JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis

# E-Commerce - Technologiestudie

**DIPLOMARBEIT**

zur Erlangung des akademischen Grades

**DIPLOMINGENIEUR**

in der Studienrichtung

**INFORMATIK**

Angefertigt am Institut für Informationsverarbeitung und Mikroprozessortechnik

Institut für Informationsverarbeitung und Mikroprozessortechnik

Betreuung:

o. Univ. Prof. Dr. Jörg R. Mühlbacher

Von:

**Gernot Ritz**

Linz, Dezember 2002

# Danksagung

Eine erfolgreiche Durchführung einer Studie lässt sich wie ein Weg beschreiben, auf welchem einem verschiedenste Menschen begleiten oder an geeigneter Stelle mit Tat und Rat zur Seite stehen. Besonders dann, wenn eine solche Arbeit parallel zu einer Vollberufstätigkeit durchgeführt wird, bedarf es dieser Helfer, da man sonst die benötigte Energie nur sehr schwer aufbringen könnte.

Daher möchte ich mit meinem Dank bei dem Initiator dieser Arbeit, Herrn o. Univ. Prof. Dr. Jörg R. Mühlbacher, beginnen, welcher mich zum Bearbeiten dieses Themas anspornte und ferner die finanziell abgegoltene Zusammenarbeit mit der Firma Wellness-Soft anbahnte.

Besonderer Dank gebührt auch Frau Dr. Dipl.-Ing. Susanne Reisinger, welche die Idee und die Grundkonzepte für das enthaltene Sport-Portal-Fallbeispiel erarbeitete und mir immer wieder mit ihrer praxisbezogenen und kameradschaftlichen Hilfe unter die Arme griff.

Ferner möchte ich mich auch bei Herrn Dr. Dipl.-Ing. Michael Sonntag und allen weiteren Mitarbeitern des FIM bedanken, die zum Gelingen dieser Arbeit beigetragen haben.

Nicht weniger verbunden bin ich auch meinem Studienkollegen Georg Kotek und den Mitarbeitern der Firma Wellness-Soft, welche Teile der beiden erstellten Fallbeispiele mitentwickelt haben und mir in vielen konstruktiven Gesprächen weiterhalfen.

Bedanken möchte ich mich auch bei allen meinen Freunden und Kollegen, welche ich nicht alle namentlich erwähnen kann, deren unterschiedlichste Hilfestellungen mir jedoch sehr wichtig waren.

Abschließend möchte ich noch einen besonderen Dank meiner Mutter, meinen Brüdern, meiner (verstorbenen) Oma und meiner Frau aussprechen, welche mich während meiner gesamten Studienzeit immer wieder motiviert und tatkräftig unterstützt haben.

DANKE!

# Kurzfassung

**E-Commerce** ist die Reaktion der Wirtschaft des jungen 21. Jahrhunderts auf den Trend zur Kommunikations- und Dienstleistungsgesellschaft. Die ständig wachsende Nutzung elektronischer Medien, im Speziellen des Internets, ruft dabei laufend neue Anbieter und Anwendungen auf den Markt. Zu schon länger bekannten Begriffen wie „Online-Shopping“ und „Internet-Banking“ gesellen sich neue wie „Application-Service-Providing“ (ASP).

Die vorliegende Arbeit beschäftigt sich sowohl theoretisch als auch praktisch (anhand von Fallstudien) mit dem Thema „E-Commerce Technologien“.

Nach einer kurzen Einführung und Begriffsdefinition im ersten Kapitel werden im zweiten die wichtigsten **Basiskonzepte** im Zusammenhang mit dem **Internet** aufgeführt und kurz besprochen. Der Hauptaugenmerk innerhalb beider Abschnitte liegt dabei auf der Behandlung allgemeiner Strukturen und Zusammenhänge, welche zur Entwicklung von ausgereiften E-Commerce-Sites von Bedeutung sind. Dabei wird bewusst nur auf das Wesentlichste eingegangen, um eine gemeinsame Wissensbasis zu schaffen.

Im darauf anschließenden Kapitel werden speziell ausgewählte **Internet-Protokolle** etwas näher betrachtet, wobei besonders auf die in der dritten, vierten und 7ten Schicht des OSI-Referenzmodells platzierten Technologien eingegangen wird.

Das vierte Kapitel beschäftigt sich anschließend mit einigen der bekanntesten **serverseitige**n Programmiersprachen, mit deren Hilfe die Implementierung von komplexen E-Commerce-**Web-Applikationen** leichter durchführbar ist. Dabei werden die Sprachen MS-ASP, ASP.net, Fabasoft-VApps und das aus dem Opensource stammende PHP4 genauer betrachtet.

Abschließend folgen **zwei Fallbeispiele** (Kapitel 5 und 6), welche auch auf der beigelegten CD im kompletten Quellcode zu finden sind. Im ersten Fall handelt es sich um eine Online-Shop-Lösung, im zweiten um ein Vereinssystem für Sportklubs. Beide Implementierungen wurden gemeinsam mit der Diplomarbeit von Herrn Georg Kotek, einem guten Freund und Studienkollegen von mir, durchgeführt.

# Abstract

**E-Commerce** is the reaction of the young 21st century's economy on the trend to a communication- and attendance-society. The constantly growing use of electronic media, specifically of the Internet, calls thereby constantly new offerers and applications to the market. Already longer admitted terms such as "Online-Shopping" and "Internet-Banking" are extended with new ones like "Application-Service-Providing" (ASP).

This theses deals with the theoretical and practical (via case studies) aspects of the topic "E-Commerce Technologies".

After a short introduction and definition part in the first chapter, the most important **basic concepts** regarding the **Internet** will be discussed briefly. The special focus lies on the analysis of general structures, which are important for the development of e-Commerce-Sites. Thereby only the most important concepts will be discussed, in order to create a common knowledgebase.

The next chapter deals with specifically selected **Internet-protocols**. Especially some of the third, fourth and seventh layer of the OSI reference model will be discussed.

In the fourth chapter some of the most common **server-based programming** languages such as MS-ASP (Microsoft Active Server Pages), ASP.net, Fabasoft VApps and PHP4 will be introduced.

Finally, **two case studies** (chapters 5 and 6) will finish this theses. The first solution deals with an online-shop while the second system realizes a platform for sportclubs. Both implementations were co-developed by my colleague Georg Kotek.

# Inhaltsverzeichnis

# 1 E-Commerce (EC)

Electronic-Commerce („elektronischer Handel“) hat sich in den letzten vier Jahren zu einem Modewort für die unterschiedlichsten unternehmerischen Einsatzgebiete entwickelt. Dieses Kapitel soll einen **kurzen Überblick** über die verschiedenen Facetten des E-Commerce geben, welche im Folgenden von Bedeutung sind.

## 1.1 Definition

Einleitend eine kurze Definition, was unter dem Begriff E-Commerce zu verstehen ist:

**„E-Commerce bezeichnet Geschäfte, welche über ein elektronisches Medium teilweise oder ganz abgewickelt werden“**, nach [1]

Da diese Definition sehr weit gefasst ist, werden im weiteren Verlauf Kriterien, mit denen es gelingt elektronische Geschäfte genauer zu klassifizieren, angeführt.

## 1.2 Elektronische Medien

Was versteht man unter „Elektronischen Medien“? Bereits in „old-economy“-Unternehmen[1] wurden seit geraumer Zeit Telefon-Telefax zur Kommunikation und Radio-Fernsehen als Werbeträger verwendet, ohne dass man dabei von E-Commerce gesprochen hat. Erst mit der Integration neuartiger Medien zur Datenkommunikation, insbesondere des **Internet**s, wurde der Begriff des Electronic-Commerce geboren.

## 1.3 Geschäftsformen

An einem elektronischem Geschäft können sowohl Verbraucher, Unternehmen und Behörden involviert sein. Je nachdem, wer als Käufer oder Verkäufer auftritt, unterscheidet man zwischen:

| to | **Consumer** | **Business** | **Government** |
|---|---|---|---|
| **Consumer** | **C-2-C** | **C-2-B** | **C-2-G** |
| **Business** | **B-2-C** | **B-2-B** | **B-2-G** |
| **Government** | **G-2-C** | **G-2-B** | **G-2-G** |

[1] Im Gegensatz zu „new-economy“-Unternehmen, welche mit Informations- und Kommunikations-Technologien ihr Geld erwirtschaften, handeln „old-economy“-Unternehmen vorwiegend mit traditionellen Produkten.

Da derzeit die am häufigsten eingesetzten E-Commerce-Lösungen in den Bereichen B-2-B und B-2-C zu finden sind, wird im Anschluss besonders auf diese beiden Geschäftsformen eingegangen.

### 1.3.1 C-2-C

**Consumer-to-Consumer** - E-Commerce (oder auch Customer-to-Customer) dient dem privaten Handel zwischen einzelnen Endverbrauchern. Hierunter fallen z.B. Internet-Anwendungen, wie etwa Online-**Kleinanzeigenbörsen**, welche das Suchen, das Kaufen, das Verkaufen oder das Tauschen von Gegenständen bzw. Tätigkeiten (z.B.: „Suche Privat-nachhilfe in Spanisch ...“) erleichtern.

### 1.3.2 C-2-B

**Consumer-to-Business** beschäftigt sich mit dem Handel zwischen privaten Verbrauchern und Unternehmen. Typische Beispiele dieser Gattung wären z.B.: Ankaufsportale für Gebrauchtwagen oder **Jobbörsen**.

### 1.3.3 C-2-G

**Consumer-to-Government** (oder auch **C-2-A**dministration) kümmert sich um die Modernisierung der Kommunikation zwischen den Bürgern und den Behörden. Somit ist es machbar, z.B die **Steuerabwicklung** über das Internet durchzuführen.

### 1.3.4 B-2-C

**Business-to-Consumer** gewann besonders durch die breite Nutzung des **World Wide Web** (**WWW** oder **W3**) massiv an Bedeutung. Die Hauptaufgaben liegen im Vertrieb von Waren und Dienstleistungen direkt über das Internet (z.B.: **Online-Shops**), sowie im Anbieten von Online-Serviceleistungen an den Endkunden (z.B.: **Internet-Banking**). Dabei spielt die Akquisition neuer Kunden eine wesentliche Rolle, was allerdings oft mit hohen Marketing-Investitionen (Werbung) verbunden ist. Ferner stellt eine B-2-C - Internet-Repräsentanz auch ein Aushängeschild einer Firma im WWW dar, weshalb die Anforderungen an die Qualität der technischen und grafischen Gestaltung meist sehr hoch angesetzt werden.

Im Gegensatz zum traditionellen Vertrieb ist es dank dem Internet nun auch möglich, die Servicequalität sowohl vor (z.B.: durch den Zugang **24h pro Tag** und die gebotene Aktualität) als auch nach einer Transaktion (z.B.: durch weiteren Produktsupport oder Updatemöglichkeiten) zu steigern und durch individuelle **Personalisierung** und Zusatz-Dienste (z.B. eines Mitgliederforums) die Kundenbindung noch weiter zu verstärken. Auch **Shopping-Portale**, welche einer virtuellen Shopping-City gleichkommen und zum besseren

Preisvergleich genützt werden können, haben sich auf diesem neuen und wesentlich transparenteren Markt etabliert.

### 1.3.5 B-2-B

**Business-to-Business** - E-Commerce dient zur Anbahnung und Durchführung von unternehmensübergreifenden Handelsaktionen, welche weitgehendst über das Medium Internet abgewickelt werden. Das Aufgabenspektrum von B-2-B umfasst sowohl die globale Kommunikation mit Handelspartnern als auch die Abbildung ganzer Geschäftsprozesse, beispielsweise für Beschaffung, Marketing und Vertrieb.

Die erste in den siebziger Jahren von Großunternehmen verwendete Technik im B-2-B war der Austausch von Geschäftsdaten (wie Bestellungen oder Aufträge) über **EDI** (Electronic Data Interchange) und proprietäre VAN-Netzwerke (Value Added Networks). Durch die Entwicklung der Virtual Privat Networks (VPNs) im Internet wurden allerdings die VANs immer unpopulärer. Auch das relativ starre Datenmodell von EDI wird mittlerweile von offeneren Standards wie **XML** (Extensible Markup Language) bzw. ebXML (Electronic Business using XML) [18] immer stärker verdrängt.

Durch die weltweite Durchsetzung des Internets entwickelten sich neue Möglichkeiten Produkt- und Unternehmensinformationen zu transportieren. **Internet-Kataloge** stellten eine wesentlich günstigere und aktuellere Alternative zu konventionellen Mitteln wie Papier, Telefon oder Fax dar.

Als der nächste große Durchbruch innerhalb dieses E-Commerce-Sektors mag die Einführung von **E-Markets** (elektronische Märkte, siehe Abbildung 1) im Jahr 1999 genannt werden. Diese bieten die Möglichkeit, auf virtuellen Handelsplattformen (**E-Portale**) mit mehreren Marktteilnehmern interaktiv zu agieren. Informationsbereitstellung und geschäftliche Transaktionen werden durch dieses Konzept per Internet firmenübergreifend ermöglicht.

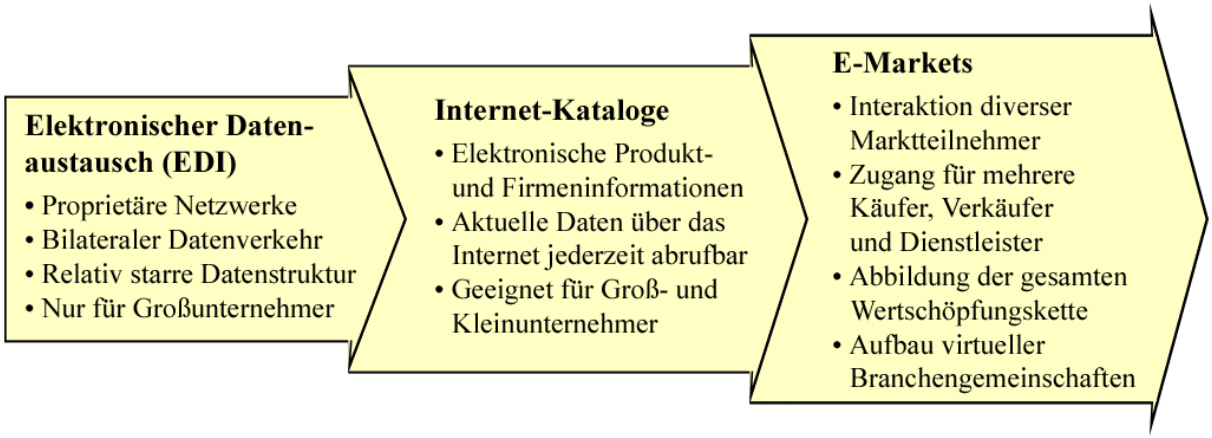



Abbildung 1: Die Entwicklung des B-2-B, nach [4]

Der Ausbau des B-2-B verspricht Kostenvorteile bei den unterstützten Geschäftsbereichen und wird ferner auch zu gewissen Veränderungen innerhalb der traditionellen Wertschöpfungsketten führen, da sich Zulieferer nun allgemein auf einem wesentlich transparenteren Markt bewähren müssen.

### 1.3.6 B-2-G

**Business-to-Government** (oder auch **B-2-A**dministration) beschäftigt sich mit dem modernen Datenaustausch zwischen Firmen und Behörden. So können auch Unternehmen z.B. ihre Steuerabwicklung über das Internet durchführen.

### 1.3.7 G-2-C

**Government-to-Consumer** (oder auch **A**dministration**-2-C**) nennt man z.B. die Möglichkeit, Förderungen, Unterstützungen (z.B. Sozialhilfe) und andere Angebote der Behörden an die Bürger über elektronischen Weg anzufordern.

### 1.3.8 G-2-B

**Government-to-Business** (oder auch **A**dministration**-2-B**) bezeichnet man den Handel zwischen Behörden und Firmen über das Internet. So könnten z.B. Ausschreibungen im WWW durchgeführt werden.

### 1.3.9 G-2-G

**Government-to-Government** (oder auch **A-2-A**dministration) intensiviert die interne Kommunikation zwischen einzelnen Behörden (z.B. Bund und Land) über das Internet.

## 1.4 Weitere Entwicklungen

Ein weiterer, gerade in letzter Zeit hinzugekommener und stark wachsender Bereich im E-Commerce nennt sich **Application-Service-Providing** (**ASP**). Hierbei geht es nicht vorrangig um den konventionellen Verkauf, sondern um das Vermarkten von speziell entwickelten Anwendungen, welche im World-Wide-Web zur Verfügung gestellt werden.

Gängige Beispiele sind Web-Mail, SMS-Dienste, Bonus-Seiten, Elektronische Telefonbücher, Straßenverzeichnisse und vieles mehr. Aber auch derzeit noch untypische Web-Anwendungen wie Textverarbeitung, Tabellenverarbeitung, ... wären als ASP-Versionen vorstellbar!

## 1.5 Fazit

Nachdem nun die verschiedensten Geschäftsformen inkl. diverser möglicher Ausprägungen des E-Commerce angesprochen wurden und somit ein kurzer Überblick über die zahlreichen Möglichkeiten innerhalb dieses neuen Marktes gegeben ist, wendet sich nun das nächste Kapitel den nötigen technischen Grundstrukturen zu, um EC im Internet auch erfolgreich umsetzen zu können.

### 1.5.1 Weiterführende Literatur

Da das Hauptaugenmerk dieser Arbeit auf den technologischen Möglichkeiten zur Umsetzung von EC liegt und sich darum nicht umfassend mit den Strategien für EC auseinandersetzt, wird abschließend noch auf diverse empfehlenswerte EC-Basis-Literatur hingewiesen:

- **Heidi Hudak**, *Diplomarbeit „Einführung von E-Commerce in Klein- und Mittelbetrieben - Methoden, Risken und Chancen“* (2001) [2]
- **Jörg Krause**, *Praxishandbuch Electronic Commerce* (1999) [3]
- **Dirk Schneider**, **Gerd Schnetkamp**, *E-Markets - B2B-Strategien im EC* (2000) [4]

# 2 Internet-Technologie für EC

Dieses Kapitel beschäftigt sich mit einigen Basiskonzepten für EC im Zusammenhang mit dem Internet. Beginnend mit einem im **WWW** typischen Modell zur **Vernetzung von Kunden und Anbietern**, welches in jeweils abgeänderter Form beinahe in jeder EC-Lösung zu finden ist, über die nötigen Einzelkomponenten für ein modernes EC-System, bis hin zur Einengung auf die wesentlichsten Standards und Protokolle für die Kommunikation und den Datenaustausch.

## 2.1 Internet-EC-Modell

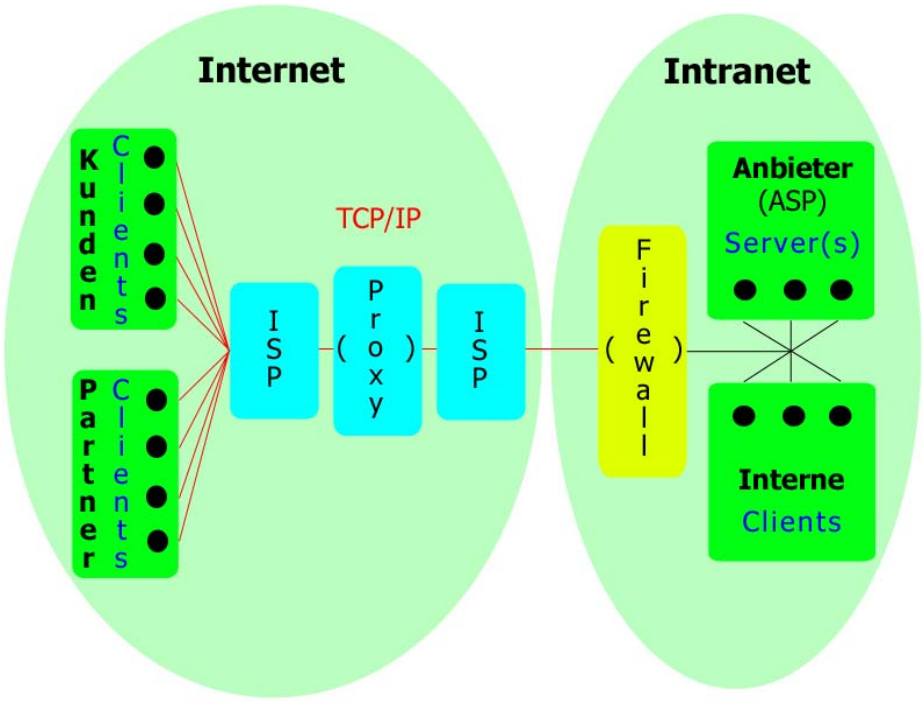



Abbildung 2: Client-Server - EC-Modell

In erster Linie teilt dieses **E-Commerce-Modell** die Verbindung von Kunden (oder auch Partnern) und Anbieter in zwei miteinander verbundene Hauptnetze. Man unterscheidet zwischen dem **Internet**, dem weltumspannenden WAN (wide area network), durch welches modernes E-Commerce erst möglich wurde, und den **firmeneigenen Intranets** der Anbieter.

Damit Kunden, Partner und auch Anbieter aber erst einmal Anschluss an das Internet bekommen, benötigen sie einen frei wählbaren **ISP** (Internet Service Provider). Dieser stellt den Internetzugang zur Verfügung, der technisch beliebig ausgeführt sein kann (herkömmliches Modem, ISDN, ADSL, Kabelmodem, Funkmodem, Strommodem, ATM, usw.). Nur das im Internet benutzte **TCP/IP-**Netzwerkprotokoll (siehe Kapitel 3.2) ist Pflicht.

Nachdem nun aber Internet und Intranet miteinander verbunden sind, sollten unbedingt präventive Sicherheitsmaßnahmen gesetzt werden, um den unberechtigten Zugriff auf interne Resourcen zu verhindern. Eine geeignete Möglichkeit ist die Überwachung, Protokollierung und Einschränkung der Kommunikation durch eine **Firewall**. Dadurch kann z.B. der Datenaustausch anhand von Filterregeln auf bestimmte IP-Adressen und Ports begrenzt werden. Auch die vollständige Überwachung der korrekten Einhaltung einzelner Protokolle (siehe Kapitel 3) kann durch eine intelligente Firewall übernommen werden.

Ferner können im Internet auch sogenannte **Proxy**-Server zum Einsatz kommen. Sie dienen dazu, Daten, welche zwischen den Internet-Clients und den Anbieter-Servern ausgetauscht werden, zu analysieren und zu cachen (zwischenspeichern), um z.B. die Antwortzeiten zu beschleunigen. Für jedes Internet-Service (http, ftp, ...) ist ein eigener Proxy-Server nötig, welcher die einzelnen Anfragen der Clients und die dazugehörigen Antworten der Server auf Zulässigkeit und wiederholtes Auftreten überprüft. So ist es möglich, dass oft besuchte Internetseiten von einem Proxy direkt retourniert werden, ohne den eigentlich dafür zuständigen Server erneut zu kontaktieren.

## 2.2 Clients

Ein Client hat vor allem die Aufgabe, Daten anzuzeigen und Eingaben zurück an den Server zu senden. Im EC haben sich kaum proprietäre Clients (Eigenentwicklungen für spezielle Einsatzgebiete) sondern hauptsächlich **Web-Browser** (Thin-Clients) etabliert.

Web-Browser kommunizieren über das **HTTP**-Protokoll (HyperText Transfer Protocol, siehe Kapitel 3.4) mit den Web-Servern und stellen Daten im **HTML**-Format (HyperText Markup Language) dar. HTML ist ein Standard, welcher vom W3C (WWW-Consortium) genormt wurde (Anfang 1998 in der Version 4.0), allerdings unterstützen nicht alle Web-Browser den vollen Funktionsumfang.

Wird der Browser verwendet für die Kommunikation mit **a)** Kunden, spricht man häufig auch von einer dynamischen **Website** bzw. Frontoffice, oder mit **b)** Partnern sowie firmeninternen Clients, spricht man hingegen oft von einem **Backoffice** (für administrative Zwecke).

Kontakt zum jeweiligen Server wird über einen **URL** (Uniform Resource Locator) hergestellt, welcher bei Websites unter anderem aus einem Domainnamen bzw. bei internen Backoffice-Anwendungen aus einer konkreten IP-Adresse besteht.

**Von einem guten Web-Browser werden zur Zeit folgende Merkmale verlangt:**

- **HTML** 4.0 (+ **DOM** Level 2 für DHTML)

  Alle logischen Elemente einer Web-Seite werden durch eine Abfolge von HTML-Befehlen beschrieben, welche dann vom Web-Browser dargestellt werden können.

Eine der wichtigsten Eigenschaften dabei ist die Möglichkeit, über Verweise zu anderen HTML-Dokumenten eine flexible Vernetzung von Inhalten zu ermöglichen.

Das DOM (Document Object Model) bildet alle HTML-Elemente (Dokument, Bilder, Verweise, Formulare, ...) einer geladenen Web-Seite in einem standardisierten Objektbaum am Web-Client ab, um dann z.B. über eine Skriptsprache darauf dynamisch (zur Zeit der Anzeige → DynamicHTML) zugreifen zu können.

- **CSS** Level 2 (Cascading Style Sheets)

  CSS ist eine unmittelbare HTML-Erweiterung, um zahlreiche Formateigenschaften (Schriftbild, Farben, Rahmen, Positionen, ...) der benutzten HTML-Befehle zentral definieren zu können. So ist es möglich einheitliche und professionelle Web-Designs wesentlich leichter zu erstellen und zu warten.

- **Cookies**

  Cookies sind clientseitig abgelegte Daten, welche von einem Server gesetzt werden können und bei jeder weiteren Kommunikation automatisch mitgesandt werden.

- **Javascript** 1.2 (JScript beim Microsoft Internet Explorer)

  Um „intelligente" HTML-Seiten zu ermöglichen, welche z.B. Formulareingaben validieren bevor sie zum Server gesendet werden, wurde die von der Syntax her an C angelehnte Skriptsprache Javascript entwickelt, welche auf das DOM des Web-Browsers zugreifen kann.

**Optional werden auch noch weitere Browser-Erweiterungen angeboten, wie z.B.:**

- **Applets** (Java 2)

  Java ist eine plattformunabhängige, objektorientierte Programmiersprache, welche von der Firma SUN erstmals 1995 vorgestellt wurde. Fertige Java-Programme, die im Fenster des Web-Browsers ablaufen nachdem sie vom angegebenen Server heruntergeladen wurden, nennt man Applets.

- **ActiveX** (von Microsoft)

  Das Gegenstück von Microsoft zu Java-Applets heißt ActiveX. Diese clientseitig lauffähigen Programme sind aber leider nicht plattformunabhängig und können daher nur auf Microsoft-Systemen eingesetzt werden.

- **Flash** (von Macromedia)

  Die Flash-Technik wurde für grafisch aufwendige Web-Seiten (oder Teile davon) entwickelt und erlaubt die Darstellung von animierten Vektor- und Pixelgrafiken inklusive Ton (sogenannte „Flash-Filme"). Auch interaktive Flash-Elemente sind dank der integrierten Sprache „Action-Script" möglich.

Die zwei gängigsten Browser sind sicherlich der Microsoft Internet Explorer (IE) und der Netscape Navigator (rückläufig). Leider haben beide eigene Anpassungen an den unterstützten Standards getätigt, welche teilweise zueinander inkompatibel sind (z.B. gibt es in beiden DOM-Ausführungen einige abweichende Eigenschaften und Methoden). Auch die jeweils unterschiedlichen Versionen weisen oft ein differentes Verhalten auf. Um wirklich fehlerfreie Web-Seiten zu erhalten, müssen entweder der gemeinsame Teiler der angebotenen Möglichkeiten verwendet oder für jeden einzelnen Browser eigens optimierte Versionen erstellt werden.

Da sich diese Arbeit aber nicht speziell mit dem Thema **Web-Client-Programmierung** auseinandersetzt, sondern sich eher auf die serverseitige Programmierung konzentriert, verweise ich hier auf ein besonders empfehlenswertes Freeware-Dokument, welches auch auf der beigelegten CD enthalten ist:

- **Stefan Münz**, *SelfHTML* (27 Oktober 2001) [5]

## 2.3 Anbieter (ASP)

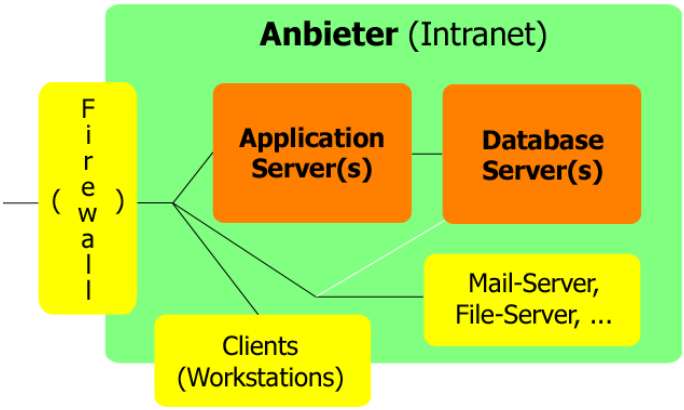



Abbildung 3: Anbieter-Netzwerk-Struktur

**E-Commerce-Anbieter** wie auch **Application-Service-Provider** benötigen im Regelfall ein **firmeneigenes Netzwerk** (Intranet), welches sich dadurch auszeichnet, dass alle Rechner (Server und Workstations) innerhalb dieses Netzes bekannt sind und jeweils gewisse Aufgaben und Rechte besitzen. Die Verbindung des Intranets mit dem Internet wird meist durch eine **Firewall** geschützt, damit nur gewollter Datenverkehr von und nach außen möglich ist.

Dieses Intranet unterscheidet sich von einem herkömmlichen Firmennetz vor allem durch die Existenz von einem **Web-Applikations-Server** (Application-Server) und dem zugehörigen **Datenbank-Server** (Database-Server). Auch mehrere Server der jeweiligen Art sind, bei

Bedarf von mehr Rechenleistung, im Clusterbetrieb[2] denkbar. Will man sich jedoch mit weniger Geschwindigkeit zufrieden geben, könnte man auch die verschiedenen Server auf einem einzelnen physikalischen Rechner installieren.

**Folgende 3 Faktoren sollten in solchen EC-Systemen zueinander abgeglichen werden:**

- **Bandbreite** der Internetanbindung:

  Darunter versteht man die maximal verfügbare Datentransfergeschwindigkeit zwischen dem Internet und den eigenen Rechnern im Intranet, welche vom jeweils gewählten Internet-Service-Provider zur Verfügung gestellt wird.

- **Maximale Clientanzahl** zur gleichen Zeit:

  Dabei handelt es sich um die zu erwartende Maximalanzahl von Benutzern, welche im selben Zeitraum auf die angebotenen Services Zugriff erhalten sollen. Durch Evaluierung dieser Größe können dann auch Rückschlüsse auf die benötigten Speicherressourcen der Server gezogen werden.

- **Antwortgeschwindigkeit** (Response-Time):

  Ein weiterer wichtiger Leistungswert ist die Zeit, in der das EC-Gesamtsystem im Stande ist, eine beliebige Clientanfrage zu bearbeiten und die dazugehörige Antwort zu retournieren. Dabei spielt nicht nur die Rechenleistung der Server eine Rolle, sondern auch die durchschnittliche Länge (Datenmenge) der einzelnen Antworten.

### 2.3.1 Datenbank-Server

Da die Datenbasis in EC-Lösungen eine der wesentlichsten Rollen spielt, wird vor dem Applikation-Server zuerst auf den **DB-Server** eingegangen.

Die wichtigste Frage, welche man sich stellen muss, ist, welches System man einsetzen möchte. Viele verschiedene DB-Applikationen sind am Markt vertreten und warten auf ihren Einsatz (siehe [7]). Damit man die richtige Entscheidung treffen kann, sind folgende grundlegende Punkte zu klären:

- Wie sieht die bestehende **Firmen-Infrastruktur** aus? Welche(s) Betriebssystem(e) soll(en) zum Einsatz kommen: z.B.

