MICROSOFT®

ACTIVE SERVER PAGES

Seminararbeit zur Vorlesung *Verteilte und Parallele Systeme I, Prof. Jung, Prof. Berrendorf*

FH Bonn Rhein Sieg SS 2002

Autoren: Ahmed El Safty, Youssef Yasmine

Inhaltsverzeichnis

1 VORWORT

Diese Seminararbeit entstand im Rahmen der Veranstaltung „*Verteilte und Parallele System*“ gelesen von Prof. Berrendorf und Prof. Jung.

Im Laufe der Vorbereitungen haben wir festgestellt, daß das Themengebiet *Active Server Pages* in dem vorgegebenen Umfang nicht vollständig erfasst werden kann. Somit mussten wir uns auf Teilgebiete beschränken und versuchten uns bei der Bearbeitung nicht in Details zu verlieren. Auf Erweiterungen und Programmierdetails, wie die Anbindung einer Datenbank, wurde ganz verzichtet. Streckenweise werden Themen ausgeführt, die mit ASP unmittelbar nicht zu tun haben. Trotzdem waren solche Ausführungen, wie z.B das HTTP Protokoll, unerlässlich um ASP verständlich zu erklären. Noch zu erwähnen ist, daß wir auf praktische Übungen mit ASP gänzlich verzichten mussten, da uns die benötigte Hard- und Software fehlte.

Im Anschluss an ASP bietet diese Seminararbeit einen kurzen Ausblick auf die ASP.NET Technologie. Dieses Kapitel ist nur darauf bedacht, wichtige Unterschiede und Neuerungen zu ASP zu nennen. Um auf ASP.NET genauer einzugehen, wären weiterer Kenntnisse über das Microsoft .NET Framework nötig, welche im Rahmen dieser Arbeit aber nicht betrachtet werden können.

Ahmed El Safty, Youssef Yasmine

19. Juni 2002

2 EINFÜHRUNG

2.1 Was ist ASP?

ASP - Active Server Pages. Ein Konsequenz aus den wachsenden Anforderungen an Websites? Ein Antwort auf das CGI - Common Gateway Interface? Diese Annahmen sind sicher zutreffend, aber unzureichend um ASP zu erfassen. Es sollte im Vorfeld schon festgehalten werden, dass es sich bei ASP um Microsoft's proprietäre Lösung zur Entwicklung von dynamischen Webseiten handelt und nicht um einen vereinbarten Standard. Man sollte annehmen, dass dieser Umstand Probleme, die Kompatibilät betreffend, mit sich bringt.

2.1.1 Historie

Ein kurzer Blick auf die Historie des Webs erklärt warum ASP, CGI, JSP, PERL, PHP, DHTML etc. immer mehr Einfluß auf die Webseiten Programmierung genommen haben. Ursprünglich wurde das WWW dem Nutzer als reiner Informationsdienst präsentiert, der aus statischen, mittels Hyperlinks verbundenen Dokumenten bestand. Dieses Art der Präsentation wird Hypertext genannt und wird mittels der **H**yper**t**ext **M**arkup **L**anguage realisiert. HTML ist ohne Konkurrenz die Sprache des Webs. Jede Ausgabe zum Browser geschieht durch HTML. Diese Sprache hat maßgeblichen Einfluss auf die Layoutmöglichkeiten und damit das Erscheinungsbild einer Webseite. Der klassische Hypertext, geschrieben in HTML, kann jedoch den Forderungen nach der interaktiven Einbeziehung des Users, der Nutzung des Webs als Medium der Werbung und Präsentation nicht gerecht werden. Der Versuch HTML gänzlich durch einen neueren Standard zu ersetzen, ist zum Scheitern verurteilt, da ein erheblicher Anteil der heutigen Web Präsenzen noch immer in Plain HTML geschrieben sind und vom Begriff Dynamik weit entfernt sind.

Im Laufe der Jahre sind so verschiedenen Systeme entstanden, die außerhalb HTML arbeiten und den Webservern die benötigte Interaktivität verleihen. Ein genanntes Beispiel ist CGI. CGI definiert eine Schnittschnelle zwischen den CGI Programmen und den Webservern, auf denen die CGI Programmen, nach einer Aufforderung des Browers, ausgeführt werden. Für den einfachen Webdesigner, der umfangreiche Anwendungen schreibt, sind CGI Applikationen jedoch oft zu aufwendig. Trotzdem soll*te* auf Interaktivität nicht verzichtet werden.

2.1.2 ASP – Skriptsprache?

Ein anderes weit verbreitetes Mittel sind die sogenannten Skriptsprachen, die kurz gesagt einfache Anweisungsprachen sind, die ein übergeordnetes System in seiner Ausführung steuern. Sie sind leicht zu erlernen und einfach strukturiert, besitzen teilweise aber auch komplexe Befehle. Ausserdem sind sie überwiegend maschinen- und plattformunabhängig.

Ist ASP also eine Skriptsprache? Nein. ASP ist eine Art Technologie oder Lösung, die verschiedene Skriptsprachen zum Einsatz bringen kann, vorzugweise aber VBScript oder JScript. ASP besitzt darüber hinaus, noch sogennante Objekte, in der Art von Befehlen, die die serverspezifischen Aufgaben übernehmen. Es sollte klar herausgestellt werden, das ASP die Befehle der Skriptsprachen nur auf dem Server ausführt. ASP ist also eine Art Umgebung für Server Side Skripting.

2.1.3 ASP – Gestaltungstool?

Noch zu erwähnen ist, dass ASP kein Gestaltungstool ist. Das heißt, dass ASP dem Browser nur pures HTML als Ergebnis liefert, zusätzlich der diversen Erweiterungen, die dann vom Browser lokal ausgeführt werden (z.B. Applets). Daraus kann man schließen, das ASP Seiten keine bestimmten Browser brauchen, um angezeigt zu werden, mit Ausnahme der genannten Erweiterungen, die der Browser unterstützen muss. ASP ist in HTML integriert und ordnet sich der Logik der HTML Seite unter. Dazu aber später mehr.

2.2 Wie funktioniert ASP?

Bevor die Funktionsweise von ASP erläutert wird, sollte man sich darüber klar werden, wo die Unterschiede zwischen serverseitiger und clientseitiger Progammierung liegen.

