

# Dynamische Inhalte in Webseiten

Ein Leitfaden zum Einsatz von Technologien  
zur dynamischen Seitengenerierung

Seminarvortrag bei Prof. Dr. Häuslein  
FH Wedel, Sommersemester 2001

Autor / Copyright: Sebastian Eichner  
eMail: [sebastian\\_sp@net-eye.de](mailto:sebastian_sp@net-eye.de)

# Inhaltsverzeichnis

1	Einleitung.....	3
2	Überblick.....	4
2.1	Begriffserläuterung.....	4
2.2	Aufgabenstellung.....	4
2.3	Problematik der Aufgabe.....	5
2.3.1	Kategorisierung der technischen Ansätze.....	5
2.3.2	Typische Anwendungsbereiche.....	5
3	Technischer Hintergrund.....	7
3.1	Statische Webseiten.....	7
3.2	Dynamische Webseiten.....	7
3.2.1	Clientseitige Techniken.....	9
3.2.1.1	DHTML/JavaScript.....	9
3.2.1.2	Applets.....	9
3.2.1.3	Plugins.....	10
3.2.2	Serverseitige Techniken.....	10
3.2.2.1	CGI.....	10
3.2.2.2	FastCGI.....	11
3.2.2.3	Webserver Erweiterungen.....	12
3.2.2.4	Application Server.....	13
3.2.2.5	Weitere Techniken.....	13
4	Praktische Anwendungsfälle und Lösungen.....	14
4.1	Einfache dynamische Inhalte.....	15
4.2	Manuell pflegbare Inhaltsbereiche.....	15
4.3	Einbindung von externem Content.....	17
4.4	Anbindung an Datenbanken.....	18
4.5	Anbindung an vorhandene Applikationen.....	19
4.6	Neuentwicklung komplexer, modularer und internetbasierter Anwendungen.....	20
5	Quellenverzeichnis.....	23

## 1 Einleitung

Mit der zunehmenden Verbreitung des Internets hat die Erzeugung von dynamischen Inhalten immer größere Bedeutung erlangt. Internet-Sites mit aktuellen Nachrichten, Shop-Systeme mit Artikel-datenbanken, redaktionell betreute Webseiten, aber auch einfachste Inhalte wie ein Zugriffszähler oder eine Datumseinblendung basieren auf dem Einbau von veränderlichen Inhalten in Webseiten.

Die dabei verwendeten Technologien haben sich seit den Anfängen mit CGI-Skripten sehr schnell weiterentwickelt. Wie können dynamische Inhalte generiert werden? Welche Probleme können mit welchen Technologien gelöst werden? Diese Seminararbeit soll zum einen etwas Hintergrundwissen vermitteln und zum anderen einen konkreten Leitfaden zur Unterstützung bei der Suche nach Problemlösungen anbieten.

## 2 Überblick

### 2.1 Begriffserläuterung

Eine eindeutige Definition für den Begriff "Dynamische Inhalte im Internet" ist leider schwer zu finden, da er meistens ohne vorhergehende Festlegung seiner Bedeutung benutzt wird.

Abgrenzung durch Darstellungsaspekte

Am einfachsten und naheliegendsten ist eine Abgrenzung zu den statischen Inhalten, beziehungsweise zu statischen Internetseiten, wenn man die Darstellung im Browser betrachtet: Während statische Seiten immer identisch aussehen, können sich dynamische Seiten von Aufruf zu Aufruf unterscheiden, abhängig vom Eintreten bestimmter Bedingungen (z.B. bestimmte Browsersoftware, personalisierte Seiten) oder bestimmter Aktionen (z.B. Einpflegen einer neuen Nachricht durch einen Redakteur).

Abgrenzung durch technische Aspekte

Betrachtet man den Ablauf der Erzeugung dynamischer Inhalte/Webseiten, läßt sich eine Abgrenzung durch technische Aspekte vornehmen: Statische Seiten sind HTML<sup>1</sup>-Dateien, die der Server direkt ausliest und ohne Veränderungen an den Client (Browser) weiterreicht. Dynamische Seiten dagegen werden entweder in einem von der Server-Software abhängigen Dateiformat abgelegt und vor der Auslieferung durch eine Software mit dem dynamischen Inhalt gefüllt (z.B. Template-Ansätze und eingebettete Skriptsprachen wie PHP<sup>2</sup>), oder werden komplett durch eine Software erzeugt (z.B. bei CGI<sup>3</sup>).

Anmerkung: Der letzte Abschnitt gilt streng genommen nicht für clientseitiges JavaScript<sup>4</sup> und Applets<sup>5</sup>, mit denen sich in sehr engen Grenzen auch ein dynamischer Inhalt ohne spezielle Serversoftware realisieren läßt (z.B. Einblenden des aktuellen Datums). Diese Technik hat allerdings in der Praxis nur geringe Bedeutung.

### 2.2 Aufgabenstellung

Ziel dieser Seminararbeit ist es, einen Überblick über die verbreitetsten Techniken zur Erzeugung dynamischer Inhalte für Webseiten zu geben. Dabei soll eine Art Leitfaden entstehen, der Hinweise zur Auswahl einer Technik zur Lösung eines bestimmten Problems gibt.

---

1 Hypertext Markup Language: Eine einfache Auszeichnungssprache zur plattformunabhängigen Darstellung von Informationen

2 PHP Hypertext Preprocessor: In HTML eingebettete Skriptsprache

3 Common Gateway Interface: Eine API zur Kommunikation zwischen einem Server und externen Programmen, siehe Kapitel 3.2.2.1

4 In Webseiten eingebettete Skriptsprache, die durch den Browser ausgeführt werden soll, standardisiert als ECMAScript

5 In Webseiten eingebettetes, spezielles Java-Programm, das durch Java-kompatible Browser ausgeführt werden kann

## 2.3 Problematik der Aufgabe

Ein Leitfaden zu dynamischen Webseiten sollte vor allem zwei Dinge beinhalten:

Zwei Hauptbestandteile der Aufgabe

- eine Kategorisierung der verschiedenen Möglichkeiten, dynamische Inhalte zu generieren, um das technische Hintergrundwissen zu vermitteln
- eine Aufstellung der typischen Anwendungsbereiche für dynamische Inhalte, um die Beurteilung einer zu lösenden Aufgabe zu ermöglichen

Die jeweiligen Ziele und dabei entstehende Probleme werden in den folgenden Absätzen genauer erläutert.

### 2.3.1 Kategorisierung der technischen Ansätze

Da es inzwischen eine unüberschaubare Anzahl an Möglichkeiten zur dynamischen Seitenerzeugung gibt, können hier nur die wichtigsten und bekanntesten berücksichtigt werden. Problematisch dabei ist, dass diese Lösungsansätze zu unterschiedlich sind, um sie alle bestimmten Kategorien zuordnen zu können. In Kapitel 3 wird die "klassische" Einteilung aufgezeigt, in die sich die meisten Ansätze nach technischen Kriterien ihrer Implementation einordnen lassen. Dadurch soll dem Leser ein Überblick über die generelle Funktionsweise dieser Techniken gegeben werden.

