . mysql_error() .
33        "\n\nHurtig beheben!";
34   mail($admin_mail,"CATASTOPHIC ERROR: DB Software",
35       "$meldung");
36 die("Ein schwerwiegender Fehler ist aufgetreten;\n
37    der Systemadministrator wurde informiert");
38 }
39
40 $anzahl_rows = mysql_num_rows( $result );
41 ?>
42
43 <p>&Uuml;ber diese <? echo $anzahl_rows; ?> Programme
44   gibt es Informationen:</p>
45 <table bgcolor="#ccffcc" border="1">
46   <tr>
47     <th>Programmname</th>
48     <th>Homepage</th>
```

```
49  </tr>
50
51 <?php
52 while( $row = mysql_fetch_row( $result ) ) {
53  list( $name, $homepage) = $row;
54  echo "<tr><td>" . $name .
55       "</td><td><a href=\"" . $homepage . "\">" .
56       $homepage . "</a></td></tr>";
57 }
58 ?>
59 </table>
60 </body>
61 </html>
```

In diesem Beispiel sind bereits alle wichtigen PHP-Funktionen zur Datenbankabfrage enthalten. In den Zeilen 4-8 werden zunächst die Datenbank-Zugriffsinformationen in Variablen abgespeichert und die E-Mail-Adresse des Administrators angegeben. In Zeile 9 wird dieser erste PHP-Block geschlossen, Ausgaben werden durch diese PHP-Anweisungen nicht produziert. Wie erkennbar ist, können PHP-Statements und HTML-Tags bunt gemischt werden (die Zeilen 10 bis 15 enthalten lediglich triviale HTML-Tags). In Zeile 17 wird wieder ein PHP-Block geöffnet; in Zeile 18 mit *mysql_connect()*; eine Verbindung zur Datenbank hergestellt. Scheitert die Verbindung aus irgendeinem Grund (z. B. wegen eines falschen Kennworts), »stirbt« die Verbindung (*or die()*) mit der Meldung »*Kann keine ... herstellen*«. Im Fehlerfall werden von PHP außerdem zusätzliche Informationen beigesteuert:

```
Warning: MySQL Connection Failed:
Keine Zugriffsberechtigung für Benutzer:
'nobody@localhost'. (Verwendetes Passwort: Ja)
in /usr/local/www/htdocs/test/MySQL-Software01.php
on line 28
Kann keine Verbindung zur Datenbank herstellen
```

Wenn dem späteren Endbenutzer solche Informationen vorenthalten werden und dieser nur die selbst definierten Fehlermeldungen zu Gesicht bekommen soll, schreiben Sie einfach ein @ vor die entsprechende Funktion (*@mysql_connect()*).

In der Variablen *$query* (Zeilen 24-25) wird zunächst das SQL-Statement abgelegt. (Strings können problemlos in der nächsten Zeile fortgeschrieben werden, wenn der auch abgespeicherte Zeilenumbruch – wie hier – keine Probleme bereitet.) In

Zeile 26 wird dann das SQL-Statement an MySQL übergeben. Wir verwenden die Funktion *mysql_db_query()*, mit der wir gleichzeitig auch die zu verwendende Datenbank auswählen können.

Sollte bei der Übermittlung der Query ein Fehler auftreten (die Funktion *mysql_db_query()* hat *FALSE* zurückgegeben), so fangen wir das ab (Zeile 28: *if(!$result){}*), erstellen eine Fehlermeldung (Zeilen 29-33) für den Administrator, welche die MySQL-Fehlernummer und den MySQL-Fehlertext enthält, und stellen ihm das via E-Mail zu (Zeilen 34-35). Im Fehlerfall erhält der Administrator folgende E-Mail:

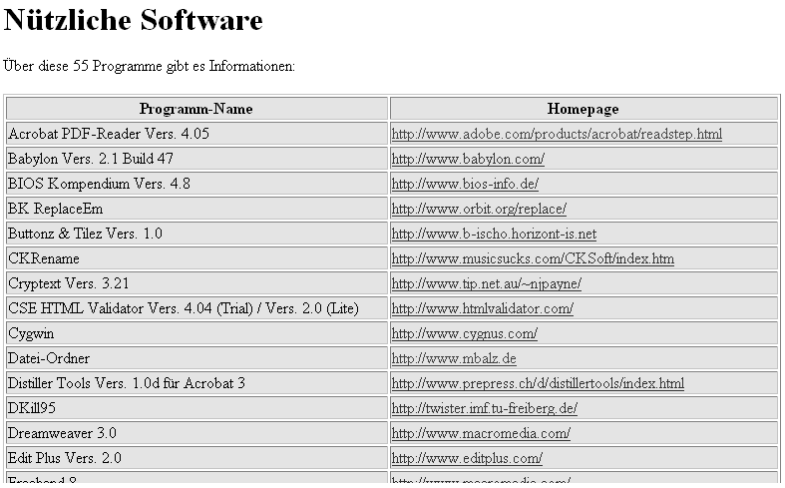
```
Subject: CATASTROPHIC ERROR: DB Software
Es gibt Probleme!