2.2.1 Server Side – Client Side

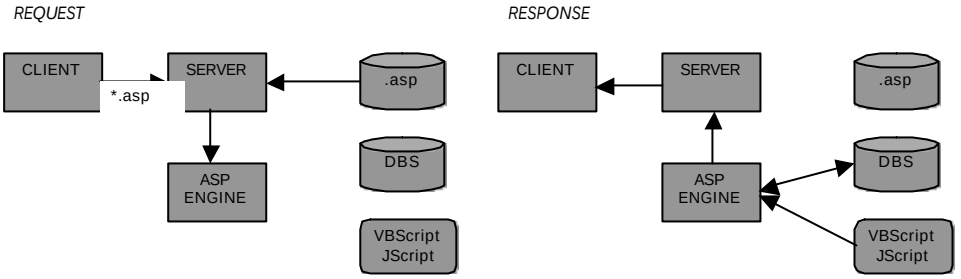
Serverseitige Programmierung bezieht sich auf alle Aktionen die nur auf dem Server (d.h. wo die Webseite als solches liegt) ablaufen können, also etwa die Verarbeitung von Formulareingaben, das Suchen in Datenbanken oder das Generieren von dynamischen Inhalten. Der Server kann dabei nur auf Daten zurückgreifen, die ihm vom Client (also dem Browser) übermittelt wurden. Im Quelltext der Seite findet sich so gut wie nie der eigentliche Programmcode, sondern immer nur die Resultate.

Clientseitige Programmierung bezieht sich dagegen auf Aktionen mit dem vom Server erhaltenen HTML-Output im Browser des Nutzers (d.h. findet tatsächlich nur LOKAL statt). Dieser Output kann auch vom Server generierte oder veränderte JavaScripts beinhalten. Ein wichtiger Unterschied zwischen clientseitiger- und serverseitiger Programmierung liegt darin, daß der Nutzer zwar JavaScript und andere clientseitige Sprachen abschalten kann, serverseitige Aktionen aber immer ausgeführt werden.

2.2.2 Request - Response

Im folgenden dienen die zwei Grafiken dazu, die Funktionsweise einer Client Anforderungen und der dazugehörigen Antwort des Servers zu

verdeutlichen.



Der Client sendet dem entsprechendem Webserver ein Dokument, welches mit der Endung .asp gekennzeichnet ist und erwartet das dieser die Active Server Page generiert. Nach der Aufforderung liest der Server die Seite aus der Platte aus und reicht sie zur Weiterverarbeitung an die ASP Engine. Die ASP Engine kümmert sich um den Ablauf des verwendeten Skripts, ob es sich nun um VBScript oder JScript oder einer anderen Skriptsprache handelt. Sie geht dabei von oben nach unten durch und führt die gefunden Befehle aus und erstellt eine HTML Seite. Dabei werden gegebenfalls weitere Ressourcen in Anspruch genommen, in der Abbildung etwa eine Datenbank. Als Ergebnis erhält der Browser des Clients ein HTML Dokument mit der Endung .asp indem allerdings die Skript Programmteile nicht mehr erkennbar sind. Die erstellte Webseite enthält eventuell andere clientseitige Anweisungen, wie JavaApplets, die auf dem Clientrechner ausgeführt werden.

2.3 Technische Vorraussetzungen

Wie in der Einleitung schon festgestellt, handelt es sich bei ASP um eine Entwicklung von Microsoft. Und wie bei den meisten „Softwareschmieden“ die Ihre Produkte kommerziell verbreiten, verstehen sich die eigenen Entwicklungen am besten mit den eigenen Produkten. So ist es nicht verwunderlich, das ASP nur auf Servern von Microsoft läuft, wenige Ausnahmen mal ausgeschlossen.

2.3.1 Konfiguration[1]

ASP ist Bestandteil des Internet Information Server (IIS) 4.0 von Microsoft, der mit Windows NT Server 4.0 oder den späteren verfügbaren Service Packs ausgeliefert wird. ASP ist sozusagen ein kostenloses Feature der Webserver von Microsoft. Man muss aber nicht unbedingt einen Serverbetriebssystem benutzen, um ASP Anwendungen zu entwickelen. Mit installiertem Personal Web Server 1.0 kann man Windows 95 auch als Entwicklungsumgebung nutzen.

Folgende Konfigurationen sind möglich:

[1] Es sollte darauf hingewiesen werden, das die Quellen nicht top aktuell sind und heute bestimmt andere Konfiguration möglich sind.

- Windows NT Server 4 mit IIS 2.0 oder 3.0
- Windows NT Server 4 mit Windows NT 4.0 Option Pack (IIS 4)
- Windows NT Workstation 4 mit Peer Web Server 2.0
- Windows 95 mit Personal Web Server 1.0

Natürlich sollte die Entwicklung mit Windows 2000 oder Windows XP gleichermaßen funktionieren, das sie die gleiche Plattform wie Windows NT nutzen.

2.3.2 Serverbetriebssysteme

Wie oben schon erwähnt gibt es wenige Alternativen zu den Microsoft Serverbetriebssystemen für ASP. Es gibt jedoch einige Anbieter die eine ASP-Engine für VBScript und JScript unter den Webservern von O´Reilly und Netscape anbieten. Ein bekanntes Produkt ist ChiliASP. Selbstverständlich bietet Microsoft selber kein Produkt, welches die ASP-Engine erweitert um andere System zu unterstützen.

2.3.3 Hardwarevorraussetzungen[2]

Im Grunde reicht für die Entwicklung jeder PC. Es ist aber zu empfehlen ein verteiltes Betriebssystem wie Windows NT einzusetzen, zumindest die Workstation. Ferner sollte für die Entwicklungsumgebung genügend Festplattenspeicher zur Verfügung stehen, da ja der Webserver, Windows NT und die ASP Erweiterungen installiert werden.

Abschliessen kann man noch hinzufügen das NT auch auf DECs Alpha und den PowerPc Plattformen läuft, jedoch sollte auch hier lieber eine PC Plattform benutzt werden. Zum einen unterscheiden sich die versch. Plattformen bei der Geschwindigkeit nicht wirklich, zum anderen sind PC Komponenten signifikant billiger und verfügbarer.

3 ERSTE SCHRITTE MIT ASP

In diesem Abschnitt gehen wir auf die Erstellung eines ASP Dokuments ein...