Allerdings sagen diese Einteilungen wenig über die Leistungsfähigkeiten der einzelnen Lösungen aus. Mit jedem der technischen Ansätze lassen sich weite Anwendungsbereiche abdecken, abhängig davon, wie geschickt die Entwickler die spezifischen Eigenheiten des gewählten Ansatzes ausnutzen beziehungsweise verbessern.

Es läßt sich also keine 1:1-Zuordnung zwischen den Techniken und den Anwendungsfeldern herstellen.

### 2.3.2 Typische Anwendungsbereiche

Anstatt einzelne Typen von Websites, z.B. Community-Sites, mit deren Technik zu analysieren, ist es sinnvoller, allgemeine Anwendungsbereiche, sozusagen "Bausteine" dynamischer Websites, und die dazugehörigen Anforderungen herauszuarbeiten.

Die Anwendungsbereiche, zu denen in Kapitel 4 Lösungsvorschläge gemacht werden, sind aus einer Betrachtung der typischen Einsatzfelder für dynamische Webseiten hervorgegangen. Zu jedem Anwendungsbereich gibt es einige besonders relevante Kriterien, anhand derer die möglichen Lösungen beurteilt werden.

Analysiert man das Konzept einer Website, so kann man fast alle dynamischen Bereiche einem der dort genannten Anwendungsfelder zuordnen. Dadurch können Rückschlüsse auf die einsetzbare Software gezogen werden.

Dabei sollte aber berücksichtigt werden, dass es für die Auswahl einer Lösung noch zahlreiche andere relevante Aspekte gibt. So spielt das vorhandene Know-How eine entscheidende Rolle: Wer sich mit einem System auskennt, wird ungern auf ein anderes wechseln, auch wenn es nach objektiven Kriterien besser geeignet wäre. Weiterhin haben in der Praxis die persönlichen Erfahrungen und Vorlieben sowie aktuelle Trends ein nicht zu unterschätzendes Gewicht.

## 3 Technischer Hintergrund

### 3.1 Statische Webseiten

Noch sind vermutlich die meisten im Internet vorhandenen Webseiten statisch. Wird eine solche Seite von einem Browser, also einem Client, angefordert, schickt dieser eine entsprechende URL<sup>6</sup> an den Server. Der Server identifiziert anhand der URL die gesuchte Datei (HTML, Bilder, etc.), liest sie ein und schickt sie unverändert an den Browser zurück.

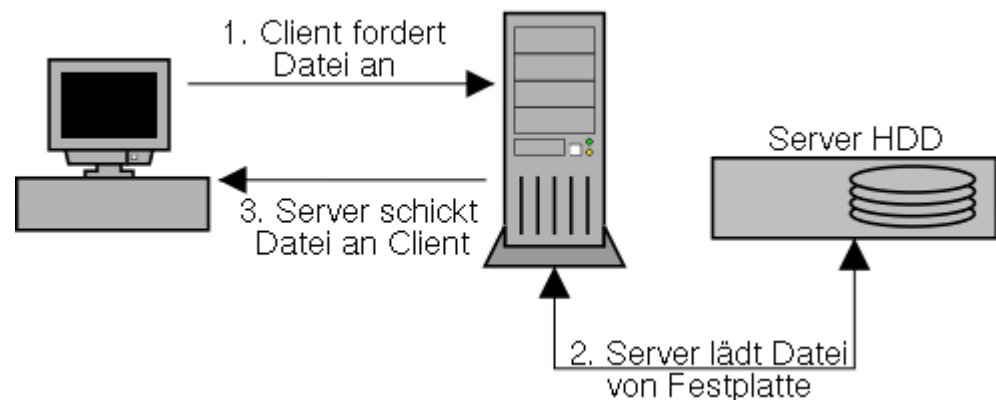


Abbildung 1: Auslieferung statischer Seiten

### 3.2 Dynamische Webseiten

Gestiegene Anforderungen führen zu dynamischen Webseiten

Mit der steigenden Verbreitung und damit größeren Bedeutung des Internets haben sich auch die Anforderungen verändert. In vielen Fällen reicht es nicht mehr, nur eine statische Datei auszuliefern. Unterschiedliche Browsersoftware, personalisierte Seiten, datenbankgestützte Shopsysteme und Internetseiten mit sich ständig ändernden, aktuellen Inhalten müssen technisch umgesetzt werden.

Dynamische Seiten können client- oder serverseitig realisiert werden

Um das zu realisieren gibt es prinzipiell zwei Ansätze. Zum einen kann immer noch eine sich nicht verändernde Datei vom Server ausgeliefert werden, deren Inhalt erst auf dem Client durch aktive Bestandteile wie zum Beispiel JavaScript nachträglich modifiziert werden (clientseitige Generierung).

Zum anderen muss die angeforderte Datei nicht sofort an den Client ausgeliefert werden, sondern kann erst durch ein Programm verändert oder sogar erst bei der Anforderung durch ein Programm erzeugt werden (serverseitige Generierung). Eine URL bezeichnet also nicht mehr unbedingt eine konkrete Datei, sondern vielmehr eine vom Server bereitzustellende Informationseinheit.

<sup>6</sup> Uniform Resource Locator: Bezeichnung einer Ressource, die über ein in der URL festgelegtes Protokoll (bei Webseiten HTTP) angefordert werden kann.