  1. MS-Windows NT-Derivat (4.0, 2000, XP, ...)
  2. Unix-Derivat (Linux, BSD, Solaris, HP-UX, AIX, ...)

[2] Man spricht von einem Clusterbetrieb, wenn die eingehenden Anfragen verschiedener Clients durch ein statistisches Verfahren (Load-Balancing - Lastverteilung) auf die einzelnen Server aufgeteilt werden. Dadurch wird die Gesamtverarbeitung parallelisiert und die Antwortzeiten der einzelnen Anfragen verkürzt.

- Wie hoch ist das **Budget**, welches für die DB ausgegeben werden darf? Besitzt die Firma eventuell schon erworbene Lizenzen: für z.B.
  1. MS-SQL Server
  2. Sybase Adaptive Server
  3. SQL Anywhere Studio
  4. Borland InterBase
  5. Ingres II
  6. Oracle
  7. IBM DB2
  8. Informix SQL Server
  9. MySQL (Open-Source)
- Wie komplex, speicher- und rechenintensiv ist die benötigte **Datenstruktur** bzw. das geschätzte **Datenvolumen**:
  1. Stamm-Daten (Lieferanten, Produkte, Preise, ...)
  2. Laufende-Daten (Kunden, Bestellungen, Transaktionen, ...)
  3. Gesuchte-Daten (Abfragen, Statistiken, ...)
- Unterstützt die Datenbank alle **Funktionen**, welche von der angestrebten EC-Lösung benötigt werden (z.B. Locking / Transactions, Backups, ...)?
- Existieren vernünftige **Schnittstellen** zu dem angestrebten Web-Applikations-Server (ODBC ... Open DataBase Connectivity, JDBC ... Java DataBase Connectivity oder z.B. native DB-Treiber)?
- Ist es aus Sicherheitsgründen sinnvoll oder sogar zwingend erforderlich, eine eigene **Firewall** vor den DB-Server zu stellen (Abbildung 4), wodurch zusätzliche Kosten entstehen würden?

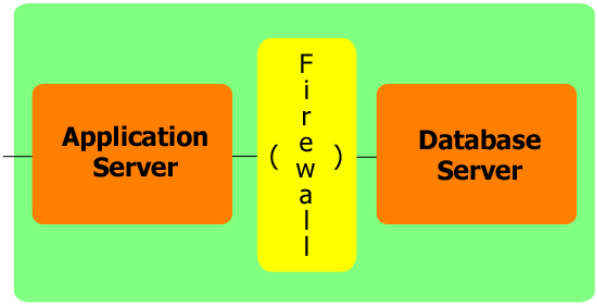



Abbildung 4: Firewall zwischen einzelnen Servern

### 2.3.2 Web-Applikations-Server

Um eine geeignete Aufbereitung der benötigen Daten für die Clients zu ermöglichen (damit sie z.B. für eine Bearbeitung vernünftig dargestellt und anschließend ausgewertet werden können) ist ein Web-Applikations-Server notwendig. Als Basis für einen solchen dient im Allgemeinen ein **Standard-Web-Server**, wie z.B.:

- MS-IIS (Internet Information Server),
- Apache (Open-Source)
- Xitami (Freeware)
- Novell Web-Server
- Netscape Enterprise-Server

Diese Web-Server werden dann mittels einer geeigneten Software zu einem lauffähigen Applikations-Server ausgebaut. Diese spezielle Software wird auch als **Middleware** bezeichnet, da sie meist die Aufgabe des Verbindungsstücks zwischen der DB und dem Web-Server einnimmt.

Die erste Möglichkeit hierfür wurde 1992 durch die Einführung der **CGI**-Schnittstelle (Common Gateway Interface) im NCSA-Web-Server (National Center for Supercomputing Applications) zur Verfügung gestellt. Mit ihrer Hilfe kann man, anstelle der Ausgabe von statischem HTML-Code (deren Inhalt sich nicht selbstständig anpassen kann), die Web-Seiten durch eigens für diesen Zweck entwickelte Middleware-Programme dynamisch am Server generieren lassen. Die verwendete Programmiersprache ist dabei frei wählbar (wie etwa C, C++, Perl, ...). Nur die CGI-Spezifikationen [19] zum Datenaustausch mit dem Web-Server müssen eingehalten werden.

Als Konkurrenz zur CGI-Schnittstelle wurden noch weitere **Server-APIs**, wie z.B. die ISAPI-Schnittstelle (Internet Server Application Programming Interface) für Microsoft-Produkte oder die NSAPI-Schnittstelle (Netscape Server Application Programming Interface), entwickelt, welche speziell auf die Web-Server der jeweiligen Hersteller optimiert wurden und dadurch eine wesentlich höhere Performance bei der Generierung von dynamischen Inhalten erreichen können.

Auch eigene **serverseitige Programmiererweiterungen** wurden geschaffen, wie z.B.:

- Java-Servlets
- ASP
- ASP.net
- PHP
- Python

Auf einige dieser Server-Erweiterungen und Programmiersprachen, welche für die Entwicklung von Middleware-Applikationen besonders geeignet erscheinen, wird im **Kapitel 4** näher eingegangen.

Als kurzes Beispiel wird nun noch eine **typische Kommunikation** eines Web-Clients mit einem Web-Applikations-Server dargestellt:

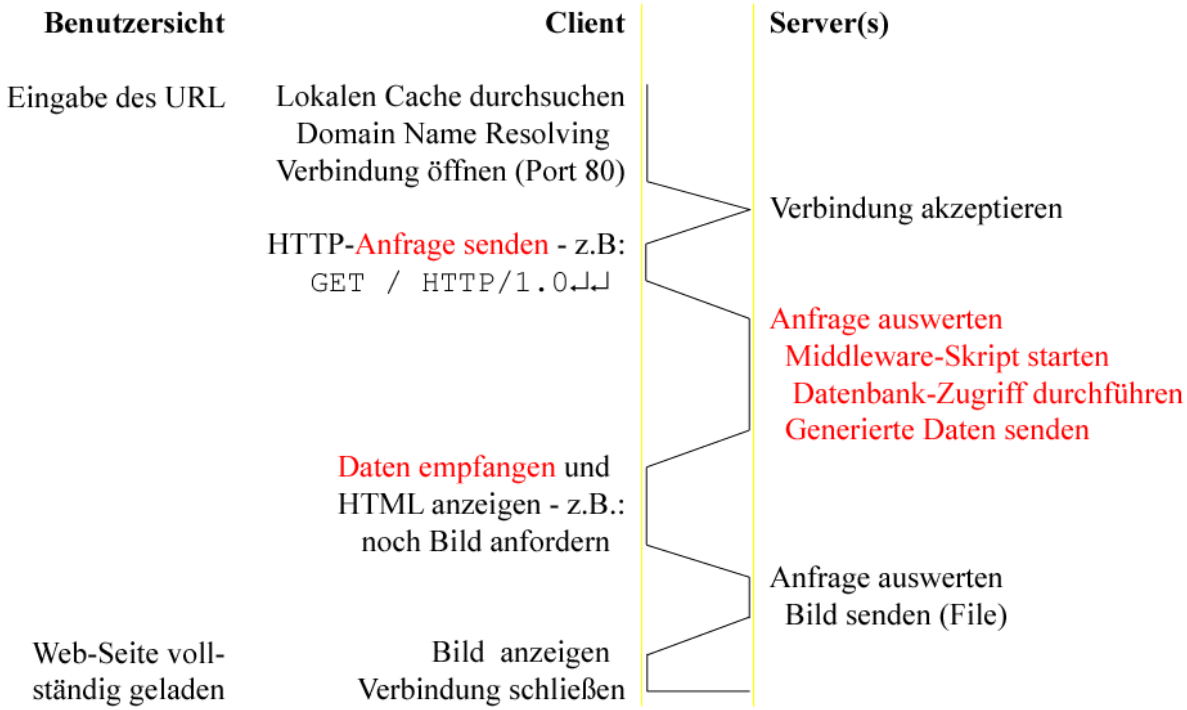



Abbildung 5: Client-Server-Kommunikation, nach [6]

1. Am Beginn gibt der Benutzer den URL der gewünschten EC-Homepage im Adressfeld seines Web-Clients ein.
2. Der Internet-Browser prüft daraufhin, ob er diese Web-Seite durch einen eventuell früheren Aufruf bereits in seinem lokalen Zwischenspeicher abgelegt hat. Wenn dies nicht der Fall ist, wird der Domainname im URL, z.B. durch Nachfrage bei einem DNS-Server (Domain Name Server), in eine konkrete IP-Adresse aufgelöst. Danach kontaktiert er den gefundenen Web-Server.
3. Im Normalfall akzeptiert dieser Server dann die Verbindung mit dem Client.
4. Nun erst kann der Web-Browser die eigentliche Anfrage nach der Homepage senden. Dies geschieht mit Hilfe des HTTP-Protokolls (siehe Kapitel 3.4.1).
5. Der Web-Applikations-Server wertet nun diese Anfrage aus, startet z.B. ein Skript-Programm (Middleware die auch DB-Zugriffe enthalten kann), welches dann die Generierung des HTML-Codes durchführt und sendet anschließend die fertige Web-Seite zurück zum Client.

6. Nachdem nun der Browser die HTML-Seite empfangen hat, wird festgestellt, ob noch weitere Daten (Bilder, Applets, ...) vom Server benötigt werden. Im gegebenen Fall werden diese dann nacheinander angefordert, so lange bis alle nötigen Daten zur vollständigen Darstellung der Homepage am Client eingelangt sind.

### 2.3.3 Live und Staging

Um solche EC-Lösungen (Middleware inklusive DB) möglichst sicher, konsistent und erweiterungsfähig zu gestalten, werden sie meist redundant installiert. Die Anbieter-Netzwerk-Struktur (siehe Abbildung 3) unterscheidet sich dabei im Wesentlichen nur um die Verdoppelung des Web-Applikations-Servers und des Datenbank-Servers (Abbildung 6).

Da gewisse Änderungen (Bugfixes, Updates, grafische Redesigns, ...) nicht sofort am laufenden System (Live) entwickelt werden dürfen, ist für Entwicklungsarbeiten eine zweite EC-Instanz (Staging) sinnvoll.

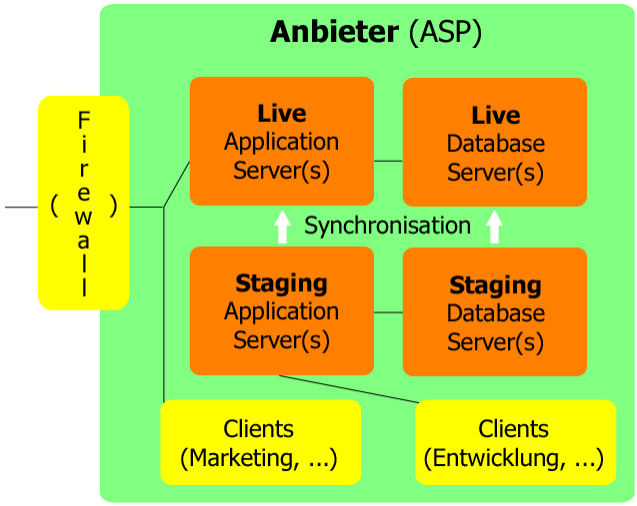



Abbildung 6: Live und Staging

Bei Verwendung eines „Live und Staging“-Konzeptes sollte man allerdings nie vergessen, Mechanismen zum Abgleich der beiden getrennten Systeme vorzusehen (z.B. durch Vergleich von Zeitstempeln), da zu einem gegebenen Zeitpunkt das Live-System schließlich die Änderungen im Staging-System übernehmen sollte!

### 2.3.4 Web-Services

Middleware kann aber nicht nur für die clientseitige Benutzung interessant sein. Es können auch spezielle Server mit Web-Diensten für andere ASP-Server (Abbildung 7) entwickelt werden, welche gewisse Teilprobleme einer Web-Applikation lösen. Solche Web-Services können auch firmenübergreifend anderen Geschäftspartnern angeboten werden, wie z.B. SMS-Gateways, Payment-Services, Werbe-Services, ...

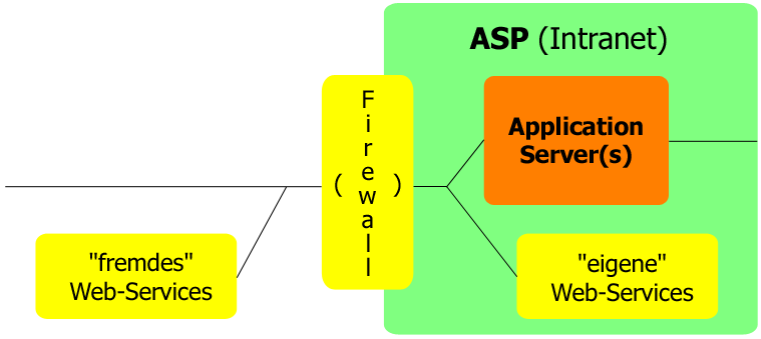



Abbildung 7: Web-Services

Zur Entwicklung und Verwendung solcher Web-Dienste kann auf eine Reihe offener **Standards** zurückgegriffen werden, wie z.B.:

- **XML** (Extensible Markup Language)

  Für Web-Dienste ist XML heute eine der grundlegendsten Basistechnologien, da es als universelles und frei definierbares Format zum Austausch beliebiger Daten besonders geeignet ist.

- **SOAP** (Simple Object Access Protocol)

  SOAP standardisiert ein Datenaustauschformat (basierend auf XML) für Web-Service-Aufrufe. Damit ist unter anderem die Übergabe der gewünschten Service-Methode und der dazugehörigen Ein-/Ausgabeparameter möglich.

- **WSDL** (Web Services Description Language)

  Mit dem WSDL-Format kann man die Schnittstellen der einzelnen Web-Services, welche dann z.B. über SOAP angesprochen werden können, in Form eines XML-Dokuments beschreiben.

- **UDDI** (Universial Description, Discovery and Integration)

  Per UDDI ist es nun möglich nach Web-Services zu suchen und strukturierte Informationen (in XML) über die gefundenen Dienste und deren Anbieter zu erhalten (im einfachsten Fall z.B.: Informationen zum Anbieter selbst und einen URL zur WSDL-Beschreibung).

In welcher Programmiersprache diese Web-Dienste aber dann wirklich realisiert werden, ist völlig unwesentlich und kann vom Anbieter frei gewählt werden.

## 2.4 Fazit

Das am Anfang angeführte Internet-EC-Modell wurde nun Schritt für Schritt durch Zerlegung in seine Einzelkomponenten (Web-Clients, Web-Application-Server, DB-Server, ...) verfeinert, um so einen guten Überblick über die einzelnen Bestandteile (inkl. der dazugehörigen Technologien) für EC-Systeme zu geben.

Im weiteren Verlauf konzentriert sich die Arbeit nun auf die Beschreibung einiger Kommunikationsprotokolle (siehe Kapitel 3) und Middleware-Sprachen (siehe Kapitel 4), welche nötig sind, um serverbasierte Web-Anwendungen (EC-Lösungen) auch vernünftig entwickeln zu können.

### 2.4.1 Weiterführende Literatur

Abschließend noch einige Literaturhinweise, welche im Zusammenhang mit dem Thema dieses Kapitels (Internet-Technologie für EC) von Bedeutung sind:

- **Dr. Rainer Lischka**, *E-Commerce – Techniken für Handel im Internet* (1999) [6]
- **Stefan Münz**, *SelfHTML* (27 Oktober 2001) [5]
- **Meinhardt Schmidt, Thomas Demmig**, *SQL* (2001) [7]
- **Henning Behme, Stefan Mintert**, *XML in der Praxis* (2000) [8]

# 3 Ausgewählte Internet-Protokolle

EC lässt sich nur dann vernünftig realisieren, wenn alle beteiligten Teilsysteme aufeinander abgestimmte Protokolle zum Datenaustausch benutzen. Im Internet haben sich einige für Electronic-Commerce als besonders relevant herausgestellt. Die wichtigsten Kandidaten, welche zum weiteren Verständnis notwendig sind, werden im folgenden Abschnitt kurz besprochen.

## 3.1 Das OSI-Referenzmodell

Da zur Einteilung des Problembereichs der computervermittelten Kommunikation das OSI 7-Schichtenmodell, welches von der International Organization for Standardization (ISO) 1974 entwickelt wurde, sehr oft als Referenz herangezogen wird, folgt auch dieses Kapitel diesem Modell.

Im Wesentlichen werden 7 Ebenen zwischen der Anwendung durch den Endbenutzer und der tatsächlichen physikalischen Netzverbindung gezogen, um einzelne Aufgabenbereiche der Datenvermittlung getrennt behandeln zu können.

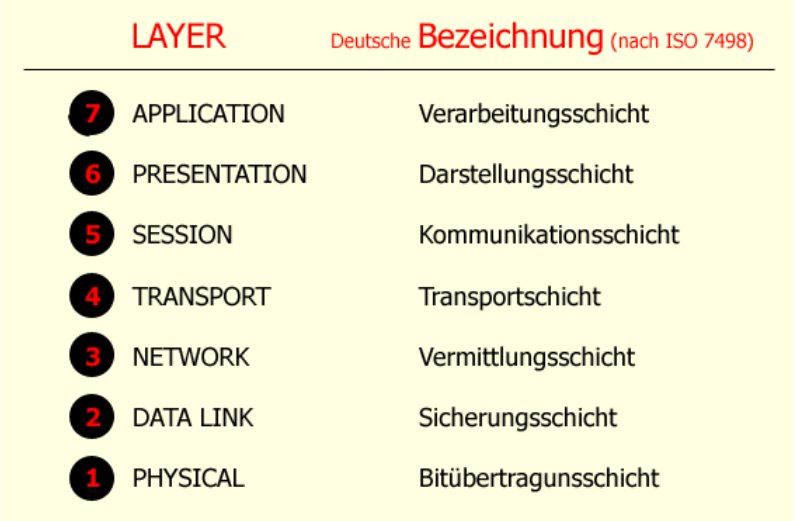



Abbildung 8: Das OSI-Modell, nach [20]

Die Sender zu Empfänger Kommunikation (Endpunkt zu Endpunkt) wird durch die Schichten 4 bis 7 behandelt. Die Layer 1-3 sind für die Kommunikation zwischen physikalisch unmittelbar benachbarten Geräten (Vermittlungspunkt zu Vermittlungspunkt) zuständig.

Das Hauptaugenmerk dieser Arbeit liegt in der dritten, vierten und 7ten Schicht des OSI-Referenzmodells.

## 3.2 EC über TCP/IP (OSI-3,4)

Ende der 60er bzw. Anfang der 70er Jahre entstand bei der US-Behörde DARPA (Department of Defense Advanced Research Project Agency) eine Protokollfamilie, welche speziell für den Einsatz in WANs (Wide-Area-Networks) konzipiert war. Folgende Ziele (nach [3]) sollten bei deren Entwicklung erreicht werden:

- Die geschaffenen Protokolle müssen **allgemeine Gültigkeit** haben, sodass verschiedenste Systeme, unabhängig von Hersteller und Herkunft, miteinander kommunizieren können.
- Das verwendete Betriebssystem bzw. die zugrunde liegende Plattform darf keine maßgebende Rolle spielen (**Interoperabilität**).
- Das Netzwerk soll möglichst einfach und doch umfassend **konfigurierbar** sein.
- TCP/IP muss auch bei möglichen Teilstörungen der Übertragungsleitungen funktionsfähig bleiben (**Robustheit**).

Durch die Fähigkeit, zwischen vielen ungleichen Systemen innerhalb von WANs und LANs (Local-Area-Network) Daten effizient auszutauschen, wurde TCP/IP zu der Basistechnologie für das heutige Internet und somit auch für den Electronic-Commerce im Web.

In den folgenden zwei Unterpunkten werden nun das

**IP .....** Internet Protocol **(OSI Schicht 3)** und das

**TCP ..** Transmission Control Protocol **(OSI Schicht 4)**

(die Komponenten von TCP/IP) getrennt behandelt (nach [11] und [20]).

### 3.2.1 IP (Internet Protocol)

Zu den wesentlichsten Aufgaben von IP gehört das Internet-Adressierungs-Schema, die **Vermittlung** der Daten (zwischen Subnet-Data-Link-Layer und Transportschicht) und das Routing der Datagramme (dem IP-Datenpaketformat).

Da sich das Internet aus vielen einzelnen lokalen Netzwerken (Subnetze) zu einem globalen, weltumspannenden Netz zusammensetzt, können IP-Pakete nur dann direkt an die gewünschte IP-Adresse gesendet werden, wenn sie im selben Subnetz aufzufinden sind. Liegt das adressierte Ziel aber außerhalb, muss das Datagram weitergeleitet werden. Dieser Prozess wird als **Routing** bezeichnet. Dazu wird im lokalen Gateway des Subnetzes nachgesehen (im sogenannten Forwarding-Table des Routers), ob die gesuchte IP-Adresse bekannt ist und falls nicht, wird das Datenpaket an das nächste eingetragene Gateway (next hop) übergeben. Dieser Vorgang wird solange fortgesetzt bis das Datagram direkt an die gewünschte Zieladresse gesendet werden kann.

IP selbst ist ein **verbindungsloses Service** (connectionless service), welches sich nicht darum kümmert, ob die gesendeten Datenpakete auch tatsächlich ankommen. Auch wird keine Garantie abgegeben, in welcher Reihenfolge die gesendeten IP-Pakete beim Empfänger eintreffen. So kann es vorkommen, dass zwei hintereinander abgesendete Datagramme beim Empfänger in zeitlich umgekehrter Folge ankommen.

Jede **IP-Adresse** adressiert immer genau eine einzelne Netzwerk-Schnittstelle (oft auch Host-Adresse genannt). In dem zur Zeit eingesetzten **IPv4** (Version 4) ist eine Adresse stets 32 Bit lang und besitzt folgende Struktur:

| **Klassenkennung** | **Netzadresse** | **Rechneradresse** |
|---|---|---|
| Aufteilung in vier Netzklassen | Subnetz innerhalb dieser Klasse | Der einzelne Host-Rechner |

Dabei bauen sich die IP-Adressen der vier unterschiedlichen Klassen folgendermaßen auf:

| | **Klassenkennung** | **Netzadresse** (Netze) | **Rechneradresse** (Hosts) |
|---|---|---|---|
| Klasse **A** | 1 Bit: `binär [0]` | 7 Bit (max. 128) | 24 Bit (max. 16777216) |
| Klasse **B** | 2 Bit: `binär [10]` | 14 Bit (max. 16384) | 16 Bit (max. 65536) |
| Klasse **C** | 3 Bit: `binär [110]` | 21 Bit (max. 2097152) | 8 Bit (max.256) |
| Klasse **D** | 4 Bit: `binär [1110]` | 28 Bit (Multicast-Adressen für Rechnergruppen) | |

Tabelle 1: IP-Adressräume, nach [20]

Insgesamt ergeben sich somit theoretisch maximal 3758096384 mögliche Host-Adressen[3]. Da aber in der Praxis nicht alle Netze optimal ausgelastet werden können, liegt die reale Maximalanzahl merklich tiefer. Daher wird in Zukunft IPv4 zu wenig Raum für neue Rechneradressen zur Verfügung stellen können. Dieses Problem soll durch die Weiterentwicklung **IPv6** (mit 128 Bit Adressen) gelöst werden.

Heute wird eine IP-Adresse üblicherweise in der **dotted-octet**-Notation (Oktett mit Trennpunkten) angegeben. Dazu werden jeweils 8 Bit der obigen Adressbildung (siehe Tabelle 1) gruppiert und als Dezimalzahl getrennt durch einen Punkt hintereinander-geschrieben, z.B.:

| Binär-Adresse: | `00001010` | `00000000` | `00000000` | `00000001` |
|---|---|---|---|---|
| Dotted-Octed: | 10. | 0. | 0. | 1 |

[3] ohne die Multicast-Adressen mitzuzählen, welche nicht einzelne Netzwerk-Schnittstellen adressieren, sondern für spezielle Broadcasting-Anwendungen eigene Gruppen von Host-Rechnern definieren.

Ferner sind die Adressbereiche 10.x.x.x, 172.16-31.x.x und 192.168.x.x für private-networks reserviert. Diese privaten IP-Adressen sind standardmäßig vom IP-Routing ausgeschlossen. Solche sogenannten privaten LANs (werden auch in vielen Firmen-Intranets eingesetzt) können aber z.B. mittels eines NAT-Gateway-Routers (Network-Address-Translation) wieder an das Internet angebunden werden. Dabei erfolgt bei Anfragen an das Internet eine Umsetzung der privaten Adressen auf eine öffentliche IP-Adresse (die dazugehörigen Antworten werden über eine im Gateway dynamisch hinterlegte Tabelle wieder an den jeweiligen privaten Host-Rechner retourniert).

### 3.2.2 TCP (Transmission Control Protocol)

Um auch **verbindungsorientierte** Dienste (connection-oriented services) im Internet zu realisieren, wird TCP für den Aufbau von logischen Verbindungen, auch Ende-zu-Ende-Kontrolle (end-to-end) genannt, zwischen kooperierenden Computern verwendet.

Die allgemeine Aufgabe besteht im Aufbau, in der Verwaltung und dem Abbau dieser logischen Leitungen. TCP garantiert dabei die richtige Reihenfolge und den korrekten **Transport** aller gesendeten Datenpakete.

Jede Anwendung, welche eine solche **TCP/IP**-Verbindung benutzt, wird dabei durch eine sogenannte **Port**-Nummer identifiziert. Diese kann entweder dynamisch vom Client bestimmt oder statisch mit einem vordefinierten Service assoziiert werden. Die Kombination einer IP-Adresse mit einer Port-Nummer nennt man dann **Socket**. Um nun eine logische Verbindung eindeutig identifizieren zu können, benötigt man ein **Socket-Paar**, z.B.:

1. <Server IP-Adresse> : <feste Service-Port-Nummer> ( Bsp.: 10.0.0.1:23 )
2. <Client IP-Adresse> : <zugeteilte Port-Nummer> ( Bsp.: 10.0.0.2:3044 )

Standardmäßig sind alle Port-Nummern unterhalb von 1024 privilegiert. Einige der genormten Internet-Services und deren TCP/IP-Port, welcher dann nicht explizit angegeben werden muss, sind in der folgenden Tabelle aufgelistet:

| Port-Nummer | Service |
|---|---|
| 21 | FTP |
| 25 | SMTP |
| 80 | HTTP |
| 110 | POP3 |
| 118 | SQL Services |
| 220 | IMAP3 |

Tabelle 2: TCP/IP-Ports, nach [20]

## 3.3 Email: SMTP und POP3 (OSI-7)

Bekanntermaßen war die Möglichkeit Emails auszutauschen eine der ersten Internet-Technologien, welche dem Electronic-Commerce zu starkem Wachstum verhalf.

Viele Arten des Briefverkehrs einer Firma können per elektronischer Post wesentlich schneller durchgeführt werden, wodurch sich der Zeitraum zwischen gewissen Anfragen und ihrer Beantwortung erheblich reduzieren lässt.

### 3.3.1 SMTP (Simple Mail Transfer Protocol)

Üblicherweise wird zum Versenden und zum Weiterleiten von Emails das SMTP-Protokoll (Port 25) benutzt. Sendet ein Client eine Nachricht zum SMTP-Server, übermittelt dieser die Email zum Mail-Server der angegebenen Empfänger-Adresse. Zwischen diesen beiden Servern können aber auch noch Mail-Relays bzw. Mail-Gateways liegen, welche die abgesendete Post weiterleiten müssen (store and forward).

Eine Kommunikation eines Clients mit einem SMTP-Server könnte z.B. folgendermaßen aussehen (nach [21], siehe **RFC821** und RFC822):

| | |
|---|---|
| ⇒ Nachdem der Client eine TCP/IP-Verbindung über den Port 25 mit dem Server aufgebaut hat, meldet sich dieser: | |
| • Server: | `220 smtp.mail.at ESMTP service (Netscape Messaging Server 4.15 Patch 3 (built Sep 28 2000))` |
| ⇒ Der Client meldet sich daraufhin mit seinem Domain-Namen an, was vom Server bestätigt wird: | |
| • Client: | `HELO domain.at` |
| • Server: | `250 smtp.mail.at` |
| ⇒ Nun wird beginnend mit dem Empfänger über den Absender bis hin zum eigentlichen Mail-Text alle benötigten Daten zum Versand der Nachricht dem Server übergeben, welcher dabei jede „Datenzeile“ wieder einzeln bestätigt: | |
| • Client: | `Mail From:<gernot.ritz@mail.at>` |
| • Server: | `250 Sender <gernot.ritz@mail.at> Ok` |
| • Client: | `RCPT To:<ritz@mail.at>` |
| • Server: | `250 Recipient <ritz@mail.at> Ok` |
| • Client: | `data` |
| • Server: | `354 Ok Send data ending with <CRLF>.<CRLF>` |

| | |
|---|---|
| • Client: | `From: Gernot Ritz <gernot.ritz@mail.at>`<br>`To: <ritz@mail.at>`<br>`Date: Fri, 22 Mar 2002 12:00:00 +0100`<br>`Return-Path: <gernot.ritz@mail.at>`<br>`Subject: Ein kurzes eMail`<br><br>`Der Inhalt dieses eMails!`<br>`.` |
| • Server: | `250 Message received: GTE0V400.7GI` |
| ⇒ Abschließend meldet sich der Client noch vom SMTP-Server ab: | |
| • Client: | `quit` |
| • Server: | `221 smtp.mail.at ESMTP server closing connection` |

Um nicht nur einfache Textnachrichten versenden zu können, wurde das MIME-Format (Multipurpose Internet Mail Extensions) eingeführt. Hiermit ist es möglich, innerhalb einer Nachricht mehrere Teilbereiche mit unterschiedlichen Inhalten zu definieren (multi-part messages). Dadurch können z.B. auch sogenannte HTML-Emails und Anhänge (attachments) übertragen werden.

### 3.3.2 POP3 (Post Office Protocol)

Die zur Zeit am häufigsten zu findenden Email-Server richten sich für die Abholung der Nachrichten durch den Mail-Client nach dem POP3-Protokoll (Port 110). Der POP3-Server selbst empfängt dabei alle Emails per SMTP und speichert sie für das spätere Abholen durch den Benutzer zwischen.