3.1 Einbindung ins HTML Dokument

Wie eingangs schon erwähnt, wurde ASP entwickelt um Interaktion zwischen Client und Server zu ermöglichen. Somit kann man sagen, das ASP ähnlich dem CGI eine Schnittstelle zwischen Client und Server bildet. Diese Schnittstelle muss vorher den Kommunikationspartnern bekannt gemacht werden, d.h. der Client muss wissen, wie der Dienst auf einem Server zu

[2] Empfehlung von Jörg Krause, siehe Quellen

erreichen ist; der Server muss wissen, wie die Client-Nachricht zu entpacken und zu bearbeiten ist.

Weiterhin haben wir festgestellt, das sich ASP der HTML Struktur unterordnet. Aus diesen Gründen werden ASP Skripte in einer HTML-Seite eingebunden. Es werden zwei Möglichkeiten angeboten:

3.1.1 Links

http://server/skript.asp[?f1=val1&f2=val+2]

Das obige Beispiel ist ein ganz normaler und einfacher Aufruf eines ASP-Skriptes. Der Client ruft einen Server auf und fordert eine Seite auf. Der Client weiß aber nicht, ob es sich um eine reine HTML Seite oder eine Programm handelt. Der Server dagegen weiß anhand der Endung .asp, dass es sich um ein ausführbares Programm handelt. Statt wie im normalen Fall, die Datei zu finden und den Code an den Client zu senden, ruft der Server die ASP-Engine auf, die das Skript ausführt, das Skript erzeugt dann Ausgaben in HTML-Format, diese Ausgabe wird dann dem Server zurückgegeben, der dann dieses HTML-Code an den Client zurückschickt. Der Client (Browser) kann dann dieses Ergebnis formatieren und ausgeben.

3.1.2 Formulare

Die zweite Möglichkeit besteht darin, eine Eingabemaske für den Benutzer zur Verfügung zu stellen. Diese Masken werden Formulare genannt, und können mit dem Container <*FORM*> erzeugt werden.

<form name="nm" action="link" method="[post | get]">

<Input> <!-- weitere Eingabefelder -->

</form>

Der Attribut *name* dient dazu dem Skript zu informieren, um welchen Form es sich handelt, falls das Skript mehrere Forms auf einer Seite bearbeiten soll.

Das Attribut *action* spezifiziert das Skript, an das die eingegebene Daten übertragen werden, durch seine URL-Addresse.

Das Attribut *method* legt fest, wie die Daten vom Browser an den Server übertragen werden, und kann zwei Werte nehmen, *POST* oder *GET*.

***GET**:* Der Browser hängt die Formulardaten einfach an die URL des Skripts an; ein '?' trennt den Skriptnamen von den Daten. GET wird für kurte Strings verwendet. Die meisten WWW-Browser verkraften nicht mehr als 512 Zeichen in der URL.

POST: Das Skript liest dann die Daten von der Standarteingabe, genau so, als würde der Anwender die Daten nacheinander eintippen. *POST* wird für

umfangreiche Daten verwendet.

Die verwendete Methode kann man anhand der Variable *REQUEST_METHODE* der Collection ServerVariables im Objekt *Request* abfragen

3.2 Skriptsprachen

Die Erweiterung des Webservers um die Funktionalität der Active Server Pages erlaubt die Ausführung einer (oder mehreren) Skriptsprachen. Vorher möchte man den Begriff „Skriptsprache“ definieren.

Skriptsprachen werden nicht explizit kompiliert oder mit Bibliotheken verbunden. Dies geschieht hinter den Kulissen; Ein Interpreter oder eine Engine liest den Quellcode durch, checkt es auf Fehlern und führt es aus. Ein veränderter Quelltext verändert sein Verhalten bei der nächsten Ausführung.

Typedeklarationen sind für Variablen freiwillig oder gar nicht möglich.

„Höhere“ Skriptsprachen bieten komplexere Datenstrukturen (Listen, Assoziative Listen) und dazu gehörige Operationen.

ASP unterstützt standardmäßig VBScript und JScript. Die Firma ActiveStat hat ein Softwarepaket entwickelt, mit dem die Nutzung von Python und Perl ermöglicht wird. In diesem Dokument wird auf VBScript eingegangen.

3.3 Aufbau von Skripts

Bevor man sich mit einer konkreten Programmiersprache beschäftigt, sollte man den Aufbau von ASP-Skripts genauer betrachten. ASP-Skripts sind normalerweise in HTML-Code eingebettete Befehlsfolgen und Anweisungen. ASP-Applikationen bestehen u.a. aus folgenden Elementen:

3.3.1 HTML-Code

3.3.2 Script-Delimiters

Auch Script-Seperatoren oder Begrenzer genannt. Sie dienen dazu, einzelne Segmente des Codes untereinander abzugrenzen

3.3.3 Script-Code

Die verwendete Sprache wird in der ersten Zeile (vorgegeben in der global.asa) mit der Direktive *<%@ LANGUAGE="Skriptsprache" %>* angegeben. ASP-Anweisungen sind durch ***<%*** *und* ***%>*** eingeschlossen und

so von statischen Elementen unterschieden.

3.3.4 Beispiel 1

```
<%@ LANGUAGE="VBSCRIPT" %>
<html>
<head>
<title>ASP mit VBScript</title>
</head
<body>
<% Response.Write "<B>Hello World !!</B>" %>
<%= "<HR>" %>
</body>
</html>
```

3.3.5 Beispiel 2

```
<SCRIPT RUNAT=“SERVER“ LANGUAGE=“VBScript“>
function SayHello(){
response.Write(“Hello !!“);
}
</SCRIPT>
...
<% CALL SayHello %>
```

3.4 VBScript

VBScript wird im ASP-Installationspaket ausgeliefert. Die Sprache basiert auf VisualBasic. VisualBasic ist weit verbreitet und wird durch die Anwendung in den Microsoft Office-Produkten stark unterstützt.