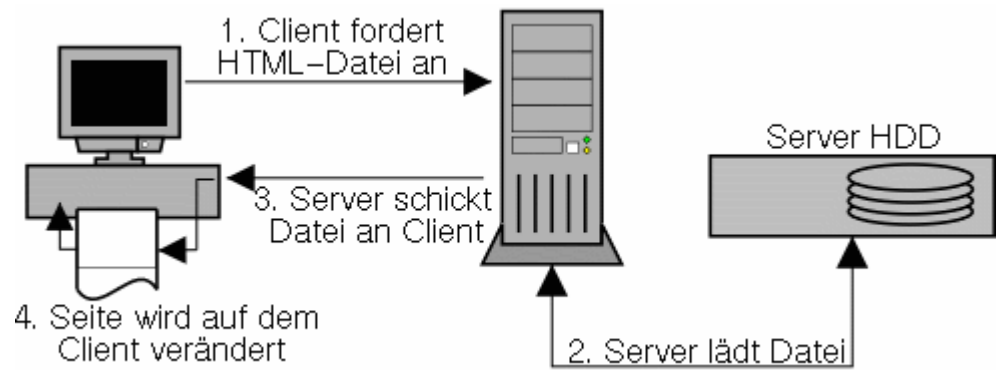


Abbildung 2: Clientseitige Generierung

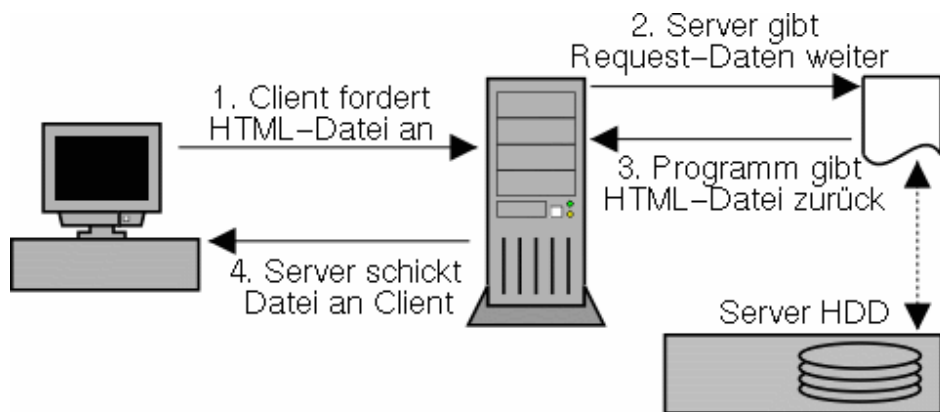


Abbildung 3: Serverseitige Generierung

Wie in der folgenden Grafik gezeigt, teilen sich client- und serverseitige Erzeugung von dynamischen Inhalten weiter in verschiedene Techniken auf.

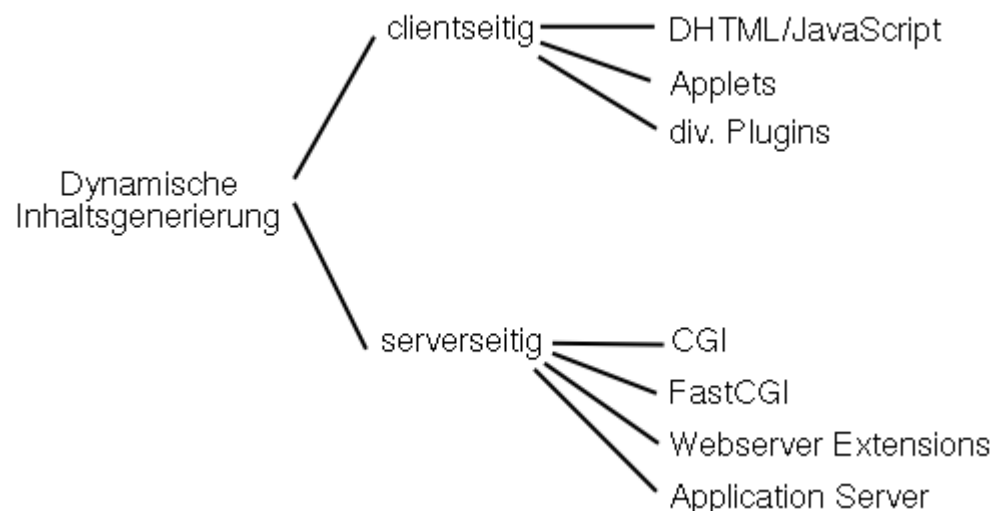


Abbildung 4: "Klassische" Techniken zur dynamischen Inhaltsgenerierung

## 3.2.1 Clientseitige Techniken

### 3.2.1.1 DHTML/JavaScript

Mit DHTML/JavaScript kann der HTML-Text nachträglich verändert werden

Mit Hilfe von JavaScript und allen unter dem Sammelbegriff DHTML<sup>7</sup> zusammengefassten Techniken kann der Inhalt von HTML-Dokumenten nachträglich verändert werden. Die Anweisungen hierzu sind in den HTML-Text eingebettet und werden durch den Browser, falls er über diese Fähigkeit verfügt, ausgeführt. Dadurch ist es zum Beispiel möglich, ein aktuelles Datum auszugeben, abhängig von der Browsersoftware bestimmte Aktionen auszuführen oder über Cookies<sup>8</sup> benutzerspezifische Informationen abzuspeichern und wieder auszulesen.

Große Probleme mit der Kompatibilität

Diese Technik ist allerdings in der Praxis problematisch, da JavaScript nicht immer vorhanden und aktiviert ist. Außerdem ist die Implementation der DHTML-Fähigkeiten von Browser zu Browser unterschiedlich und zum Teil fehlerhaft. Das Benutzen dieser Funktionen wird damit zum Glücksspiel und kann zu ungewollten Resultaten führen.

### 3.2.1.2 Applets

Java-Applets ermöglichen komplexe Programme

Applets sind spezielle Java<sup>9</sup>-Programme, die über eine im HTML-Text eingebettete Referenz von Java-fähigen<sup>10</sup> Browsern nachgeladen und ausgeführt werden. Diesen Programmen steht prinzipiell der gesamte Funktionsumfang von Java zur Verfügung. Damit verfügen Applets über sehr weitreichende Fähigkeiten wie Grafik (2D und 3D), Netzwerkintegration und Interaktion mit dem Benutzer. Über Applets lassen sich zum Beispiel leicht Ticker realisieren, die aktuelle Nachrichten von einem Server beziehen und anzeigen.

Einsatz problematisch

Nachteilig wirkt sich allerdings die Tatsache aus, dass Java nicht in allen Browsern integriert und aktiviert ist. Damit entsteht hier, wie bei JavaScript eine Unsicherheit, ob das Applet überhaupt ausgeführt werden kann. Außerdem gibt es bei einigen Browsern immer noch Stabilitäts- und Performanceprobleme mit Applets, was die Nutzbarkeit weiter einschränkt.

<sup>7</sup> Dynamic HTML: DHTML ist keine standardisierte HTML-Version, vielmehr verwendet man es als Sammelbegriff für unterschiedliche Techniken, die über statischen HTML-Inhalt hinausgehen. Dazu gehört vor allem die Manipulation des Inhaltes über eine spezielle API (das Document Object Model).

<sup>8</sup> Cookies sind kleine Dateien, die auf dem Client gespeichert werden und in denen über das HTTP-Protokoll oder JavaScript Informationen abgespeichert werden können.

<sup>9</sup> Java ist eine objektorientierte, plattformunabhängige Sprache, die zu sog. Bytecode kompiliert und auf einer Virtuellen Maschine (JVM) ausgeführt wird.

<sup>10</sup> Voraussetzung für einen Java-fähigen Browser ist eine in den Browser integrierte Laufzeitumgebung (Java Runtime Environment)

### 3.2.1.3 Plugins

Über Plugins, also Zusatzsoftware, die im Browser installiert werden kann, läßt sich nahezu jede Fähigkeit nachrüsten. Dazu muss der Benutzer allerdings meistens explizit die nötige Software installieren, die zudem spezifisch für jede Plattform und gegebenenfalls für einzelne Browser programmiert werden muss. Damit schränkt sich der Kreis möglicher Nutzer sehr stark ein, so dass Webseiten, die auf Plugins angewiesen sind, nicht besonders verbreitet sind. Diese Variante wird daher nur zur Vollständigkeit erwähnt, ohne tiefer darauf einzugehen.

## 3.2.2 Serverseitige Techniken

Seit den Anfängen der Dynamischen Seitengenerierung haben sich die dafür verwendeten Servertechniken ständig weiterentwickelt. Dabei sind sie immer leistungsfähiger und komplexer geworden. In diesem Abschnitt soll ein Überblick über die gängigen technischen Konzepte hinter der Serversoftware gegeben werden.