Das Übertragen der Post vom Server zum Client könnte dann z.B. so aussehen (nach [21], siehe **RFC1939**):

| | |
|---|---|
| ⇒ Nachdem der Client eine TCP/IP-Verbindung über den Port 110 mit dem Server aufgebaut hat, meldet sich dieser: | |
| • Server: | `+OK Netscape Messaging Multiplexor ready` |
| ⇒ Nun identifiziert sich der Client mittels Benutzernamen und Kennwort beim POP3-Server, um Zugriff auf das gewünschte Mail-Konto zu erhalten: | |
| • Client: | `USER gr1971` |
| • Server: | `+OK password required for user gr1971` |
| • Client: | `PASS secret` |
| • Server: | `+OK Maildrop ready` |

| | |
|---|---|
| ⇒ Ab diesem Zeitpunkt kann der Client verschiedene Befehle beim Server ausführen. Hier wird als erstes der Status des Mail-Accounts (eine empfangene Nachricht und insgesamt 768 Bytes Speicherbedarf) abgefragt: | |
| • Client: | `STAT` |
| • Server: | `+OK 1 768` |
| ⇒ Der zweite Befehl bewirkt das Auflisten der Nachrichtenlängen der einzelnen empfangenen Mails: | |
| • Client: | `LIST` |
| • Server: | `+OK scan listing follows`<br>`1 768`<br>`.` |
| ⇒ Auch der benötigte Speicherplatz jedes einzelnen Mails kann abgefragt werden: | |
| • Client: | `LIST 1` |
| • Server: | `+OK 1 768` |
| • Client: | `LIST 2` |
| • Server: | `-ERR No such message` |
| ⇒ Zum Abfragen einer Nachricht (Länge, Mail-Header und Inhalt) kann der nun folgende Befehl verwendet werden: | |
| • Client: | `RETR 1` |
| • Server: | `+OK 768 octets`<br>`Return-Path: <gernot.ritz@mail.at>`<br>`Received: from smtp.mail.at ([194.24.128.102]) by`<br>`webmail.mail.at`<br>`(Netscape Messaging Server 4.15) with ESMTP id`<br>`GTE0ZP00.4SX for <ritz@mail.at>;`<br>`Fri, 22 Mar 2002 19:35:49 +0100`<br>`Received: from domain.at ([212.241.68.236]) by`<br>`smtp.mail.at`<br>`(Netscape Messaging Server 4.15) with SMTP id`<br>`GTE0V400.7GI for <ritz@mail.at>;`<br>`Fri, 22 Mar 2002 19:33:04 +0100`<br>`From: "Gernot Ritz" <gernot.ritz@mail.at>`<br>`To: <ritz@mail.at>`<br>`Date: Fri, 22 Mar 2002 12:00:00 +0100`<br>`Return-Path: <gernot.ritz@mail.at>`<br>`Subject: Ein kurzes eMail`<br>`Date: Fri, 22 Mar 2002 19:35:48 +0100`<br>`Message-ID: <GTE0V400.7GI@smtp.mail.at>`<br><br>`Der Inhalt dieses eMails!`<br>`.` |

| | |
|---|---|
| ⇒ Nachdem nun das Mail abgefragt wurde, kann es bei Bedarf vom POP3-Server gelöscht werden: | |
| • Client: | `dele 1` |
| • Server: | `+OK message deleted` |
| ⇒ Am Ende der Sitzung meldet sich der Client wieder vom Server ab: | |
| • Client: | `Quit` |
| • Server: | `+OK` |

## 3.4 Web: HTTP (OSI-7)

Für den modernen EC sind Emails alleine allerdings schon lange nicht mehr ausreichend. Heutzutage ist die Kommunikation zwischen Thin-Clients bzw. Web-Browsers und den EC-Servern der Firmen eine der wichtigsten Angelegenheiten. Dieser Datenaustausch über das Web wird durch das HTTP-Protokoll beherrscht.

### 3.4.1 HTTP (HyperText Transfer Protocol)

Um z.B. ein HTML-Dokument, welches auf einem Web-Server abgelegt ist, durch einen Web-Client abfragen zu können, benötigt man ein „**Anforderungs-Antwort**“-Protokoll. Das HTTP-Protokoll (Port 80) stellt zu diesem Zweck für jede Dokumentanforderung (**request**) genau eine Verbindung über TCP her. Nach der Übermittlung exakt dieses angeforderten Dokuments (**response**) wird die Verbindung dann automatisch wieder geschlossen.

HTTP existiert zur Zeit in zwei Evolutionsstufen, welche in folgenden RFCs (siehe [21]) beschrieben sind:

- **HTTP/1.0 - RFC1945** und **HTTP/1.1 - RFC2068**

#### HTTP-Request

Eine einfache Zeichenkette, welche sich an folgende Syntax hält, stellt einen korrekten HTTP-Request eines Clients an einen Web-Server dar (nach [20]):

```
[HTTP-Method] [Identifier] [HTTP-Version]
[optional: Request-Header(s)]

[optional: Request-Data]
```

**Die Version 1.1 kennt folgende [HTTP-Method]-Anforderungen:**

| | |
|---|---|
| **GET** | Anforderung des durch den [Identifier] referenzierten Objektes. |
| HEAD | Fordert nur den HEAD des [Identifier] ref. Objektes an. |
| OPTIONS | Fordert Informationen über den Server oder über Objekte an. |
| **POST** | Übermittlung von [Request-Data] an das durch den [Identifier] adressierten Objektes. Wird bei **HTML-FORMs** mit METHOD=POST angewandt, um Formulareingaben zurück an den Application-Server zu senden. |
| PUT | Lädt ein File auf den Server hoch (UPLOAD). |
| DELETE | Löscht ein Objekt (normalerweise deaktiviert). |
| TRACE | Diagnostik-Methode |

Tabelle 3: HTTP-Methods

**Als optionale [Request-Headers] sind u.a. folgende erlaubt:**

| Request-Header | Beispiel |
|---|---|
| Accept: [MIME-type]/[MIME-subtype] | Accept: image/gif, image/jpeg, */* |
| Accept-Charset: [Zeichensatz] | Accept-Charset: iso-8859-1,*,utf-8 |
| Accept-Language: [Sprache] | Accept-Language: en, de |
| Authorization: [Schema] [Daten] | Authorization: user username:pwd |
| **Content-length:** [Länge (in Bytes) von [Request-Data] bei Post & Put] | Content-length: 24 |
| **Content-type:** [MIME-type]/ [MIME-subtype] | Content-Type: application/ x-www-form-urlencoded |
| **Cookie:** [Cookie-Data] | Cookie: userID=1234 |
| Date: [Datum & Zeit] | Date: Sunday, 17-Mar-02 00:00:00 GMT |
| If-Modified-Since: [Datum & Zeit] | If-Modified-Since: Sun, 17 Mar 2002 … |
| Referer: [URL des Aufrufes] | Referer: http://www.fim.uni-linz.ac.at |
| User-Agent: [Browser] | User-Agent: Mozilla/3.0 |

Tabelle 4: HTTP-Request-Headers

Nachdem nun der Web-Server einen solchen HTTP-Request (siehe auch 3.4.2) eines beliebigen Web-Clients empfangen hat, antwortet er nach einem ebenfalls im HTTP-Protokoll genau definierten Schema.

### HTTP-Response

Die Syntax der Server-Anwort an den Client sieht nun folgendermaßen aus (nach [20]):

```
[HTTP-Version] [Status-Code] [Reason-String]
[optional: Response-Header(s)]

[Response-Data]
```

**Einige mögliche `[Status-Codes]` und `[Reason-Strings]` einer HTTP-Antwort:**

| **200** | **Ok** |
|---|---|
| 302 | Found |
| 400 | Bad Request |
| 401 | Unauthorized |
| 403 | Forbidden |
| 404 | Not found |

Tabelle 5: HTTP-Status-Codes

**Als `[Response-Headers]` sind u.a. folgende wesentlich:**

| Response-Header | Bespiele |
|---|---|
| **Cache-control:** [Request/Response-Cache-Direktiven] | Cache-Control: no-cache, must-revalidate |
| **Content-length:** [Länge (in Bytes) von [Response-Data]] | Content-Length: 100 |
| **Content-type:** [MIME-type]/[MIME-subtype] | Content-Type: text/html<br>Content-Type: application/x-pdf |
| Expires: [Datum und Zeit] | Expires: Mon, 26 Jul 1997 05:00:00 … |
| Last-modified: [Datum und Zeit] | Last-Modified: Sun, 17 Mar 2002 … |
| **Location:** [Umlenkung auf diesen URL] | Location: http://www.redirect.at |
| Pragma: [spezielle Direktiven] | Pragma: no-cache |
| Server: [Web-Server] | Server: Xitami |
| **Set-Cookie:** [Cookie-Definition] | Set-Cookie: userID=1234; path=/; domain=www.cookie.at expires=Sun, 17-Mar-02 00:00:00 GMT |
| WWW-Authenticate: [Schema] realm="[Bereich]" | WWW-Authenticate: Basic realm="Backoffice" |

Tabelle 6: HTTP-Response-Headers

### 3.4.2 HTTP-Beispiele

Zum besseren Verständnis nun noch ein paar kurze Beispiele, wie z.B. die Kommunikation zwischen Client und Server mittels HTTP aussehen kann:

**A)** Eine **statisch** am Web-Server **abgelegte** HTML-**Seite** wird abgerufen:

⇒ Der Web-Client sendet einen HTTP-GET-Request mit einer Anfrage nach der Web-Seite „static.html“ zum Web-Server:

- Client:

```
GET /static.html HTTP/1.0 ↵↵
```

⇒ Daraufhin antwortet der Web-Server mit folgendem HTTP-Response, welcher unter anderem die dort **abgelegte HTML-Seite** (mit der Länge von 129 Bytes) beinhaltet:

- Server:

```
HTTP/1.0 200 Ok
Server: Xitami
Content-Type: text/html
Content-Length: 129
Last-Modified: Fri, 29 Mar 2002 13:44:44 GMT

<html>
 <head>
  <title>Static Page</title>
 </head>
 <body>
  <H1>Dies ist eine statische Web-Seite</H1>
 </body>
</html>
```

**B)** Eine **dynamisch generierte Seite** wird von einem Application-Server abgerufen:

⇒ Der Web-Client sendet wieder einen HTTP-GET-Request, diesmal mit einer Anfrage nach der Web-Seite „dynamic.php“, zum Web-Server:

- Client:

```
GET /dynamic.php HTTP/1.0 ↵↵
```

⇒ Nun antwortet der Server mit folgendem HTTP-Response, welcher die gerade mittels PHP (siehe Kapitel 4.6) **berechnete HTML-Seite** (mit der Länge von 203 Bytes) beinhaltet. Ferner wird dem Client dabei mitgeteilt, dass er diese Seite nicht zwischenspeichern darf (durch die Angabe von Pragma, Cache-Control und eines abgelaufenen Expires). Dies stellt sicher, dass der Web-Client immer wieder die aktuelle Version dieser Seite vom Server anfordert.

- Server:

```
HTTP/1.0 200 Ok
Content-Length: 203
Content-Type: text/html
Pragma: no-cache
Cache-Control: no-cache, must-revalidate
Last-Modified: Mon, 01 Apr 2002 14:33:00 GMT
Expires: Mon, 26 Jul 1997 05:00:00 GMT
X-Powered-By: PHP/4.1.1
Server: Xitami
```

```
<html>
 <head>
  <title>Dynamic Page</title>
 </head>
 <body>
  <H1>Dies ist eine dynamische Web-Seite</H1>
  <p>Ihr Name ist ???.</p>
  <a href="?form=name">Namen angeben</a>
 </body>
</html>
```

- Der Benutzer klickt auf den „Namen angeben“-Link auf der zuvor geladenen Seite:

⇒ Daraufhin wird ein HTTP-GET-Request abgesetzt, diesmal die Anfrage nach der Seite „dynamic.php?form=name“:

- Client:

```
GET /dynamic.php?form=name HTTP/1.0 ↵↵
```

⇒ Der Server übergibt nun dem bereits einmal aufgerufenen PHP-Skript „dynamic.php“ den GET-Parameter (siehe 3.4.3) „form“ mit dem Wert „name“, woraufhin ein neuer HTTP-Response (mit einem HTML-FORM zur Eingabe des Benutzernamens) zum Web-Client gesendet wird:

- Server:

```
HTTP/1.0 200 Ok
Content-Length: 278
Content-Type: text/html
Pragma: no-cache
Cache-Control: no-cache, must-revalidate
Last-Modified: Mon, 01 Apr 2002 14:33:03 GMT
Expires: Mon, 26 Jul 1997 05:00:00 GMT
X-Powered-By: PHP/4.1.1
Server: Xitami

<html>
 <head>
  <title>Dynamic Page</title>
 </head>
 <body>
  <H1>Dies ist eine dynamische Web-Seite</H1>
  <form method="post" action="?">
   <p>Ihr Name <input type="text" name="name">
   <input type="submit" value="Speichern"></p>
  </form>
  <a href="?">Zur&uuml;ck</a>
 </body>
</html>
```

- Nun gibt der Benutzer z.B. den Namen „Gernot“ in das HTML-Formular ein und sendet es ab. Der Server antwortet daraufhin mit dem Setzen eines **Cookie**s und fordert zusätzlich eine URL-Umlenkung (**Redirect**) an. Der Web-Client speichert nun das empfangene Cookie lokal ab und setzt selbstständig einen neuen GET-Request auf den Redirect-URL ab. Dabei muss noch bemerkt werden, dass ab diesem Zeitpunkt die zuvor gesetzten

Cookie-Daten automatisch bei allen HTTP-Anforderungen an diesen Server wieder mitinkludiert werden:

⇒ Nun folgt der HTTP-POST-Request des Clients, welcher durch das Absenden des HTML-FORMs an den Server geschickt wird. Der eingegebene Benutzername wird dabei als Content (Request-Data) mitübergeben[4]:

- Client:

```
POST /dynamic.php? HTTP/1.0 ↵
Content-type: application/x-www-form-urlencoded ↵
Content-length: 11 ↵↵
name=Gernot
```

⇒ Der Server antwortet mit einem HTTP-Response, welcher den Benutzernamen wieder als Cookie zum Client retourniert (Set-Cookie) und einen URL-Redirect (Location), in diesem Fall auf sich selbst (zum Refresh der Web-Seite), anfordert:

- Server:

```
HTTP/1.0 302 Found
Content-Length: 0
Content-Type: text/html
Set-Cookie: user=Gernot
Pragma: no-cache
Cache-Control: no-cache, must-revalidate
Last-Modified: Mon, 01 Apr 2002 14:33:09 GMT
Expires: Mon, 26 Jul 1997 05:00:00 GMT
X-Powered-By: PHP/4.1.1
Server: Xitami
Location: http://www.webserver.at/dynamic.php
```

⇒ Nachdem der Web-Client nun das gerade empfangene Cookie in seinen lokalen Speicher abgelegt hat, führt er (ohne direktes Mitwirken des Benutzers) den vom Server (durch Angabe eines neuen Location-URLs) angeforderten HTTP-Request durch. Diese neue Anforderung beherbergt, so wie alle zukünftigen, bereits die zuvor gespeicherten Cookie-Daten:

- Client

```
GET /dynamic.php HTTP/1.0 ↵
Cookie: user=Gernot ↵↵
```

⇒ Abschließend sendet der Server noch die aktuelle Web-Seite mit dem im Cookie übergebenen Namen des Benutzers per HTTP-Response zurück zum Web-Client:

- Server

```
HTTP/1.0 200 Ok
Content-Length: 206
Content-Type: text/html
Pragma: no-cache
Cache-Control: no-cache, must-revalidate
Last-Modified: Mon, 01 Apr 2002 14:33:10 GMT
Expires: Mon, 26 Jul 1997 05:00:00 GMT
X-Powered-By: PHP/4.1.1
Server: Xitami
```

[4] Würde etwa ein zweites Formularfeld mit der Bezeichnung "password" existieren, müsste die letzte Zeile des HTTP-Requests z.B. folgendermaßen aussehen: `name=Gernot&password=Kennwort`

```
<html>
 <head>
  <title>Dynamic Page</title>
 </head>
 <body>
  <H1>Dies ist eine dynamische Web-Seite</H1>
  <p>Ihr Name ist Gernot.</p>
  <a href="?form=name">Namen angeben</a>
 </body>
</html>
```

### 3.4.3 URL (Uniform Resource Locator)

Um im Internet auf jede einzelne Internetsource zugreifen zu können, wird eine Art Signatur benötigt, welche den Standort und das Zugangsprotokoll exakt definiert. Diese Aufgabe übernimmt der URL.

Der Aufbau eines URL sieht folgendermaßen aus (nach [20]):

```
[Zugangsprotokoll]://[Hostadresse][optional: :Port]/[Pfad]
```

Mögliche Zugangsprotokolle sind z.B.:

- http
- https ..... http over Secure Socket Layer (SSL)
- ftp ..... File Transfer Protocol
- telnet
- file ..... Dateien auf lokalem Rechner
- mailto ..... mailto:Email-Adresse
- news ..... news:UseNet-Gruppenname

Bei der Hostadresse handelt es sich entweder um einen Domain-Namen oder direkt um eine IP-Adresse. Eine Verbindung zum Server wird dann standardmäßig über die Portnummer des angegebenen Zugangsprotokolls aufgebaut, außer der optionale Port-Parameter überschreibt den Ursprungswert.

Der Pfad des URLs enthält meist nicht nur den vollen Ziel-Pfad des angeforderten Objektes, sondern auch einen eventuellen Filenamen (andernfalls wird ein Standard-Filename wie z.B. default.html, default.*, index.html oder index.* verwendet). Zusätzlich kann der URL-Pfad auch weitere Parameter, welche durch ein Fragezeichen getrennt an den Filenamen angehängt werden können, zum Server transportieren, z.B.:

```
URL?parameter1=wert&parameter2=wert&...
```

Wichtig ist auch die Tatsache, dass innerhalb eines URL-Pfades nicht alle US-ASCII-Zeichen zulässig sind. Es existieren gewisse reservierte Zeichen, welche im Bedarfsfall speziell kodiert (**urlencoded**) werden müssen. Die wichtigsten Zeichen sind in der folgenden Übersetzungstabelle enthalten:

| Zeichen | urlencoded |
|---|---|
| Leerraum | %20 |
| / | %2F |
| \ | %5C |
| & | %26 |
| @ | %40 |
| % | %25 |

Tabelle 7: urlencoded-Characters

## 3.5 Fazit

Aufbauend auf dem OSI-7-Schichtenmodell wurden in diesem Kapitel die für EC unbedingt nötigen Protokolle zur Kommunikation zwischen Clients und Servern (der EC-Anbieter) besprochen. Besonderes Augenmerk wurde dabei auf das HTTP-Protokoll gelegt, da dies für Entwickler von Internet-Applikationen von außerordentlich großer Bedeutung ist.

Ausgehend von diesem Wissen beschäftigt sich das nächste Kapitel nun mit einigen Middleware-Sprachen und den nötigen Konzepten zum erfolgreichen Erstellen solcher Internet-Anwendungen.

### 3.5.1 Weiterführende Literatur

Abschließend sei noch eine Empfehlung eines Buches angeführt, welches schlicht als das Standardwerk über Netzwerktechnologien zählt:

- **Andrew S. Tanenbaum**, *Computer-Netzwerke* (1992) [10]

# 4 Serverbasierte Web-Anwendungen

Erst durch **dynamisch generierte Web-Seiten** ist EC mittels Thin-Clients sinnvoll möglich. Dazu werden alle benötigten Daten vom **Application-Server** durch eine eigene **Middleware (OSI Schicht 7)** für die Clients geeignet aufbereitet und die zurückgesendeten Eingaben anschließend ausgewertet. Da die Kommunikation im Web-Bereich aber mittels des bereits besprochenen **HTTP**-Protokolls ohne direkten Zusammenhalt der einzelnen erzeugten Seiten erfolgt, muss die Logik der Middleware die jeweils für sich abgeschlossenen Anfragen des Clients an den Server zu einer gesamten Anwendung (**Web-Application**) zusammenführen.

## 4.1 Web-Application-Basics

Im Allgemeinen gibt es drei verbreitete Techniken eine Web-Anwendung geschlossen zu halten, um z.B. die weitere Ausgabe aller oder einzelner Web-Seiten durch frühere getätigte Aktionen bzw. Eingaben der einzelnen Anwender zu beeinflussen:

a) **Session-ID** im **URL** weiterreichen

b) **Session-ID** in einem versteckten **Form-Feld** durchreichen

c) **Session-ID** in einem **Cookie** ablegen

In allen Fällen (a und b können auch gemischt werden) wird bei der ersten Anfrage eines Clients eine eindeutige Kennung (**unique Session-ID**) am Server generiert und ab diesem Zeitpunkt als Client-Kennung jedes Mal mitübertragen, z.B.:

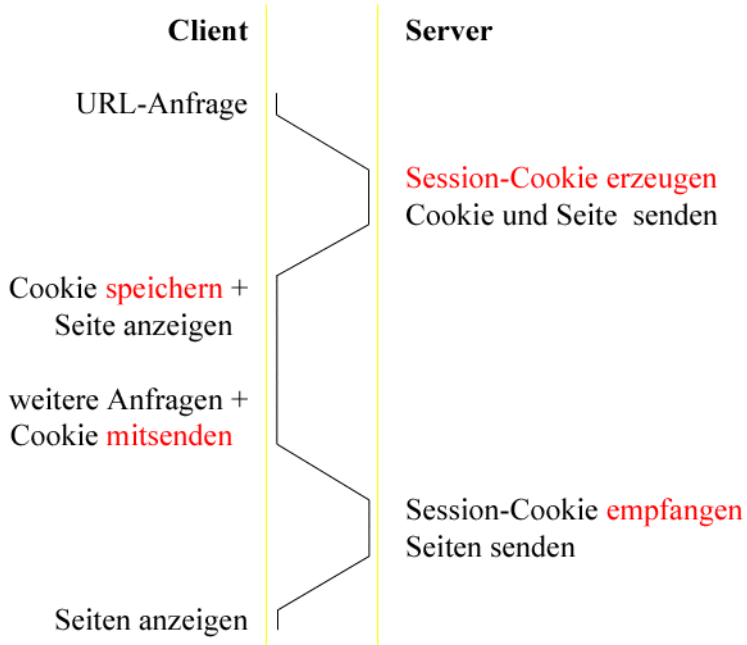



Abbildung 9: Session-Cookie

Besonders geeignet für typische Web-Anwendungen stellt sich die Cookie-Technik (siehe Abbildung 9 und Kapitel 3.4.2) heraus, da nach dem Setzen eines Cookies die gespeicherten Daten jedes Mal automatisch vom Client mitgesendet werden, egal ob es sich nun um einen GET- oder POST-Request handelt.

Um nicht nur eine Verknüpfung zwischen einem beliebigen Client und der Web-Anwendung herzustellen, sondern einen Ankerpunkt mit genau einem bereits bekannten Benutzer (durch eine frühere Anmeldung) zur Verfügung zu haben, kann man auch eine **User-ID** statt oder zusätzlich zur Session-ID benutzen. Dadurch kann man Daten einzelner Benutzer in einer **Datenbank** sammeln und somit den Inhalt der Web-Seiten und den Verlauf der Web-Anwendung noch gezielter steuern (**Personalisierung**).

## 4.2 Middleware-Aufbau

Im Gegensatz zu statischen Web-Seiten, welche einfach vom Server zur Verfügung gestellt werden und sich eher selten ändern, werden dynamische Seiten durch die Middleware ständig neu berechnet, um den Inhalt an den aktuellen Zustand anzupassen.

Um sicherzustellen, dass der Client auch immer die korrekte Seite empfängt und nicht eine alte Version, welche durch einen Proxy-Server oder im Client-Cache selbst zwischengespeichert wurde, muss bei allen nötigen Seiten immer erst einmal der HTTP-Header passend modifiziert werden (siehe Kapitel 3.4.1).

Danach kann die Middleware den empfangenen **Client-Request auswerten** und **dann** abhängig von diesem den neuen **Server-Request berechnen** und absenden.

### 4.2.1 Business- und Darstellungs-Logik

Moderne Middleware wird heutzutage oft in die zwei logische Teilaufgaben der Business-Logik und der Darstellungs-Logik zerlegt, welche nacheinander ausgeführt werden. Zusätzlich kommt dann noch die Datenbank-Anbindung hinzu:

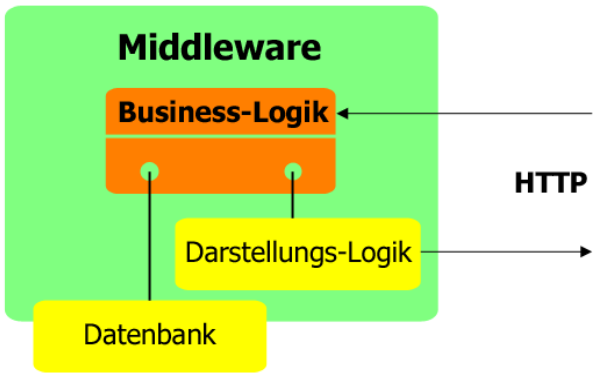



Abbildung 10: Middleware-Aufbau

1. **Die Business-Logik** (Verarbeitungs-Logik) kümmert sich in etwa um:
   - **Auswertung** der empfangenen URL-Parameter bzw. HTML-FORM-Eingaben
   - **Ablaufsteuerung** für die gesamte Web-Anwendung
   - Durchführung aller benötigten **Datenbank-Zugriffe**
   - **Berechnung aller** nötigen **Daten** für die Ausgabe (z.B.: im XML-Format)
2. **Die Darstellungs-Logik** (Presentation-Layer) ist zuständig für:
   - **Konvertierung** der Business-Logik-Daten in das Format des Clients (z.B.: HTML)
   - **Absendung** des Server-Requests

### 4.2.2 Datenbankanbindung

Generell unterscheidet man bei der Datenbankanbindung der Middleware an die DB in **Native-Interfaces** und in **Application-Program-Interfaces** (nach [20]).

Wird die Datenbank direkt über die produktspezifischen Libraries (z.B.: von Oracle, Informix, SQL-Server, ...) angesprochen, spricht man von Native-Calls. Will man aber zu einem späteren Zeitpunkt auf eine DB eines anderen Herstellers umsteigen, sind meistens weitreichende Änderungen in der Middleware notwendig.

Im Unterschied zu einer herstellerspezifischen Schnittstelle ist ein API (Application-Program-Interface) eine zumindest teilweise abstrakte Kommunikationsebene. Damit muss man sich als Middleware-Entwickler nicht mehr so stark um die Eigenheiten der tatsächlichen DB-Anbindung kümmern. Da aber von einem API selbst wieder die Native-Libraries der einzelnen Datenbanken benutzt werden, sollte auf die Liste der unterstützten DB-Interfaces geachtet werden.

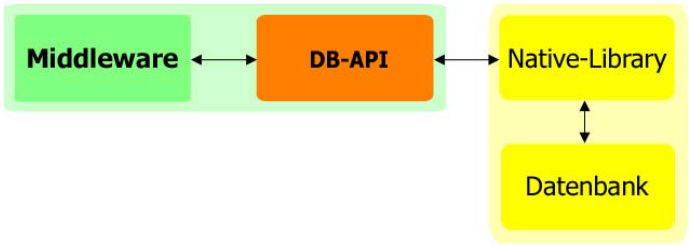



Abbildung 11: Datenbankanbindung

Die bekanntesten standardisierten **DB-APIs** für den Zugriff auf heterogene Datenbanken sind:

- **ODBC** (Open Database Connectivity)
- Microsoft **OLE DB** und **ADO** (ActiveX Data Objects)
- **JDBC** (Java Database Connectivity)

## 4.3 MS-ASP

Bei den MS-**Active-Server-Pages** handelt es sich um eine serverseitige Anbindung einer **Skript-Sprache**[5] (Server Side Scripting), welche besonders einfach und doch vielseitig zur Entwickelung von Middleware eingesetzt werden kann.

Aufbauend auf dem Internet Information Server (IIS) von Microsoft und einer geeigneten Datenbankanbindung, z.B. an den MS-SQL Server, erhält man durch MS-ASP einen leistungsfähigen Applikations-Server, welcher sich bei mehr Rechenleistungsbedarf auch noch weiter zu einem Server-Cluster mit Load-Balancing ausbauen lässt.

### 4.3.1 ASP-Features

ASP-Scriptcode kann an jeder beliebigen Position innerhalb einer am Server fest abgelegten Web-Seite (mit der Dateiendung: asp) mittels der Tags `<%` und `%>` eingebunden werden. Diese in statischen HTML-Dokumenten standardmäßig nicht verwendeten Tags dienen dem Öffnen und Schließen von ASP-Skript-Blöcken, welche dann dynamisch generiertes HTML (anstelle des ASP-Codes) zum Client senden können:

⇒ Dieses Beispiel beherbergt einen einzigen ASP-Skript-Block, welcher bei jeder Anforderung dieser Web-Seite mittels des Befehls `Response.Write` und der Funktion `Time()` eine Textzeile mit der aktuellen Uhrzeit des Servers zum Client sendet:

```
<HTML>
 <HEAD><TITLE>Example: A</TITLE></HEAD>
 <BODY><% Response.Write "Es ist gerade " & Time() & " Uhr." %></BODY>
</HTML>
```

Zusätzlich steht noch folgende verkürzte Syntax zur Verfügung für den Fall, dass es sich lediglich um die Ausgabe eines reinen ASP-Ausdrucks handelt und keine entsprechende Programmlogik benötigt wird:

⇒ Da das obige Beispiel lediglich eine berechnete Zeichenkette ausgibt (einen relativ einfachen ASP-Ausdruck), kann anstelle des Tags `<%` gefolgt vom Befehl `Response.Write` die alternative Schreibweise `<%=` angewendet werden:

```
<HTML>
 <HEAD><TITLE>Example: B</TITLE></HEAD>
 <BODY><%="Es ist gerade " & Time() & " Uhr."%></BODY>
</HTML>
```

[5] Skript-Programme benötigen prinzipiell keine explizite Compilierung, werden aber meist zur Laufzeit durch einen JIT-Compiler (just-in-time) beschleunigt.

Neben dieser eben beschriebenen Eigenschaft, dass sich ASP-Scripts direkt in den HTML-Code der benötigten Web-Seiten einbinden lassen, um somit den Inhalt der einzelnen Seiten jederzeit dynamisch ändern zu können, verfügt ASP noch über eine ganze Menge weiterer **Features** (nach [12]), z.B.:

- Datenbankanbindung durch Unterstützung von ADO (siehe 4.2.2) einfach & effizient
- HTTP-Header können direkt manipuliert werden
- HTML-Formulare und URLs (sprich POST / GET) können leicht ausgewertet werden
- Vollständige Cookie-Unterstützung
- Automatisches Session-Management
- Erweiterbar durch fremde oder eigene serverseitige ActiveX-Objekte

### 4.3.2 ASP-VBScript

Die Standard-Skript-Sprache von ASP ist von Haus aus VBScript. Auf Wunsch wird aber auch JavaScript über den Tag `<%@ LANGUAGE=JAVASCRIPT %>` unterstützt. Selbst innerhalb einer einzigen Seite kann man die benutzte Sprache jederzeit wechseln.

Bei **VBScript** handelt es sich um eine relativ fehlertolerante und schnell erlernbare Sprache. Die Anzahl der Befehlsworte ist leicht zu überblicken und die Variablen müssen vor ihrer Verwendung nicht eigens typisiert werden. Die Microsoft-eigene Sprache unterstützt unter anderem die Datentypen String, Integer, Currency, Double, Date, Boolean, Arrays und (**ActiveX**-) Objects, welche intern alle durch den **Variablentyp Variant** repräsentiert werden. Leider reicht die Objektorientierung von VBScript nur zum Anlegen und Benutzen von ActiveX-Objekten aus und nicht zum Definieren eigener Klassen samt ihrer Methoden.