Fehlerhaftes SQL:

SELECT name, homepage FROMMMM soft ORDER BY name

Fehler-Nr:    1064
Fehlermeldung: Fehler in der Syntax bei 'soft ORDER
  BY name' in Zeile 1.

Hurtig beheben!
```

In den Zeilen 36-37 wird eine Information für den Endanwender in die HTML-Datei ausgegeben und dann die weitere Abarbeitung des Skripts abgebrochen. (Das bedeutet auch, dass die HTML-Datei nicht ordnungsgemäß geschlossen wird. Wenn Sie sich gerade in einer HTML-Tabellenbearbeitung befinden, kann das Probleme machen: Der Browser stellt die Fehlermeldung auf dem Bildschirm nicht dar. Deswegen ist es besser, sich eine eigene Fehlerfunktion zu bauen, die vorsichtshalber zunächst schließende Tabellen-Tags sowie zum Ende einen schließenden Body- und HTML-Tag ausgibt. Das schadet nicht und hilft häufig).

In Zeile 40 wird ermittelt, wie viele Datensätze die Anfrage ergeben hat. Dieser Wert wird in Zeile 43 dazu verwendet, den Anwender über die Größe der Datenbank zu unterrichten. Wie Sie sehen, können Sie PHP-Statements an beliebigen Stellen im HTML-Code unterbringen (auch, was häufig nützlich ist, in Attributen von Tags).

Anschließend wird mit HTML ein Tabellenkopf produziert (Zeilen 45-49). Die *while*-Schleife in Zeile 52 holt nun zeilenweise alle Datensätze aus der Datenbank und schreibt sie in das Array *$row*. Mit *list()* werden die Datensätze in ihre Bestand-

teile zerlegt und diese den Variablen *$name* und *$homepage* zugewiesen. Der abschließende *echo*-Befehl gibt jeweils eine Tabellenzeile aus, wobei der Url in der Variablen *$homepage* auch gleich dem *href*-Attribut des <*a*>-Tags zugewiesen wird (damit ein Klick des Anwenders auf den so erzeugten Link ausreicht, um ihn zur Homepage der ausgewiesenen Software zu befördern.). Die Anführungszeichen der *href*-Option des <*a*>-Tags müssen mit Backslashs gequotet werden. Der Aufruf der Datei MySQL-*Software01.php* (die Sie auch von der Web-Site zum Buch herunterladen können) erzeugt die in Abb. 13.2 dargestellte Ausgabe.