### 3.2.2.1 CGI

CGI als flexibler Standard mit hoher Verbreitung aber Performancenachteilen

CGI ist die Abkürzung für "Common Gateway Interface" und bezeichnet eine Standard-API<sup>11</sup> zur Kommunikation zwischen Webserver und externen Programmen. CGI-Skripte sind wohl der älteste Ansatz für dynamische Webseiten. Dabei startet der Server das über die URL angegebene CGI-Skript in einem neuen Prozess und übergibt diesem in Umgebungsvariablen und über die Standard-eingabe Daten aus dem HTTP-Request. Das CGI-Skript schreibt die von ihm erzeugte Datei auf die Standardausgabe, die der Server dann an den Client weiterleitet. CGI-Skripte können also nicht nur HTML-Seiten generieren, sondern sind prinzipiell unabhängig vom Ausgabemedium. So werden zum Beispiel häufig Bilder über CGI-Skripte dynamisch generiert.

Der große Vorteil von CGI-Skripten ist die Flexibilität. Sie können in jeder Sprache programmiert werden, die das Auslesen von Umgebungsvariablen sowie das Konzept der Standardeingabe und -ausgabe unterstützt. In der Praxis bedeutet das: Nahezu alle Sprachen können hierfür benutzt werden. Häufig anzutreffen sind Perl<sup>12</sup>-Dialekte, C-Programme, Shellskripte sowie zahlreiche andere Skriptsprachen.

<sup>11</sup> Application Programming Interface: eine definierte Schnittstelle zur Programmierung

<sup>12</sup> Practical Extraction and Report Language: Eine inzwischen sehr komplexe Sprache, die ursprünglich vor allem zur Verarbeitung von Texten gedacht war und dort immer noch hervorragende Möglichkeiten bietet.

Diese Flexibilität bringt allerdings auch Nachteile mit sich. Da jedes CGI-Skript in einem neuen, eigenen Prozess gestartet wird, geht jedes Mal Zeit für die Initialisierung verloren. Außerdem können Ressourcen wie Datenbankverbindungen nicht von mehreren Prozessen genutzt werden, sondern müssen jedes Mal neu erstellt und beendet werden<sup>13</sup>. CGI-Skripte können außerdem keine Informationen von einem Aufruf zum nächsten übernehmen. Sie werden daher als zustandslos bezeichnet. Anwendungen, die Informationen zwischenspeichern müssen, wie zum Beispiel ein Warenkorb, brauchen dazu externe Unterstützung durch Datenbanken oder das Speichern in Dateien.

### 3.2.2.2 FastCGI

Weiterentwicklung des CGI-Standards für bessere Performance

Um die oben genannten Nachteile von CGI-Skripten zu kompensieren, wurden verschiedene Verbesserungen an CGI vorgenommen. Eine der verbreitetsten Weiterentwicklungen ist FastCGI.

Dabei wird beim ersten Aufruf eines Skriptes der entsprechende Prozess gestartet, aber nach Abarbeitung dieses Aufrufes nicht sofort beendet. Stattdessen läuft das Skript weiter und wartet auf den nächsten Aufruf. Der Overhead für die Erzeugung des Prozesses bei jedem Aufruf entfällt also. FastCGI-Skripte sind "persistent", im Gegensatz zu normalen CGI-Skripten können sie Informationen von einem Aufruf zum nächsten zwischenspeichern.

Außerdem läuft die Kommunikation zwischen Webserver und Skript, zum Beispiel um Request-Daten zu übergeben, nicht über die Umgebungsvariablen, sondern über eine bidirektionale Verbindung (TCP<sup>14</sup> oder Pipes<sup>15</sup>). Dadurch wird eine verbesserte Performance erreicht und die Skripte können, bei zu hoher Last auf dem Webserver, auf einen zweiten Rechner ausgelagert werden.

Ein weiterer Geschwindigkeitsvorteil ergibt sich daraus, dass ein Skript Ressourcen wie eine Datenbankverbindung nur zu Beginn ein Mal öffnen muss und sie danach für alle folgenden Zugriffe weiterverwenden kann.

Einfache Portierung von CGI-Skripten

FastCGI-Skripte können wie CGI-Skripte in einer Vielzahl von Sprachen geschrieben werden. Normale CGI-Skripte können daher meist mit wenig Aufwand zu FastCGI-Skripten portiert werden.

<sup>13</sup> Datenbankverbindungen müssen zu Beginn aufgebaut und initialisiert werden. Der dabei anfallende Kommunikationsaufwand kann sich bei sehr vielen Verbindungen zu einer signifikanten Belastung sowohl des Web- als auch des Datenbankservers entwickeln.

<sup>14</sup> Transmission Control Protocol: verbindungsorientiertes Transportprotokoll (OSI-Layer 4)

<sup>15</sup> Pipe: Technik zur Interprozesskommunikation, vor allem in UNIX-Systemen verwendet

### 3.2.2.3 Webserver Erweiterungen

Integration der dynamischen Funktionen in den Webserver

Solange nur einzelne dynamische Seiten in einer Website vorkamen, liess sich mit CGI- beziehungsweise FastCGI-Skripten ein gutes Ergebnis erzielen. Mit der steigenden Komplexität entstanden aber auch große Websites, die vollständig aus dynamisch generierten Seiten bestanden. Damit wurde der Performance-Aspekt und die Reduzierung der Webserver-Last immer wichtiger.

Ein wesentlicher Zeitvorteil kann erzielt werden, wenn die Skriptaufrufe direkt durch den Webserver bearbeitet werden, statt sie an einen externen Prozess weiterzureichen. Genau das ist möglich, wenn man den Webserver über zusätzliche Module erweitert. Funktionalitäten, die nicht durch den Server abgedeckt werden, können über solche Erweiterungen nachgerüstet werden.

Minimierung des Kommunikationsaufwandes

So gibt es für den weit verbreiteten Apache-Webserver zum Beispiel zahlreiche Module, die Skriptsprachen wie Perl in den Server integrieren. Ein mit dem Modul `mod_perl` erweiterter Apache kann jedes Perl-Programm, mit kleineren Veränderungen, als Bestandteil des Serverprozesses laufen lassen. Dadurch entfällt die verhältnismäßig langsame Kommunikation mit einem externen Prozess.

Allerdings bringt dieser Ansatz auch Probleme mit sich, da ein fehlerhaftes Servermodul die Stabilität des gesamten Servers beeinträchtigen kann. Normalerweise werden deshalb nur grundlegende Fähigkeiten über Servermodule nachgerüstet.

### 3.2.2.4 Application Server

Spezielle Server mit webtypischen Zusatzfunktionen

Einen anderen Ansatz zur Performancesteigerung und einfacherer Entwicklung von dynamischen Webseiten stellt die Verwendung spezieller Application Server dar. Ein Application Server ist ein separater Prozess, der wie FastCGI-Skripte über mehrere Aufrufe bestehen bleibt und dessen Funktionsumfang durch Module erweitert werden kann. Die Programme, die letztendlich die dynamischen Inhalte generieren, laufen innerhalb des Application Servers.