Zur **Programmablaufsteuerung** stehen folgende Konstrukte zur Verfügung:

- IF – THEN – ELSE – END IF
- SELECT CASE – CASE – CASE ELSE – END SELECT
- FOR [EACH] – NEXT
- DO [WHILE] – LOOP [UNTIL]
- SUB – END SUB
- FUNCTION – END FUNCTION
- ON ERROR

Um z.B. das HTTP-Beispiel B) des Kapitels 3.4.2 (dynamische Seite) mit Hilfe eines ASP-VBScripts zu realisieren, könnte man folgendes **Beispiel-Skript (`dynamic.asp`)** verwenden:

```
<%
 ' ================================================================================
```

- Um das **Cachen** dieser dynamischen Seite am Client und auf eventuellen Proxies zu **vermeiden**, wird der HTTP-Header durch nachfolgenden Befehl passend modifiziert. (Diese Seite ist somit vor 24 × 60 Minuten in der Vergangenheit abgelaufen. → Das WWW erstreckt sich über alle Zeitzonen.)

```
 Response.Expires = -1440
 ' ================================================================================
```

- Bevor das Skript nun die aktuelle Seite berechnet, werden noch alle zum Server übertragenen **Parameter (GET und POST)** ausgewertet. Falls der Benutzer im Formular der zuletzt generierten Seite seinen Namen eingegeben hat, wird dieser in einem Cookie am Client gespeichert und eine URL-Umlenkung zur selben ASP-Seite angefordert („Client-Refresh“). Ansonsten wird die Web-Seite abhängig von den zuvor empfangenen Parametern zusammengestellt:

```
 ' ==== HANDLE-REQUEST
 form = Request.QueryString("form")
 user = Request.Cookies("user")
 IF user = "" THEN user = "???"
 name = Trim (Request.Form("name"))
 IF name <> "" THEN
  Response.Cookies("user") = name
  Response.Redirect("dynamic.asp")
 END IF
 ' ================================================================================
%>
<html>
 <head>
  <title>Dynamic Page</title>
 </head>
 <body>
  <H1>Dies ist eine dynamische Web-Seite</H1>
<%
 ' ================================================================================
 IF form = "name" THEN
%>
  <form method="post" action="?">
   <p>Ihr Name <input type="text" name="name">
   <input type="submit" value="Speichern"></p>
  </form>
  <a href="?">Zur&uuml;ck</a>
<%
 ' ================================================================================
 ELSE
%>
  <p>Ihr Name ist <%=user %>.</p>
  <a href="?form=name">Namen angeben</a>
<%
 END IF
 ' ================================================================================
%>
 </body>
</html>
<%
 ' ================================================================================
%>
```

Für Web-Entwickler, die sich genauer mit ASP und VBScript beschäftigen wollen, sei abschließend noch auf ein besonders empfehlenswertes Microsoft-Press-Buch hingewiesen:

- **Tobias Weltner**, *Active Server Pages lernen und beherrschen* (1999) [12]

## 4.4 ASP.net

Mit der offiziellen Einführung der neuen Microsoft **.NET-Technologie** im Jahr 2002 wurde auch eine neue ASP-Variante Namens ASP.net vorgestellt. Allerdings hat diese neue Version nur mehr sehr wenig mit dem alten ASP gemeinsam. So basiert ASP.net auf einem vollkommen **objektorientiert**en Programmieransatz, bei welchem zu jedem Bedienelement auf einer Web-Seite auch ein entsprechendes Objekt (inkl. Felder und Methoden) existiert. Auch wird standardmäßig keine interpretierte Skript-Sprache mehr zur Programmierung eingesetzt, sondern eine **compilierte Sprache**[6], wie z.B. VB.net oder **C#**[7], wodurch es möglich wird, auf die gesamte Klassenbibliothek von .NET zugreifen zu können (nach [23]).

### 4.4.1 Web Forms

Neben allen Features, welche auch vom „gewöhnlichen“ ASP bekannt sind, beherrscht das neue ASP.net vor allem einen völlig neuen Ansatz bei der Entwicklung von webbasierten Benutzerschnittstellen. Aber auch diverse Erweiterungen, z.B. am Session-Management, wurden in ASP.net integriert ( neben einem cookie-gestützten Verfahren, wie in Abbildung 9, wird nun auch eine URL basierte Variante angeboten. Ein solcher URL transportiert die aktuelle Session-ID innerhalb einer geklammerten Zeichenkette, platziert nach dem Hostnamen (nach [24]), z.B.: http://localhost/(v10es5voleyb55wuw55b1uuk)/session.aspx ).

Doch nun zu der neuen serverseitigen Unterstützung bei der Generierung und Auswertung von webbasierten Formularen, den sogenannten **Web Forms**. Wurden in herkömmlichen ASP-Applikationen noch die dynamischen Teile mithilfe von Response-Methoden ausgegeben und die vom Client empfangenen Eingaben durch Auswertung der Response-Collection verarbeitet, geschieht dies in ASP.net nun völlig anders. Jedem Objekt eines Formulars wird dabei am Server ebenfalls automatisch ein **Control-Object** zur Seite gestellt. Dabei wird grob zwischen 2 Arten von Objekten unterschieden (nach [25]):

- **HTML-Controls**

  dienen der direkten Unterstützung bekannter HTML-Konstrukte, wie z.B.:

---

[6] Es wird ein CIL-Code (Common Intermediate Language) erzeugt, welcher später just-in-time übersetzt wird.

[7] C# ist eine neuentwickelte Sprache aus dem Hause Microsoft, welche eine konsequente Weiterentwicklung von C++, Java und VB (Visual Basic) darstellt und speziell für die .NET-Technologie entworfen wurde. Die Syntax dieser neuen Programmiersprache sieht dem von C++ und Java dadurch sehr ähnlich.

- HtmlTextArea ..... dient der Unterstützung von ..... <textarea>
- HtmlSelect ..... ..... <select>
- HtmlTable ..... ..... <table>
- HtmlAnchor ..... ..... <a>

- **Web-Controls**

  sind erweiterte Objekte, welche nicht sofort 1:1 in HTML umgesetzt werden, sondern jeweils durch eine eigene objektspezifische Ausgabelogik (inkl. automatischer Anpassung an die verschiedensten Web-Browser) in die zu erzeugende Web-Seite integriert werden, z.B.:

  - asp:TextBox ..... Texteingabe
  - asp:DropDownList ..... Auswahlliste
  - asp:Button ..... Schaltfläche
  - asp:Hyperlink ..... Verweis
  - asp:Label ..... Beschriftung

Alle **ASP.net-Objekte** haben aber Folgendes gemeinsam:

- Sie können direkt über Programmcode (am Server) ihren Inhalt und je nach Objekt auch ihr Aussehen (Farbe, Beschriftung, Größe, ...) verändern.
- Pro Objekt können Ereignisse und dazugehörige serverseitige Bearbeitungsroutinen definiert werden. Die dazu nötige Client-Server-Synchronisation führt ASP.net dabei selbstständig durch (Parameterübergaben inkl. der jeweiligen Objekt-Zuordnung, Erhaltung aller Benutzereingaben (View-State) innerhalb einer Web Form).

Am besten lässt sich diese von ASP.net benutzte, ereignisbasierte Webprogrammierung anhand eines kurzen **Beispiel**s **(`example.aspx`)** verdeutlichen:

```
1  <%@ Page Language="C#" %>
2
```

- Als erstes steht die serverseitige Ereignisroutine des später in Zeile 18 & 19 definierten Objekts „send“. Sie modifiziert die nachfolgende HTML-Ausgabe durch Setzen eines neuen Anzeigetextes für das Objekt „output“ von Zeile 20:

```
3  <script runat="server">
4   void send_click(object sender, EventArgs e)
5   {
6    output.Text="Ihre letzte Eingabe war: "+input.Text;
7   }
8  </script>
9
```

- Nun folgt der HTML-Teil dieser Web-Seite inklusive der ASP.net-Objekte (Web Form), welche (wie auch schon die Ereignisfunktion) durch das zusätzliche Attribut `runat="server"` gekennzeichnet werden:

```
10  <html>
11   <head>
12    <title>ASP.net</title>
13   </head>
14   <body>
15    <h1>Web-Form-Example</h1>
16    <form method="post" runat="server">
17     <hr>Name: <asp:TextBox id="input" runat="server"/>
18     <asp:Button id="send" runat="server" text="OK"
19          OnClick="send_click"/>
20     <hr><asp:Label id="output" runat="server"/>
21    </form>
22   </body>
23  </html>
```

Würde man diese Web-Seite durch einen Client aufrufen, erhielte man folgendes Eingabeformular:

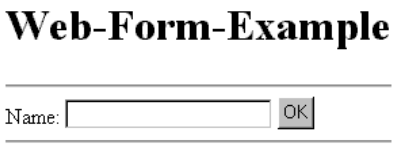



Abbildung 12: Leeres Web Form

Nachdem man dann seinen Namen (z.B. Gernot) eingegeben und das Formular mithilfe des Buttons „OK“ abgesendet hat, wird am Server die Ereignisroutine „send_click“ aufgerufen. Diese liest die Benutzereingabe aus dem Objekt „input“ aus und modifiziert daraufhin die nächste HTML-Ausgabe zum Client durch Setzen eines neuen Anzeigetextes („Ihre letzte Eingabe war: Gernot“) für das Label-Object „output“:


Abbildung 13: Web Form nach dem Absenden

Dieses Konzept ist so mächtig, dass sich alle Teile (Objekte) einer Web-Seite dynamisch am Server verändern lassen (z.B. ist auch das Hinzufügen von Tabellen-Zeilen zu einem HtmlTable-Object möglich). Ferner kann durch den gezielten Einsatz von verschiedenen Ereignisroutinen dann auch noch der Ablauf der Web-Applikation beliebig gesteuert werden.

Um sich mit dieser neuen Technologie konkreter vertraut zu machen, sei abschließend noch folgendes Buch empfohlen, welches sich neben ASP.net und C# auch mit allen weiteren Komponenten der .NET-Plattform beschäftigt:

- **Beer, Birngruber, Mössenböck, Wöß**, *Die .NET-Technologie* (2002) [23]

## 4.5 Fabasoft-VApps

Das Framework **Fabasoft-VApps** (Virtual-Applications) wurde speziell für das wachsende ASP-Marktsegment von der in Österreich ansässigen Firma Fabasoft entwickelt. Da durch VApps Web-Formulare für Thin-Clients sehr effizient automatisch **generiert** werden können, sind sie besonders für **datenorientierte ASP-Anwendungen** geeignet. Dabei kümmert sich das Framework selbständig um die Kommunikation mit dem Client und dem Zusammenhalt der einzelnen VApps.

### 4.5.1 Designkriterien von VApps

Folgende Merkmale zeichnen die VApps[8] von Fabasoft aus:

- VApps besitzen auf dem Server keinen dauerhaft gespeicherten Zustand, da dieser hex-kodiert zum Web-Client übertragen wird. Dadurch ist ein optimales Load Balancing innerhalb einer Server-Farm gesichert.
- Alle dargestellten Seiten werden aus einer universellen Beschreibung generiert. Dadurch sind VApps leicht wartbar und erweiterbar. Sie können durch die generische Interpretierung auch automatisch in verschiedenen Formaten ausgegeben werden: HTML, WAP, ...
- Die Darstellungslogik, die Businesslogik und das objektorientierte Datenmodell sind voneinander weitgehend getrennt.

### 4.5.2 Allgemeiner Aufbau

Zum Verständnis des Aufbaus von VApps bedarf es zuerst einer kurzen Einführung in die darunterliegenden objektorientierten **Fabasoft-Components**.

Im wesentlichen gilt, dass in Fabasoft-Components alles aus Objekten und den zugehörigen Objektklassen besteht, welche je eine weltweit eindeutige Adresse (COO.x.x.y.y)[9] besitzen. Jedes Objekt beinhaltet Eigenschaften bzw. Eigenschaftslisten, welche verschiedene Typen

---

[8] Fabasoft Components Version 4.0 RC1 (Vorserien-Version) vom 25.09.2000

[9] x.x steht für eine von Fabasoft vergebene Domain-Adresse, y.y wird vom System automatisch generiert

(z.B.: Zeichenkette, Zahl, Datum, Objektzeiger, ...) aufnehmen können. Die Components-Objektklassen unterstützen einfache Vererbung.

Eigene Klassen können jederzeit selbst entwickelt oder abgeleitet werden. Dadurch ist man in der Lage, das komplette **Datenmodell**, welches für eine Anwendung benötig wird, **objektorientiert** zu designen und benötigt keine herkömmliche Datenbankanbindung mehr.

Es stehen eine Vielzahl von vorgefertigten **Klassensammlung**en zur Verfügung, z.B.: Dokumentenmanagement, Kundenbeziehungs-Management, ...

Eine **Fabasoft-VApp** ist eine Sammlung von verketteten Objekten, welche mindestens folgende Objektklassen benutzt:

- **Anwendungsformat**
  Definiert die Art der Ausgabe (HTML, WAP, ...)
- **Anwendungssteuerung**
  Das Bindeglied zwischen dem gewählten *Anwendungsformat* und einer konkreten *Anwendung*. So kann eine VApp dann z.B. über folgenden URL gestartet werden:
  .../fscasp/content/bin/fscext40.dll?dx=COO.1.980.1.1220&ax=COO.1.980.1.1344
  Wobei **dx** die Adresse der *Anwendungssteuerung* und **ax** die der *Anwendung* darstellt.
- **Anwendung**
  Innerhalb einer Anwendung können Schritte (Schrittobjekte) definiert werden, welche sequentiell abgearbeitet werden und die Businesslogik beschreiben:
  1. Aufruf einer *Anwendung* oder *Anwendungssicht*
  2. Starten einer *Aktion* zur Manipulation von Objekt-Daten oder VApp-Variablen
  3. *IF, WHILE, TRY*
  4. Eigene *Skriptkomponentenobjekte*, welche in VB oder JavaScript implementiert werden können und ein genormtes Interface zu VApps besitzen.
- **Anwendungssicht**
  Beschreibt eine komplette VApp-Seite zur Ein-/Ausgabe von Daten am Client. Sie selbst besteht aus einer Sammlung von *Sichten*, welche strukturiert ausgegeben werden. Auch Verzweigungen, z.B. zu weiteren Anwendungssichten, können mittels definierbarer Schritte (siehe auch *Anwendung*) angegeben werden. Diese Verzweigungen folgen einem Stack-Konzept, sodass man auch zu früheren Seiten zurückkehren kann.
- **Sicht**
  Hier wird beschrieben, wie einzelne Eigenschaften bzw. Eigenschaftslisten formatiert ausgegeben werden sollen. *Sichten* können auch verschachtelt eingesetzt werden.

### 4.5.3 Hybridlösungen

VApps können zusätzlich zu dx & ax (*Anwendungssteuerung* & *Anwendung*) auch noch weitere Parameter im URL entgegennehmen, um von außen steuerbar zu bleiben. Beim Beenden der VApp ist es auch möglich zusätzlich einen frei definierbare URL aufzurufen. Somit ist es denkbar, z.B. eine frame-basierte Web-Anwendung zu gestalten, welche in einem Frame eine zentrale Navigation besitzt und in einem anderen Frame eine beliebige VApp ansteuert, die im Anschluss eigenständig abläuft.

Ergänzend bietet Fabasoft-Components ein Interface zur Kommunikation mit MS-ASP des Internet-Information-Servers an. Dadurch ist es mittels Active-Server-Pages (welche mehr Freiheit in der Web-Darstellung erlauben) möglich, auf die in Components abgelegten Daten zuzugreifen.

Setzt man nun Active-Server-Pages ein, um diverse Fabasoft-VApps aufrufen, bzw. auch umgekehrt, erhält man eine Art Hybridlösung zwischen beiden Technologien! Dadurch können nun auch Teile eines Web-Portals, welche durch die relativ starre Darstellungsstruktur der VApps nur schwer realisierbar wären, mit Hilfe von Active-Server-Pages flexibler implementiert werden.

## 4.6 PHP4

In der großen Fülle an verfügbaren **serverseitige**n **Web-Skriptsprache**n, hat sich neben ASP inzwischen auch die Sprache PHP einen guten Namen gemacht.

1994 schuf Rasmus Lerdorf die erste Version von PHP (Personal-Home-Page Tools), welche er den **Open-Source**-Jüngern frei zur Verfügung stellte. Daraus entstand 1995 die 2.Version (PHP / Form Interface), die bereits professionellen Einsatz fand. Anfang 1999 war das **kostenlos**e, unter der **GNU-Lizenz** veröffentlichte PHP bereits bei der Versionsnummer 3 angekommen und entwickelte sich 2000 mit Hilfe von ZEND (Zeev Suraski und Andi Gutman) zum jetzigen **PHP4** (nach [13]).

### 4.6.1 PHP-Features

PHP4 ist für viele Plattformen (Betriebssysteme als auch Web-Server) verfügbar. Es vereint die Einfachheit von ASP mit der bekannten Syntax von C und der Mächtigkeit von Perl (Reportsprache zur Behandlung von Zeichenketten). **Objektorientiert**es Programmieren ist ebenso möglich wie eine saubere Modularisierung. So können beliebige Klassen vollständig in PHP4 implementiert und innerhalb der eigenen Web-Seiten verteilt benutzt werden.

Um einen groben Überblick über die vielen Features von PHP4 zu geben, seien nun einige der wichtigsten aufgelistet (nach [13]):

- Datenbankanbindung zu allen gängigen DB-Systemen
- HTTP-Header-Manipulation
- HTML-Formular- und URL-Auswertung (POST / GET)
- Komplette Cookie-Unterstützung
- Vollständiges Session-Management
- PHP-Klassen und Objekte inklusive einfacher Vererbung
- eine bestechend umfangreiche Funktionsbibliothek
- Verarbeitung von regulären Ausdrücken
- Grafiken per PHP am Server generieren
- E-Mails per Skript versenden
- XML-Unterstützung

### 4.6.2 PHP-Sprachaufbau

Genau wie bei ASP-Seiten kann PHP und HTML innerhalb des selben Dokumentes gemischt verwendet werden. Dabei gibt es diverse Möglichkeiten PHP-Code einzubetten:

⇒ Das nachfolgende Beispielskript dient lediglich der Darstellung der verschiedenen Einbettungsmöglichkeiten von PHP-Code. Dabei werden folgende Tags zur PHP-Einklammerung verwendet: `<? ?> <% %> <script> </script>`

```
<HTML>
 <HEAD>
  <TITLE>
   Examples
  </TITLE>
 </HEAD>
 <BODY>
  <?php     print "Version A |"; ?>
  <?        print "Version B |"; ?>
  <? $version_c = "Version C |"; ?>
  <?=$version_c ?>

  <script language="php">
            print "Version D |";
  </script>
  <%        print "Version E |"; %>
  <% $version_f = "Version F  "; %>
  <%=$version_f %>
 </BODY>
</HTML>
```

⇒ Die erzeugte Ausgabe entspricht der folgenden Zeichenkette:

```
Version A | Version B | Version C | Version D | Version E | Version F
```

Ebenfalls analog zu ASP muss bei PHP für **Variablen kein fixer Typ** vorgegeben werden. Jede Variable kann intern entweder einen String-, Integer-, Double-, Array- oder Object-Typ aufnehmen. Andere Typen wie Date oder Boolean werden z.B. als Integer gespeichert.

Da PHP von der Syntax her sehr stark an C angelehnt wurde, sieht der PHP-Code seinem Vorbild sehr ähnlich. Einer der wesentlichsten Unterschiede besteht allerdings darin, dass Variablen immer ein **$**-Zeichen vorangestellt haben.

Zur PHP-**Programmablaufsteuerung** gibt es sowohl **C-ähnliche** als auch neue Konstrukte:

- if (…) {…} elseif (…) {…} else {…}
- switch (…) {case …: … break; … default: …}
- for (…;…;…) {…}
- while (…) {…}
- do {…} while (…)
- foreach (… as …) {…}
- function …(…) {… return …}
- class … extends … {var … function …}
- @…; // hide error: $php_errormsg

Das folgende kurze **Beispiel-Skript (`dynamic.php`)** implementiert nun mittels PHP das im Kapitel 3.4.2 Unterpunkt B) (dynamische Web-Seite) angegebene Verhalten:

```
1   <?
2    // ============================================================================
```

- Definition einer PHP-Funktion, welche durch Modifizierung des **HTTP-Header**s sicherstellt, dass die gewünschte Version dieser generierten Seite auch stets beim Client ankommt.

```
3    function http_nocache()
4    {
5     header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");          // Date in the past
6     header("Last-Modified: ".gmdate("D, d M Y H:i:s")." GMT"); // always modified!
7     header("Cache-Control: no-cache, must-revalidate");        // HTTP/1.1
8     header("Pragma: no-cache");                                // HTTP/1.0
9    }
10   // ============================================================================
```

- Analog zur ASP-Implementierung im Kapitel 4.3.2 werden auch in diesem Skript nun die **GET- und POST-Parameter** ausgewertet. Falls der Benutzer seinen Namen beim letzten Aufruf dieser Web-Seite eingegeben hat, wird ebenfalls ein Cookie gesetzt und anschließend ein Redirect durchgeführt.

Ansonsten wird die eigentliche Web-Seite generiert:

```
11   $form = isset($HTTP_GET_VARS["form"])    ? $HTTP_GET_VARS["form"]    :    "";
12   $user = isset($HTTP_COOKIE_VARS["user"]) ? $HTTP_COOKIE_VARS["user"] : "???";
13   $name = isset($HTTP_POST_VARS["name"])   ? $HTTP_POST_VARS["name"]   :    "";
14   // ============================================================================
15   http_nocache();
16   // ============================================================================
17   if (trim($name)!="") {
18    setcookie("user",trim($name));
19    header ("Location: ".$SERVER_URL."dynamic.php");
20    exit;
21   }
22   // ============================================================================
23  ?>
24  <html>
25   <head>
26    <title>Dynamic Page</title>
27   </head>
28   <body>
29    <H1>Dies ist eine dynamische Web-Seite</H1>
30  <?
31   // ============================================================================
32   if ($form == "name") {
33  ?>
34    <form method="post" action="?">
35     <p>Ihr Name <input type="text" name="name">
36     <input type="submit" value="Speichern"></p>
37    </form>
38    <a href="?">Zur&uuml;ck</a>
39  <?
40   }
41   // ============================================================================
42   else {
43  ?>
44    <p>Ihr Name ist <?=$user ?>.</p>
45    <a href="?form=name">Namen angeben</a>
46  <?
47   }
48   // ============================================================================
49  ?>
50   </body>
51  </html>
52  <?
53   // ============================================================================
54  ?>
```

Abschließend seien für Interessierte, welche sich eingehender mit PHP4 beschäftigen wollen, zwei ausführliche Referenzen (ein Buch und ein Online-Dokument) und eine Kurzübersicht besonders empfohlen:

- **Jörg Krause**, *PHP - Grundlagen und Lösungen* (2000) [13]
- http://www.php.net/docs.php, *PHP: Documentation* (12.5.2002) [15]
- http://www.selfphp.info/, *SelfPHP* (3.10.2002) [26]
- **Mark Kronsbein, Thomas Weinert**, *PC Spicker PHP4* (2001) [14]

## 4.7 Konzepte für Web-Anwendungen

Im Kapitel 4.1 und 4.2 wurden ja bereits zwei der wichtigsten Basiskonzepte angeführt, ohne die es nicht möglich ist vernünftige Web-Anwendungen zu realisieren. Alle in den darauf folgenden Kapiteln angeführten serverseitigen Programmiersprachen sind auch in der Lage, diese angesprochenen Konzepte weitreichend zu unterstützen.

So ist es möglich, die Businesslogik und die Darstellungslogik (siehe Kapitel 4.2) durch geeignete Klammerung (Script-Tags) bzw. Code-Modularisierung (z.B.: bei ASP und PHP) voneinander zu trennen oder die HTML-Ausgabe überhaupt generisch durch den Programmcode erzeugen zu lassen (z.B. ASP.net, Fabasoft VApps). Auch bieten alle behandelten Sprachen ein geeignetes Sessionmanagement (siehe Kapitel 4.1), um die eigentlich verbindungslosen Web-Seiten zu einer geschlossenen Applikation zusammenführen zu können. Sessionbezogene Daten (Variablen) können dabei direkt am Applikations-Server gehalten werden, um **z.B.** diverse Benutzereingaben abzulegen, die aktuelle **Sprachwahl** zu speichern oder einen gesamten **Warenkorb** zu realisieren. Bei Verwendung von Cookies ist es sogar möglich die einzelnen Clients längerfristig (über mehrere Sitzungen hinweg) wiederzuerkennen. Setzt man diese Techniken zusammen mit einer Datenbank gezielt ein (z.B. indem man gesammelte Informationen über einzelne Benutzer in der DB ablegt), ergeben sich höchst interessante Möglichkeiten für Web-Anwendungen.

Im EC ist es dabei besonders wichtig, durch **intelligente Konzepte** neue Anreize für die umworbenen Kunden zu schaffen, damit diese bewegt werden, ihre bisherigen Geschäftsgewohnheiten zugunsten der jeweiligen EC-Lösung zu verändern (nach [27]):

- **Personalisierung**

  Durch die Möglichkeit sich als Benutzer bei einer EC-Web-Applikation anmelden zu können (im einfachsten Fall durch einen frei wählbaren Namen (**nickname**) und ein **Kennwort**), können diverse Eingaben des daraufhin bekannten Kunden in einer DB gesammelt und wieder abgerufen werden. Dadurch kann sich diese Web-Anwendung durch Umsetzung z.B. folgender Konzepte auf die einzelnen Benutzer einstellen:

  1. **Konfigurator**: Dem Benutzer können mehrere **Designs** (Skins, Farben, ...) und Sprachversionen der Web-Seite angeboten werden, aus welchen er frei wählen kann. Dadurch kann die Akzeptanz der Benutzeroberfläche der jeweiligen EC-Lösung erheblich gesteigert werden (positive Erfahrung).
  2. **Intelligente Angebote:** Sind in einem EC-System zu allen Produkten intern geeignete Interessensinformationen gespeichert (z.B.: Artikel „Laufschuh X“ gehört zu den Profilen „Laufsport“ und „Freizeit“), kann man durch Aufzeichnen (**Logging**) und Auswerten der vom Benutzer früher bereits angesehenen Artikel (**Surfverhalten**) Rückschlusse auf die Interessen des

Kunden (**Kundenprofil**) ziehen. Dadurch ist eine Web-Anwendung in der Lage zielgerichtete Angebote (z.B. auf der Startseite) anbieten zu können.

Dieses Konzept muss natürlich nicht auf Produkte allein beschränkt bleiben. Es ist genauso denkbar einzelne Web-Seiten und Web-Links verschiedenen **Interessensgebiete**n zuzuordnen und somit weitere Faktoren zur Kundenprofilerkennung zu gewinnen. Auch durch Benutzerfragebögen (z.B. bei einer Kundenerstanmeldung) kann man Informationen über Hobbys, u.s.w. erhalten.

3. **Intelligente Werbung:** Auch kundenprofilbezogene Werbebanner folgen dem im vorigen Punkt erläuterten Konzept.
4. **Nachbetreuung:** Durch Speicherung aller getätigten Transaktionen eines Kunden (z.B. seine Einkäufe, Downloads, ...) können durch die Web-Applikation benutzerspezifische Hinweise auf geeignetes Zubehör, neue Softwareupdates, aktuelle Gerätetreiber, u.s.w. gegeben werden.

- **Added Services**

  Durch Zusatzdienste, welche nicht unbedingt direkt mit der EC-Lösung zusammenhängen müssen, kann der Kunde angeregt werden die Web-Site regelmäßig aufzusuchen. Solche „Added Services“ können z.B. sein:

  1. **Mitgliederforum:** So können sich z.B. die Kunden einer EC-Lösung gegenseitig bei Problemen mit gekauften Produkten helfen. Dies ist günstig für den EC-Anbieter und praktisch für alle Benutzer.
  2. **Produktbewertungen:** Dadurch kann jeder Kunde seine eigene Meinung zu einem Produkt „online“ äußern (bzw. das Produkt benoten). Dadurch wird z.B. anderen Benutzern zu einer (hoffentlich) besseren Kaufentscheidung verholfen, was die Zufriedenheit der Kunden des Anbieters steigert.
  3. **Newsboard:** Wird auf einer EC-Web-Site zusätzlich ein ständig aktualisierter Informationsbereich integriert, welcher sich allein mit spezifischen Themen rund um das Angebot dieser EC-Lösung beschäftigt (z.B.: Hinweise auf Messen, ...), kann auch die Besuchsfrequenz der Kunden erhöht werden.
  4. **Games:** Kleine Online-Spiele und Quizzes regen auch zum öfteren Webbesuch an.
  5. **Downloads**: Prospekte, Desktop Wallpapers, Screensavers bieten auch Anreize.

Die aufgezählten Konzepte, welche helfen sollen Web-Anwendungen interessanter und intelligenter zu gestalten, sind natürlich beliebig kombinierbar und erweiterbar. Dieses Unterkapitel soll aber keine vollkommene Auflistung aller für Web-Applikationen benutzbaren Konzepte beinhalten, sondern versteht sich vielmehr als Wegweiser zu neuen, innovativen EC-Lösungen.

## 4.8 Fazit

Ziel dieses Kapitels war einerseits die Darstellung diverser systemübergreifender Konzepte, welche für Entwickler von EC-Web-Applikationen zum Basiswerkzeug gehören sollten, und andererseits die Schaffung eines groben Überblicks über einige verfügbare Web-Programmiersprachen (Programmierumgebungen) und deren Features.

Damit schließt der (vorwiegend) theoretische Teil dieser Arbeit. Die nächsten zwei Kapitel wenden sich nun jeweils einem vollständig implementierten EC-Fallbeispiel zu. Zum besseren Verständnis der darin gezeigten Code-Beispiele sind allerdings fortgeschrittene Kenntnisse über HTML, Javascript, ASP, VBScript, SQL und VApps erforderlich.

### 4.8.1 Weiterführende Literatur

Abschließend seien noch die wichtigsten Literaturempfehlungen wiederholt, welche auch schon bei den einzelnen Abschnitten dieses Kapitels angeführt wurden:

- **Tobias Weltner**, *Active Server Pages lernen und beherrschen* (1999) [12]
- **Beer, Birngruber, Mössenböck, Wöß**, *Die .NET-Technologie* (2002) [23]
- **Jörg Krause**, *PHP - Grundlagen und Lösungen* (2000) [13]

# 5 Fallbeispiel: Online-Shop

Dieses Kapitel widmet sich nun einer konkreten **Implementierung eines EC-Systems**, welches in Zusammenarbeit mit der Diplomarbeit von Herrn Georg Kotek entstanden ist (siehe [28]). Dabei handelt es sich um einen speziellen Online-Shop, der vorwiegend **für den Online-Vertrieb** entwickelt wurde. Darunter versteht man die Möglichkeit, Produkte, welche in elektronischer Weise übertragbar sind (Software, elektronische Bücher, Bilder, Musik, Filme, ...) auch vollständig online erwerben zu können. Das Shop-System ist aber auch in der Lage „greifbare“ Produkte (Hardware, Bücher, CD’s, ...) zu vertreiben und den Shop-Betreiber später bei der Auslieferung per Post oder Spedition weitreichend zu unterstützen.

## 5.1 Projektbeschreibung

Am Anfang jeder Entwicklung muss aber zunächst die Frage nach der zugrunde liegenden **System-Plattform** beantwortet werden. Für diesen EC-Prototypen sollte eine Middleware-Sprache benutzt werden, welche relativ einfach zu verstehen ist, um die implementierten Techniken leicht nachvollziehen zu können. Aus diesem Grund fiel die Entscheidung auf folgende Basiskonfiguration:

- MS-Windows NT, 2000 oder XP
- MS-Internet-Information-Server inklusive **ASP-VBScript**
- MS-SQL-Server 7 / 2000 mit ADO-Datenbankanbindung

Anfangs waren darüber hinaus noch einige der MS-Site-Server - Erweiterungen (siehe [16], bieten Funktionen für z.B.: Personalisierung, Werbebanner, Analysen, ...) geplant, welche sich für diese Lösung leider als wenig hilfreich herausstellten. Erstens ist die Installation des Site-Servers kompliziert und sehr fehleranfällig und zweitens wäre eine solche Implementierung des Online-Shops nicht mehr so transparent wie eine reine MS-ASP - Lösung.