Nützliche Software

Über diese 55 Programme gibt es Informationen:

Programm-Name	Homepage
Acrobat PDF-Reader Vers. 4.05	http://www.adobe.com/products/acrobat/readstep.html
Babylon Vers. 2.1 Build 47	http://www.babylon.com/
BIOS Kompendium Vers. 4.8	http://www.bios-info.de/
BK ReplaceEm	http://www.orbit.org/replace/
Buttonz & Tilez Vers. 1.0	http://www.b-ischo.horizont-is.net
CKRename	http://www.musicsucks.com/CKSoft/index.htm
Cryptext Vers. 3.21	http://www.tip.net.au/~nipayne/
CSE HTML Validator Vers. 4.04 (Trial) / Vers. 2.0 (Lite)	http://www.htmlvalidator.com/
Cygwin	http://www.cygnus.com/
Datei-Ordner	http://www.mbalz.de
Distiller Tools Vers. 1.0d für Acrobat 3	http://www.prepress.ch/d/distillertools/index.html
DKill95	http://twister.imf.tu-freiberg.de/
Dreamweaver 3.0	http://www.macromedia.com/
Edit Plus Vers. 2.0	http://www.editplus.com/

Abb. 13.2: Datenbankabfrage mit Skript MySQL-Software01.php

Das Skript ist so noch nicht sonderlich nützlich. Eine Liste aller Einträge hätten wir einfacher auch ohne Datenbank erstellen können. Interessanter wird es, wenn die PHP-Seite mit Formularelementen angereichert wird, die es ermöglichen, Einfluss auf das zu nehmen, was ausgegeben werden soll. Wir benötigen also Formularelemente. Das folgende Beispiel soll lediglich das Prinzip deutlich machen. Selbstverständlich sollte es in einer realen Anwendung weiter verfeinert werden. Beim ersten Aufruf der Seite soll nun keine Liste mehr ausgegeben, statt dessen soll eine Auswahlbox angezeigt werden. Mit Hilfe dieser Auswahlbox können potenzielle Anwender eine Softwarerubrik auswählen, für die Informationen aus der Datenbank geholt werden sollen. Außerdem soll es möglich sein, nach frei wählbaren Begriffen zu suchen (»Gibt es Informationen zu 'emacs'«?).

Es wäre natürlich sehr unpraktisch, wenn jedes Mal, wenn eine Softwarerubrik geändert wird oder in die Datenbank zusätzliche Rubriken aufgenommen werden, auch die PHP-Seite geändert werden müsste. Daher wird der Inhalt der Auswahlbox nicht fest in der PHP-Seite verdrahtet, sondern seinerseits aus der Datenbank geholt. Wir realisieren dies mit einer kleinen Funktion, die nach dem Konfigurationsblock eingefügt wird:

```
1  # Funktion Aufzaehlung:
2  # Liest Werte fuer Selectbox aus DB aus
3  function Aufzaehlung($tabelle, $var) {
4    global $db_benutzer, $db_passwort, $db_name,
5          $db_server_host;
6    mysql_connect("$db_server_host", "$db_benutzer",
7                "$db_passwort")
8      or die("Kann keine Verbindung zur
9             Datenbank herstellen");
10   $query = "select schluessel, name from $tabelle
11            order by name";
12   $result = mysql_db_query("$db_name", $query);
13   if ($var == "") {
14     echo '<OPTION VALUE="0" SELECTED>Bitte
15           ausw&auml;hlen!</OPTION>';
16   }
17   while ( $cols =  mysql_fetch_row($result)) {
18     if ($var == $cols[0]) { $selected = " SELECTED"; }
19     else { $selected = ""; }
20     echo '<OPTION VALUE="' . $cols[0] . '"' .
21         $selected . '>' . $cols[1] . '</OPTION>';
22   }
23 }
```

In einer PHP-Funktion sind zunächst alle verwendeten Variablen lokal. Da auf außerhalb der Funktion gebildete Variablen zugegriffen werden soll (Authentifikationsdaten), sind diese zunächst mit dem Statement *global* innerhalb der Funktion sichtbar zu machen (Zeilen 4-5). In Zeile 6 wird eine Verbindung zur Datenbank etabliert, und über diese werden aus der Schlüsseltabelle *rubrik* die Rubrikwerte und Rubrikbezeichnungen geholt (Zeile 12). Sofern noch keine Rubrik ausgewählt worden war (beim ersten Aufruf) wird »händisch« ein Wert »*Bitte auswählen*« in die Liste eingefügt (damit der Benutzer weiß, was zu tun ist, Zeilen 13-16). Die *while*-Schleife (Zeilen 17-22) sorgt schließlich für den Aufbau der Auswahlbox.

Unterhalb der Überschrift (*<h1>...</h1>*) wird Code zur Erstellung der Auswahlelemente eingefügt:

```
1  <h1>N&uuml;tzliche Software</h1>
2  <form action="<? echo $PHP_SELF ?>"
3    method="post" target="">
4  <p>
5  <select name="rubrik">
6  <% aufzaehlung(rubrik, $rubrik) %></select>
7  oder: suche nach ...