Dadurch können ihnen zahlreiche Dienste zentral zur Verfügung gestellt werden. So unterhält ein Application Server typischerweise einen ganzen Pool an offenen Datenbankverbindungen, die er auf Anforderung einzelnen Skripten zur Verfügung stellt. Weiterhin können für Webanwendungen häufig benötigte Funktionen wie Transaktionsmanagement, Benutzerauthentifizierung und Bezahlungssysteme durch den Server bereitgestellt werden. Dieser kann dann alle Zugriffe zentral koordinieren und optimieren.

Die meisten Application Server stellen inzwischen auch Möglichkeiten zum Load-Balancing, also der Verteilung der Zugriffe auf mehrere Rechner, und zur Erhöhung der Ausfallsicherheit, zum Beispiel durch Fallback-Systeme, zur Verfügung.

### 3.2.2.5 Weitere Techniken

Es gibt noch viele andere Techniken, die sich nicht sinnvoll in Gruppen einordnen lassen

Die hier aufgeführten vier Unterteilungen kann man als "klassische" Einteilung sehen, die auch die historische Entwicklung widerspiegelt. Neben diesen Technologien existieren zahlreiche weitere, die sich nicht einer gemeinsamen Gruppe zuordnen lassen. Dazu gehören zum Beispiel neue, XML<sup>16</sup>-/XSLT<sup>17</sup>-basierte Systeme wie Cocoon<sup>18</sup>, Informationsaustauschformate wie RDF<sup>19</sup> und herstellerspezifische Serversoftware mit integrierten Entwicklungsumgebungen wie Apple WebObjects. Trotzdem erhält man mit den vorhergehenden Punkten einen ersten, hilfreichen Überblick über die wichtigsten Techniken.

---

16 Extensible Markup Language: Eine Untermenge von SGML zur Beschreibung strukturierter Daten.

17 Extensible Style Language Transformations: Programmiersprache, um XML-Dokumente zu transformieren und darzustellen.

18 XML-basiertes Publishing-Framework der Apache Foundation (<http://xml.apache.org/cocoon/index.html>)

19 Resource Description Framework: W3C-standardisiertes, XML-basiertes Austauschformat für Meta-Daten

## 4 Praktische Anwendungsfälle und Lösungen

Lösungsansätze für verschiedene "Bausteine" einer dynamischen Website

Für dynamische Inhalte beziehungsweise dynamische Webseiten gibt es sehr viele, unterschiedliche Anwendungen. Selten gleichen sich zwei Projekte so sehr, dass man eine bereits verwendete Lösung einfach übernehmen kann. Um trotzdem brauchbare Hinweise geben zu können, wann welche Lösungsansätze sinnvoll sind, werden in diesem Kapitel verschiedene typische "Bausteine" dynamischer Webseiten und deren Anforderungen geschildert. Zu den Anforderungen werden dann jeweils passende Vorschläge gemacht.

Spezifische Anforderungen des konkreten Projektes müssen berücksichtigt werden

Dabei sollte daran gedacht werden, dass diese Hinweise sich aus allgemeinen Überlegungen ergeben und es durchaus sein kann, dass ein konkretes Projekt andere Anforderungen stellt. Hier können nur Vorschläge gemacht werden, mit welchem Ansatz sich eine bestimmte Anforderung in den meisten Fällen umsetzen lässt.

Außerdem sollte berücksichtigt werden, dass eine Website in der Regel aus mehreren Elementen besteht, für die möglicherweise jeweils unterschiedliche Lösungen ideal wären. Trotzdem wird man nur eine davon als gemeinsame Basis für alle Bestandteile verwenden.

## 4.1 Einfache dynamische Inhalte

*Beispiele: Zugriffszähler, Datums- und Zeiteinblendung, "Joke of the day"*

Dieser Abschnitt bezieht sich auf kleine, dynamische Bereiche in einzelnen Webseiten. Für derartige Funktionen gibt es im Internet auch viele Archive, aus denen fertige Skripte kopiert werden können.

Wichtig für diese einfachsten Anwendungen sind vor allem folgende Punkte:

- einfache Programmierung
- flexible Einsatzmöglichkeiten
- geringe Voraussetzungen (server-/clientseitig)

CGI-Skripte als flexible  
und einfache Lösung

Auf diesem Gebiet haben wohl die CGI-Skripte die größte Verbreitung. Sie lassen sich einfach und flexibel in einer fast beliebigen Programmiersprache umsetzen und können von praktisch jedem Webserver ausgeführt werden. Die durchzuführenden Operationen sind von geringer Komplexität und lassen sich mit den durch Programmiersprache und Betriebssystem zur Verfügung gestellten Mitteln realisieren.

Da inzwischen Skriptsprachen wie PHP in den meisten Webservern integriert sind, wird häufig auch für einfache Probleme auf sie zurückgegriffen.

Notfalls clientseitige  
Techniken

Wer ganz ohne Server-Techniken auskommen muss, kann auch clientseitige Techniken wie DHTML/JavaScript oder einfache Applets benutzen. Dabei treten aber die in Kapitel 3.2.1 genannten Kompatibilitätsprobleme und Einschränkungen auf.

## 4.2 Manuell pflegbare Inhaltsbereiche

*Beispiele: aktuelle Nachrichten, Ankündigungen, Artikelsortiment, Preislisten, Newsletter auf Webseiten, redaktionell betreute Webseiten*

Der neben Datenbank-Anbindung vermutlich häufigste Grund für die dynamische Generierung von Webseiten ist eine einfache Pflegebarkeit der Inhalte und manuelle Aktualisierung via Browser. Dabei werden üblicherweise die editierbaren Bereiche in einer Datenbank gespeichert und von dort in die dynamischen Seiten eingefügt.

Das kann die "Willkommenseite" der privaten Homepage mit aktuellen Nachrichten, eine regelmäßig aktualisierte Preisliste, oder auch eine umfangreiche Website sein, die von einer mehrköpfigen Redaktion betreut wird. Mit den unterschiedlichen Einsatzbereichen ergeben sich also zum Teil noch spezielle Anforderungen.

Allgemein wichtige Faktoren:

- gute Performance
- einfache Anbindung an Datenbanken
- keine clientseitigen Voraussetzungen

und speziell bei größeren Websites mit mehreren Redakteuren:

- Mehrbenutzerfähigkeit
- Unterstützung des redaktionellen Ablaufs (Workflow)
- Trennung von Inhalt, Logik und Layout
- gute Skalierbarkeit, Erweiterbarkeit

Skriptsprachen für Projekte von kleiner bis mittlerer Größe

Für Projekte von kleiner bis mittlerer Größe werden sehr oft eingebettete Skriptsprachen wie PHP und ASP verwendet, die als Servermodule realisiert sind. Durch die direkte Integration der Skripte in den HTML-Quelltext sind sie einfach zu erstellen, die Anbindung an Datenbanken ist problemlos. Bei PHP sind die Funktionen hierfür bereits fest in die Sprache integriert, bei ASP sehr flexibel über die Active Data Objects (ADO) gelöst.