### 5.1.1 Entwurf

Nach der Festlegung der System-Plattform, wurde **vor Beginn** der Implementierungsarbeiten zum angestrebten Online-Shop-System zunächst ein **Grobentwurf** erstellt. Dabei wurde im ersten Schritt das Gesamtprojekt in drei aufbauende Komponenten zerlegt, auf welche im weiteren Verlauf dieses Kapitels noch genauer eingegangen wird. Diese **Komponenten** sind:

- die **Datenbank**-Struktur inkl. diverser DB-Prozeduren
- die **Backoffice**-VBScripts für die Administration (**Intranet**)
- die **Website**-VBScripts (Frontoffice) für die Kunden des **Internet**-Shops

Danach wurden die gewünschten **Funktionalitäten bzw. Features**, welche das **Backoffice** dem Betreiber des Systems zur Verfügung stellen soll, fixiert:

- Mehrbenutzersystem mit frei definierbaren Zugriffsrechten
- Shop-Konfiguration: Backoffice-Benutzer, Produktgruppen, Sprachen, multilinguale Shop-Texte, Länder / Währungen, Umrechnungskurse, Versandkosten / Steuern
- komfortable Bearbeitung der Produkt-Stammdaten
- umfangreiche Kunden-Stammdaten-Verwaltung
- Logistik-Unterstützung für folgende Bereiche: Online-Vertrieb, Versandwesen, Rechnungswesen, Kreditkartenabrechnung.
- vielseitiges Analyse-Werkzeug: Produkt-, Warenkorb-, Download-, Bestellungs-, Kreditkarten-, Umsatz-, Kunden-, Länder- und Lieferzeit-Auswertungen
- erweiterungsfähiges Hilfesystem

Anschließend wurde auch der **Funktionsumfang**, welchen der Kunde des **Online-Shop**s bereitgestellt bekommt, definiert:

- kategorisierter Internet-Produktkatalog (auch für nicht registrierte Kunden)
- Online-Registrierung für neue Shop-Kunden
- freie flexible Produktsuche
- Support-, Informations- und Kontakt-Webseiten
- Warenkorb- und Bestellungssystem für registrierte Kunden
- Download-Bereich für online gekaufte Produkte

### 5.1.2 Zukunft

Auch zukünftige Erweiterungsmöglichkeiten, die für einen Online-Shop als besonders interessant gelten, sollten für nachfolgende Shop-Revisionen leicht integrierbar bleiben.

Dabei wurde bereits über folgende **Zusatzfeatures** nachgedacht:

- personalisierte Angebote durch Auswerten des Surfverhaltens der einzelnen Kunden
- Unterstützung von E-Cash
- erweiterte Zahlungsmöglichkeiten (z.B. über einen Paymentserver)
- besondere Angebote z.B. per SMS, MMS oder Email propagieren

Doch nun zurück zum aktuellen Shop-Entwurf, welcher nachfolgend genau besprochen wird.

## 5.2 Datenbank

Bevor nun auf alle Tabellen der Shop-DB im Einzelnen präziser eingegangen wird, gibt die Abbildung 14 zunächst einmal einen Überblick über alle intern verwendeten **DB-Relationen**:

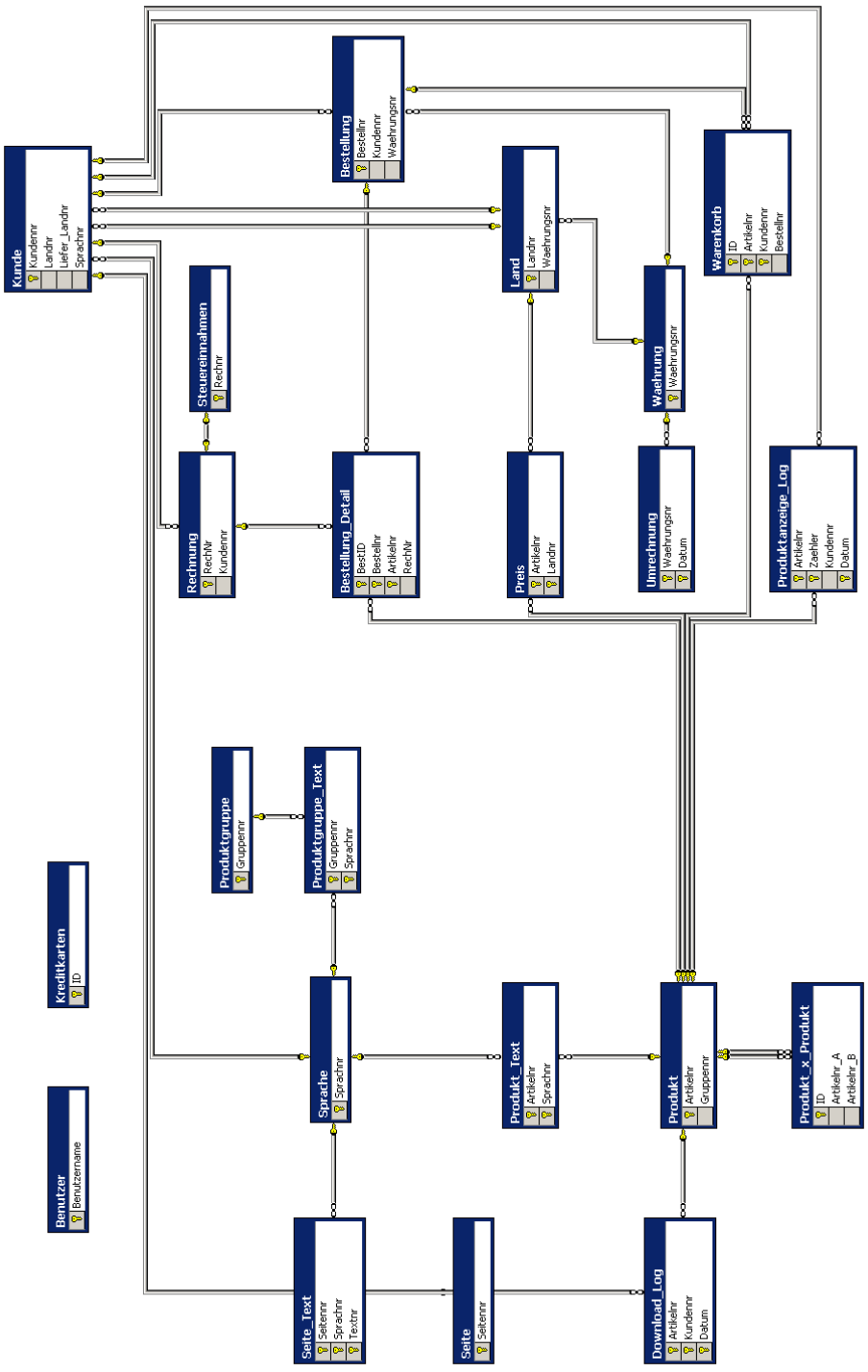



Abbildung 14: Online-Shop-Datenbankrelationen (inkl. aller Primär-Schlüsselwörter)

### 5.2.1 DB-Tabellen

| Benutzer | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ Benutzername | varchar(20) |
| Vorname | varchar(20) |
| Nachname | varchar(20) |
| Passwort | varchar(20) |
| Berechtigungsstufe | varchar(10) |

Alle Backoffice-Benutzer des Systems werden in dieser Tabelle abgelegt. Als **Primary-Unique-Key** ♦ (siehe Legende[10]) dient der Benutzername. Im Kapitel 5.4.1 werden die Berechtigungsstufen erläutert.

| Sprache | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ Sprachnr | int |
| Sprache | varchar(20) |

Um die Multilingualität der Shop-Website zu ermöglichen, müssen hier die zu unterstützenden Sprachen definiert sein.

| Land | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ Landnr | int |
| Land | varchar(30) |
| Postgebuehr | money |
| Nachnahmegeb | money |
| → Waehrungsnr | int |
| Steuern | int |

Um den Online-Shop länderübergreifend einsetzen zu können, müssen einige länderspezifische Daten in dieser Tabelle gespeichert sein.
Die beiden abgelegten Gebühren sind exklusive Steuern (%) anzugeben.

| Produkt | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ Artikelnr | varchar(20) |
| Artikelname | varchar(80) |
| Herstellerlink | varchar(80) |
| → Gruppennr | int |
| Downloadlink | varchar(80) |
| Bestand | int |
| Bestand_Min | int |
| Datum | datetime |
| Aktiv | varchar(1) |

Alle Produkte, ob zur Zeit gerade aktiv ("1") oder inaktiv bzw. ausgelaufen ("0"), werden hier gesammelt. Jeder Artikel muss einer Produktgruppe zugeordnet werden. Der angegebene Artikelname dient nur zur internen Bezeichnung im Backoffice. Für Online-Vertriebsprodukte wird zusätzlich ein FTP-Downloadlink benötigt.

| Produkt_Text | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ Artikelnr | varchar(20) |
| ♦ Sprachnr | int |
| Artikelbezeichnung | varchar(80) |
| Artikelbeschreibung | varchar(800) |
| Bild | varchar(80) |

Pro Sprache und Produkt muss ein Datensatz in dieser Tabelle gespeichert werden. Dadurch ist das System in der Lage sprachabhängige Texte und Bilder zu den vorhandenen Artikeln darzustellen.

[10] **Legende**: ♦ Primär-Schlüsselwort ♦ reine Sequenz-ID → Fremdschlüssel (Referenz)

| Produkt_x_Produkt | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ ID | int |
| → Artikelnr_A | varchar(20) |
| → Artikelnr_B | varchar(20) |

Für künftige Shoprevisionen wurde diese Tabelle bereits zur Identifikation beliebiger Produkt-Verknüpfungen vorgesehen (z.B.: Digicam mit entsprechendem Zubehör).

| Produktgruppe | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ Gruppennr | int |
| Gruppe | varchar(30) |

Hier werden die unterschiedlichen Produktgruppen (z.B.: Anwendersoftware, Spiele, ...) inkl. Backoffice-Namen definiert.

| Produktgruppe_Text | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ Gruppennr | int |
| ♦ Sprachnr | int |
| Gruppenname | varchar(50) |

Wegen der Multilingualität muss pro Sprache und Produktgruppe eine korrekte Gruppenbezeichnung vorhanden sein.

| Produktanzeige_Log | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ ID | int |
| ♦ Artikelnr | varchar(20) |
| ♦ Zaehler | int |
| → Kundennr | int |
| ♦ Datum | datetime |

Diese Tabelle dient für Statistik- und Personalisierungsaufgaben. Durch sie ist es möglich, pro Datum und Artikel die Web-Zugriffe (anonyme oder die eines bekannten Kunden) mitzuzählen.

| Seite | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ Seitennr | int |
| Zugriffe | int |
| Einstellungen | varchar(30) |

Jede Webseite des Online-Shops benötigt eine interne Seitennummer. Für die Zukunft wurden ein Parametrisierungsfeld und ein Zugriffszähler vorgesehen.

| Seite_Text | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ ID | int |
| ♦ Seitennr | int |
| ♦ Sprachnr | int |
| ♦ Textnr | int |
| Text | text |

Um auch bei eigentlich statischen Texten der Website Multilingualität zu erreichen, werden diese Texte per Referenz über die Seitennummer und einer zugehörigen Textnummer übersetzt.

| Waehrung | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ Waehrungsnr | int |
| Waehrung | varchar(20) |

Damit auch unterschiedliche Währungen (wie z.B.: EUR, USD, ...) unterstützt werden, müssen diese hier definiert sein.

| Umrechnung | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ ID | int |
| ♦ Waehrungsnr | int |
| Kurs | money |
| ♦ Datum | datetime |

Pro Tag und Währung muss ein aktueller Umrechnungskurs eingespeichert werden.

| Preis | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ ID | int |
| ♦ Artikelnr | varchar(20) |
| Preis | money |
| ♦ Landnr | int |

Pro Artikel und Land (Währung) wird in dieser Tabelle der Preis exkl. Steuern abgelegt. Wird ein Artikel in einem Land nicht vertrieben, hat er keinen solchen Eintrag.

| Rechnung | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ RechNr | int |
| → Kundennr | int |
| Datum | datetime |
| Steuer | int |

Hier werden alle ausgestellten Rechnungen archiviert. Dabei wird auch die zur Zeit gerade gültige Steuer (%) mitgesichert.

| Steuereinnahmen | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ ID | int |
| ♦ Rechnr | int |
| Betrag | money |

Zu jeder Rechnung muss in dieser Tabelle der in der Kundenwährung abzuführende Steuerbetrag eingetragen werden.

| Warenkorb | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ ID | int |
| ♦ Artikelnr | varchar(20) |
| ♦ Kundennr | int |
| Anzahl | int |
| Bestellnr | int |
| Datum | datetime |
| Ausgewaehlt | char(1) |
| Versand | char(1) |

Alle von einem Kunden in den Warenkorb gelegten Produkte werden hier gespeichert. Im Feld `"Versand"` wird angegeben, ob es sich um einen Online- oder Speditions-Artikel ("O" / "S") handelt. Nur ausgewählte Einträge ("J" / "N") sind bei einer eventuell folgenden Bestellung zu berücksichtigen.

| Bestellung | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ Bestellnr | int |
| → Kundennr | int |
| Vertriebsart | varchar(20) |
| → Waehrungsnr | int |
| Versandkosten | money |
| Datum | datetime |
| Zeit | datetime |
| Titel | varchar(20) |
| Nachname | varchar(30) |
| Vorname | varchar(30) |
| Strasse | varchar(20) |
| Nr | varchar(20) |
| Plz | varchar(10) |
| Ort | varchar(30) |
| Landnr | int |
| Tel | varchar(30) |
| Handy | varchar(30) |
| Fax | varchar(30) |
| Email | varchar(60) |
| Geburtsdatum | datetime |
| Kreditkarte | varchar(30) |
| Kartennummer | varchar(20) |
| Gueltigkeit | varchar(10) |
| Liefer_Titel | varchar(20) |
| Liefer_Vorname | varchar(30) |
| Liefer_Nachname | varchar(30) |
| Liefer_Strasse | varchar(20) |
| Liefer_Nr | varchar(20) |
| Liefer_Plz | varchar(10) |
| Liefer_Ort | varchar(30) |
| Liefer_Landnr | int |

Diese Tabelle enthält alle getätigten Bestellungen der Shop-Kunden. Folgende Vertriebsarten werden dabei unterstützt:

"O+K" ... Online mit Kreditkarte
"O+Z" ... Online mit Zahlschein
"S+K" ... Spedition mit Kreditkarte
"S+Z" ... Spedition mit Zahlschein
"S+N" ... Spedition per Nachnahme

Eine Mischbestellung aus Online- und Speditionsvertrieb kann nur durch Aufspaltung in zwei Bestellungen erfolgen. Zusätzlich müssen die Kundenstammdaten-Felder zu jeder Bestellung kopiert werden, um den zu dieser Zeit gültigen Stand zu sichern. Die einzelnen Artikel der Bestellungen werden dann separat in der Tabelle "Bestellung_Detail" gespeichert.

| Bestellung_Detail | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ BestID | int |
| ♦ Bestellnr | int |
| ♦ Artikelnr | varchar(20) |
| Anzahl | int |
| Preis | money |
| Rabatt | int |
| Status | varchar(20) |
| Lieferdatum | datetime |
| → RechNr | int |
| Bearbeiter | varchar(20) |
| Verrechnet | varchar(3) |

Hier werden nun die einzelnen Positionen aller Bestellungen archiviert. Wird später eine Rechnung ausgestellt, muss noch die Rechnungsnummer eingetragen werden. (Die zur Rechnung gehörigen Kunden-stammdaten können dann aus der Tabelle "Bestellung" abgeleitet werden.)

| Download_Log | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ ID | int |
| ♦ Artikelnr | varchar(20) |
| Zaehler | int |
| ♦ Kundennr | int |
| ♦ Datum | datetime |

In dieser Tabelle werden alle gestarteten Produkt-Downloads der Kunden mitprotokolliert.

| Kunde | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ Kundennr | int |
| Titel | varchar(20) |
| Nachname | varchar(30) |
| Vorname | varchar(30) |
| Strasse | varchar(20) |
| Nr | varchar(20) |
| Plz | varchar(10) |
| Ort | varchar(30) |
| → Landnr | int |
| Tel | varchar(30) |
| Handy | varchar(30) |
| Fax | varchar(30) |
| Email | varchar(60) |
| Geburtsdatum | datetime |
| Kreditkarte | varchar(30) |
| Kartennummer | varchar(20) |
| Gueltigkeit | varchar(10) |
| ♦ Benutzername | varchar(20) |
| Passwort | varchar(20) |
| Bonitaet | varchar(20) |
| Rabatt | int |
| Liefer_Titel | varchar(20) |
| Liefer_Vorname | varchar(30) |
| Liefer_Nachname | varchar(30) |
| Liefer_Strasse | varchar(20) |
| Liefer_Nr | varchar(20) |
| Liefer_Plz | varchar(10) |
| Liefer_Ort | varchar(30) |
| → Liefer_Landnr | int |
| → Sprachnr | int |
| Datum | datetime |
| Aktiv | varchar(1) |
| OnlineZahlung | char(1) |
| OfflineZahlung | char(1) |

Hier werden alle Kundenstammdaten gesammelt. Wie auch bei den Stammdaten der Produkte werden in dieser Tabelle nie Einträge gelöscht, sondern nur auf inaktiv gesetzt. Der eindeutige Benutzername inkl. Passwort der registrierten Kunden wird zur Wiederanmeldung am Online-Shop benötigt.

| Kreditkarten | |
|---|---|
| **Feldname** | **Datentyp** |
| ♦ ID | int |
| Kartenname | varchar(50) |

Die in dieser Tabelle gespeicherten Kreditkartennamen werden vom Shop-System zur Auswahl gestellt.

### 5.2.2 DB-Stored-Procedures

Softwaretechnisch werden in den ASP-Seiten des Online-Shop-Systems zwei verschiedene Arten der DB-Abfrage bzw. DB-Manipulation benutzt:

1. Direktes Absetzen von **SQL-Kommandos**: Das hat den Vorteil, dass die Abfragen direkt im Source der ASP-Scripts zu erkennen sind und die jeweiligen SQL-Statements direkt von der Business-Logik modifiziert bzw. generiert werden können. Leider sind aber solche Datenbank-Zugriffe im Verhältnis zur 2. Methode bei weitem nicht so performant.
2. Benutzung von **Stored-Procedures** des **MS-SQL-Server**s: Diese werden direkt in der DB gespeichert, automatisch vorkompiliert (dadurch sehr effizient) und später nur mehr aufgerufen (durch ASP). Die DB-Zugriffe werden mit Hilfe der Stored-Procedures besser gekapselt und somit der Wiederverwertungsgrad und die Wartungsfreundlichkeit gesteigert. Ein weiterer Pluspunkt ist, dass alle solchen DB-Prozeduren implizit innerhalb einer eigenen Transaktion ausgeführt werden, was bei Mehrbenutzersystemen wie diesem besonders wichtig ist (Konsistenz der Daten).

Folgende **SQL-Prozeduren** wurden für das **Backoffice** des Online-Shops entwickelt:

| | |
|---|---|
| Analyse_Bestellungen | Konfig_Texte_Sprachen |
| Analyse_Bester_Kunde | Konfig_Texte_Sprachnr |
| Analyse_Bester_Tag | Konfig_Texte_Text |
| Analyse_Bestes_Land | Konfig_Texte_Textnr |
| Analyse_Bestes_Produkt | Konfig_Texte_Uebersicht |
| Analyse_Download | Konfig_Texte_Update |
| Analyse_Download_Kunde | Konfig_Versandkosten_Aendern_Anzeigen |
| Analyse_Kreditkarten | Konfig_Versandkosten_Uebersicht |
| Analyse_Lieferzeit | Konfig_Versandkosten_Update |
| Analyse_Produkt_Anzeige | Kunde_Select |
| Analyse_Umsatz | Logistik_Kreditkartenabr |
| Analyse_Umsatz_Details | Logistik_Kreditkartenabr_Kartennamen |
| Analyse_Umsatz_Kunde | Logistik_Onlineabrechnung |
| Analyse_Warenkorb | Logistik_Onlineabrechnung_Rechnung |
| Analyse_Warenkorb_Kunde | Logistik_Rechnungssuche |
| Analyse_Warenkorb_Verkauft | Rechnung_Detail |
| Write_Produkt | Umrechnung_Read |
| Benutzerverwaltung_Benutzer | Umrechnung_Read_Waehrungen |
| Konfig_Land_Waehrung_Aendern_Anzeigen | Umrechnung_Update |
| Konfig_Land_Waehrung_Insert | Umrechnung_Write |
| Konfig_Land_Waehrung_Uebersicht | Vertrieb_Bestellungen |
| Konfig_Land_Waehrung_Update | Vertrieb_Bestellungen_Bearbeitung |
| Konfig_Shop_Sprachen_Aendern_Anzeigen | Vertrieb_Lieferschein |
| Konfig_Shop_Sprachen_Insert | Vertrieb_Lieferschein_Update |
| Konfig_Shop_Sprachen_Uebersicht | Vertrieb_Lieferschein_Zurueck |
| Konfig_Shop_Sprachen_Update | Vertrieb_Rechnung |
| Konfig_Texte_Insert | |

Tabelle 8: Backoffice - SQL-Prozeduren

Weiters wurden auch diverse **SQL-Prozeduren** für die **Shop-Website** implementiert:

| | |
|---|---|
| Bestellung_Write | Warenkorb_Produktwahl |
| Download_Read | Warenkorb_Read |
| Kunde_Update_PaymentOffline | Warenkorb_Update |
| Kunde_Update_PaymentOnline | Warenkorb_Versand |
| Produktanzeige_Log_Write | Warenkorb_Write |
| Seite_Text_Read | Write_Download_Log |
| Warenkorb_Bestellen | Write_Kunde |
| Warenkorb_Delete | |

Tabelle 9: Shop-Website - SQL-Prozeduren

Einige der integrierten DB-Zugriffe (direkte SQL-Kommandos & Stored-Procedures) werden in den Abschnitten 5.4 und 5.5 in Verbindung mit den dort ausgesuchten ASP-Scripts gezeigt. Eine Beschreibung aller SQL-Prozeduren ist in der Diplomarbeit von Georg Kotek enthalten!

## 5.3 Sitemap

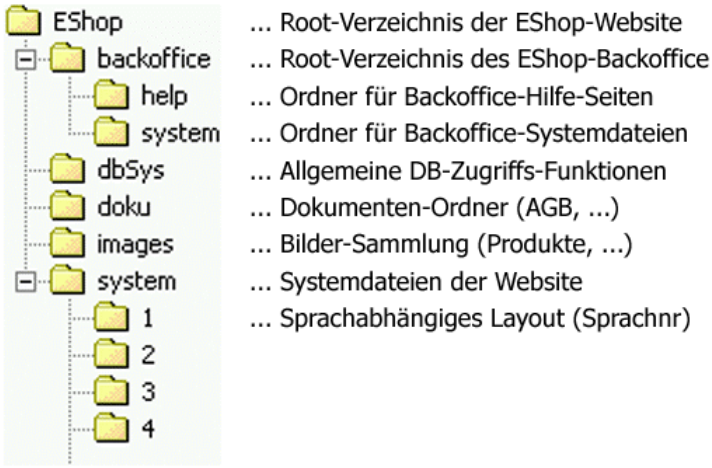



Abbildung 15: Interne Verzeichnisstruktur des Online-Shop-Systems

### EShop

| | |
|---|---|
| default.asp | ... **Shop-Website-Startskript** (siehe 5.5.1) |
| home.asp | ... Begrüßungsseite |
| anmelden.asp | ... Anmeldung von bereits bekannten Kunden |
| benutzer.asp | ... Neuanmeldung und Kundendatenänderungen |
| produkte.asp | ... Produktgruppen > Produkte > Details |
| suche.asp | ... Produktsuche per Suchtext > Details |
| warenkorb.asp | ... Warenkorb (für angemeldete Kunden!) |
| bestellen.asp | ... Bestellung der Produkte im Warenkorb |
| download.asp | ... Download der bereits gekauften Produkte |
| support.asp | ... Support-Seite (Links, ...) |
| kontakt.asp | ... Kontakt-Seite (Firma, ...) |
| info.asp | ... Informations-Seite (AGBs, ...) |
| definfo.asp | ... Copyright-Vermerk am Fuße der Webseiten |
| FatalError.asp | ... Fehleranzeige bei Datenbankproblemen |

**EShop/backoffice**

| | |
|---|---|
| default.asp | ... **Backoffice-Startskript** (siehe 5.4.1) |
| produkte.asp | ... Stammdatenverwaltung: Produkte |
| kunden.asp | ... Stammdatenverwaltung: Kunden |
| logistik.asp | ... Diverse Logistikfunktionen |
| Logistik_Kreditkartenabr.asp | |
| Logistik_Onlineabrechng.asp | |
| Logistik_Rechnungssuche.asp | |
| Rechnung.asp | |
| vertrieb.asp | |
| Vertrieb_Lieferschein.asp | |
| analysen.asp | ... Diverse Analysefunktionen |
| Analyse_Bestellung.asp | |
| Analyse_Bester_Kunde.asp | |
| Analyse_Bester_Tag.asp | |
| Analyse_Bestes_Land.asp | |
| Analyse_Bestes_Produkt.asp | |
| Analyse_Download.asp | |
| Analyse_Download_Kunde.asp | |
| Analyse_Kreditkarten.asp | |
| Analyse_Lieferzeit.asp | |
| Analyse_Produkt_Anzeige.asp | |
| Analyse_Umsatz.asp | |
| Analyse_Umsatz_Details.asp | |
| Analyse_Umsatz_Kunde.asp | |
| Analyse_Warenkorb.asp | |
| Analyse_Warenkorb_Kunde.asp | |
| konfig.asp | ... Shop-Konfigurationsmodule |
| Konfig_Benutzer.asp | |
| Konfig_Land_Waehrung.asp | |
| Konfig_Shop_Gruppen.asp | |
| Konfig_Shop_Sprachen.asp | |
| Konfig_Texte.asp | |
| Konfig_Umrechnung.asp | |
| Konfig_Versandkosten.asp | |
| hilfe.asp | ... Backoffice-Hilfeseiten |
| FatalError.asp | ... Anzeige eines internen Fehlers |

## 5.4 Backoffice

Dieses Kapitel beschäftigt sich nun etwas intensiver mit der **Backoffice-Implementierung** des Online-Shops. Dabei wird nur eine ausgesuchte Untermenge der Gesamtentwicklung näher erläutert, um einen Einblick in die Funktionsweise des Systems zu geben, ohne sich zu sehr in Details zu verlieren. Zum weiterem Studium des komplett funktionsfähigen **EShop**s liegen alle Sourcen gepackt im „Projekte“-Ordner auf der beigelegten CD.

### 5.4.1 System

Da das Backoffice aus diversen einzelnen Shop-Wartungsmodulen, wie Produkte, Kunden, Logistik, Analysen, Konfiguration und Hilfe, besteht, wurden gemeinsam benötigte Funktionen in systemspezifische **ASP-Scripts** ausgelagert (welche durch den ASP-Befehl `#include` in die einzelnen ASP-Seiten der Wartungsmodule eingebettet werden können). Dadurch wird die Modularität und Wartbarkeit des Online-Shops erheblich gesteigert.

Dabei stellt die Datei `dbV2.asp` (welche ihrerseits das Skript `dbConnection.asp` verwendet) die wichtigsten **Basisfunktionen** zum **Datenbankzugriff** zur Verfügung. Diese Funktionen sind so universell ausgeführt, dass sie nicht nur für das Backoffice sondern auch von den Shop-Website-Skripts (siehe Kapitel 5.5) genutzt werden können.

**EShop/dbSys/dbConnection.asp**

- Hier werden der **ADO-Connection-String** zum SQL-Server ( MS-Datenlink zur Shop-DB: `install | eshop_dev` ) **und einige Voreinstellungen** definiert.

  Dabei beziehen sich die `Landnr, Waehrungsnr, Sprachnr, unknownuser` ( `=kundennr` im Produktanzeige_Log falls Kunde / Client nicht bekannt ist ) auf Einträge in der Shop-DB.

```
<%
 ' ============================================================================
 ' # Datenbankverbindung
 ' ----------------------------------------------------------------------------
 dbConnection="Provider=SQLOLEDB.1;Integrated Security=SSPI;" & _
              "Persist Security Info=False;User ID=sa;"       & _
              "Initial Catalog=install;Data Source=localhost"
 ' ============================================================================

 ' ============================================================================
 ' # Konstanten definieren
 ' ----------------------------------------------------------------------------
 Landnr=1
 Land="Österreich"
 Waehrungsnr=1
 Waehrung="ATS"
 Sprachnr=2
 Sprache="Deutsch"
 DBError="Datenbankfehler!"
 DateError="Fehler bei der Datumseingabe festgestellt!"
 InputError="Fehlerhafte Eingabe"
 NoResults="Es wurden keine Ergebnisse gefunden!"
 unknownuser=1
 ' ============================================================================
%>
```

**EShop/dbSys/dbV2.asp**

- Die nun folgende **Sammlung an ASP-Funktionen** dient im Allgemeinen der Abstraktion zwischen Business-Logik und Datenbank (bzw. ADO).