3.4.1 Kommentare

Wie jede Sprache, enthält VBScript-Quelltext neben Codeanweisung Kommentare, die oft den kryptischen Code erläutern. VBScript kennt zwei Kommentararten. Das altbekannte REM (steht für remark) stammt aus der ursprünglichen Basic-syntax. Später entstand das Kommentarzeichen „'“ - der Apostroph. Mit diesen Kommentararten lässt sich eine Zeile kommentieren. Kommentare auch mitten in der Zeile stehen, allerdings nicht bei Ausdrücken, die Ausgaben erzeugen.

```
<%
REM Diese Zeile ist ein Kommentar
i= i+1 ' Dieser Ausdruck i um eins
%>
<%= name ' schreibt name und funktioniert nicht %>
```

3.4.2 Variablen

Ohne Variablen haben Programmiersprachen keine Bedeutung. Variablen sind das Gedächtnis von Programmen. Dort werden Werte abgelegt, die man später aufruft und eventuell manipuliert oder ausgibt. Intern sind Variablen nur Namen, die einen bequemen Zugriff auf einen Speicherplatz im Computer verweisen. Variablen können aus Buchstaben und Zahlen bestehen und müssen mit einem Buchstaben beginnen. Das einzige zulässige Zeichen ist das Unterstrich. Die maximale Länge beträgt 255 Zeichen. Um die Nutzung und Prüfung nicht dem Zufall zu überlassen, können Variablen mit dem Schlüsselwort DIM explizit erzeugt werden. Variablennamen müssen in ihren Geltungsbereich eindeutig sein.

3.4.2.1 Beispiele

```
DIM name
DIM A, B
A = 7
name = “Bill“
```

3.4.2.2 Geltungsbereich

Der Geltungsbereich einer Variable ist in zwei Stufen, Private und Public, einstellbar. Die beiden Schlüsselworte ergänzen den DIM-Befehl und ermöglichen es, den Geltungsbereich einer Variable zu beschränken. Variablen sind lokal, wenn sie nur innerhalb einer Prozedur gelten und nach dem Verlassen der Prozedur verschwinden. Das Schlüsselwort für diese Verhaltensweise ist Private. Ohne Angabe des Schlüsselwortes sind alle Variablen global. Also von allen Prozeduren gleichermaßen erreichbar und veränderbar. Man kann explizit das Schlüsselwort Public verwenden.

3.4.2.3 Lebensdauer

Definiert wird in der Programmiertechnik auch die Lebensdauer einer Variable. Das ist die Spanne von der Erzeugung der Variable mit *DIM* bis zur Verlassen des Geltungsbereichs. Lokale Variablennamen leben solange, wie die Programmausführung sich innerhalb der Prozedur befindet.

3.4.3 Arrays

Arrays sind Sammlungen von Variablen mit einem gemeinsamen Name. Der Inhalt von Arrays wird über Indizes (beginnend mit 0) angesprochen.

DIM name(9) ' 10 Speicherplätze

name(0) = "Bill"

DIM name(9,1) '2D-Array mit 10 Speicherplätze

REDIM name(4)

REDIM name()

3.4.4 Datentypen

Es gibt in VBScript keine fest definierbaren Datentypen und strenge Typprüfungen wie in anderen Programmiersprachen. Der einzige Datentyp, der existiert, wird Variant genannt. Dieser Datentyp kann entweder eine Zahl oder eine Zeichenkette sein. VBScript wird die Typprüfung nur dann vornehmen und möglicherweise einen Laufzeitfehler erzeugen, wenn zwei unverträgliche Typen in einer Operation verbunden werden. z.B. Division einer Zahl durch String. VBScript erkennt den Datentyp automatisch. Die Verwendung von Strings kann durch "" erzwungen werden.

3.4.5 Konstanten

Konstanten sind in allen Programmiersprachen nützlich, um feste, immer wieder benötigte Werte mit verständlichen Begriffen zu umschreiben. Das erhöht die Lesbarkeit des Programms und erleichtert vor allem die Änderungen. Konstanten werden mit dem Schlüsselwort *CONST* deklariert und sofort mit einem Wert belegt.

<% CONST max_element = 100 %>

3.4.6 Zeit und Datum

VBScript enthält viele Funktionen zur Abfrage der aktuellen Zeit und des aktuellen Datums.

Es ist jetzt genau *<%= NOW %> (Serverzeit)
*

Heute ist der *<%= DATE %>
*

Und es ist: *<%= TIME %>
*

Der aktuelle Monat ist : *<%= MONTH(DATE) %>
*

Die Stunde ist: *<%= HOUR(TIME) %>
*

3.4.7 Formatierung

Nicht immer ist die Ausgabe der ermittelten Werte so, wie man es sich vorstellt. VBScript stellt Funktionen für die Formatierung von Zahlen, Datum und Währung bereit.

<% preis = 49.80 %>

Das Buch kostet: *<%= FORMATCURRENCY(preis) %>*

3.4.8 Mathematische Operatoren und Funktionen

3.4.8.1 Operatoren

Mit Operatoren werden zwei Werte, Konstanten oder Variablen verknüpft. Die mathematischen Operatoren repräsentieren die Grundrechenarten und zwei erweiterte Operatoren für Modulo- und Potenzberechnungen.

*+, -, *, /, Mod, ^*

3.4.8.2 Funktionen

VBScript ist nicht unbedingt die Sprache mit der besten mathematischen Unterstützung. Für elementare Berechnungen bietet sie zahlreiche Funktionen zur Verfügung. u.a. *COS(), SIN(), LOG(), EXP() ..*

Für kommerzielle Anwendungen stehen ein paar Rundungsfunktionen zum Einsatz bereit. u.a. *ABS(), INT(), ROUND(), ...*

3.4.9 Zufallszahlen

RND() gibt eine Zufallszahl zwischen 0 und 1 zurück. So könnte beispielsweise der Wert 0,77865123 erzeugt werden. Zufallszahlen werden häufig für die Generierung von Schlüssel benötigt. Der Startwert der *RND()*-Funktion ist immer unsinnigerweise immer derselbe. VBScript löst das Problem so um, indem die Funktion *RANDOMIZE()* aufgerufen wird. Nur so wir der Startwert der *RND()*-Funktion verschoben.

3.4.10 Zeichenkettenoperatoren und -funktionen

Neben den Mathematischen Operatoren lassen sich auch Zeichenketten mit entsprechenden Funktionen und Operatoren verändern

3.4.10.1 Operatoren

Zeichenketten können mit den Operatoren “&“ oder “+“ konkateniert werden. Bei Verwendung von Variablen mit dem zweiten Operator wertet zuerst VBScript die Zeichenkette aus und versucht, mathematische Operation auszuführen. Enthält die Zeichenkette nur Zahlen, wird eine Addition ausgeführt.