Skriptsprachen haben inzwischen durch fortgeschrittene Interpreter und leistungsfähige Caching-Mechanismen eine Geschwindigkeit erreicht, die für die meisten Webseiten ausreicht. Werden die Seiten nur selten geändert gibt es außerdem die Möglichkeit, im Falle einer Inhaltsänderung nur ein Mal aus allen dynamischen Skriptseiten statische HTML-Seiten zu erzeugen und diese an die Clients ausliefern.

Freie Content-Management-Systeme

Auf Basis dieser Skriptsprachen gibt es inzwischen auch gute frei verfügbare Content Management Systeme, wie zum Beispiel Midgard (PHP+MySQL), die für viele Anwendungen bereits ausreichen.

Clientseitige Techniken schließen sich normalerweise aus, da sie zu viele Unwägbarkeiten enthalten und clientseitig auf besonderen Voraussetzungen basieren.

Erheblich höhere Anforderungen bei größeren Projekten

Bei größeren Websites sind die Anforderungen deutlich höher. Hier geht es nicht primär um einfache Erstellung, sondern um ausbaufähige, mehrschichtige Softwaresysteme, die den Arbeitsablauf innerhalb einer Redaktion oder eines Unternehmens unterstützen. Softwaretechnische Eigenschaften und eine detaillierte Planung haben hier eine erheblich größere Bedeutung.

Deshalb sind "einfache" Skriptsprachen wie PHP ungeeignet. Auch ASP läßt sich hier nur als Instrument zur HTML-Generierung einsetzen, die eigentliche Programmlogik sollte in externe Objekte (z.B. COM<sup>20</sup>-Komponenten) ausgelagert werden.

Große Projekte erfordern komplexe Eigenentwicklung oder professionelles CMS

Daher kommen bei einer Eigenentwicklung meistens komplexe Application Server zum Einsatz. In Kapitel 4.6 wird darauf näher eingegangen.

Als Alternative zur Programmierung einer eigenen Software können auch fertige Content Management Systeme (CMS) verwendet werden. Diese sind speziell auf die Erstellung von Seiten mit dynamischen, einfach pflegbaren Inhalten ausgelegt und übernehmen neben der Speicherung und Archivierung von Information meistens auch die Serverfunktion. Die verschiedenen Hersteller bieten sehr unterschiedliche Leistungsmerkmale und Schwerpunkte. Daher kann in dieser Seminararbeit nicht genauer auf einzelne CMS eingegangen werden. Eine Marktanalyse hinsichtlich der eigenen, speziellen Anforderungen ist hier unerlässlich.

### 4.3 Einbindung von externem Content

*Beispiele: News-Ticker, Zusammenfassung von Inhalten für Suchmaschinen, Content-Syndication<sup>21</sup>*

Dynamische Webseiten erhalten ihre Attraktivität durch ständig wechselnde Inhalte. Um diese Inhalte nicht alle selbst generieren zu müssen beziehungsweise Inhalte mehreren Websites zur Verfügung zu stellen, benötigt man ein einheitliches Austauschformat für Informationen. Dabei stehen folgende Kriterien im Vordergrund:

- einfache Integration in Webseiten
- auf offenen Standards basierend
- Erweiterbarkeit

Ein allgemein verwendbares Austauschformat muss sich ohne großen Aufwand in bestehende Webseiten integrieren lassen und sollte möglichst geringe zusätzliche Anforderungen an den Client oder Server stellen. Es sollte auf offenen Standards basieren, um eine möglichst hohe Verbreitung zu erzielen. Damit auch spezielle Anforderungen berücksichtigt werden können, sollte das Format flexible Erweiterungsmöglichkeiten vorsehen.

"Lightweight Content-Syndication" über RSS

Mit dem "RDF Site Summary" (RSS<sup>22</sup>) gibt es einen Standard, der diese Bedingungen erfüllt und sich inzwischen weitgehend durchgesetzt hat. RSS basiert auf dem W3C-standardisierten "Resource Description Framework" (RDF), das ein allgemeines XML-basiertes Framework für Metadaten im Internet beschreibt.

<sup>21</sup> Content Syndication: Der Handel mit Inhalten, die in der Regel gegen Entgelt zur Verfügung gestellt werden.

<sup>22</sup> RDF Site Summary 1.0-Standard: <http://purl.org/rss/1.0/>

RSS wurde ursprünglich von Netscape für das "My Netscape"-Konzept entworfen. Kurze Zeit danach verselbstständigte sich die Entwicklung, heute wird RSS als leichtgewichtiges Austauschformat für Nachrichten, Newsgroups und ähnliches verwendet. RSS-Inhalte lassen sich sehr einfach in Seiten einbinden, beispielsweise als Nachrichtenliste oder Ticker. Die Umsetzung in HTML lässt sich mit allen serverseitigen Skriptsprachen, mit Server-Modulen oder auch clientseitig mit JavaScript durchführen.

Die letzte Version RSS 1.0 erlaubt außerdem die einfache Erweiterung mit eigenen Modulen durch XML-Namespaces. So lassen sich beispielsweise die weitverbreiteten bibliographischen Dublin-Core-Angaben verwenden.

#### 4.4 Anbindung an Datenbanken

*Beispiele: User-Login-Verwaltung, Stichwortsuche über die Website, Warenkörbe, Reservierungs- und Bestellsysteme*

Webseiten als Datenbankschnittstelle

Sehr oft stellt eine Website eine Schnittstelle, ein "Frontend" zur Interaktion mit einer Datenbank dar. Dem Benutzer wird eine einfache Möglichkeit geboten, Datenbanken abzufragen (z.B. Suchfunktionen) oder sie zu verändern, beispielsweise durch das Aufgeben einer Bestellung.

Technisch mit allen Sprachen möglich

Datenbankzugriffe über eine Webseite stellen heute technisch kein Problem mehr dar. Bei jeder Sprache oder Software für dynamische Webseiten ist diese Möglichkeit vorgesehen, beziehungsweise meistens ein elementarer Bestandteil. Für die Auswahl sind daher in den meisten Fällen andere Faktoren ausschlaggebend.

Die grundlegenden Anforderungen lassen sich dabei recht kurz zusammenfassen:

- einfache, effiziente Datenbankanbindung
- Möglichkeit, spezifische Fähigkeiten der gewählten Datenbank voll auszunutzen
- flexible Anbindung, um die Anwendung möglichst mit einer beliebigen Datenbank verwenden zu können

Kompromiss zwischen Performance und Flexibilität

Die letzten beiden Anforderungen stehen sich dabei im allgemeinen gegensätzlich gegenüber. Nutzt man datenbankspezifische Features aus, um eine optimale Performance zu erreichen, geht die Datenbankunabhängigkeit in der Regel verloren. Entscheidet man sich für Datenbankunabhängigkeit und verwendet die speziellen Fähigkeiten nicht, so erzielt man nicht die optimale Performance oder muss die entsprechende Funktion in der Anwendung selbst implementieren. Es gilt also, einen Kompromiss zwischen Performance und Wiederverwendbarkeit zu finden.