8  <input type="text" name="such_string"
9    value="<? echo $such_string ?>"
10   size="30" maxlength="120">
11 </p>
12 <input type="Submit" name="action" value="Suchen">
13 </form>
```

In Zeile 2 wird zunächst ein HTML-Formular angelegt. Als auszuführendes Skript wird *$PHP_SELF* angegeben. Diese Variable enthält den Namen des aufgerufenen PHP-Skripts, das Formular ruft sich also selbst auf. Zeile 5 und 6 bauen die Auswahlbox auf, Zeile 8-10 ein Eingabefeld für einen Suchstring. In Zeile 12 wird schließlich der Submit-Button ausgegeben. Wird dieser vom Anwender angeklickt, erhält die Variable *$rubrik* den Wert des ausgewählten Eintrags der Auswahlbox und die Variable *$such_string* den String, der in das Editfeld eingetragen wurde. Diese Werte werden dem aufgerufenen Skript übergeben (die PHP-Seite ruft sich – mit veränderten Werten – selbst auf). Die Ausgabe einer Liste machen wir nun davon abhängig, ob eine der genannten Variablen gesetzt ist. Ist dies nicht der Fall (z. B. beim ersten Aufruf der Seite), wird keine Liste ausgegeben, sondern die Aufforderung, eine Wahl zu treffen:

```
1  if(!$rubrik and !$such_string){
2    die ("Bitte treffen Sie Ihre Wahl!\n</body></html>");
```

Abschließend bedarf es nur noch einer Änderung der SQL-Query, und unser Formular funktioniert wie gewünscht:

```
1 $query = "SELECT name, homepage FROM soft
2   WHERE rubrik = '$rubrik' OR
3     (name like '%$such_string%' and '$such_string' != '')
4   ORDER BY name";
```

Unsere Änderungen führen zu der in Abb. 13.3 dargestellten Ausgabe. Wenn Sie das Formular testen wollen, finden Sie es unter dem Namen MySQL-*Software02.php* auf der Web-Site zum Buch.

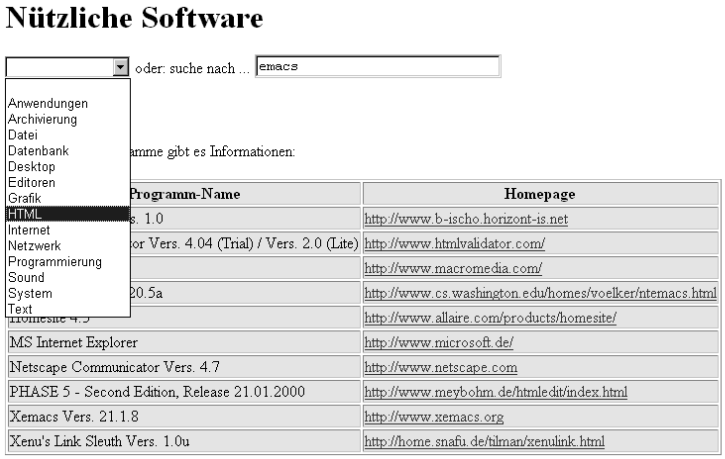


Abb. 13.3: Datenbankabfrage mit Skript MySQL-Software02.php

Die Erweiterung des Formulars, so, dass es sich auch zur Eingabe eignet, bringt nun nichts grundsätzlich Neues mehr und soll daher hier nicht dargestellt werden. Die Datei *MySQL-Software03.php* (sie wissen schon: auf der Web-Site zum Buch finden Sie auch diese Datei) enthält eine ausgefeiltere Eingabemaske, die sich sowohl zur Neudefinition von Rubriken und Stichwörtern als auch zur Neuaufnahme von Softwaretiteln eignet. Verwenden Sie den Quelltext als Anregung für eigene Experimente. Mit dieser Einführung sollten Ihnen die ersten Schritte erleichtert werden. Eine intensivere Auseinandersetzung mit den Sprachkonstrukten von PHP und ein Studium des Handbuchs oder eines Buches zu PHP kann sie freilich nicht ersetzen.