  Am Anfang der Backoffice-Entwicklung wurde die DB direkt über ADO angesprochen. Nur folgende 2 Funktionen zum korrekten Auf- bzw. Abbau einer Datenbankverbindung wurden global zur Verfügung gestellt:

  ```
  db_open(…)      ... Datenverbindung herstellen
  db_close(…)     ... Datenverbindung beenden
  ```

  Einige „frühe“ Shop-Scripts benutzen diese 1ste Version des DB-Zugriffs.

```
<!--#include file="dbConnection.asp"-->

<%
 ' ============================================================================
 ON ERROR RESUME NEXT
 ' ============================================================================

 ' ============================================================================
 ' # Function: Datenbankverbindung herstellen (Version 1)
 ' # Vars: IN:
 ' #       OUT: nameConn, kommando, RS
 ' ----------------------------------------------------------------------------
 sub db_open (nameConn, kommando, RS)
  SET nameConn = Server.CreateObject("ADODB.Connection")
  SET kommando = Server.CreateObject("ADODB.Command")
  SET RS = Server.CreateObject("ADODB.RecordSet")
  nameConn.Open dbConnection
  SET kommando.ActiveConnection = nameConn
 end sub
 ' ============================================================================
 ' # Function: Datenbankverbindung beenden (Version 1)
 ' # Vars: IN: nameConn, RS
 ' #       OUT:
 ' ----------------------------------------------------------------------------
 sub db_close (nameConn, RS)
  SET RS=nothing
  nameConn.Close
  SET nameConn=nothing
 end sub
 ' ============================================================================

```

- In der weiter fortgeschrittenen Entwicklungsphase des Online-Shops wurden dann allerdings neue, leistungsstärkere **Datenbank-Zugriffsfunktionen** eingeführt:

```
opendb(…)          ... Datenverbindung herstellen
closedb(…)         ... Datenverbindung beenden
runsql(…)          ... SQL-Befehl starten
endsql(…)          ... SQL-Befehl beenden
runproc(…)         ... SQL-Prozedur starten
endproc(…)         ... SQL-Prozedur beenden
```

  Durch diese 2te Version der Zugriffsroutinen ist es wesentlich einfacher, sowohl einzelne **SQL-Kommandos** wie auch **Stored-Procedures** abzusetzen.

```
 ' ============================================================================
 ' # Function: Datenbankverbindung herstellen (Version 2)
 ' # Vars: IN:
 ' #       OUT: db
 ' ----------------------------------------------------------------------------
 sub opendb(db)
  SET db=Server.CreateObject("ADODB.Connection")
  db.Open dbConnection
 end sub
 ' ============================================================================
 ' # Function: Datenbankverbindung beenden (Version 2)
 ' # Vars: IN: db
 ' #       OUT:
 ' ----------------------------------------------------------------------------
 sub closedb(db)
  db.Close
  SET db=Nothing
 end sub
```

```
50  ' ========================================================================
51  ' # Function: SQL-Befehl starten (V2) (Auto-CheckForError)
52  ' # Vars: IN: sql
53  ' #       OUT: db, rs
54  ' ------------------------------------------------------------------------
55  sub runsql(db,rs,sql)
56   opendb db
57   SET rs=db.Execute(sql)
58   CheckForError
59  end sub
60  ' ========================================================================
61  ' # Function: SQL-Befehl beenden (V2)
62  ' # Vars: IN: db, rs
63  ' #       OUT:
64  ' ------------------------------------------------------------------------
65  sub endsql(db,rs)
66   SET rs=Nothing
67   closedb db
68  end sub
69  ' ========================================================================
70  ' # Function: SQL-Prozedur starten (V2)
71  ' # Vars: IN:
72  ' #       OUT: db, RS, command
73  ' ------------------------------------------------------------------------
74  sub runproc(db,RS,command)
75   opendb db
76   SET command = Server.CreateObject("ADODB.Command")
77   SET RS = Server.CreateObject("ADODB.RecordSet")
78   SET command.ActiveConnection = db
79   command.CommandType = &H0004 'Stored Procedure
80  end sub
81  ' ========================================================================
82  ' # Function: SQL-Prozedur beenden (V2)
83  ' # Vars: IN: db, RS
84  ' #       OUT:
85  ' ------------------------------------------------------------------------
86  sub endproc(db,RS)
87   SET RS=nothing
88   closedb db
89  end sub
90  ' ========================================================================
91
```

- Zusätzlich wurden auch noch oft benötigte **Konvertierungsfunktionen** integriert, um die Kommunikation mit der SQL-DB und einem HTML / JS - Client zu vereinfachen:

```
myCond(…)          ... Texte  für SQL-Abfragen konvertieren
myNum(…)           ... Zahlen für SQL-Abfragen konvertieren
mySQLDate(…)       ... Datum  für SQL-Abfragen konvertieren
mydate(…)          ... Datum  für HTML-Ausgabe konvertieren
myHTML(…)          ... Zeichenkette für  HTML  encodieren
myURL(…)           ... Zeichenkette für  URLs  encodieren
myJS(…)            ... Zeichenkette für Jscript umwandeln
```

```
92  ' ========================================================================
93  ' # Function: String-Konvertierung fuer SQL-Conditions
94  ' # Vars: IN: s, len
95  ' #       OUT:
96  ' # RetValue: s ohne ' Zeichen und auf len gekürzt
97  ' ------------------------------------------------------------------------
98  function myCond(s,len)
99   IF len=0 THEN  myCond=Replace(s,"'","")
100  IF len<>0 THEN myCond=left(Replace(s,"'",""),len)
101 end function
```

```
102  ' ========================================================================
103  ' # Function: Numeric-Konvertierung fuer SQL-Conditions
104  ' # Vars: IN: s
105  ' #       OUT:
106  ' # RetValue: s mit einheitlichen . Komma
107  ' ------------------------------------------------------------------------
108  function myNum(s)
109   myNum=Replace(s,",",".")
110  end function
111  ' ========================================================================
112  ' # Function: Datum-SQL-Konvertierung
113  ' # Vars: IN: d
114  ' #       OUT:
115  ' # RetValue: d auf M.D.Y konvertiert (wenn moeglich, sonst NULL)
116  ' ------------------------------------------------------------------------
117  function mySQLdate(d)
118   mySQLdate="NULL"
119   IF isDate(d) THEN mySQLdate="'"&Month(d)&"."&Day(d)&"."&Year(d)&"'"
120  end function
121  ' ========================================================================
122  ' # Function: Datum-HTML-Konvertierung
123  ' # Vars: IN: d
124  ' #       OUT:
125  ' # RetValue: d auf D.M.Y konvertiert (wenn moeglich, sonst "")
126  ' ------------------------------------------------------------------------
127  function mydate(d)
128   mydate=""
129   IF isDate(d) THEN mydate=Day(d)&"."&Month(d)&"."&Year(d)
130  end function
131  ' ========================================================================
132  ' # Function: String-HTML-Konvertierung
133  ' # Vars: IN: h
134  ' #       OUT:
135  ' # RetValue: h HTML-Encoded
136  ' ------------------------------------------------------------------------
137  function myHTML(h)
138   myHTML=Server.HTMLEncode(h & "")
139  end function
140  ' ========================================================================
141  ' # Function: String-URL-Konvertierung
142  ' # Vars: IN: u
143  ' #       OUT:
144  ' # RetValue: u URL-Encoded
145  ' ------------------------------------------------------------------------
146  function myURL(u)
147   myURL=Server.URLEncode(u & "")
148  end function
149  ' ========================================================================
150  ' # Function: Javascript-String-Konvertierung
151  ' # Vars: IN: j
152  ' #       OUT:
153  ' # RetValue: j mit LF+CR auf \n konvertiert
154  ' ------------------------------------------------------------------------
155  function myJS(j)
156   myJS=Replace(Replace(j,Chr(13),"\n"),Chr(10),"")
157  end function
158  ' ========================================================================
159
```

- Um die multilingualen Texte der Shop-Website leichter anzeigen zu können, gibt es auch noch eine „**Übersetzungsroutine**“ ( `getText(…)` ), die mit Hilfe der SQL-Prozedur[11] `Seite_Text_Read` sehr effizient abgearbeitet wird.

[11] siehe im Anschluss an dieses ASP-Skript

- Am Ende des ASP-Scripts wurde noch eine **Fehlerbehandlungsfunktion** ( `CheckForError(…)` ) integriert, welche z.B. im Falle eines Datenbank-Fehlers den Web-Browser auf die speziell dafür vorgesehene Web-Seite `FatalError.asp` umlenkt.

```
160  ' ========================================================================
161  ' # Function: Hole sprachabhängige Textzeile aus DB
162  ' #       IN: seitennr (global), sprachnr (global), textnr
163  ' #      OUT:
164  ' # RetValue: Textzeile HTML-Encoded (<br>,<b>,</b> werden belassen)
165  ' #           (Falls Text in sprachnr undefiniert ist, wird GTlangnr verwendet)
166  ' # Constant: GTlangnr, GTlandnr
167  ' ------------------------------------------------------------------------
168  GTlangnr=Sprachnr
169  GTlandnr=Landnr
170  ' ------------------------------------------------------------------------
171  function GetText(textnr)
172   runproc GTdb,GTrs,GTcmd
173    GTcmd.CommandText="Seite_Text_Read"
174    GTcmd.Parameters.Append GTcmd.CreateParameter("RETURN",3,&H0004)
175    GTcmd.Parameters.Append GTcmd.CreateParameter("@seitennr",3,1,4,seitennr)
176    GTcmd.Parameters.Append GTcmd.CreateParameter("@sprachnr",3,1,4,sprachnr)
177    GTcmd.Parameters.Append GTcmd.CreateParameter("@sprachnr",3,1,4,GTlangnr)
178    GTcmd.Parameters.Append GTcmd.CreateParameter("@sprachnr",3,1,4,textnr)
179    SET GTrs=GTcmd.Execute()
180    CheckForError
181    IF NOT GTrs.EOF THEN
182        GetText=myHTML(GTrs("Text"))
183     GetText=Replace(GetText,"&lt;br&gt;","<br>")
184     GetText=Replace(GetText,"&lt;b&gt;" ,"<b> ")
185     GetText=Replace(GetText,"&lt;/b&gt;","</b>")
186    END IF
187   endproc GTdb,GTrs
188  end function
189  ' ========================================================================
190
191  ' ========================================================================
192  ' # Function: Prüft ob in ASP-Errorobjekt ein Fehler registriert ist
193  ' #           Tritt ein Fehler auf, wird auf die sprachabhängige Seite
194  ' #           FatalError.asp umgeleitet.
195  ' #       IN: sprachnr (global)
196  ' #      OUT:
197  ' ------------------------------------------------------------------------
198  sub CheckForError()
199   IF err THEN
200    err.Clear
201    %>
202    <script>
203     document.location.href='FatalError.asp?Sprachnr=<%=sprachnr%>&Err=<%=err%>';
204    </script>
205    <%
206   END IF
207  end sub
208  ' ========================================================================
209  %>
```

**Stored-Procedure: Seite_Text_Read**

- Diese SQL-Prozedur **liefert** den **sprachbezogenen Text** (`@sprachnr`) für eine Textstelle (`@textnr`) einer spezifischen Web-Seite (`@seitennr`).

Existiert in der DB der verlangte Text nicht in der gewünschten Sprache, wird stattdessen die Standardsprache (`@standard`) verwendet.

```
1   CREATE PROCEDURE Seite_Text_Read (
2   @seitennr INT,
3   @sprachnr INT,
4   @standard INT,
5   @textnr INT) AS
6   BEGIN
7
8     IF NOT EXISTS (SELECT * FROM Seite_Text AS ST WHERE ST.Seitennr=@seitennr AND
      ST.Sprachnr=@sprachnr AND ST.Textnr=@textnr)
9       SELECT * FROM Seite_Text AS ST WHERE ST.Seitennr=@seitennr AND
        ST.Sprachnr=@standard AND ST.Textnr=@textnr
10    ELSE
11      SELECT * FROM Seite_Text AS ST WHERE ST.Seitennr=@seitennr AND
        ST.Sprachnr=@sprachnr AND ST.Textnr=@textnr
12  END
13
14  GO
```

Die **Zugriffskontrolle** (nur Shop-Personal darf Zugriff auf das eigene Backoffice haben) und die **Navigation** innerhalb der einzelnen Wartungsmodule stellt ein weiteres zentrales Thema dar, welches durch das nachfolgende System-Script zentral gelöst wurde.

Da das gesamte Backoffice ohne HTML-Frames auskommen sollte, aber doch alle Module dieselbe Navigationsleiste am oberen Rand und einen unten platzierten Standardbalken besitzen sollten, wurden Routinen geschaffen, die **auf allen Backoffice-Seiten** gleich eingebunden und verwendet werden können (siehe 5.4.2 und 5.4.3).

### EShop/backoffice/system/navi.asp

- Die erste **ASP-Funktion** dient der **Zugriffskontrolle** für alle Wartungsmodule. Wird eine Seite angefordert, ohne dass sich zuvor ein Benutzer angemeldet hat und somit die ASP-**Sessionvariable** `"user"` korrekt initialisiert ist, wird man auf das Anmeldeformular (`default.asp`) umgeleitet.

```
1   <%
2   ' ============================================================================
3   ' # Berechtigungskontrolle für Backoffice (Statt <html> verwenden!!)
4   ' ----------------------------------------------------------------------------
5    sub html()
6     Response.Expires=0
7     IF session("user")="" THEN Response.Redirect "default.asp"
8   %><html>
9   <meta http-equiv="Content-Script-Type" content="text/javascript"><%
10   end sub
11  ' ============================================================================
12
```

- Die zweite **ASP-Funktion** dient zur Erzeugung der **Navigationsleiste**. Wird dabei `"anmelden" als` Funktionsparameter mitgegeben, resultiert dies in der

Ausgabe des folgenden **Seiten-Kopf**s:


Bei anderen Eingabewerten (siehe Listing Zeile 15) sieht die Ausgabe dann z.B. folgendermaßen aus :


Dabei verschiebt sich je nach dem mitgegebenen Funktionswert der hellmarkierte Kreis rund um das aktuelle Menüsymbol inklusive des fettgeschriebenen Menütextes.

Zusätzlich prüft diese ASP-Routine noch, ob der angemeldete Backoffice-Benutzer überhaupt die Berechtigung besitzt diesen Menüpunkt anzuwählen. Dabei wird das **Berechtigungsstufe**-Feld der **Benutzer**-Tabelle mit der ASP-**Sessionvariable** `"level"` verglichen. Folgendes Schema kommt dabei zum Einsatz:

```
varchar:  "0"|"1"   "0"|"1"   "0"|"1"    "0"|"1"      "0"|"1"     "0"|"1"
→ Modul: Produkte Kunden Logistik Analysen Konfiguration Hilfe
```

Besteht keine Berechtigung, wird die Seitenausgabe automatisch mit der Ausgabe einer Fehlermeldung beendet.

```
13  ' ================================================================================
14  ' # Navigationsleiste für Backoffice (Statt <body> verwenden!!)
15  ' # (menu) = anmelden|willkommen|produkte|kunden|logistik|analysen|konfig|hilfe
16  ' --------------------------------------------------------------------------------
17   sub body(menu)
18    prvLevel=session("level")
19    prvError=0
20    IF menu="anmelden" THEN
21     prvS="<td background='system/oben.gif' width=384>" & _
22          "<a href='javascript:document.login.submit();'>" & _
23          "<img src='system/anmelden.gif' width=64 height=50 border=0></a></td>"
24    ELSE
25     prvS="<td><a href='produkte.asp'"
26     IF menu="produkte" THEN
27      IF mid(prvLevel,1,1)="0" THEN prvError=1
28      prvS=prvS+"><img src='system/produkte2.gif'"
29     ELSE
30      prvS=prvS+" onmouseover='change(4,1);' onmouseout='change(4,0);'>" & _
31                "<img src='system/produkte0.gif'"
32     END IF
33     prvS=prvS+"width=64 height=50 border=0 alt='Produkte'></a></td>"
34     prvS=prvS+"<td><a href='kunden.asp'"
35     IF menu="kunden" THEN
36      IF mid(prvLevel,2,1)="0" THEN prvError=1
37      prvS=prvS+"><img src='system/kunden2.gif'"
38     ELSE
39      prvS=prvS+" onmouseover='change(5,3);' onmouseout='change(5,2);'>" & _
40                "<img src='system/kunden0.gif'"
```

```
41      END IF
42      prvS=prvS+"width=64 height=50 border=0 alt='Kunden'></a></td>"
43      prvS=prvS+"<td><a href='logistik.asp'"
44      IF menu="logistik" THEN
45       IF mid(prvLevel,3,1)="0" THEN prvError=1
46       prvS=prvS+"><img src='system/logistik2.gif'"
47      ELSE
48       prvS=prvS+" onmouseover='change(6,5);' onmouseout='change(6,4);'>" & _
49                 "<img src='system/logistik0.gif'"
50      END IF
51      prvS=prvS+"width=64 height=50 border=0 alt='Logistik'></a></td>"
52      prvS=prvS+"<td><a href='analysen.asp'"
53      IF menu="analysen" THEN
54       IF mid(prvLevel,4,1)="0" THEN prvError=1
55       prvS=prvS+"><img src='system/analysen2.gif'"
56      ELSE
57       prvS=prvS+" onmouseover='change(7,7);' onmouseout='change(7,6);'>" & _
58                 "<img src='system/analysen0.gif'"
59      END IF
60      prvS=prvS+"width=64 height=50 border=0 alt='Analysen'></a></td>"
61      prvS=prvS+"<td><a href='konfig.asp'"
62      IF menu="konfig" THEN
63       IF mid(prvLevel,5,1)="0" THEN prvError=1
64       prvS=prvS+"><img src='system/konfig2.gif'"
65      ELSE
66       prvS=prvS+" onmouseover='change(8,9);' onmouseout='change(8,8);'>" & _
67                 "<img src='system/konfig0.gif'"
68      END IF
69      prvS=prvS+"width=64 height=50 border=0 alt='Konfiguration'></a></td>"
70      prvS=prvS+"<td><a href='hilfe.asp'"
71      IF menu="hilfe" THEN
72       IF mid(prvLevel,6,1)="0" THEN prvError=1
73       prvS=prvS+"><img src='system/hilfe2.gif'"
74      ELSE
75       prvS=prvS+" onmouseover='change(9,11);' onmouseout='change(9,10);'>" & _
76                 "<img src='system/hilfe0.gif'"
77      END IF
78      prvS=prvS+"width=64 height=50 border=0 alt='Hilfe'></a></td>"
79     END IF
80    %><body text="#000000" bgcolor="#FFFFFF"
81     link="#000000" vlink="#000000" alink="#000000">
82    <script>
83     var item = new Array();
84     item[ 0]=new Image(); item[ 0].src="system/produkte0.gif";
85     item[ 1]=new Image(); item[ 1].src="system/produkte1.gif";
86     item[ 2]=new Image(); item[ 2].src="system/kunden0.gif";
87     item[ 3]=new Image(); item[ 3].src="system/kunden1.gif";
88     item[ 4]=new Image(); item[ 4].src="system/logistik0.gif";
89     item[ 5]=new Image(); item[ 5].src="system/logistik1.gif";
90     item[ 6]=new Image(); item[ 6].src="system/analysen0.gif";
91     item[ 7]=new Image(); item[ 7].src="system/analysen1.gif";
92     item[ 8]=new Image(); item[ 8].src="system/konfig0.gif";
93     item[ 9]=new Image(); item[ 9].src="system/konfig1.gif";
94     item[10]=new Image(); item[10].src="system/hilfe0.gif";
95     item[11]=new Image(); item[11].src="system/hilfe1.gif";
96     function change(x,s) {document.images[x].src=item[s].src;}
97    </script>
98    <table border=0 cellspacing=0 cellpadding=0 width=680 align=center>
99     <tr>
100     <td><img src="system/links0.gif" width=8 height=50 border=0></td>
101     <td><img src="system/backoffice.gif" width=256 height=50 border=0></td>
102     <td><img src="system/rechts0.gif" width=8 height=50 border=0></td>
103     <td width=8></td>
104     <td><img src="system/links1.gif" width=8 height=50 border=0></td>
105     <%=prvS%>
106     <td><img src="system/rechts1.gif" width=8 height=50 border=0></td>
107    </tr>
108   </table><br>
```

```
109
110  <table border=0 cellspacing=0 cellpadding=0 width=640 align=center>
111  <tr align=center><td><%
112    IF prvError<>0 THEN%>
113  <h1>Keine Berechtigung</h1>
114   Fragen Sie Ihren System-Administrator
115   um eine Berechtigung f&uuml;r diesen Bereich zu erhalten.
116  <br><br>
117  <% endbody("FEHLER: Keine Berechtigung!!!")%>
118  </html><%
119     Response.End
120    END IF
121   end sub
122  ' ========================================================================
123
```

- Beendet wird jede Backoffice-Seite mit dieser **ASP-Funktion**. Sie gibt einen kurzen **Statusbalken** mit einer eventuellen Fehlermeldung aus: z.B.


```
124  ' ========================================================================
125  ' # Schlußzeile für Backoffice (Statt </body> verwenden!!)
126  ' # (error) = error | "" | "OKAY!!!"
127  ' ------------------------------------------------------------------------
128   sub endbody(error)
129    IF session("user")<>"" THEN
130     prvS1="Benutzer: "+session("user")
131     prvS2="<a href='default.asp'><img src='system/abmelden.gif'" & _
132           " width=54 height=12 border=0 alt='Abmelden'></a>"
133    ELSE
134     prvS1="Anmeldung"
135     prvS2=" "
136    END IF
137  %></td></tr>
138  </table>
139
140  <table border=0 cellspacing=0 cellpadding=0 align=center width=680>
141   <tr bgcolor=#bcddda>
142    <td><small> </small></td>
143    <td><small><%=prvS1%></small></td>
144    <td align=center><small><b>
145      <%=Server.HTMLEncode(error & "")%> 
146     </b></saml></td>
147    <td align=right valign=bottom><small><%=prvS2%></small></td>
148    <td><small> </small></td>
149   </tr><tr>
150    <td><img src="system/links2.gif" width=8 height=8 border=0></td>
151    <td background="system/unten.gif" width=200>
152     <img src="system/unten.gif" width=8 height=8 border=0></td>
153    <td background="system/unten.gif" width=464>
154     <img src="system/unten.gif" width=8 height=8 border=0></td>
155    <td background="system/unten.gif" width=200>
156     <img src="system/unten.gif" width=8 height=8 border=0></td>
157    <td><img src="system/rechts2.gif" width=8 height=8 border=0></td>
158   </tr>
159  </table>
160  <% IF (error<>"" AND error<>"OKAY!!!") THEN %>
161  <script>window.alert("<%=error%>");</script>
162  <% END IF %>
163  </body><%
164   end sub
165  ' ========================================================================
166  %>
```

Nachdem nun alle ASP-Scripts aufgeführt wurden, welche diverse seitenübergreifende Funktionen für das gesamte System bereitstellen, folgt nun das **Startskript** und zugleich **Anmeldeformular** für den **Backoffice**-Bereich.

Diese ASP-Seite führt außerdem auch beispielhaft die Einbettung (siehe nächstes Listing Zeile 1 und 2) sowie die Benutzung einiger der zuvor genannten systembezogenen Funktionen vor, z.B.: prüft die DB-SQL-Abfrage in Zeile 13-18 des Scripts, ob eine angegebene Person dem System überhaupt bekannt ist.

Als Backoffice-Benutzer erhält man nun durch den Aufruf dieser ASP-Seite (mittels der Eingabe der Web-Adresse des Online-Shops gefolgt von der Zeichenkette `/backoffice` ) die Möglichkeit, sich mit seinem **Benutzernamen und Kennwort** (z.B.: admin / admin) Zugang zur Shop-Verwaltung zu beschaffen.

Dabei sieht der **Ablauf** für den Anwender folgendermaßen aus:

| | | | | |
|---|---|---|---|---|
| **(1) Anmeldeformular** | → | *Anmelden* ⇒ Eingabe akzeptiert | → | **(2) Willkommen** |
| | → | *Anmelden* ⇒ Eingabe inkorrekt | → | (1) |

**EShop/backoffice/default.asp**

```
1   <!--#include file=  "system/navi.asp"-->
2   <!--#include file="../dbSys/dbV2.asp"-->
```

- Im ersten Schritt wird versucht, eventuelle Eingabedaten (Benutzer, Kennwort) eines **vorangegangene**n Anmeldeversuches einzulesen und zu kontrollieren. Wird diese **Anmeldung** akzeptiert, werden die ASP-**Sessionvariablen** `"user"` und `"level"` gesetzt und folgende Ausgabe erzeugt:


Abbildung 16: Backoffice - Willkommen

```
3   <%
4   ' ================================================================================
5   ' # Start-Default-Login-Seite
6   ' --------------------------------------------------------------------------------
7   ' # Login-Daten einlesen:
8    user=myCond(Request.Form("user"),20)
9    pwd=myCond(Request.Form("pwd"),20)
10   try=Request.Form("try")
11   IF try="" THEN try=0
12  ' --------------------------------------------------------------------------------
13  ' # Login prüfen:
```

```
14    sql="SELECT count(*) AS 'Anzahl' FROM Benutzer WHERE" & _
15        " Benutzername='"&user&"' and Passwort='"&pwd&"'"
16    runsql db,rs,sql
17     count=rs("Anzahl")
18    endsql db,rs
19    ' # Login durchführen, falls Benutzername eingegeben und korrekt:
20    IF user<>"" and count=1 THEN
21     sql="SELECT Berechtigungsstufe FROM Benutzer WHERE" & _
22         " Benutzername='"&user&"' and Passwort='"&pwd&"'"
23     runsql db,rs,sql
24      level=rs("Berechtigungsstufe")
25     endsql db,rs
26     session("level")=level
27     session("user")=user
28      ' # Benutzer begruessen:
29   %><html>
30   <head>
31    <title>Willkommen</title>
32   </head>
33
34   <%body "willkommen"%>
35
36   <h1>Willkommen</h1>
37   <b>Benutzer:</b> <%=session("user")%><br><br>
38
39   <%endbody ""%>
40
41   </html>
```

- Wird dieses ASP-Skript **ohne POST-Parameter** (z.B. das erste Mal) aufgerufen, wird eine **leere Anmeldung** ausgegeben. Ist zur Zeit aber bereits ein Benutzer angemeldet, so wird durch diesen Aufruf auch die ASP-**Sessionvariable** `"user"` gelöscht und somit dieser **Benutzer abgemeldet**.

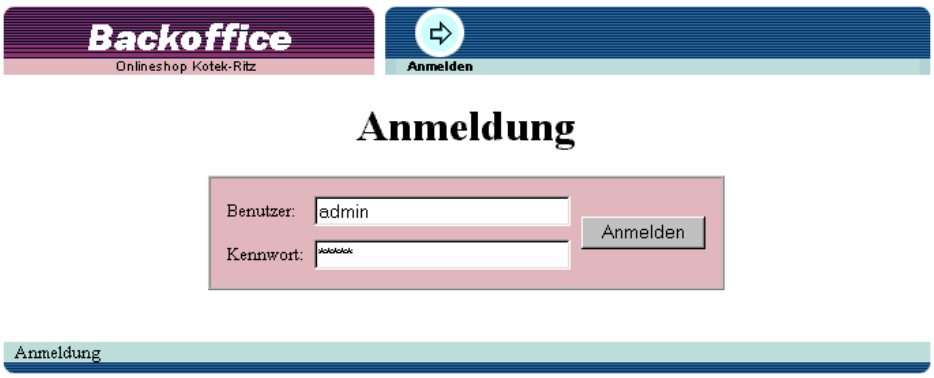



Abbildung 17: Backoffice - Anmeldung

```
42   <%
43   ' ------------------------------------------------------------------------
44    ELSE
45      ' # Login-Maske anzeigen:
46     session("user")=""
47   %><html>
48   <head>
49    <title>Login</title>
50   </head>
51
52   <%body "anmelden"%>
53
54   <form METHOD="POST" ACTION="default.asp" name="login">
```

```
55  <input type=hidden name="try" value=<%=try+1%>>
56  <h1>Anmeldung</h1>
57  <table border=1 cellspacing=0 cellpadding=8 align=center>
58   <tr bgcolor=#e1b6bd><td>
59    <table border=0 cellspacing=0 cellpadding= 4><tr>
60     <td><small>Benutzer:</small></td>
61     <td><input type=text name="user" size=25 tabindex=1></td>
62     <td rowspan=2><input type=submit value="Anmelden" tabindex=3></td>
63    </tr><tr>
64     <td><small>Kennwort:</small></td>
65     <td><input type=password name="pwd" size=25 tabindex=2></td>
66    </tr></table>
67   </td></tr>
68  </table>
69  <small> </small>
70  </form>
71
72  <%
73   status=""
74   IF try>0 THEN
75    status="FEHLER: Anmeldung fehlgeschlagen!"
76   END IF
77   endbody status%>
78
79  </html>
80  <%
81   END IF
82  ' ================================================================================
83  %>
```

Aufbauend auf diese System-Scripts wurden insgesamt **34 Backoffice-Scripts** zur Online-Shop-Wartung entwickelt. Davon werden im weiteren Verlauf dieser Arbeit zwei der mächtigsten Module genauer untersucht, um einen guten Einblick in die generelle Funktionsweise zu geben.

### 5.4.2 Produkte

Das erste Wartungsmodul dient zum **Anlegen** und zum **Suchen** von Shop-Produkten sowie zur **Pflege** aller im Online-System aktuell bzw. auch früher einmal angebotenen Artikeln.

Über die Produktwartung können selbstverständlich auch alle sprachabhängigen, artikelbezogenen Texte und länder-/währungsabhängigen Preise mitgepflegt werden.

Das Webinterface hält sich dabei an folgenden groben **Ablauf (** → Seiten-**Modi )**:

| | | | | |
|---|---|---|---|---|
| (①) **Such-/Anlegemaske** | → | *Suchen* | → | (②) |
| | → | *Anlegen* | → | (③) |
| … | | | | … |
| (②) **Suchergebnisliste** | → | *Zurück* | → | (①) |
| | → | < *Gehe zu Seite* > | → | (②) |
| | → | *Artikelnr aus Liste* | → | (③) |
| … | | | | … |
| (③) **Detailmaske** | → | *Zurück* | → | (①) |
| | → | *Speichern* | → | (③) |

**EShop/backoffice/produkte.asp**

```
1   <!--#include file="../dbSys/dbV2.asp"-->
2   <!--#include file=  "system/navi.asp"-->
```

- Am Anfang werden mögliche (GET- und POST-) Werte eines vergangenen Seitenaufrufes gesichert. Durch die **Variable** `"action"` wird dabei der aktuell angeforderte **Befehl** ( *suchen, anlegen,* ...) erkannt, bzw. der gültige Seiten-**Modus** ( Masken: suchen, detail, ...) definiert.
- Des weiteren werden im Anschluss auch spezielle seitenbezogene ASP-Funktionen definiert, welche diverse komplexere bzw. mehrmals benötigte Routinen vorweg kapseln. Dies wiederum steigert die Wartbarkeit des gesamten Programmcodes dieser ASP-Seite.

```
3   <%
4   ' ================================================================================
5   ' # Produktstammdaten-Seite
6   ' --------------------------------------------------------------------------------
7    html
8   %>
9   <head>
10   <title>Produkte</title>
11  </head>
12
13  <%body "produkte"%>
14
15  <h1>Produkte</h1>
16  <form METHOD="POST" ACTION="produkte.asp" name="produkte"><%
17  ' --------------------------------------------------------------------------------
18   status=""
19  ' --------------------------------------------------------------------------------
20  ' # Eingabe-Daten einlesen:
21   artnr=myCond(Request.QueryString("artnr"),20)    ' # GET- MESSAGE ?
22   IF artnr="" THEN
23    action=Request.Form("action")                   ' # POST-MESSAGE !
24    inaktiv=Request.Form("inaktiv")
25    artikelnr=myCond(Request.Form("artikelnr"),20)
26    sprachnr=myCond(Request.Form("sprachnr"),0)
27    artikelbez=myCond(Request.Form("artikelbez"),80)
28    gruppennr=myCond(Request.Form("gruppennr"),0)
29    landnr=myCond(Request.Form("landnr"),0)
30    preis=myCond(Request.Form("preis"),0)
31    IF action="detail" or action="aendern" THEN
32     artikelbesch=myCond(Request.Form("artikelbesch"),800)
33     bild=myCond(Request.Form("bild"),80)
34     link=myCond(Request.Form("link"),80)
35     downloadlink=myCond(Request.Form("downloadlink"),80)
36     datum=Request.Form("datum")
37    END IF
38   ELSE                                              ' # GET - MESSAGE !
39    action="detail"
40   END IF
41  ' --------------------------------------------------------------------------------
42  ' # Erzeuge sql-where-statement:
43   function where(expr,cmp)
44    prvC1="="
45    prvC2=""
46    IF cmp=1 THEN
47     prvC1=" LIKE "
48     prvC2="%"
```

```
49      END IF
50      prvS1=""
51      prvS2="WHERE "
52      IF artikelnr<>"" THEN
53       prvS1=prvS1 & prvS2 & "P.Artikelnr='" & artikelnr & "'"
54       prvS2=" AND "
55      END IF
56      IF sprachnr<>"" THEN
57       prvS1=prvS1 & prvS2 & "T.Sprachnr='" & sprachnr & "'"
58       prvS2=" AND "
59      END IF
60      IF artikelbez<>"" THEN
61       prvS1=prvS1 & prvS2 & "(T.Artikelbezeichnung" & prvC1 & _
62             "'" & artikelbez & prvC2 & "' OR P.Artikelname" & prvC1 & _
63             "'" & artikelbez & prvC2 & "')"
64       prvS2=" AND "
65      END IF
66      IF gruppennr<>"" THEN
67       prvS1=prvS1 & prvS2 & "P.Gruppennr='" & gruppennr & "'"
68       prvS2=" AND "
69      END IF
70      IF landnr<>"" THEN
71       prvS1=prvS1 & prvS2 & "M.Landnr='" & landnr & "'"
72       prvS2=" AND "
73      END IF
74      IF PREIS<>"" THEN
75       prvS1=prvS1 & prvS2 & "M.Preis=" & myNum(preis) & ""
76       prvS2=" AND "
77      END IF
78      IF inaktiv="" THEN
79       prvS1=prvS1 & prvS2 & "P.aktiv=1"
80       prvS2=" AND "
81      END IF
82      prvS1=prvS1 & prvS2 & "P.Artikelnr=T.Artikelnr AND P.Artikelnr=M.Artikelnr"
83      prvS2=" AND "
84      IF expr<>"" THEN
85       prvS1=prvS1 & prvS2 & expr
86      END IF
87      where=prvS1
88     end function
89    ' ------------------------------------------------------------------------------
90    ' # Baue <options> fuer frei definierbare Selektion:
91     sub options(optnr,sql)
92      %><option value=""></option><%
93      runsql prvDB,prvRS,sql
94       DO UNTIL prvRS.EOF
95        prvSel=""
96        IF optnr<>"" AND IsNumeric(optnr) THEN
97         IF CInt(optnr)=prvRS(0) THEN prvSel=" selected"
98        END IF
99        %><option value="<%=prvRS(0)%>"<%
100       %><%=prvSel&">"&myHTML(prvRS(1))%></option><%
101       prvRS.MoveNext
102      LOOP
103     endsql prvDB,prvRS
104    end sub
105   ' ------------------------------------------------------------------------------
106   ' # Baue <options> fuer Sprach-Selektion:
107    sub optsprache(sprachnr)
108     prvSQL="SELECT Sprachnr,Sprache FROM Sprache ORDER BY Sprache"
109     options sprachnr,prvSQL
110    end sub
111   ' ------------------------------------------------------------------------------
112   ' # Baue <options> fuer Produktgruppen-Selektion:
113    sub optgruppe(gruppennr)
114     prvSQL="SELECT Gruppennr,Gruppe FROM Produktgruppe ORDER BY Gruppe"
115     options gruppennr,prvSQL
116    end sub
```

```
117 ' ------------------------------------------------------------------------
118 ' # Baue <options> fuer Land-Selektion:
119  sub optland(landnr)
120   %><option value=""></option><%
121   prvSQL="SELECT Landnr,Land,Waehrung " & _
122          "FROM Land L, Waehrung W WHERE L.Waehrungsnr=W.Waehrungsnr " & _
123          "ORDER BY Land"
124   runsql prvDB,prvRS,prvSQL
125    DO UNTIL prvRS.EOF
126     prvSel=""
127     IF landnr<>"" AND IsNumeric(landnr) THEN
128      IF CInt(landnr)=prvRS(0) THEN prvSel=" selected"
129     END IF
130     %><option value="<%=prvRS(0)%>"<%
131     %><%=prvSel&">"&myHTML(prvRS(1))&" - "&myHTML(prvRS(2))%></option><%
132     prvRS.MoveNext
133    LOOP
134   endsql prvDB,prvRS
135  end sub
```

- Im folgenden Abschnitt wird ein eventuell empfangener **Befehl** (`action`) **bearbeitet** ( *anlegen, aendern, suchen* ). Dabei werden auch die zur jeweiligen Durchführung benötigten Eingabewerte auf Korrektheit und Vollständigkeit überprüft (z.B.: muss die Eingabe eines Produktpreises stets eine korrekte Zahl repräsentieren und darf nicht eine beliebige Zeichenkette sein).