Einfache Anbindung mit Skriptsprachen

Auch bei der Datenbankanbindung sind Skriptsprachen sehr beliebt, da man Datenbankinhalte direkt in die Webseite schreiben beziehungsweise Benutzereingaben aus dem Skript direkt in die Datenbank speichern kann.

Mit Active Server Pages kann die Datenbankanbindung über "Active Data Objects<sup>23</sup>" (ADO) sehr flexibel und trotzdem effizient gehalten werden, da sie eine einheitliche Abstraktionsebene bieten, die trotzdem noch den Zugriff auf spezielle Funktionen zulässt.

PHP stellt für viele Datenbanken spezifische Funktionen zur Verfügung. Dadurch kann der Entwickler alle Features nutzen und die optimale Geschwindigkeit erzielen. Allerdings verliert er damit die Flexibilität, eine einheitliche Schnittstelle wie ADO gibt es für PHP noch nicht.

JDBC-Schnittstelle für Java-Anwendungen

Java bietet mit der JDBC<sup>24</sup>-Schnittstelle eine einheitliche und leistungsfähige Möglichkeit, auf verschiedene Datenbanken zuzugreifen und ist in etwa mit ADO zu vergleichen.

Application Server mit eigener Abstraktionsschicht

Application-Server verfügen meistens über eine eigene Abstraktionsschicht für Datenbankzugriffe. Damit muss sich die Anwendung nicht mehr an der speziellen Datenbank orientieren, stattdessen ist die Schnittstelle des Application Servers maßgebend.

## 4.5 Anbindung an vorhandene Applikationen

*Beispiele: Web-Interface für eine bestehende Anwendung, Zugriffsmöglichkeiten für Außendienstmitarbeiter, Integration der Ergebnisse einer Anwendung in die Webseite*

Mit dem Aufbau einer Website werden nicht immer alle dahinter stehenden Anwendungen neu entwickelt. Bestehende Anwendungen, die zum Beispiel in C++, COBOL oder Java implementiert sind, sollen in diesem Fall Inhalte für eine Webseite liefern oder sogar via Internet "fernbedienbar" sein.

Die Wahl einer Serversoftware beziehungsweise Programmiersprache hängt dabei sehr stark von der anzubindenden Anwendung ab. Allgemein muss die Serverseite folgende Bedingungen erfüllen:

- sinnvolle Schnittstelle zur anzubindenden Anwendung  
oder
- Möglichkeit zur direkten Integration der Anwendung in die Seitengenerierung

---

<sup>23</sup> Active Data Objects: von Microsoft standardisierte, auf OLE DB aufbauende API zum einheitlichen Zugriff auf unterschiedliche Datenquellen

<sup>24</sup> Java Database Connectivity: durch Sun standardisierte API zum Zugriff auf Datenbanken und weitere Datenquellen (<http://java.sun.com/products/jdbc/>)

- Effizienz der Schnittstelle
- eventuell: standardisierte Schnittstelle (zur Wiederverwendung)
- einfache I/O–Umsetzung in das Ausgabeformat (hier: HTML)

Shellbasierte Anwendungen über Standard-I/O anbinden

Für einfache, "shellbasierte" Anwendungen bieten sich vor allem die Perl–Varianten wegen ihrer hervorragenden Fähigkeiten zur Verarbeitung von Textausgaben an. Auch mit den anderen Skriptsprachen lassen sich diese Anwendungen leicht anbinden, da der vom Programm ausgegebene Text mehr oder weniger direkt in die Webseite eingefügt werden kann. Die verwendete Schnittstelle ist hier die Standardein–/–ausgabe.

Kombination aus Skriptsprache und Software–Komponenten

Viele Anwendungen, vor allem auf Windows–Plattformen, können ihre Funktionalitäten über ein Komponenten–Modell (z.B. COM) zur Verfügung stellen. Einige Skriptsprachen können diese Komponenten direkt ansprechen, COM–Komponenten können beispielsweise in ASP sehr einfach genutzt werden, um den ansonsten sehr begrenzten Funktionsumfang quasi beliebig zu erweitern. Auch Standards wie CORBA<sup>25</sup>, zum Aufruf von Methoden in entfernten Objekten lassen sich in den meisten Skriptsprachen nutzen. Dadurch lassen sich die Vorteile einer Skriptsprache, wie die einfache HTML–Generierung, mit den softwaretechnischen Möglichkeiten der in höheren Programmiersprachen entwickelten Anwendungen kombinieren, wobei natürlich Performanceverluste durch den erhöhten Kommunikationsaufwand berücksichtigt werden müssen.

Erweiterung der Applikation um Web–Fähigkeiten

Eine weitere Möglichkeit ist, die anzubindende Anwendung selbst "web–fähig" zu machen. Dabei erweitert man die Applikation in ihrer ursprünglichen Programmiersprache um Funktionen, die entweder die Kommunikation in HTML oder die direkte Einbettung der Anwendung in die Webseite erlaubt. Beispiele für die direkte Einbettung wäre bei Java die Implementation der Applet–Schnittstelle oder für VisualBasic–Anwendungen die Umsetzung als ActiveX–Control.

Integration über Application Server

Auch in diesem Aufgabenbereich sei auf das große Feld der Application Server verwiesen, die zahlreiche Möglichkeiten bieten, externe Anwendungen in die dynamische Seitengenerierung einzubinden. So unterstützt zum Beispiel der ColdFusion<sup>26</sup>–Server in der Version 5 die Integration über COM, CORBA, EJB<sup>27</sup> sowie die Erweiterung durch Java–Servlets, Java–Klassen und C/C++.

## 4.6 Neuentwicklung komplexer, modularer und internetbasierter Anwendungen

*Beispiele: Neuentwicklung von Software für eCommerce–Unternehmen, Community–Sites, ...*

<sup>25</sup> Common Object Request Broker Architecture: Beschreibung eines Frameworks für verteilte Anwendungen in heterogenen Umgebungen

<sup>26</sup> <http://www.coldfusion.com>

<sup>27</sup> Enterprise Java Beans: durch Sun standardisiertes, serverseitiges Komponentenmodell für die J2EE–Plattform

Plattformauswahl für eine Neuentwicklung ist eine strategische Entscheidung

Beim Aufbau einer dynamischen Website in Verbindung mit der Neuentwicklung der dahinterstehenden Software, handelt es sich um sehr langfristige Entscheidungen. Neben der Entscheidung für eine Programmiersprache und den damit verbundenen Schwierigkeiten, stehen vor allem IT-strategische Überlegungen im Vordergrund. Welche Zukunft hat eine bestimmte Plattform? Welche Weiterentwicklungen sind möglich? Toleriert man die Abhängigkeit von einer Software und einem Unternehmen oder setzt man auf verbreitete Standards? Für diesen Abschnitt ist es am schwierigsten, allgemeine Kriterien zu finden, dennoch lassen sich einige wichtige Punkte herausgreifen:

- Zukunftssicherheit
- Ausbaufähigkeit
- Effizienz (in Entwicklung und Produktivbetrieb)
- Eignung aus softwaretechnischer Sicht (v.a. verwendbare Programmiersprachen, saubere Mehrschicht-Architektur)
- Einschätzung der technischen Weiterentwicklung
- gegebenenfalls: Portabilität

Skriptsprachen ungeeignet

Betrachtet man die Anforderungen, so sind die Skriptsprachen, vor allem aus softwaretechnischer Sicht, ungeeignet für große Projekte, da eine klare Trennung in mehrere Funktionsschichten schwierig zu realisieren ist. Skriptsprachen spielen hier höchstens noch in Verbindung mit einer komponentenbasierten Software, wie in Kapitel 4.5 erläutert, eine Rolle.