13.2.5 Datenbankabstraktion

Abschließend möchten wir auf die Möglichkeit hinweisen, sich von vornherein nicht auf eine Datenbank festzulegen, in dem Bibliotheken oder Klassen verwendet werden, die von der konkreten verwendeten Datenbank abstrahieren und die datenbankspezifischen Befehle in einem Treiber kapseln. Eine Bibliothek, die dies ermöglicht, ist die PHPLib.

Hauptgeschäft der PHPLib ist es, Sessions und Sessionvariablen zur Verfügung zu stellen, also Variablen, die ihren Wert am Ende einer Seite behalten und auf die nächste Seite mitgeschleppt werden. Darauf aufbauend offeriert die Bibliothek eine bessere und flexiblere Benutzerauthentifikation sowie ein flexibles System zur Kontrolle von Zugriffsrechten. (1. Ein Benutzer authentifiziert sich. 2. Jedem Benutzer können Rechte zugeordnet sein, die steuern, auf welche Dateien dieser Benutzer zugreifen kann bzw. welche Seiten er zu Gesicht bekommt etc.). Quasi als Abfallprodukt stellt die PHPLib eine auch unabhängig von der sonstigen Funktionalität benutzbare Klasse zur Datenbankabstraktion zur Verfügung. Wenn Sie Ihre Datenbankanbindung mit Hilfe dieser Klasse realisieren, können Sie problemlos die Datenbank wechseln. Unterstützt werden derzeit: mSQL, MySQL, ODBC, Oracle 7, Oracle OCI 8 Interface, PostgreSQL und Sybase). PHPLib ist freie Software (unter der GNU Library General Public Licence). Die PHPLib finden Sie auf `http://phplib.netuse.de`.

Eine weitere Möglichkeit zur Datenbankabstraktion bietet die class.DBI von Bill Adams. class.DBI ist eine PHP-Klasse zur Datenbankabstraktion. Syntaktisch lehnt sie sich an die PerlDBI an. Wer schon mit dieser gearbeitet hat, wird rasch auch mit PHP class.DBI zurechtkommen. Die Klasse ist nur rudimentär dokumentiert, ersatzweise kann die Perl-Dokumentation – *man dbi* – herangezogen werden. Es ist nicht die gesamte Funktionalität von Perl DBI/DBH implementiert. Zurzeit werden die folgenden Datenbanken unterstützt: Informix, MySQL, PostgreSQL, Sybase. An einem ODBC-Treiber wird gearbeitet. Auch class.DBI ist Open-Source-Software. Downloadhinweise finden Sie auf den Web-Seiten zum Buch.

Last, but not least soll auf dbwrap von Ralf Geschke hingewiesen werden. Auch dbwrap, die PHP-Datenbank-Wrapperklasse, bietet in der jetzigen Ausführung nur eine Schnittstelle zu MySQL, eine Anpassung an andere Datenbanken ist allein durch den Austausch oder die Erweiterung der Klasse dbwrap möglich (das müsste dann allerdings bei Bedarf noch geleistet werden), eine Änderung des Applikations-Codes wird somit vermieden. Auch dbwrap ist Open-Source-Software. Und auch für diese Bibliothek finden Sie den Download-Link auf der Web-Seite zum Buch.

13.3 PerlDBI und Embperl

Den Zugriff auf eine Datenbank kann nicht nur mit PHP, sondern – natürlich – auch mit Perl realisiert werden. Wenn Sie sich gut mit Perl auskennen, ist dies eventuell eine Alternative zu PHP. Sinnvollerweise verwendet man das Perl Data-

base Interface PerlDBI. Hierbei handelt es sich um ein Datenbank-Interface, das eine einheitliche Schnittstelle (Funktionen, Variablen) für eine Vielzahl von Datenbanken zur Verfügung stellt. Wenn sich im Verlaufe eines Projekts die Notwendigkeit ergibt, die Datenbank zu wechseln, ist dies problemlos und ohne Änderungen der Web-Seiten möglich.