<%= 17 & 3 %> ergibt 173

<%= 17 + 3 %> ergibt 20

<%= “17“ + “3“ %> ergibt 173

3.4.10.2 Funktionen

Da Zeichenketten häufig zerlegt werden müssen, dienen in VBScript vor allem die drei Funktionen *LEFT(), MID(), RIGHT(),* die den linken, mittleren und rechten Teil einer Zeichenkette zurückgeben. Bei anderen Programmiersprachen wird die Funktion substr() benutzt. *LEN()* gibt die Länge eines Strings zurück. u.a. werden folgende Funktionen angeboten: *REPLACE(), LCASE(), UCASE(), INSTR(), STRCOMP()...*

4 OBJEKTE UND KOMPONENTEN

4.1 Objekte

Im Laufe der Entwicklung von Rechnersystemen haben sich viele Programmiersprachen nebeneinander etabliert. Jede Sprache für sich wurde weiter perfektioniert und an die gestiegenen Bedürfnisse der Programmierer angepasst. Am Anfang wurde linear und prozedural programmiert. Dabei wurde der Code von oben nach unten ausgeführt und nach Verzweigung mit Sprungbefehlen die Ausführung an andere Stelle fortgesetzt. Um bestimme Programmstücke mehrfach benutzen zu können, sind Unterprogramme, auch Prozeduren genannt, zu verwenden. Objekte sind Sammlungen von Programmcode (wie Prozeduren) mit zusätzlichen Variablen, die bestimmte innere Zustände und Eigenschaften haben.

4.2 Objektorientierte Programmierung

In der Welt der Objektorientierten Programmierung (OOP) werden die Funktionen eines Objekts Methoden genannt. Die Methoden beschreiben das Verhalten eines Objekts. Das Objekt selbst wird in einem Quelltext auch nie verwendet, sondern eine daraus abgeleitete Instanz. Es wird also nicht das Objekt selbst, sondern eine Klasse, deren Variablen auf das neue Objekt vererbt werden, zur Verfügung gestellt. Die Variablen beschreiben die Eigenschaften des Objekts.

VPScript setzt OOP nicht ganz so konsequent um. VisualBasic ist eine klassische prozedurale Sprache und tut sich mit der Aufnahme der komplette Funktionalität der Objekte schwer. Trotzdem stehen Objekte zur Verfügung. VBScript definiert noch eine zusätzliche Erweiterung - die Kollektionen. Das sind Sammlungen von Objekten unter einem gemeinsamen Namen.

4.3 Objektsyntax

die normale Objektsyntax besteht immer aus dem Objektname, einem Punkt als Trennzeichen und den Namen der Methode oder Eigenschaft.

<% Response.Write(“Ausgabe“) %>

4.4 Eingebaute Objekte

In VBScript sind sieben Objekte eingebaut worden. Fünf davon sind bereits von VisualBasic übernommen. Sie dienen den elementaren Ein- und Ausgabeoperationen und ersetzen die entsprechenden Befehle der Oberflächensteuerung von VisualBasic. Das ist notwendig, denn die Erzeugung eines Dialogfeldes darf für den Inhaber eines Skripts nicht möglich sein, andererseits macht es keinen Sinn, Windows-typische Dialoge zu programmieren, wenn die Ausgabesprache HTML die meisten Elemente gar nicht beherrscht.

4.4.1 Request

Fordert Informationen vom Browser an bzw. enthält die Informationen, die von einem HTML-Formular übertragen wurde.

4.4.2 Response

Sendet Informationen zum Browser, vor allem zur direkten Ausgabe von Text aus VBScript heraus.

4.4.3 Server

Steuert die ASP-Umgebung und dient beispielsweise der Objektsteuerung und Kontrolle der TimeOut-Zeiten.

4.4.4 Err

Steuert die Laufzeitfehler.

4.4.5 Session

Speichert Informationen über die aktuelle Sitzung mit Hilfe von Cookies.

4.4.6 Application

Verteilt Informationen zwischen den verschiedenen Nutzern einer Sitzung. Damit wird die Interaktion zwischen gleichzeitig präsenten Nutzern einer Webseite möglich.

4.4.7 ObjectContext

Steuert Transaktionen, die vom Microsoft Transaktion Server MTS verwaltet werden

Die in den Objekten enthaltenen Methoden und Eigenschaften werden durch die Punktschreibweise aufgerufen[3]

Objekt.Methode Parameter

Objekt.Eigenschaft Parameter

4.5 Übersicht über mitgelieferte Komponenten (Collections)

Komponenten sind ähnlich wie die eingebauten Objekten Sammlungen von Methoden und Eigenschaften. Der wesentliche Unterschied ist die Einsatzbreite. Komponenten sind sehr viel stärker spezialisiert und auf eine ganze bestimmte Aufgabe zugeschnitten

Mit der Version 2.0 von der Active Server Pages (IIS 4) werden neun Komponenten geliefert.

4.5.1 Die Werbebanner-Komponente (*Ad Rotator*)

wird verwendet, um einen rotierenden Banner z erzeugen und zu verwalten.

4.5.2 Browsereigenschaften-Komponente (*Browser Capabilities*)

ist eine Komponente, die bei der Auswertung der Browsereigenschaften verwendet wird. Man kann damit die HTML-Ausgabe an die Fähigkeiten der Browser anpassen.

4.5.3 Die Inhaltsverbindungs-Komponente (*Content Linking*)

vereinfacht die Navigation zwischen mehreren Seite.

4.5.4 Die Zähler-Komponente (*Counter*)

wird verwendet, um Hits auf einem Web zählen.

4.5.5 Die Seitenzähler-Komponente (*Page Counter*)

ergänzt die Zähler-Komponente um Funktionen zur Registrierung von Hits auf einzelnen Seite eines Webs.

[3] Klammern sind nicht immer notwendig

4.5.6 Die Inhaltsrotations-Komponente (*Content Rotator*)

ist die Basis für einen Newsservice. Im Gegensatz zur Banner-Komponente wird HTML-Text ausgegeben.

4.5.7 Die Zugriffstest-Komponente (*Permission Checker*)

verwaltet geschützte Seiten.

4.5.8 Das ActivX-Data-Object (*ADO*)

ist eine spezielle Komponente, die die gesamte Funktionalität des Zugriffs auf Datenbanken beinhaltet.

4.5.9 Die Dateizugriffs-Komponente (*File Access*)

ist eine neue Komponente im IIS 4, die enthält die verschiedenen Objekte, mit denen auf Datei- und Verzeichnissystem des Servers zugegriffen werden kann.