  Generell wird zwischen dem Anlegen eines Produktes, der Änderung eines Artikels und der Suche nach einem oder mehreren Produkten unterschieden.

```
136 ' ------------------------------------------------------------------------
137 ' # Anlegen: Eingaben vollständig?
138  IF action="anlegen" and (artikelnr="" or sprachnr="" or artikelbez="" or _
139     gruppennr="" or landnr="" or preis="") THEN
140   status="FEHLER: Formular unvollständig ausgefüllt!!!"
141   action=""
142  END IF
143 ' ------------------------------------------------------------------------
144 ' # Anlegen durchfuehren:
145  IF action="anlegen" THEN
146   IF IsNumeric (preis) THEN
147    runproc db,RS,command
148    command.CommandText = "Write_Produkt"
149    command.Parameters.Append command.CreateParameter _
150     ("@Artikelnr", 200, 1, 20, artikelnr)
151    command.Parameters.Append command.CreateParameter _
152     ("@Artikelname", 200, 1, 80, artikelbez)
153    command.Parameters.Append command.CreateParameter _
154     ("@Gruppennr", 3, 1, 4, gruppennr)
155    ret="0"
156    SET RS=command.Execute()
157    CheckForError
158    ' # Erfolgreich angelegt?
159     IF NOT RS IS nothing THEN
160      SET RS=RS.NextRecordset()
161         IF NOT RS.EOF THEN ret=RS("ret")
162     END IF
163    endproc db,RS
164    IF ret="1" THEN
165     ' # Wenn Produkt angelegt wurde, fertigstellen und weiter auf Detailanzeige:
166     sql="INSERT INTO Produkt_Text (Artikelnr, Sprachnr, Artikelbezeichnung)" & _
167         " VALUES ('"&artikelnr&"','"&sprachnr&"','"&artikelbez&"')"
168     runsql db,rs,sql
169     endsql db,rs
```

```
170      sql="INSERT INTO Preis (Artikelnr, Preis, Landnr)" & _
171          " VALUES ('"&artikelnr&"',"&myNum(preis)&",'"&landnr&"')"
172      runsql db,rs,sql
173      endsql db,rs
174      artnr=artikelnr
175      action="detail"
176     ELSE
177      status="FEHLER: Artikel bereits vorhanden!!!"
178      action=""
179     END IF
180    ELSE
181     status="FEHLER: Fehlerhafte Eingabe!!!"
182     action=""
183    END IF
184   END IF
185  ' ------------------------------------------------------------------------------
186  ' # Produktdaten aendern: Eingaben korrekt?
187   IF action="aendern" THEN
188    IF gruppennr="" OR artikelbez="" OR _
189       (sprachnr="" AND (artikelbesch<>"" OR bild<>"")) OR _
190       (landnr<>"" AND preis="") OR _
191       (landnr="" AND preis<>"") THEN
192     status="FEHLER: Formular unvollständig ausgefüllt!!!"
193     action="detail"
194    END IF
195    IF (landnr<>"" AND preis<>"" AND NOT IsNumeric (preis)) THEN
196     status="FEHLER: Fehlerhafte Eingabe!!!"
197     action="detail"
198    END IF
199   END IF
200  ' ------------------------------------------------------------------------------
201  ' # Produktdaten aendern durchfuehren:
202   IF action="aendern" THEN
203    aktiv=1
204    IF inaktiv<>"" THEN aktiv=0
205    update=""
206    IF sprachnr="" THEN update="Artikelname='" & artikelbez & "', "
207    sql="UPDATE Produkt SET " & _
208     "Aktiv='" & aktiv & "', " & update & _
209     "Gruppennr='" & gruppennr & "', " & _
210     "Herstellerlink='" & link & "'," & _
211     "Downloadlink='" & downloadlink & "' " & _
212     "WHERE Artikelnr='" & artikelnr & "'"
213    runsql db,rs,sql
214    endsql db,rs
215    ' # Produktpreis aendern:
216    IF landnr<>"" THEN
217     sqlA="SELECT count(*) AS 'Anzahl' FROM Preis WHERE" & _
218          " Artikelnr='"&artikelnr&"' AND Landnr='"&landnr&"'"
219     sqlB="INSERT INTO Preis (Artikelnr, Preis, Landnr)" & _
220          " VALUES ('"&artikelnr&"',"&myNum(preis)&",'"&landnr&"')"
221     sqlC="UPDATE Preis SET Preis="&myNum(preis)&" WHERE" & _
222          " Artikelnr='"&artikelnr&"' AND Landnr='"&landnr&"'"
223     runsql db,rs,sqlA
224      anzahl=rs(0)
225     endsql db,rs
226     IF anzahl<>0 THEN sqlB=sqlC
227     runsql db,rs,sqlB
228     endsql db,rs
229    END IF
230    ' # Produkttext aendern:
231    IF sprachnr<>"" THEN
232     sqlA="SELECT count(*) AS 'Anzahl' FROM Produkt_Text WHERE" & _
233          " Artikelnr='"&artikelnr&"' AND Sprachnr='"&sprachnr&"'"
234     sqlB="INSERT INTO Produkt_Text (Artikelnr, Sprachnr," & _
235          " Artikelbezeichnung, Artikelbeschreibung, Bild)" & _
236          " VALUES ('"&artikelnr&"', '"&sprachnr&"'," & _
237          " '"&artikelbez&"', '"&artikelbesch&"', '"&bild&"')"
```

```
238     sqlC="UPDATE Produkt_Text SET Bild='"&bild&"'," & _
239          " Artikelbezeichnung='"&artikelbez&"'," & _
240          " Artikelbeschreibung='"&artikelbesch&"' WHERE " & _
241          "Artikelnr='"&artikelnr&"' AND Sprachnr='"&sprachnr&"'"
242     runsql db,rs,sqlA
243      anzahl=rs(0)
244     endsql db,rs
245     IF anzahl<>0 THEN sqlB=sqlC
246     runsql db,rs,sqlB
247     endsql db,rs
248    END IF
249    status="OKAY!!!"
250    artnr=artikelnr
251    action="detail"
252   END IF
253  ' ------------------------------------------------------------------------------
254  ' # Produkt suchen: Eingaben untersuchen!
255   IF action="suchen" THEN
256    IF preis<>"" and not IsNumeric(preis) THEN
257     status="FEHLER: Fehlerhafte Eingabe!!!"
258     action=""
259    ELSE
260     sql="SELECT count(DISTINCT P.Artikelnr) AS 'Anzahl' FROM" & _
261         " Produkt P, Produkt_Text T, Preis M " & where("",1)
262     runsql db,rs,sql
263      anzahl=rs(0)
264     endsql db,rs
265     IF anzahl=0 THEN
266      status="FEHLER: Keinen Eintrag gefunden!!!"
267      action=""
268     END IF
269     ' # Falls genau ein Produkt gefunden wird, sofort auf Detailanzeige gehen:
270     IF anzahl=1 THEN
271      sql="SELECT DISTINCT P.Artikelnr FROM" & _
272          " Produkt P, Produkt_Text T, Preis M " & where("",1)
273      runsql db,rs,sql
274       artnr=rs("Artikelnr")
275      endsql db,rs
276      action="detail"
277     END IF
278    END IF
279   END IF
```

- Die **nächsten 3 Code-Abschnitte** beherbergen die **Darstellungslogik** der einzelnen Seiten-**Modi**. Die Reihenfolge, in der sie im Programm-Listing auftauchen, entspricht dabei nicht der des Ablaufes in der Benutzerführung, was aber keinen Einfluss auf die tatsächliche Funktionsweise hat.
- Das nun folgende Programmstück erledigt die Ausgabe der **Suchergebnisliste**, nachdem die Such-/Anlegemaske (Abbildung 19) abgesendet wurde. Zum Beispiel werden nach Eingabe "`update`" im Feld „Artikelbezeichnung“ alle Produkte, in deren Artikelnamen der Ausdruck "update" vorkommt, gefunden.

  Die angezeigte Liste enthält dabei maximal 8 Einträge. Wurde eine größere Anzahl Artikel gefunden, kann man zwischen verschiedenen Ergebnisseiten durchschalten ( < *Gehe zu Seite* > ). Auch das Ändern des Sortierkriteriums ist möglich, indem man auf die jeweilige Tabellenüberschrift ( Artikelnr, Gruppe, Artikelbezeichnung, Preis ) klickt.

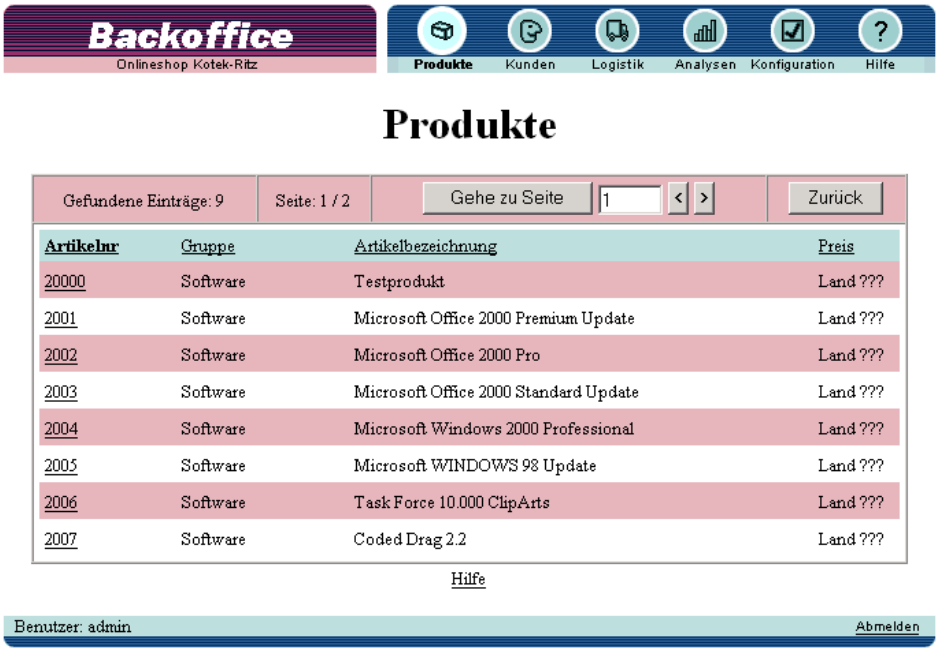



Abbildung 18: Backoffice - Produkte - Suchergebnisliste

```
280
281  ' ------------------------------------------------------------------------------
282  ' # Suchliste ausgeben:
283   IF action="suchen" THEN%>
284  <input type=hidden name="action" value="suchen">
285  <script>document.produkte.action.value="suchen";</script>
286  <input type=hidden name="inaktiv" value="<%=myHTML(inaktiv)%>">
287  <input type=hidden name="artikelnr" value="<%=myHTML(artikelnr)%>">
288  <input type=hidden name="sprachnr" value="<%=myHTML(sprachnr)%>">
289  <input type=hidden name="artikelbez" value="<%=myHTML(artikelbez)%>">
290  <input type=hidden name="gruppennr" value="<%=myHTML(gruppennr)%>">
291  <input type=hidden name="landnr" value="<%=myHTML(landnr)%>">
292  <input type=hidden name="preis" value="<%=myHTML(preis)%>"><%
293    sort=Request.Form("sort")
294    IF sort="" THEN sort="P.Artikelnr"
295    page=Request.Form("page")
296    IF (not IsNumeric(page)) or page="" THEN page=1
297    pages=fix(anzahl/8)
298    IF anzahl MOD 8 <> 0 THEN pages=pages+1
299    IF CInt(page)>pages THEN page=pages
300    IF CInt(page)<1 THEN page=1%>
301  <input type=hidden name="sort" value="<%=sort%>">
302  <table border=1 cellspacing=0 cellpadding=4 width=100%>
303   <tr bgcolor="#e1b6bd"><td align=center>
304    <small>Gefundene Eintr&auml;ge: <%=anzahl%></small>
305   </td><td align=center>
306    <small>Seite: <%=page%> / <%=pages%></small>
307   </td><td align=center>
308    <input type=submit name="suchen" value="Gehe zu Seite">
309    <input type=text name="page" size=4 value="<%=page%>">
310    <input type=button name="minus" value="<"
311     onclick="javascript:document.produkte.page.value--;
312     document.produkte.submit();">
313    <input type=button name="plus"  value=">"
314     onclick="javascript:document.produkte.page.value++;
315     document.produkte.submit();">
316   </td><td align=center>
317    <input type=button name="back" value=" Zur&uuml;ck "
318     onclick="javascript:document.produkte.action.value='';
```

```
319     document.produkte.submit();">
320    </td></tr>
321    <tr><td colspan=4>
322     <table border=0 cellspacing=0 cellpadding=4 width=100%>
323      <tr bgcolor=#bcddda><td width=16%>
324       <small>
325        <%IF sort="P.Artikelnr" THEN%><b><%END IF%>
326        <a href="javascript:document.produkte.sort.value='P.Artikelnr';
327         document.produkte.submit();">Artikelnr</a>
328        <%IF sort="P.Artikelnr" THEN%></b><%END IF%>
329       </small>
330      </td><td width=20%>
331       <small>
332        <%IF sort="Gruppe" THEN%><b><%END IF%>
333        <a href="javascript:document.produkte.sort.value='Gruppe';
334         document.produkte.submit();">Gruppe</a><br>
335        <%IF sort="Gruppe" THEN%></b><%END IF%>
336       </small>
337      </td><td width=54%>
338       <small>
339        <%IF sort="Artikelbezeichnung" THEN%><b><%END IF%>
340        <a href="javascript:document.produkte.sort.value='Artikelbezeichnung';
341         document.produkte.submit();">Artikelbezeichnung</a>
342        <%IF sort="Artikelbezeichnung" THEN%></b><%END IF%>
343       </small>
344      </td><td width=10%>
345       <small>
346        <%IF sort="Preis" THEN%><b><%END IF%>
347        <a href="javascript:document.produkte.sort.value='Preis';
348         document.produkte.submit();">Preis</a>
349        <%IF sort="Preis" THEN%></b><%END IF%>
350       </small>
351      </td></tr><%
352     order=sort
353     IF order="Preis" AND landnr="" THEN order="P.Artikelnr"
354     IF order="Artikelbezeichnung" AND sprachnr="" THEN order="Artikelname"
355     view=",Artikelname"
356     IF sprachnr<>"" THEN view=",Artikelbezeichnung"
357     IF landnr<>"" THEN view=view&",Preis"
358     sql="SELECT DISTINCT P.Artikelnr,Gruppe"&view&" FROM" & _
359         " Produkt P, Produkt_Text T, Preis M, Produktgruppe Y " & _
360         where("Y.Gruppennr=P.Gruppennr",1) & _
361         " ORDER BY " & order
362     runsql db,rs,sql
363      c=0
364      DO UNTIL rs.EOF or c>=(page-1)*8
365       rs.MoveNext
366       c=c+1
367      LOOP
368      c=1
369      DO UNTIL rs.EOF or c>8
370       IF sprachnr<>"" THEN
371        viewA=myHTML(rs("Artikelbezeichnung"))
372       ELSE
373        viewA=myHTML(rs("Artikelname"))
374       END IF
375       IF landnr<>"" THEN
376        viewB=myHTML(rs("Preis"))
377       ELSE
378        viewB="Land ???"
379       END IF
380       bg="#ffffff"
381       IF c MOD 2 = 1 THEN bg="#e1b6bd"%>
382     <tr bgcolor=<%=bg%>><td>
383      <small><a href="produkte.asp?artnr=<%=myURL(rs("Artikelnr"))%>">
384       <%=myHTML(rs("Artikelnr"))%></a> </small>
385     </td><td>
386      <small><%=myHTML(rs("Gruppe"))%> </small>
```

```
387    </td><td>
388     <small><%=viewA%> </small>
389    </td><td>
390     <small><%=viewB%> </small>
391    </td></tr><%
392      rs.MoveNext
393      c=c+1
394     LOOP
395    endsql db,rs%>
396    </table>
397   </td></tr>
398  </table>
399  <small><a href="help/produkte.asp">Hilfe</a></small><%
400   END IF
```

- Nun folgt die **Start-** bzw. **Such-/Anlegemaske** der Produkt-Wartung. Um mit ihr einen neuen Artikel anlegen zu können, muss sie vollständig ausgefüllt werden. Danach wird man auf die Detailmaske des neuen Produktes weitergeleitet, um eventuell noch weiterführende Eingaben machen zu können.

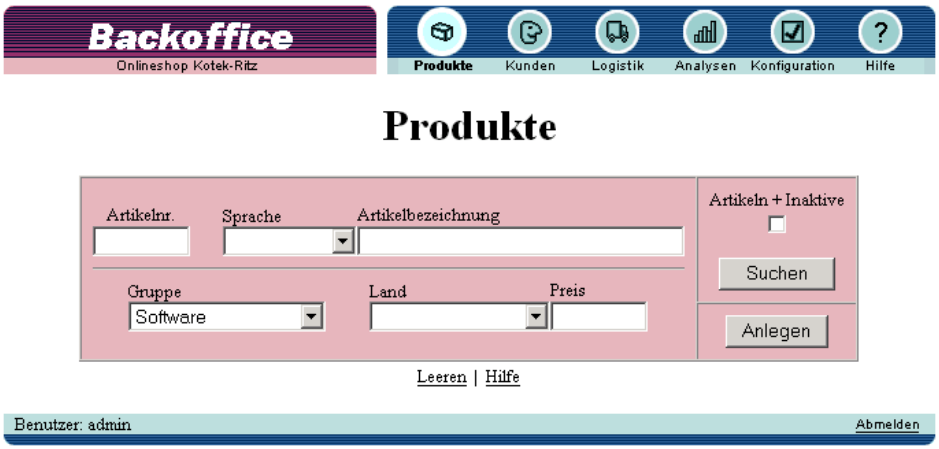



Abbildung 19: Backoffice - Produkte - Such-/Anlegemaske

```
401  ' ------------------------------------------------------------------------
402  ' # Such-/Eingabemaske ausgeben:
403   IF action="" THEN%>
404  <input type=hidden name="action" value="suchen">
405  <script>document.produkte.action.value="suchen";</script>
406  <table border=1 cellspacing=0 cellpadding=8>
407   <tr bgcolor=#e1b6bd align=center><td rowspan=2>
408    <table border=0 cellspacing=0 cellpadding=0><tr>
409     <td align=center><small>Artikelnr.</small><br>
410      <input type=text name="artikelnr" size=8
411       value="<%=myHTML(artikelnr)%>"></td>
412     <td width=24></td>
413     <td><small>Sprache</small><br><select name="sprachnr">
414      <%optsprache sprachnr%></select><br></td>
415     <td><small>Artikelbezeichnung</small><br>
416      <input type=text name="artikelbez" size=36
417       value="<%=myHTML(artikelbez)%>"></td>
418    </tr></table><hr>
419    <table border=0 cellspacing=0 cellpadding=0><tr>
420     <td><small>Gruppe</small><br>
421      <select name="gruppennr"><%optgruppe gruppennr%></select><br></td>
422     <td width=8></td>
423     <td width=24></td>
424     <td><small>Land</small><br><select name="landnr">
```

```
425      <%optland landnr%></select></td>
426     <td><small>Preis</small><br><input type=text name="preis" size=8
427      value="<%=myHTML(preis)%>"></td>
428    </tr></table>
429   </td><td>
430    <small>Artikeln + Inaktive</small><br>
431     <input type=checkbox name="inaktiv" size=20<%IF inaktiv<>"" THEN
432     %> checked<%END IF%>><br>
433    <small> </small><br>
434     <input type=submit name="suchen" value="  Suchen  "><br>
435   </td></tr>
436   <tr bgcolor=#e1b6bd align=center><td>
437    <input type=button name="anlegen" value="Anlegen"
438     onclick="document.produkte.action.value='anlegen';
439     document.produkte.submit();">
440   </td></tr>
441  </table>
442  <small><a href="produkte.asp">Leeren</a>  | 
443   <a href="help/produkte.asp">Hilfe</a></small><%
444   END IF
```

- Abschließend kommt noch die **Detailmaske** eines gewählten Produktes. Das Besondere an dem generierten HTML dieser Backoffice-Seite ist, dass für die Auswahl der sprachbezogenen Texte und der länder-/währungsabhängigen Preise ein vom Server angepasstes Javascript-Programm mitgesendet wird.

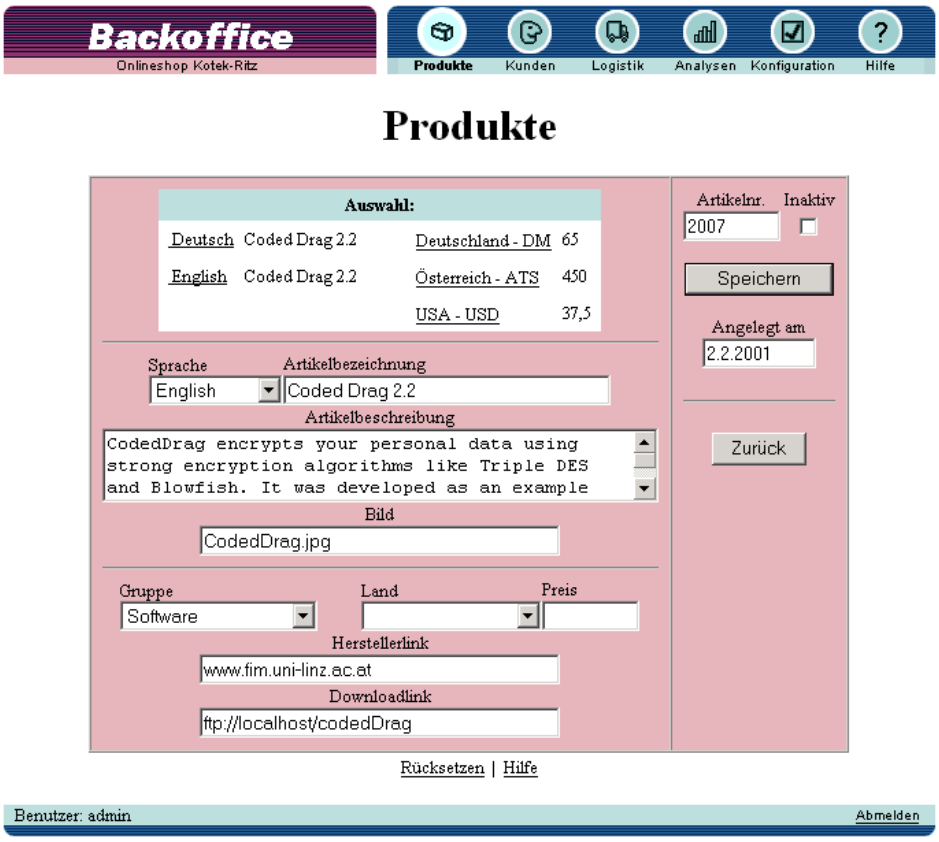



Abbildung 20: Backoffice - Produkte - Detailmaske

```
445  ' ------------------------------------------------------------------------
446  ' # Produkt-Detaildaten besorgen:
447   IF action="detail" and artnr<>"" THEN
```

```
448     sql="SELECT * FROM Produkt WHERE Artikelnr='" & artnr & "'"
449     runsql db,rs,sql
450      inaktiv=""
451      IF rs("Aktiv")<>1 THEN inaktiv="on"
452      artikelnr=rs("Artikelnr")
453      sprachnr=""
454      artikelbez=rs("Artikelname")
455      gruppennr=rs("Gruppennr")
456      landnr=""
457      preis=""
458      artikelbesch=""
459      bild=""
460      link=rs("Herstellerlink")
461      downloadlink=rs("Downloadlink")
462      datum=rs("Datum")
463     endsql db,rs
464    END IF
465   ' ------------------------------------------------------------------------
466   ' # Produkt-Detaildaten ausgeben:
467    IF action="detail" THEN%>
468   <input type=hidden name="action" value="aendern">
469   <script>
470    document.produkte.action.value="aendern";
471    function sprache(x) {
472     for (f=0,n=0; n<document.produkte.sprachnr.length; n++)
473      if (document.produkte.sprachnr.options[n].value==x) f=n;
474     return (f);
475    }
476    function land(x) {
477     for (f=0,n=0; n<document.produkte.landnr.length; n++)
478      if (document.produkte.landnr.options[n].value==x) f=n;
479     return (f);
480    }
481   </script>
482   <table border=1 cellspacing=0 cellpadding=8>
483    <tr bgcolor=#e1b6bd align=center><td>
484     <table border=0 cellspacing=0 cellpadding=4>
485      <tr bgcolor=#bcddda align=center>
486       <td colspan=5><small><b>Auswahl:</b></small><br></td>
487      </tr><%
488     sqlA="SELECT Artikelbezeichnung,Artikelbeschreibung,Bild," & _
489          "Sprache,S.Sprachnr " & _
490          "FROM Produkt_Text P, Sprache S WHERE P.Sprachnr=S.Sprachnr " & _
491          "AND Artikelnr='" & artikelnr & "' ORDER BY Sprache"
492     sqlB="SELECT Preis,Land,L.Landnr,Waehrung " & _
493            "FROM Preis M, Land L, Waehrung W WHERE M.Landnr=L.Landnr " & _
494            "AND L.Waehrungsnr=W.Waehrungsnr " & _
495            "AND Artikelnr='" & artikelnr & "' ORDER BY Land"
496     runsql dbA,rsA,sqlA
497     runsql dbB,rsB,sqlB
498     DO UNTIL rsA.EOF AND rsB.EOF
499      IF NOT rsA.EOF THEN%>
500      <tr bgcolor=#ffffff>
501       <td><small> <A HREF="#input"
502        onclick="javascript:document.produkte.artikelbez.value=
503        '<%=myHTML(rsA("Artikelbezeichnung"))%>';
504        document.produkte.artikelbesch.value=
505        '<%=myJS(myHTML(rsA("Artikelbeschreibung")))%>';
506        document.produkte.bild.value='<%=myHTML(rsA("Bild"))%>';
507        document.produkte.sprachnr.selectedIndex=sprache(<%=rsA("Sprachnr")%>);">
508       <%=myHTML(rsA("Sprache"))%></A></small></td>
509       <td><small><%=myHTML(rsA("Artikelbezeichnung"))%> </small></td><%
510       rsA.MoveNext
511      ELSE%>
512      <tr bgcolor=#ffffff>
513       <td><small> </small></td><td><small> </small></td><%
514      END IF%>
515       <td width=24><small> </small></td><%
```

```
516     IF NOT rsB.EOF THEN%>
517      <td><small><A HREF="#input"
518       onclick="javascript:document.produkte.preis.value='<%=rsB("Preis")%>';
519       document.produkte.landnr.selectedIndex=land(<%=rsB("Landnr")%>);">
520       <%=myHTML(rsB("Land"))%> - <%=myHTML(rsB("Waehrung"))%></A></small>
521      <td><small><%=myHTML(rsB("Preis"))%> </small></td>
522     </tr><%
523      rsB.MoveNext
524     ELSE%>
525      <td><small> </small></td><td><small> </small></td>
526     </tr><%
527     END IF
528    LOOP
529    endsql dbB,rsB
530    endsql dbA,rsA%>
531    </table><hr>
532    <a name="input">
533    <table border=0 cellspacing=0 cellpadding=0><tr align=center><td>
534     <table border=0 cellspacing=0 cellpadding=0><tr>
535      <td><small>Sprache</small><br><select name="sprachnr">
536       <%optsprache sprachnr%></select><br></td>
537      <td><small>Artikelbezeichnung</small><br>
538       <input type=text name="artikelbez" size=36
539        value="<%=myHTML(artikelbez)%>"></td>
540      </tr></table>
541     <small>Artikelbeschreibung</small><br>
542     <textarea name="artikelbesch" cols=48 rows=3><%
543      %><%=myHTML(artikelbesch)%></textarea>
544    </td></tr></table>
545    <small>Bild</small><br>
546    <input type=text name="bild" size=40 value="<%=myHTML(bild)%>"><hr>
547    <table border=0 cellspacing=0 cellpadding=0><tr>
548     <td><small>Gruppe</small><br>
549      <select name="gruppennr"><%optgruppe gruppennr%></select><br></td>
550     <td width=8></td>
551     <td width=24></td>
552     <td><small>Land</small><br><select name="landnr">
553      <%optland landnr%></select></td>
554     <td><small>Preis</small><br><input type=text name="preis" size=8
555      value="<%=preis%>"></td>
556    </tr></table>
557    <small>Herstellerlink</small><br>
558    <input type=text name="link" size=40 value="<%=myHTML(link)%>">
559    <br>
560    <small>Downloadlink</small><br>
561    <input type=text name="downloadlink" size=40 value="<%=myHTML(downloadlink)%>">
562   </td><td valign=top>
563    <table border=0 cellspacing=0 cellpadding=0><tr>
564     <td align=center><small>Artikelnr.</small></td>
565     <td align=center><small> Inaktiv</small></td>
566    </tr><tr>
567     <td align=center><input type=text name="artikelnr" size=8
568      value="<%=myHTML(artikelnr)%>" readonly></td>
569     <td align=center><input type=checkbox name="inaktiv" size=20
570      <%IF inaktiv<>"" THEN%> checked<%END IF%>></td>
571    </tr></table>
572    <small> </small><br>
573    <input type=submit name="aendern" value="  Speichern  "><br>
574    <small> <br>Angelegt am</small><br>
575    <input type=text name="datum" size=10
576     value="<%=myHTML(mydate(datum))%>" readonly><br>
577    <small> <hr> </small><br>
578    <input type=button name="back" value=" Zur&uuml;ck "
579     onclick="document.location.href='produkte.asp';">
580   </td></tr>
581  </table>
582  <small><a href="produkte.asp?artnr=<%=myURL(artikelnr)%>">R&uuml;cksetzen</a>
583    |  <a href="help/produkte.asp">Hilfe</a></small><%
```

```
584   END IF
585   ' ------------------------------------------------------------------------
586   %>
587   </form>
588
589   <%endbody status%>
590
591   </html>
592   <%
593   ' ========================================================================
594   %>
```

**Stored-Procedure: Write_Produkt**

- Diese SQL-Prozedur **speichert** einen neuen **Artikel** (`@artikelnr`) inkl. eines Namens in der Standardsprache des Backoffice (`@artikelname`) für eine Produktgruppe (`@gruppennr`) ab.