Eine schöne Zusammenstellung zum Perl-DBI (Treibern (DBD), Dokumentation, Mailinglisten, sonstige Ressourcen) findet sich auf `http://www.symbolstone.org/technology/perl/DBI/`. Für das PerlDBI existieren derzeit Treiber für Adabas, CSV, DB2, Empress, Illustra, Informix, Ingres, InterBase, MiniSQL, MySQL, ODBC, Oracle, PostgreSQL, SearchServer, Solid, Sybase, Unify und XBase.

Mit PerlDBI allein können zwar Datenbankzugriffe realisiert werden, es muss aber der gesamte HTML-Code durch Perl ausgegeben werden. Er kann nicht, wie im PHP-Verfahren, in die HTML-Seite eingestreut werden. Änderungen werden dadurch ziemlich erschwert, und eine sachgerechte Arbeitsteilung zwischen Programmierern und Seitenlayoutern bei der Erstellung von Web-Seiten ist nur unter erheblichem Aufwand möglich.

Interessanter wird die Angelegenheit daher, wenn man sich dazu entschließt, das Perl-Modul Embperl zu verwenden. Dieses Modul macht es möglich, Perl-Befehle in ganz gewöhnlichen HTML-Seiten zu verwenden. Die Funktionsweise ist die gleiche wie bei PHP im vorherigen Kapitel: Der Perl-Code wird in die HTML-Seite eingebettet (»embedded«), Embperl interpretiert diesen Code, führt ihn aus und schreibt gegebenenfalls das Ergebnis der Ausführung an die entsprechende Stelle der HTML-Seite. Embperl kann in klassischer Weise über das CGI verwendet werden. Wesentlich performanter wird die Angelegenheit allerdings, wenn der Web-Server Apache mit mod_perl verwendet wird. Wie mod_php wird auch mod_perl in den Web-Server Apache integriert; ein Rückgriff auf das langsame CGI-Verfahren erübrigt sich. Für den Perl-Veteranen stellt die Kombination aus Apache, mod_perl, Embperl und PerlDBI eine mächtiges und hochperformantes Werkzeug für die Datenbankintegration in Web-Angebote dar. Hier ein Auszug aus der Feature-Liste von Embperl:

- Es stellt Meta Commands zur bedingten und iterativen Verarbeitung von HTML-Dokumenten zur Verfügung.
- Es erzeugt automatisch dynamische Tabellen/Listen aus Perl Arrays oder Funktionsaufrufen (z. B. DBI fetch).
- Formulardaten, die an ein Embperl Dokument gesandt werden, sind recht einfach über einen Hash zugreifbar.

- Embperl fügt automatisch Daten aus dem Formular-Hash in HTML-Input-, -Textarea- und -Select-Tags ein.
- Es versteht HTML- und URL-Kodierung/Dekodierung.
- Embperl stellt Pro-Benutzer und Pro-Seite persistente Sessiondaten zur Verfügung. Dafür ist es lediglich nötig die Daten in einem spezielle Hash abzulegen.
- Embperl kann vollständig in Apache und mod_perl integriert werden, um die beste Performance zu erreichen. Es kann auch als CGI Skript laufen, Offline ausgeführt werden oder von anderem Perlprogrammen/-modulen aufgerufen werden.
- Das Perlmodule DBIx::Recordset bietet einen hochwertigen, einfach zu handhabenden Datenbankzugriff für Embperl.
- Wenn Sicherheit ein Problem darstellt, ist es möglich Embperl so zu konfigurieren, dass es Safe.pm nutzt. Dadurch entstehen sichere Namensräume; zudem können einzelne Perl Opcode gesperrt werden.

Nähere Informationen zu Embperl und mod_perl finden sich unter anderen auf den folgenden Web-Seiten:

- `http://perl.apache.org/`

  Homepage des Apache/Perl Integration Projects. Hier finden sich Informationen, Dokumentationen und Download-Möglichkeiten zu mod_perl und Embperl.