4.6 Die Objekte Request und Response

In diesem Abschnitt werden die beiden wichtigsten eingebauten Objekte unter die Lupe genommen. Vorher sollte man einiges über die Funktionsweise des World Wide Web und über die Prinzipien des Informationsaustausches in diesem Medium erfahren.

4.6.1 Das HTTP-Protokoll

Damit der Browser Informationen vom WWW-Server und umgekehrt erhält, bedarf es gewisser Methoden und Regeln, die einen einwandfreien Informationsfluss gewährleisten. Verantwortlich dafür zeichnet sich das HTTP-Protokoll. HTTP basiert auf Client/Server- bzw. Frage/Antwort-Regeln. HTTP besitzt drei wichtige Eigenschaften:

4.6.1.1 HTTP ist nicht verbindungsorientiert

Nachdem eine Anfrage gestartet wurde, unterbricht der Client die Verbindung zum Server und wartet auf eine Antwort.

4.6.1.2 HTTP ist medienunabhängig

Jede Art von Daten kann mittels HTTP versendet werden, solange der Client und Browser wissen, wie diese Informationen zu verarbeiten sind. Wie mit bestimmten Inhalten zu verfahren ist, beschreiben sogenannte “Standard Labeling Schemas“. Durch dieses Label bzw. Inhaltstyp werden MIME-Spezifikationen definiert.

4.6.1.3 HTTP besitzt keinen Status

Diese Tatsache resultiert in erster Linie aus dem Sachverhalt, dass HTTP

nicht verbindungsorientiert arbeitet. Server und Client wissen nur voneinander durch den Austausch von Requests (Anforderungen). Nach dem Austausch dieser Requests vergessen beide Parteien sofort einander. Aus diesem Grund können weder der Client noch der Browser Informationen zwischen verschiedenen Requests oder nach dem Besuch verschiedener Web-Sites behalten.

Ein Client-Browser, der sich den Inhalt einer bestimmten Web-Site anschauen möchte, muss diese Informationen logischerweise vom Server anfordern. Der Client-Browser sendet einen Request, der den URL (Uniform Ressource Locator) der angeforderten Seite sowie zusätzliche Informationen enthält. Diese zusätzlichen Informationen (Zeit, Browser-Version usw.) sind im sogenannten HTTP-Header enthalten.

Nachdem der Server einen Request für eine bestimmte Seite erhalten hat, geschehen verschiedene Dinge:

I. Der Server stellt fest, ob die entsprechende Seite existiert.

II. Der Server verarbeitet den HTTP-Header.

III. Der Server stellt den Typ und die Extension der jeweiligen Seite fest und ob diese vom MIME unterstützt wird.

IV. Der Server überprüft die Seite auf ausführbare ASP-Scripts - falls notwendig-, die dann an die ASP-Engine zur Ausführung weitergeleitet werden.

V. Der Server erstellt einen HTTP-Header und sendet den Inhalt. Der HTTP-Header muss den Client-Browser über zwei Dinge informieren:

    a. Der Status-Header teilt dem Client-Browser den Status seiner Anfrage mit. Der Status-Header besteht aus einem Status-Code und einer Status-Nachricht. Oft verwendete Server-Status-Codes sind in der unteren Tabelle aufgeführt.

    b. Sobald der Server damit beginnt, den Inhalt zu senden, muss der Header den Inhaltstyp näher spezifizieren. Dies ist notwendig, damit der Client-Browser weiß, wie er mit dem Inhalt umgehen muss. ContentType ist einer von diversen MIME-Typen, die per RFC 1521 definiert wurden. Eine Aufzählung der wichtigsten Typen sind in der unteren Tabelle aufgeführt.

4.6.1.4 Tabelle der Server Status Codes

Code	Beschreibung
200	**OK.** Der Request war erfolgreich und es wurden Daten versandt.
301	**Moved Permanently.** Der angeforderte Inhalt befindet sich an einer neuen Stelle. Ist der neue Standort der Information bekannt, so wird der Browser automatisch dem neuen Link folgen. Einige ältere Browser unterstützen diese Funktion nicht.
302	**Moved Temporarily.** Der angeforderte Inhalt befindet sich vorübergehend an einer neuen Stelle. Der Code wird bearbeitet wie unter 301.
400	**Bad Request.** Die Anfrage wurde vom Server nicht verstanden. Der Server vermutet, dass die Ursache der Clientseite ist.
401	**Unauthorized.** Die Anfrage erfordert eine Authentifizierung des Benutzers.
403	**Forbiden.** Die Abfrage wurde verstanden, die Ausführung ist aber verboten.
404	**Not Found.** Der angeforderte Inhalt konnte auf dem entsprechenden Server nicht lokalisiert werden. Die Nachricht wird auf dem Browser des Anwenders ausgegeben.
500	**Internal Server Error.** Der Server gibt einen internen Fehler aus, so dass der Request nicht bearbeitet werden kann. Es daran liegen, dass das Skripte (Syntaktische und Laufzeit-) Fehler enthält.

4.6.1.5 Tabelle der Content Typen

Typ/Subtyp	Beschreibung
text/html	Der Client-Browser soll HTML-Code anzeigen.
text/plain	Der Client-Browser soll nur ASCII-Code

Typ/Subtyp	Beschreibung
	anzeigen.
image/gif	Dieser Typ beschreibt eine GIF-Grafik-Datei und teilt dem Browser mit, dass er diese Grafik anzeigen soll oder eine entsprechende Applikation zu starten ist, die es ermöglicht, die Grafik anzuzeigen.

4.6.2 Das Objekt Response

Mit dem **Response**-Objekt können Informationen und Steuerungen erstellt und an den Client geschickt werden. Das Objekt hat Methoden und Eigenschaften, die die Funktionen des Webservers direkt kontrollierbar machen.

Es gibt verschiedene Wege, um die HTTP-Ausgabe zu kontrollieren bzw. zu steuern. Die folgenden ASP-Eigenschaften, Methoden und Komponenten können separat oder gemeinsam genutzt werden, um die benötigten Resultate zu erhalten:

4.6.2.1 Status-Eigenschaft

kann verwendet werden, um beispielsweise eine Authentifizierung zu erzwingen.

Response.Status = 401

4.6.2.2 ContentType-Eigenschaft

teilt dem Client mit, wie die gesendeten Daten darzustellen sind.