J2EE-Standard als Java-Plattform

Sehr interessant in diesem Bereich ist die von Sun standardisierte Java 2 Enterprise Edition (J2EE<sup>28</sup>). Die J2EE definiert eine Plattform für komponentenbasierte, mehrschichtige Anwendungen. Ein J2EE-konformer Application Server, wie z.B. Orion<sup>29</sup>, muss nach diesem Standard bestimmte Dienste wie Transaktionsmanagement, CORBA und den Java Messaging Service zur Verfügung stellen.

Eine Applikation, die auf dem J2EE-Standard basiert, muss diese Dienste nicht selber implementieren sondern kann sich darauf verlassen, dass sie durch den Server bereitgestellt werden. Das bedeutet auch, dass diese Applikation auf beliebigen J2EE-Servern lauffähig ist.

J2EE ist außerdem auf bestimmte Anwendungsmodelle ausgerichtet, die das Entwickeln einer klaren, mehrschichtigen Architektur unterstützen.

Hinsichtlich der oben genannten Aspekte Zukunftssicherheit, Ausbaufähigkeit, Softwaretechnik und Portabilität spricht also sicherlich vieles für Java und J2EE.

<sup>28</sup> Java 2 Platform, Enterprise Edition: durch Sun standardisierte Plattform für komponentenbasierte, mehrschichtige Anwendungen im Unternehmensbereich

<sup>29</sup> <http://www.orionserver.com>

Windows DNA für Micro-  
soft-basierte Systeme

Microsoft stellt im Rahmen seiner .net-Strategie mit "Windows DNA<sup>30</sup>" ein ähnliches Konzept vor. Dabei spielt ebenfalls der komponentenorientierte Ansatz und die Definition einer allgemeinen Softwareumgebung die zentrale Rolle. Wie dieser Ansatz und die dazugehörige Plattform in der Praxis zu bewerten ist, wird sich erst im Laufe der Jahre 2001/2002 zeigen, da es bis jetzt noch keine konkrete Umsetzung gibt.

Application Server als  
vielseitige Alternative

Neben diesen beiden durch Sun beziehungsweise Microsoft propagierten Plattformen könnten auch die großen Application Server von verschiedenen Herstellern eine Alternative sein. Dabei übernimmt der Application Server wieder eine Reihe allgemeiner Dienste und die clientseitige Präsentationsschicht für das Internet. Viele Hersteller liefern integrierte Pakete, in denen eine Entwicklungsumgebung, Administrationstools und ähnliches enthalten sind. Die Anwendungslogik kann unabhängig davon in einer beliebigen, von dem gewählten Server unterstützten, Programmiersprache entwickelt werden.  
Wie in Kapitel 4.5 bereits erwähnt, unterstützt ColdFusion 5 beispielsweise COM, CORBA, EJB, Java-Klassen und -Servlets sowie C/C++.

---

<sup>30</sup> Windows Distributed interNet Architecture: Plattform und Produktsuite zur Entwicklung mehrschichtiger Komponentenanwendungen unter Windows, Bestandteil von Microsofts .net-Strategie

## 5 Quellenverzeichnis

Dornfest, Rael: RSS: Lightweight Web Syndication: 7/2000,  
<http://www.xml.com/pub/a/2000/07/17/syndication/rss.html>

Dornfest, Rael: Writing RSS 1.0:  
[http://www.oreillynet.com/pub/a/network/2000/08/25/magazine/rss\\_tut.html](http://www.oreillynet.com/pub/a/network/2000/08/25/magazine/rss_tut.html)

Dyck, Timothy: Four scripting languages speed development:  
10/2000,  
<http://www.zdnet.com/eweek/stories/general/0,11011,2646052,00.html>

Farley, Jim: Microsoft .NET vs. J2EE, How Do They Stack Up?:  
[http://java.oreilly.com/news/farley\\_0800.html](http://java.oreilly.com/news/farley_0800.html)

Heitmeyer, David P.: Dynamic Content: CGI and other solutions:  
2000, <http://icg.harvard.edu/~cscie12/lecture/week13/dynamic/handout.html>

Jacob, Petra; Leue, Alf: Migration – die Herausforderung bei der  
Einführung eines Content-Management-Systems:  
<http://www.contentmanager.de/magazin/artikel?ShowID=33>

Microsoft: Microsoft Active Data Objects:  
<http://msdn.microsoft.com/library/psdk/dasdk/ados4piv.htm>

Monson-Haefel, Richard: Enterprise Java Beans, 1. Ausgabe:  
O'Reilly 1999 (ISBN: 1-56592-605-6)

ODB Software Inc.: Dynamic Web Sites Whitepaper:  
[http://www.odbsoft.com/dynamic\\_web\\_sites\\_white\\_paper.html](http://www.odbsoft.com/dynamic_web_sites_white_paper.html)

Pekowsky, Larne: Java Server Pages, 2. Ausgabe: Addison-Wesley,  
2000 (ISBN: 0-201-70421-8)

Plessel, Christian; Wilde, Erik: Dienstbare Geister, Server-Side-  
Techniken im Web – ein Überblick: iX 3/2001, Seite 88

RSS-DEV Working Group: RDF Site Summary 1.0: 3/2001,  
<http://groups.yahoo.com/group/rss-dev/files/specification.html>

Strangeland, Joannie Kervran; Shinoda, Kiko: A Windows DNA  
Primer: 12/1999, <http://msdn.microsoft.com/voices/windowsdna.asp>

Sun Microsystems: JDBC Data Access:  
<http://java.sun.com/products/jdbc/>

Sun Microsystems: Java 2 Enterprise Edition:  
<http://java.sun.com/j2ee/>

W3C: RDF Model and Syntax Specification: 2/1999,  
<http://www.w3.org/TR/REC-rdf-syntax/>

Weissinger, A. Keyton: ASP in a nutshell: O'Reilly 1999 /ISBN: 1-56592-490-8)

Willers, Michael: Weite Reise, Microsofts Weg zu .NET: c't 6/2001, Seite 252

Yerxa, Gregory: The Best Bets for Web Development: 10/1999, <http://www.nwc.com/1020/1020f1.html>