- `http://www.heise.de/ix/artikel/1999/09/137/`

  ix-Artikel von Gerald Richter: Kompakte Datenbankabfragen mit Embperl

- `http://www.ecos.de/x/index.htm/ep/r_embperl.htm`

  Deutschsprachige Embperl-Portalseite der Firma Ecos (Hier findet sich u.a. eine Übersetzung der Originaldokumentation ins Deutsche)

13.4 Java Server Pages

Die Java Server Pages, kurz JSP genannt, sind der derzeit neueste Stern am Himmel der interaktiven Web-Technologien. Es ist noch nicht sicher, aber es gibt Hinweise dafür, dass diese Technologie sich gerade im Bereich professioneller und größerer Web-Projekte durchsetzen wird. Viele Web-Anwendungen werden bereits unter Verwendung der Programmiersprache Java erstellt; verwendet man nun das JSP-Verfahren, können die bereits entwickelten Module und Klassen und das erworbene Know-how weiter genutzt werden.

Wir wollen an dieser Stelle auf diese kommende Möglichkeit hinweisen und die grundlegende Funktionsweise der JSP skizzieren; eine ausführlichere Erörterung dieser Technologie kann im Rahmen dieses Buches nicht erfolgen, denn dazu müßte auch auf die Programmiersprache Java eingegangen werden. generell ist festzuhalten, dass die Verwendung von JSP und den derzeit verfügbaren JSP-Applikations- bzw. JSP-enabled-Web-Servern in einer Produktionsumgebung (derzeit) noch nicht vorbehaltlos empfohlen werden kann. Dies wird sich aber in absehbarer Zeit ändern, und daher empfehlen wir Ihnen dringend, ein wachsames Auge auf die JSP-Entwicklung zu halten.

JSP wurde als plattformunabhängige Alternative zu Microsofts ASP entwickelt. Gleichzeitig könnte es sich auch als ernsthafte Konkurrenz für das »bessere ASP«, nämlich für PHP, erweisen. ASP dürfte mit größerer Wahrscheinlichkeit zum Auslaufmodell werden. Ob dies auch für PHP gilt, bleibt abzuwarten. PHP hat mit seinem neuen Sprachkern »Zend« eine höchst performante Lösung anzubieten, ist mit Sicherheit einfacher in der Anwendung und last but not least schneller erlernbar.

Nachdem sich die Erwartungen in Java-Applets nur teilweise erfüllt hatten, begann 1997 der Siegeszug der Java-Servlets, die mittlerweile sehr populär sind. Servlets verdrängen immer mehr CGI-basierte Lösungen. Ein prinzipieller Nachteil der CGI-Lösungen wird aber auch bei Servlets nicht gelöst: Die HTML-Anweisungen zur Generierung der Web-Seiten stehen im Sourcecode der Programmiersprache. Dies bedeutet u. a., das jede kleine Änderung im Design und Inhalt der HTML-Seite nicht ohne Neuübersetzung des Programmcodes realisiert werden kann. Der entscheidende Nachteil in professionellen Umgebungen dürfte aber sein, dass bei Verwendung von Servlets die Änderung auch des Layouts und des Inhalts einer Web-Seite gute Kenntnisse der Programmiersprache Java erfordert und daher eine Arbeitsteilung zwischen inhaltlich Verantwortlichen, Designern und Programmierern sehr erschwert wird.

Genau hier setzt die JSP-Technologie an. Das »kopfstehende« CGI-Prinzip, nämlich die Einbettung von HTML-Direktiven in den Programmcode, wird bei JSP »auf seine Füße gestellt«: Programmcode wird in HTML-Dokumente eingebettet. Das ist zwar nicht so neu, wie Sun uns glauben machen möchte (schließlich findet sich dieses Prinzip schon lange u. a. auch bei PHP), aber trotzdem eine gute, weil effiziente Lösung. Eine wesentliche Voraussetzung für professionelles Arbeiten ist damit erfüllt, denn inhaltliche Verantwortung, Design und Programmlogik z. B. zur Abfrage von Datenbanken müssen nicht mehr in einer Hand liegen, wodurch eine arbeitsteilige Produktion ermöglicht wird.