Response.ContentType = type/subtype

Die beiden Eigenschaften müssen gesetzt sein, bevor irgendwelche HTML- oder Bildschirmausgaben an den Browser gesendet werden; anderenfalls wird eine Fehlermeldung generiert.

4.6.2.3 Buffer-Eigenschaft

besagt, ob die gesendeten Daten auf dem Webserver gepuffert werden. Die Eigenschaft ist nützlich, wenn man den Programmablauf auf dem Browser sehen möchte.

Response.Buffer = TRUE

4.6.2.4 CacheControl-Eigenschaft

bewirkt, ob ein Proxy die vom ASP-Skript generierte Ausgabe speichern soll.

Response.CacheControl = "Public|Private"

4.6.2.5 Expires-Eigenschaft

bewirkt, wie lange (in Minuten) eine Seite auf dem Browser gecacht werden kann bevor sie nochmal gefordert wird.

Response.Expires = 0

4.6.2.6 Response.Write("String")

Dies ist die wichtigste Methode. Sie dient dazu Zeichenketten auszugeben.

4.6.2.7 Response.Redirect(URL)

Die Redirect Methode bezieht sich auf die Status-Eigenschaft. Diese Methode teilt dem Browser mit, wo eine neue Seite zu finden ist und gestattet eine automatische Umleitung. Die Redirect-Methode stellt zwei Dinge sicher, damit der Client-Browser den Standort der neuen Web-Seite findet. Zum einen sendet die Methode einen HTTP-Status-Header mit dem Code 302:

HTTP/1.0 302 Object Moved

Location URL

Im Anschluss daran generiert die Methode automatisch eine neue URL in der HTML-Seite. Alle anderen Ausgaben werden ignoriert. Clients ohne automatische Redirection werden ebenfalls unterstützt.

4.6.2.8 Response.BinaryWrite data

Die BinaryWrite-Methode wird dazu benutzt, Daten ohne Übersetzung direkt in den HTTP-Datenstrom zu schreiben. Diese Methode ist besonders geeignet, kleine Grafiken, Multimediadateien zu verschicken.

4.6.2.9 Response.AppedToLog String

Die AppendToLog-Methode kann benutzt werden um z.B. Benutzerinteraktion zu protokollieren. Die Ausgaben werden in die Server-Logdatei umgeleitet.

4.6.2.10 Response.Clear

Die Clear-Methode löscht jede gepufferte HTML-Ausgabe .

4.6.2.11 Response.End

Mit der End-Methode kann die Ausführung des Skripts an der Stelle abgebrochen werden.

4.6.3 Das Objekt *Request*

Auch Anforderungen lassen sich steuern. Dazu wird das Objekt Request benutzt. Das Objekt hat Funktionen und Methoden, die Informationen eines

Clients abfragen können.

4.6.3.1 Request.TotalByte

In der TotalByte-Eigenschaft ist die Gesamtzahl der Bytes enthalten, die neben dem HTTP-Header vom Client gesendet werden.

4.6.3.2 Request.BinaryRead(Count)

Die BinaryRead(count)-Methode ermittelt den Inhalt eines HTML-Formulars in binärer Form. Count gibt die Anzahl der zu lesenden Bytes an.

4.6.3.3 Request.ClientCertificate(key[code])

Die ClientCertificate-Komponente enthält Informationen über das Zertifikat eines Client.

key kann u.a. folgende Werte haben:

I. *Subject:* Beschreibung des Zertifikats
II. *Issue:* Herausgeber des Zertifikats
III. *ValidForm:* Das Datum, ab dem das Zertifikat gültig wird

4.6.3.4 Cookies Komponente

Die Cookies Komponente enthält alle in einem HTTP-Request gesendeten Cookiewerte.

4.6.3.5 Form Komponente

Die Form Komponente enthält eine Reihe von Objekten, die Formulardaten speichern. Das ist eine sehr wichtige Komponente bei der Webprogrammierung. Der Method-Wert im Formular-Code erzwingt die Benutzung dieser Komponente für die Auswertung der Daten.

Request.Form(field)[index.count]

field ist der Feldname, zu dem die Komponente den Wert finden soll.

index ist optional, wird dann benutzt wenn z.B. MultipleChoicelisten im Formular enthalten sind.

count gibt die Anzahl der Felder an, die übertragen wurden.

Beispiel

```
<%
FOR EACH feld IN Request.Form
Response.Write(feld &“ = “)
Response.Write(Request.Form(feld) & “<BR>“)
```

NEXT

%>

4.6.3.6 QueryString Komponente

Die QueryString-Komponente speichert ebenfalls die Daten, die von einem Client gesendet wurden. Der Unterschied zu der vorherigen Komponente liegt daran, dass die Daten nicht von einem Formular gesendet wurden; sonder die Daten wurden an einer URL angehängt, getrennt von dem Skriptnamen mit einem „?“ und voneinander mit einem „&“.

http://server/skript.asp?f1=val1&f2=val+2

Request.QueryString(variable)[(index)|.Count]

4.6.3.7 ServerVariables Komponente

Die ServerVariables-Komponente ermöglicht den Zugriff auf die HTTP-Header.

<%

FOR EACH feld IN Request.Form

Response.Write(feld &“ = “)

*Response.Write(Request.Form(feld) & “
“)*

NEXT

%>

4.6.3.8 REQUEST_METHOD

kann den Wert GET oder POST haben

4.6.3.9 SCRIPT_NAME

der virtuelle Pfad der aktuellen ASP-Seite. Sinnvoll eingesetzt kann diese Variable für Programme die auf beliebigen Servern laufen sollen.

4.6.3.10 SERVER_NAME

der Name des Webservers

4.6.3.11 SERVER_PORT

Portnummer des Servers

4.6.3.12 HTTP_USER_AGENT

Der Typ des Browsers, von dem die Abfrage kommt.

4.6.3.13 HTTP_REFERER

enthält die URL, von der der Benutzer seine Abfrage stellt, wenn die ASP-Seite durch Anklicken eines Hyperlinks auf einer anderen Seite erreicht wurde. Sie wird z.B. für die Auswertung von Werbeträger eingesetzt.

4.6.3.14 QUERY_STRING

enthält die Zeichenkette nach dem Fragezeichen, dem Trennzeichen für die Übertragung von Parametern zum Server.