Um mit Java Server Pages arbeiten zu können braucht man einen Server, der JSP-Seiten interpretieren kann. Seit dem Sun Ende 1999 den Code für die Servlet- und Java Server Pages-Technologie an die Apache-Gruppe lizenziert hat, ist diese die erste Anlaufstelle für JSP-Server (Projekt »Jakarta«). Unter dem Namen »Tomcat« stellte die Apache-Gruppe Anfang 2000 die Referenzimplementierung eines JSP-Servers der Öffentlichkeit vor. Der Tomcat-Server basiert auf der von Sun entwickelten JSWDK (JavaServer Web Development Kit)-Implementierung, die nicht mehr weiter entwickelt wird. Tomcat kann als eigenständiger Applikations-Server betrieben werden. Über das JServ-Modul (*mod_jserv.so*) kann der Tomcat-Server aber auch in einen Apache-Server integriert werden – natürlich das weit performantere Verfahren.

JSP-Seiten werden beim ersten Aufruf und nach jeder Änderung in ein ganz normales Java Servlet übersetzt und profitieren damit von den Laufzeiteigenschaften der Java Servlets. In Java Server Pages sind viele Funktionalitäten bereits integriert, so etwa das Erzeugen von Sessions, die automatisch erledigt werden. Das angesprochene Session-Feature basiert allerdings auf der Verwendung von Cookies, die momentan als sicherheitskritisch diskreditiert sind und daher von vielen Anwendern deaktiviert werden. In JSP-Seiten können Java-Beans verwendet werden. Bei diesen »Bohnen« handelt es sich um konfigurierbare Module, die komplexere Aufgaben verarbeiten können und so die Programmlogik in einem einfachen Aufruf verstecken. Prädestiniert für Java-Beans ist z. B. die Programmlogik für Datenbankzugriffe. Java Beans sind ein genereller Standard und werden nicht nur in Java Server Pages benutzt. Aus diesem Grund finden sich für viele Problemstellung bereits ausprogrammierte Lösungen – das viel zitierte Rad braucht nicht ständig neu erfunden werden.

Weitere Informationen zur Java Server Pages Technologie finden Sie u. a. auf den nachfolgend genannten Web-Seiten. Die Web-Recherche ergab keine einzige deutschsprachige Portalsite für JSP. Dies zeigt, dass diese Technologie noch am Anfang steht:

- `http://jakarta.apache.org/tomcat/jakarta-tomcat/src/doc/uguide/tomcat_ug.html`

  Anleitung zur Installation sowie erste Schritte bei der Anwendung des Tomcat-JSP-Servers

- `http://www.javasoft.com/products/jsp/syntax.pdf`

  JSP-Syntax-Karte

- `http://java.sun.com/products/jsp/`

  JSP-Seiten von Sun. Hier findet sich ein größerer Fundus von Dokumenten.

13.5 Zusammenfassung

In diesem Kapitel wurden Möglichkeiten der Integration von Datenbanken in Web-Angebote erörtert. Begonnen haben wir mit Java; hier sind wir insbesondere auf JDBC (*Java Database Connectivity*) eingegangen, eine standardisierte Java-Anwendungsschnittstelle zu relationalen Datenbanken. Breiteren Raum hat die Darstellung des PHP Hypertext Preprocessors eingenommen, weil es sich hierbei um eine weit verbreitete und einfach anzuwendende Technologie handelt. Ausführlich sind wir auf die Installation und Konfiguration von PHP als Apache-Modul eingegangen. Die grundlegenden Möglichkeiten zur Realisierung von Datenbankabfragen und -eingaben mittels PHP über eine Web-Formular wurde detailliert am Beispiel des RDBMS MySQL besprochen. Demgegenüber haben wir über die Möglichkeiten des Perl Database Interface (PerlDBI) und der Einbettung von Perl-Programmen in HTML-Seiten mit Embperl nur einen knappen Überblick gegeben. Auch die zukunftsträchtige Technologie der Java Server Pages ist lediglich kurz und überblicksartig besprochen worden; ihre Erörterung erfolgte zum Schluß, weil die Darstellungen im PHP-Kapitel das Verständnis von JSP erleichtern. Die von uns vorgenommene Gewichtung im Umfang entspricht der derzeitigen Verbreitung der angesprochenen Technologien zur Internetanbindung von Datenbanken. In ihrer Funktionsweise ähneln sich die Technologien, und wir meinen, dass die genauere Kenntnis einer dieser Techniken (PHP) das schnelle Einarbeiten in die anderen erlaubt, sofern die jeweiligen Basisprogrammiersprachen Perl und Java bekannt sind.