5 ASP.NET

ASP.NET ist nicht wirklich eine Evolution der ASP Version 3.0, stattdessen bietet es eine völlig neues Paradigma für serverseitiges Webscripting. Die folgenden Ausführungen sind nicht darauf bedacht die Details von ASP.NET ausschöpfend zu beschreiben. Viel eher sollen hier die wichtigsten Eigenschaften von ASP.NET betrachtet werden.

ASP.NET unterscheidet sich von ASP in zwei Punkten:

I. Im Hinblick auf die Programmierung bietet ASP.NET gegenüber ASP mehrere Erweiterungen. In diesem Punkt hat Microsoft gute Arbeit geleistet und den Kommentaren und Verbesserungsvorschlägen der ASP Programmierer bei der Entwicklung von ASP.NET Rechnung getragen.

II. ASP.NET bietet nicht nur Erweiterungen, sondern ändert einige Grundstrukturen bei der Programmierung dieser Seiten. Im Klartext heisst das, das es eine völlig neue Erfahrung sein kann, ASP.NET Seiten zu schreiben.

5.1 COMPILER

Eine grundlegende Änderung zu ASP ist, dass ASP.NET Skripts kompiliert werden. Skripte werden nicht mehr interpretiert und der Ausführungsplan wird nicht länger in den Cache geschrieben. Das hat zur Folge das ASP.NET Seiten

erhebliche Performance Steigerungen gegenüber ASP Seiten verbuchen können. ASP.NET werden vom CLR Compliant Compiler[4] kompiliert.

5.2 CACHING

Während kompilierte ASP.NET Seiten einige Verbesserungen bezüglich der Performance bieten, kann der Zugriff auf externe Daten erheblichen negativen Einfluss auf die Peformance einer Webseite haben. Beispielsweise wird eine Webseite, die eine Datenbank Tabelle anzeigen möchte, die meiste Zeit damit verbringen eine Verbindung zur Datenbank aufzubauen und die Informationen zu aquiriren. Im Versuch diesen allzu bekannten Bottleneck zu entschärfen, ist ASP.NET mit einem Data-Cache Modul ausgestattet. Dieses Data-Cache Modul erlaubt den Entwickler zu spezifizieren, welche Daten auf der ASP Seite zu „cachen“ sind und unter welchen Bedingungen der Cache geleert und eine neue Anfrage an den Datenspeicher ausgeführt werden soll.

Zum Beipiel: Wenn man im Vorfeld schon weiss, dass eine bestimmte Datenbank nur selten einen Update erhält, kann man die ASP Seite so gestalten, dass sie die Ergebnisse für 24 Stunden cacht. Dann muss man nicht bei jeder Anfrage der ASP Seite die Datenbank neu lesen. Mit ein wenig Geschick bei der Programmierung kann man es auch so handhaben, dass bei jedem Update der Datenbank der Cache ungültig wird.

5.3 Server Controls

ASP.NET Server Controls erlauben einen deklarativen Programmierstil, der es ermöglicht anspruchsvolle Seiten, die User Input, Uploads etc. enthalten, zu entwickeln. Hingegen werden ASP Seiten sehr prozedural erstellt. Jede Aufgabe wird normalerweise in einer seperaten ASP Seite realisiert und in jeder ASP Seite werden die Aufgaben sequentiell codiert. Zum Beispiel: Will man einem User ermöglichen seinen Namen in eine Textbox einzutragen und es dann auszugeben werden zwei ASP Seiten benötigt, die erste generiert ein *form tag* in der der User seinen Namen eintragen kann, die zweite würde die *form´s action* repräsentieren, die den Eintrag des Users ausgibt. Dieses Vorgehensweise nennt man post-back. Es ist wichtig zu wissen, das lange if statements benötigt werden, um sicherzustellen ob ein post-back stattgefunden hat oder nicht. Mit Server Controls geht diese wesentlich einfacher.

Beispiel für ein Server Control der eine Textbox generiert:

[4] CLR steht für Common Language Runtime und ist eine neue objektorientierte Infrastruktur die Microsoft im Hinblick auf die .NET Strategie entwickelt hat. Dieser .NET CLR Compiler enthält die gesamte Linie der Visual Studio Compilers, d.h VisualBasic, VisualC++, Java, C# und unter anderem auch einen Compiler für PERL Skripts.

```
FORM ACTION="blah.aspx" METHOD=POST RUNAT="server">
  <asp:textbox id="MyTextBox" runat="server"/>
</FORM>
```

Dieses Server Control, erstellt mit *asp:textbox* teil ASP.NET mit, das eine Textbox benötigt wird. Dann wird eine standard textbox in HTML generiert, die folgendermaßen aussieht:

```
<FORM name="ctrl2" method="post" action="intro7.aspx" id="ctrl2">
  <INPUT type="hidden" name="__VIEWSTATE" value="a0z-426043723...">

  <input name="MyTextBox" type="text" id="MyTextBox">
```

Es ist wichtig zu wissen, das die server controls reines plain HTML dem Client zurückgeben. Somit kann man mit jedem Browser ASP.NET Seiten anzeigen lassen.

5.4 Enwicklung

Im Gegensatz zu ASP ist ASP.NET eine noch sehr junge Technologie. ASP.NET wird momentan in der Version 1.0 ausgeliefert und frei verfügbar. Es ist auf folgenden Plattformen lauffähig:

- Microsoft Windows 2000 Professional and Server (SP 2 recommended)
- Microsoft Windows XP Professional
- Microsoft Windows .NET Server

Vorraussetzung ist, dass das .NET Framework auf dem Server installiert ist. Hier soll jedoch nicht näher darauf eingegangen werden.

5.4.1 ASP.NET Web Matrix

Ganz aktuell[5] bietet Microsoft ein WYSIWYG[6]-Tool an, um ASP.NET Seiten zu entwicklen. Es ist kostenfrei und kann bei asp.net heruntergeladen werden. Unter anderem bietet es ein SQL und XML Web Service Development Support an. Ein sehr nützliches Feature ist die Einbindung eines Personal Web Servers. D.h dass man Anwendungen entwickeln und direkt austesten kann, ohne dass man einen Server mit IIS braucht.

[5] Release date 17.06.2002
[6] What you see is what you get

6 LITERATURVERZEICHNIS

- Active Server Pages – Jörg Krause, Addison Wesley
- Skriptsprachen für dynamische Webauftritte – Wolfgang Dehnhardt, Hanser
- www.asp.net
- www.aspextra.de
- www.aspfree.com