  Existiert bereits ein Artikel mit dieser Artikelnummer, retourniert die Prozedur den Wert 0, ansonsten 1.

```
1   CREATE PROCEDURE Write_Produkt (
2   @artikelnr      VARCHAR(20),
3   @artikelname  VARCHAR(80),
4   @gruppennr      INT) AS
5   BEGIN
6   TRANSACTION
7     IF NOT EXISTS(SELECT P.Artikelnr FROM Produkt AS P WHERE P.Artikelnr=@artikelnr)
8       BEGIN
9         INSERT INTO Produkt(Artikelnr, Aktiv, Artikelname, Gruppennr, Datum)
10          VALUES (@artikelnr, 1, @artikelname, @gruppennr, GETDATE())
11        SELECT ret=1
12      END
13    ELSE
14      SELECT dummy=0
15      SELECT ret=0
16    IF @@error=0 COMMIT
17      ELSE ROLLBACK
18
19  GO
```

### 5.4.3 Kunden

Das zweite zu besprechende Modul dient zum **Anlegen** und zum **Suchen** von beliebigen Shop-Kunden sowie zur **Pflege** aller abgelegten personenbezogenen Daten des bereits aufgebauten Kundenstammes.

Dabei ist zu beachten, dass sich dieser Kundenstamm nicht nur aus den vom Backoffice-Personal angelegten Personen zusammensetzt, sondern es für beliebige Interessenten möglich ist, sich selbst durch Angabe ihrer Personalien auf der Online-Shop-Website anmelden zu können (siehe Abbildung 31: Shop-Website - Anmeldung).

Die Implementierung der ASP-Seite selbst ist gleich strukturiert wie bei der Produktwartung.

Das Webinterface hält sich dabei an folgenden groben **Ablauf (** → Seiten-**Modi )**:

| | | | | |
|---|---|---|---|---|
| (1) **Such-/Anlegemaske** | → | *Suchen* | → | (2) |
| | → | *Anlegen* | → | (3) |
| … | | | | … |
| (2) **Suchergebnisliste** | → | *Zurück* | → | (1) |
| | → | < *Gehe zu Seite* > | → | (2) |
| | → | *Artikelnr aus Liste* | → | (3) |
| … | | | | … |
| (3) **Detailmaske** | → | *Zurück* | → | (1) |
| | → | *Speichern* | → | (3) |

**EShop/backoffice/kunden.asp**

```
1   <!--#include file="../dbSys/dbV2.asp"-->
2   <!--#include file=  "system/navi.asp"-->
```

- **Analog** zum `produkt.asp` werden auch hier eventuell vorangegangene Eingaben sofort eingelesen. Die Variable `"action"` dient ebenfalls dem Zweck den aktuellen **Befehl** ( *suchen, anlegen,* ...) zu identifizieren bzw. den Seiten-**Modus** ( Masken: suchen, detail, ...) zu definieren.
- Zusätzlich werden seitenspezifisch angepasste ASP-Funktionen definiert, um wieder die Wartungsfreundlichkeit des Scripts zu steigern.

```
3   <%
4   ' ================================================================================
5   ' # Kundenstammdaten-Seite
6   ' --------------------------------------------------------------------------------
7    html
8   %>
9   <head>
10   <title>Kunden</title>
11  </head>
12
13  <%body "kunden"%>
14
15  <h1>Kunden</h1>
16  <form METHOD="POST" ACTION="kunden.asp" name="kunden"><%
17  ' --------------------------------------------------------------------------------
18   status=""
19  ' --------------------------------------------------------------------------------
20  ' # Eingabe-Daten einlesen:
21   knr=myCond(Request.QueryString("knr"),0)        ' # GET- MESSAGE ?
22   IF knr="" THEN
23    action=Request.Form("action")                  ' # POST-MESSAGE !
24    kundennr=myCond(Request.Form("kundennr"),0)
25    inaktiv=Request.Form("inaktiv")
26    titel=myCond(Request.Form("titel"),20)
27    nachname=myCond(Request.Form("nachname"),30)
28    vorname=myCond(Request.Form("vorname"),30)
29    strasse=myCond(Request.Form("strasse"),20)
30    nr=myCond(Request.Form("nr"),20)
31    plz=myCond(Request.Form("plz"),10)
32    ort=myCond(Request.Form("ort"),30)
33    landnr=myCond(Request.Form("landnr"),0)
34    email=myCond(Request.Form("email"),60)
```

```
35      IF action="detail" or action="aendern" THEN
36       tel=myCond(Request.Form("tel"),30)
37       handy=myCond(Request.Form("handy"),30)
38       fax=myCond(Request.Form("fax"),30)
39       geburtsdatum=myCond(Request.Form("geburtsdatum"),0)
40       kreditkarte=myCond(Request.Form("kreditkarte"),30)
41       kartennummer=myCond(Request.Form("kartennummer"),20)
42       gueltigkeit=""
43       gueltigA=myCond(Request.Form("gueltigA"),2)
44       gueltigB=myCond(Request.Form("gueltigB"),2)
45       IF isnumeric(gueltigA) AND isnumeric(gueltigB) THEN
46        IF gueltigA>0 AND gueltigA<13 THEN
47         IF gueltigA<9 THEN gueltigA="0"&(gueltigA*1)
48         IF gueltigB<9 THEN gueltigB="0"&(gueltigB*1)
49         gueltigkeit=gueltigA&"/"&gueltigB
50        END IF
51       END IF
52       benutzername=myCond(Request.Form("benutzername"),20)
53       passwort=myCond(Request.Form("passwort"),20)
54       bonitaet=myCond(Request.Form("bonitaet"),20)
55       rabatt=myCond(Request.Form("rabatt"),0)
56       liefer_titel=myCond(Request.Form("liefer_titel"),20)
57       liefer_vorname=myCond(Request.Form("liefer_vorname"),30)
58       liefer_nachname=myCond(Request.Form("liefer_nachname"),30)
59       liefer_strasse=myCond(Request.Form("liefer_strasse"),20)
60       liefer_nr=myCond(Request.Form("liefer_nr"),20)
61       liefer_plz=myCond(Request.Form("liefer_plz"),10)
62       liefer_ort=myCond(Request.Form("liefer_ort"),30)
63       datum=Request.Form("datum")
64      END IF
65     ELSE                                              ' # GET - MESSAGE !
66       action="detail"
67     END IF
68    ' ------------------------------------------------------------------------
69    ' # Erzeuge sql-where-statement:
70     function where(expr,cmp)
71      prvC1="="
72      prvC2=""
73      IF cmp=1 THEN
74       prvC1=" LIKE "
75       prvC2="%"
76      END IF
77      prvS1=""
78      prvS2="WHERE "
79      IF kundennr<>"" THEN
80       prvS1=prvS1 & prvS2 & "K.Kundennr='" & kundennr & "'"
81       prvS2=" AND "
82      END IF
83      IF titel<>"" THEN
84       prvS1=prvS1 & prvS2 & "K.Titel" & prvC1 & "'" & titel & prvC2 & "'"
85       prvS2=" AND "
86      END IF
87      IF nachname<>"" THEN
88       prvS1=prvS1 & prvS2 & "K.Nachname" & prvC1 & "'" & nachname & prvC2 & "'"
89       prvS2=" AND "
90      END IF
91      IF vorname<>"" THEN
92       prvS1=prvS1 & prvS2 & "K.Vorname" & prvC1 & "'" & vorname & prvC2 & "'"
93       prvS2=" AND "
94      END IF
95      IF strasse<>"" THEN
96       prvS1=prvS1 & prvS2 & "K.Strasse" & prvC1 & "'" & strasse & prvC2 & "'"
97       prvS2=" AND "
98      END IF
99      IF nr<>"" THEN
100      prvS1=prvS1 & prvS2 & "K.Nr='" & nr & "'"
101      prvS2=" AND "
102     END IF
```

```
103     IF plz<>"" THEN
104      prvS1=prvS1 & prvS2 & "K.Plz='" & plz & "'"
105      prvS2=" AND "
106     END IF
107     IF ort<>"" THEN
108      prvS1=prvS1 & prvS2 & "K.Ort" & prvC1 & "'" & ort & prvC2 & "'"
109      prvS2=" AND "
110     END IF
111     IF landnr<>"" THEN
112      prvS1=prvS1 & prvS2 & "K.Landnr='" & landnr & "'"
113      prvS2=" AND "
114     END IF
115     IF email<>"" THEN
116      prvS1=prvS1 & prvS2 & "K.Email" & prvC1 & "'" & email & prvC2 & "'"
117      prvS2=" AND "
118     END IF
119     IF inaktiv="" THEN
120      prvS1=prvS1 & prvS2 & "K.Aktiv=1"
121      prvS2=" AND "
122     END IF
123     IF expr<>"" THEN
124      prvS1=prvS1 & prvS2 & expr
125     END IF
126     where=prvS1
127    end function
128   ' ------------------------------------------------------------------------------
129   ' # Baue <options> fuer Land-Selektion:
130    sub optland(landnr)
131     %><option value=""></option><%
132     prvSQL="SELECT Landnr,Land FROM Land ORDER BY Land"
133     runsql prvDB,prvRS,prvSQL
134      DO UNTIL prvRS.EOF
135       prvSel=""
136       IF landnr<>"" AND IsNumeric(landnr) THEN
137        IF CInt(landnr)=prvRS(0) THEN prvSel=" selected"
138       END IF
139       %><option value="<%=prvRS(0)%>"<%=prvSel&">"&myHTML(prvRS(1))%><%
140       %></option><%
141       prvRS.MoveNext
142      LOOP
143     endsql prvDB,prvRS
144    end sub
145   ' ------------------------------------------------------------------------------
146   ' # Baue <options> fuer Kreditkarten-Selektion:
147    sub optkarte(kartenname)
148     %><option value=""></option><%
149     prvSQL="SELECT Kartenname FROM Kreditkarten ORDER BY Kartenname"
150     runsql prvDB,prvRS,prvSQL
151      DO UNTIL prvRS.EOF
152       prvSel=""
153       IF kartenname=prvRS(0) THEN prvSel=" selected"
154       %><option value="<%=prvRS(0)%>"<%
155       %><%=prvSel&">"&myHTML(prvRS(0))%></option><%
156       prvRS.MoveNext
157      LOOP
158     endsql prvDB,prvRS
159    end sub
```

- Auch in diesem Skript dient der folgende Abschnitt der **Bearbeitung** eines eventuell eingegangenen **Befehl**s (`action` = *anlegen, aendern, suchen* ). Dabei werden auch diesmal alle Eingabe-Werte vorweg auf Korrektheit und Vollständigkeit überprüft (z.B.: muss die Eingabe des Geburtsdatums eines Kunden stets ein korrektes Datum darstellen).

Unterschieden wird generell zwischen dem Anlegen eines neuen Kunden, der Datenänderung eines bekannten Kunden und der Suche nach einem oder mehreren Kundensätzen.

```
160  ' ------------------------------------------------------------------------------
161  ' # Anlegen: Eingaben vollständig?
162   IF action="anlegen" and (nachname="" or vorname="" or strasse="" or nr="" or _
163      plz="" or ort="" or landnr="" or email="") THEN
164    status="FEHLER: Formular unvollständig ausgefüllt!!!"
165    action=""
166   END IF
167  ' ------------------------------------------------------------------------------
168  ' # Anlegen durchfuehren:
169   IF action="anlegen" THEN
170    kundennr=""
171    sql="SELECT count(*) AS 'Anzahl' FROM Kunde K " & where("",0)
172    runsql db,rs,sql
173     anzahl=rs(0)
174    endsql db,rs
175    IF anzahl<>0 THEN
176     status="FEHLER: Kunde bereits vorhanden!!!"
177     action=""
178    ELSE
179     runproc db,RS,command
180     command.CommandText = "Write_Kunde"
181     command.Parameters.Append command.CreateParameter _
182      ("@Titel", 200, 1, 20, titel)
183     command.Parameters.Append command.CreateParameter _
184      ("@Nachname", 200, 1, 30, nachname)
185     command.Parameters.Append command.CreateParameter _
186      ("@Vorname", 200, 1, 30, vorname)
187     command.Parameters.Append command.CreateParameter _
188      ("@Strasse", 200, 1, 20, strasse)
189     command.Parameters.Append command.CreateParameter _
190      ("@Nr", 200, 1, 20, nr)
191     command.Parameters.Append command.CreateParameter _
192      ("@Plz", 200, 1, 10, plz)
193     command.Parameters.Append command.CreateParameter _
194      ("@Ort", 200, 1, 30, ort)
195     command.Parameters.Append command.CreateParameter _
196      ("@Landnr", 3, 1, 4, landnr)
197     command.Parameters.Append command.CreateParameter _
198      ("@Email", 200, 1, 60, email)
199     command.Parameters.Append command.CreateParameter _
200      ("@Benutzername", 200, 1, 20, "")
201     command.Parameters.Append command.CreateParameter _
202      ("@Passwort", 200, 1, 20, "")
203     knr=""
204     SET RS=command.Execute()
205     CheckForError
206     ' # Hole angelegte Kundennummer:
207      IF NOT RS IS nothing THEN
208       SET RS=RS.NextRecordset()
209       IF NOT RS.EOF THEN knr=RS("knr")
210      END IF
211     endproc db,RS
212     IF knr="" THEN
213      status=DBError
214      action=""
215     ELSE
216      ' # Wenn Kunde angelegt wurde, schalte weiter auf Detailanzeige:
217      action="detail"
218     END IF
219    END IF
220   END IF
221  ' ------------------------------------------------------------------------------
```

```
' # Kundendaten aendern: Eingaben korrekt?
 IF action="aendern" THEN
  IF nachname="" or vorname="" or strasse="" or nr="" or plz="" or ort="" or _
     landnr="" or email="" or liefer_nachname="" or liefer_vorname="" or _
     liefer_strasse="" or liefer_nr="" or liefer_plz="" or liefer_ort="" THEN
   status="FEHLER: Formular unvollständig ausgefüllt!!!"
   action="detail"
  END IF
  IF IsNumeric(rabatt) THEN
   IF CInt(rabatt)<0 or CInt(rabatt)>100 THEN
    status="FEHLER: Fehlerhafte Eingabe!!!"
    action="detail"
   END IF
  ELSE
   status="FEHLER: Fehlerhafte Eingabe!!!"
   action="detail"
  END IF
  IF geburtsdatum<>"" and not IsDate(geburtsdatum) THEN
   status="FEHLER: Fehlerhafte Eingabe!!!"
   action="detail"
  END IF
  IF (gueltigA<>"" OR gueltigB<>"") AND gueltigkeit="" THEN
   status="FEHLER: Fehlerhafte Eingabe!!!"
   action="detail"
  END IF
 END IF
' ------------------------------------------------------------------------------
' # Kundendaten aendern durchfuehren:
 IF action="aendern" THEN
  aktiv=1
  IF inaktiv<>"" THEN aktiv=0
  sql="UPDATE Kunde SET " & _
   "Aktiv='" & aktiv & "', " & _
   "Titel='" & titel & "', " & _
   "Nachname='" & nachname & "', " & _
   "Vorname='" & vorname & "', " & _
   "Strasse='" & strasse & "', " & _
   "Nr='" & nr & "', " & _
   "Plz='" & plz & "', " & _
   "Ort='" & ort & "', " & _
   "Landnr='" & landnr & "', " & _
   "Email='" & email & "', " & _
   "Tel='" & tel & "', " & _
   "Handy='" & handy & "', " & _
   "Fax='" & fax & "', " & _
   "Geburtsdatum=" & mySQLdate(geburtsdatum) & ", " & _
   "Kreditkarte='" & kreditkarte & "', " & _
   "Kartennummer='" & kartennummer & "', " & _
   "Gueltigkeit='" & gueltigkeit & "', " & _
   "Bonitaet='" & bonitaet & "', " & _
   "Rabatt='" & CInt(rabatt) & "', " & _
   "Liefer_Titel='" & liefer_titel & "', " & _
   "Liefer_Vorname='" & liefer_vorname & "', " & _
   "Liefer_Nachname='" & liefer_nachname & "', " & _
   "Liefer_Strasse='" & liefer_strasse & "', " & _
   "Liefer_nr='" & liefer_nr & "', " & _
   "Liefer_Plz='" & liefer_plz & "', " & _
   "Liefer_Ort='" & liefer_ort & "', " & _
   "Liefer_Landnr='" & landnr & "' " & _
   "WHERE Kundennr='" & kundennr & "'"
  runsql db,rs,sql
  endsql db,rs
  status="OKAY!!!"
  knr=kundennr
  action="detail"
 END IF
' ------------------------------------------------------------------------------
' # Kunden suchen: Eingaben untersuchen!
```

```
290   IF action="suchen" THEN
291    IF kundennr<>"" and not IsNumeric(kundennr) THEN
292     status="FEHLER: Fehlerhafte Eingabe!!!"
293     action=""
294    ELSE
295     sql="SELECT count(*) AS 'Anzahl' FROM Kunde K " & where("",1)
296     runsql db,rs,sql
297      anzahl=rs(0)
298     endsql db,rs
299     IF anzahl=0 THEN
300      status="FEHLER: Keinen Eintrag gefunden!!!"
301      action=""
302     END IF
303     ' # Falls genau ein Kunde gefunden wird, sofort auf Detailanzeige gehen:
304     IF anzahl=1 THEN
305      sql="SELECT Kundennr FROM Kunde K " & where("",1)
306      runsql db,rs,sql
307       knr=rs("Kundennr")
308      endsql db,rs
309      action="detail"
310     END IF
311    END IF
312   END IF
```

- Die nächsten **3 Code-Abschnitte** beherbergen wieder die **Darstellungslogik**.
- Das erste dieser Programmstücke erledigt erneut die Ausgabe der **Suchergebnisliste**, nachdem die ausgefüllten Suchfelder der Such-/Anlegemaske (Abbildung 22) mit dem Suchen-Button abgesendet wurden.

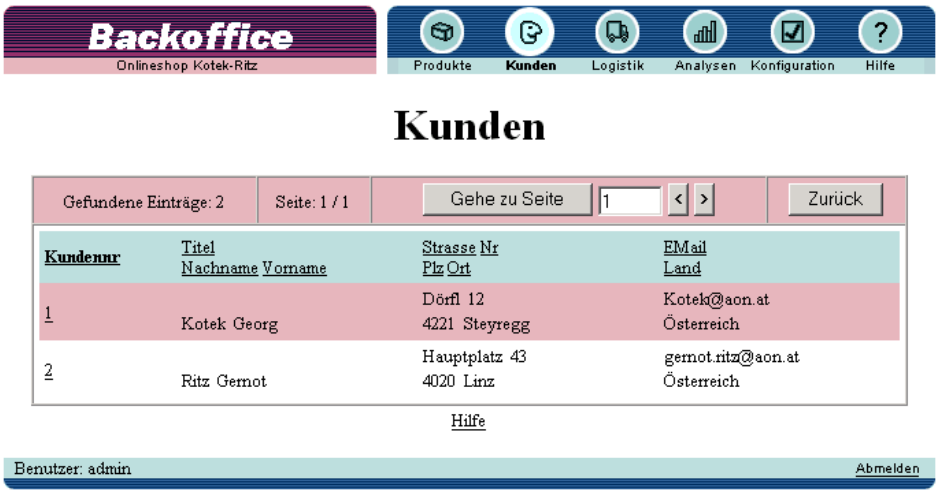



Abbildung 21: Backoffice - Kunden - Suchergebnisliste

```
313   ' ------------------------------------------------------------------------
314   ' # Suchliste ausgeben:
315    IF action="suchen" THEN%>
316   <input type=hidden name="action" value="suchen">
317   <script>document.kunden.action.value="suchen";</script>
318   <input type=hidden name="kundennr" value="<%=myHTML(kundennr)%>">
319   <input type=hidden name="inaktiv" value="<%=myHTML(inaktiv)%>">
320   <input type=hidden name="titel" value="<%=myHTML(titel)%>">
321   <input type=hidden name="nachname" value="<%=myHTML(nachname)%>">
322   <input type=hidden name="vorname" value="<%=myHTML(vorname)%>">
323   <input type=hidden name="strasse" value="<%=myHTML(strasse)%>">
324   <input type=hidden name="nr" value="<%=myHTML(nr)%>">
325   <input type=hidden name="plz" value="<%=myHTML(plz)%>">
```

```
<input type=hidden name="ort" value="<%=myHTML(ort)%>">
<input type=hidden name="landnr" value="<%=myHTML(landnr)%>">
<input type=hidden name="email" value="<%=myHTML(email)%>"><%
  sort=Request.Form("sort")
  IF sort="" THEN sort="Kundennr"
  page=Request.Form("page")
  IF (not IsNumeric(page)) or page="" THEN page=1
  pages=fix(anzahl/8)
  IF anzahl MOD 8 <> 0 THEN pages=pages+1
  IF CInt(page)>pages THEN page=pages
  IF CInt(page)<1 THEN page=1%>
<input type=hidden name="sort" value="<%=sort%>">
<table border=1 cellspacing=0 cellpadding=4 width=100%>
 <tr bgcolor="#e1b6bd"><td align=center>
  <small>Gefundene Eintr&auml;ge: <%=anzahl%></small>
 </td><td align=center>
  <small>Seite: <%=page%> / <%=pages%></small>
 </td><td align=center>
  <input type=submit name="suchen" value="Gehe zu Seite">
  <input type=text name="page" size=4 value="<%=page%>">
  <input type=button name="minus" value="<"
   onclick="javascript:document.kunden.page.value--; document.kunden.submit();">
  <input type=button name="plus"  value=">"
   onclick="javascript:document.kunden.page.value++; document.kunden.submit();">
 </td><td align=center>
  <input type=button name="back" value=" Zur&uuml;ck "
   onclick="javascript:document.kunden.action.value='';
   document.kunden.submit();">
 </td></tr>
 <tr><td colspan=4>
  <table border=0 cellspacing=0 cellpadding=4 width=100%>
   <tr bgcolor=#bcddda><td width=16%>
    <small>
     <%IF sort="Kundennr" THEN%><b><%END IF%>
     <a href="javascript:document.kunden.sort.value='Kundennr';
      document.kunden.submit();">Kundennr</a>
     <%IF sort="Kundennr" THEN%></b><%END IF%>
    </small>
   </td><td width=28%>
    <small>
     <%IF sort="Titel" THEN%><b><%END IF%>
     <a href="javascript:document.kunden.sort.value='Titel';
      document.kunden.submit();">Titel</a><br>
     <%IF sort="Titel" THEN%></b><%END IF%>
     <%IF sort="Nachname" THEN%><b><%END IF%>
     <a href="javascript:document.kunden.sort.value='Nachname';
      document.kunden.submit();">Nachname</a>
     <%IF sort="Nachname" THEN%></b><%END IF%>
     <%IF sort="Vorname" THEN%><b><%END IF%>
     <a href="javascript:document.kunden.sort.value='Vorname';
      document.kunden.submit();">Vorname</a>
     <%IF sort="Vorname" THEN%></b><%END IF%>
    </small>
   </td><td width=28%>
    <small>
     <%IF sort="Strasse" or sort="Nr" THEN%><b><%END IF%>
     <a href="javascript:document.kunden.sort.value='Strasse';
      document.kunden.submit();">Strasse</a>
     <%IF sort="Strasse" or sort="Nr" THEN%></b><%END IF%>
     <%IF sort="Nr" THEN%><b><%END IF%>
     <a href="javascript:document.kunden.sort.value='Nr';
      document.kunden.submit();">Nr</a><br>
     <%IF sort="Nr" THEN%></b><%END IF%>
     <%IF sort="Plz" THEN%><b><%END IF%>
     <a href="javascript:document.kunden.sort.value='Plz';
      document.kunden.submit();">Plz</a>
     <%IF sort="Plz" THEN%></b><%END IF%>
     <%IF sort="Ort" THEN%><b><%END IF%>
```

```
394       <a href="javascript:document.kunden.sort.value='Ort';
395        document.kunden.submit();">Ort</a>
396       <%IF sort="Ort" THEN%></b><%END IF%>
397      </small>
398     </td><td width=28%>
399      <small>
400       <%IF sort="Email" THEN%><b><%END IF%>
401       <a href="javascript:document.kunden.sort.value='Email';
402        document.kunden.submit();">EMail</a><br>
403       <%IF sort="Email" THEN%></b><%END IF%>
404       <%IF sort="Land" THEN%><b><%END IF%>
405       <a href="javascript:document.kunden.sort.value='Land';
406        document.kunden.submit();">Land</a>
407       <%IF sort="Land" THEN%></b><%END IF%>
408      </small>
409     </td></tr><%
410    order=sort
411    IF sort="Nr" THEN order="Strasse, Nr"
412    sql="SELECT Kundennr,Titel,Nachname,Vorname,Strasse,Nr,Plz,Ort," & _
413        "K.Landnr,Email,L.Landnr,Land FROM Kunde K, Land L " & _
414           where("L.Landnr = K.Landnr",1) & " ORDER BY " & order
415    runsql db,rs,sql
416     c=0
417     DO UNTIL rs.EOF or c>=(page-1)*8
418      rs.MoveNext
419      c=c+1
420     LOOP
421     c=1
422     DO UNTIL rs.EOF or c>8
423      bg="#ffffff"
424      IF c MOD 2 = 1 THEN bg="#e1b6bd"%>
425    <tr bgcolor=<%=bg%>><td>
426     <small><a href="kunden.asp?knr=<%=rs("Kundennr")%>">
427     <%=rs("Kundennr")%></a> </small>
428    </td><td>
429     <small><%=myHTML(rs("Titel"))%> <br>
430     <%=myHTML(rs("Nachname"))%> 
431     <%=myHTML(rs("Vorname"))%> </small>
432    </td><td>
433     <small><%=myHTML(rs("Strasse"))%> 
434     <%=myHTML(rs("Nr"))%> <br>
435     <%=myHTML(rs("Plz"))%> 
436     <%=myHTML(rs("Ort"))%> </small>
437    </td><td>
438     <small><%=myHTML(rs("Email"))%> <br>
439     <%=myHTML(rs("Land"))%> </small>
440    </td></tr><%
441      rs.MoveNext
442      c=c+1
443     LOOP
444    endsql db,rs%>
445    </table>
446   </td></tr>
447  </table>
448  <small><a href="help/kunden.asp">Hilfe</a></small><%
449   END IF
```

- Die **Start-** bzw. **Such-/Anlegemaske** der Kunden-Wartung: Auch hier muss die Maske vollständig (ausgenommen dem Titelfeld) ausgefüllt werden, um einen neuen Kunden anzulegen. Nach erfolgreichem Anlegen wird man daraufhin zur Detailmaske weitergeleitet.

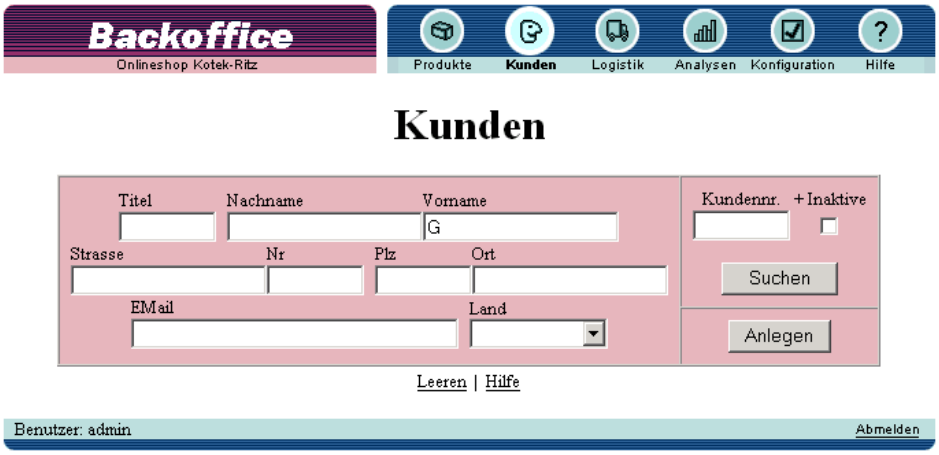



Abbildung 22: Backoffice - Kunden - Such-/Anlegemaske

```
450  ' ------------------------------------------------------------------------------
451  ' # Such-/Eingabemaske ausgeben:
452
453  <input type=hidden name="action" value="suchen">
454  <script>document.kunden.action.value="suchen";</script>
455  <table border=1 cellspacing=0 cellpadding=8>
456   <tr bgcolor=#e1b6bd align=center><td rowspan=2>
457    <table border=0 cellspacing=0 cellpadding=0><tr>
458     <td><small>Titel</small><br><input type=text name="titel" size=8
459      value="<%=myHTML(titel)%>"></td>
460     <td width=8></td>
461     <td><small>Nachname</small><br><input type=text name="nachname" size=20
462      value="<%=myHTML(nachname)%>"></td>
463     <td><small>Vorname</small><br><input type=text name="vorname" size=20
464      value="<%=myHTML(vorname)%>"></td>
465    </tr></table>
466    <table border=0 cellspacing=0 cellpadding=0><tr>
467     <td><small>Strasse</small><br><input type=text name="strasse" size=20
468      value="<%=myHTML(strasse)%>"></td>
469     <td><small>Nr</small><br><input type=text name="nr" size=8
470      value="<%=myHTML(nr)%>"></td>
471     <td width=8></td>
472     <td><small>Plz</small><br><input type=text name="plz" size=8
473      value="<%=myHTML(plz)%>"></td>
474     <td><small>Ort</small><br><input type=text name="ort" size=20
475      value="<%=myHTML(ort)%>"></td>
476    </tr></table>
477    <table border=0 cellspacing=0 cellpadding=0><tr>
478     <td><small>EMail</small><br><input type=text name="email" size=36
479      value="<%=myHTML(email)%>"></td>
480     <td width=8></td>
481     <td><small>Land</small><br><select name="landnr">
482      <%optland landnr%></select></td>
483    </tr></table>
484   </td><td>
485    <table border=0 cellspacing=0 cellpadding=0><tr>
486     <td align=center><small>Kundennr.</small></td>
487     <td align=center><small> + Inaktive</small></td>
488    </tr><tr>
489     <td align=center><input type=text name="kundennr" size=8
490      value="<%=myHTML(kundennr)%>"></td>
491     <td align=center><input type=checkbox name="inaktiv" size=20
492      <%IF inaktiv<>"" THEN%> checked<%END IF%>></td>
493    </tr></table>
494    <small> </small><br><input type=submit name="suchen"
495     value="  Suchen  "><br>
496   </td></tr>
```

```
497   <tr bgcolor=#e1b6bd align=center><td>
498    <input type=button name="anlegen" value="Anlegen"
499     onclick="document.kunden.action.value='anlegen'; document.kunden.submit();">
500   </td></tr>
501  </table>
502  <small><a href="kunden.asp">Leeren</a>  | 
503   <a href="help/kunden.asp">Hilfe</a></small><%
504   END IF
```

- Abschließend noch die **Detailmaske** eines ausgewählten Kunden.

aktuellen Stammdaten aus der Datenbank besorgt, bevor sie in der anschließend generierten Seite eingebettet werden. Durch die ASP-Variable `"knr"`, welche z.B. von der Suchergebnisliste über einen GET-Parameter weitergereicht wird, lässt sich der benötigte Datensatz leicht adressieren.

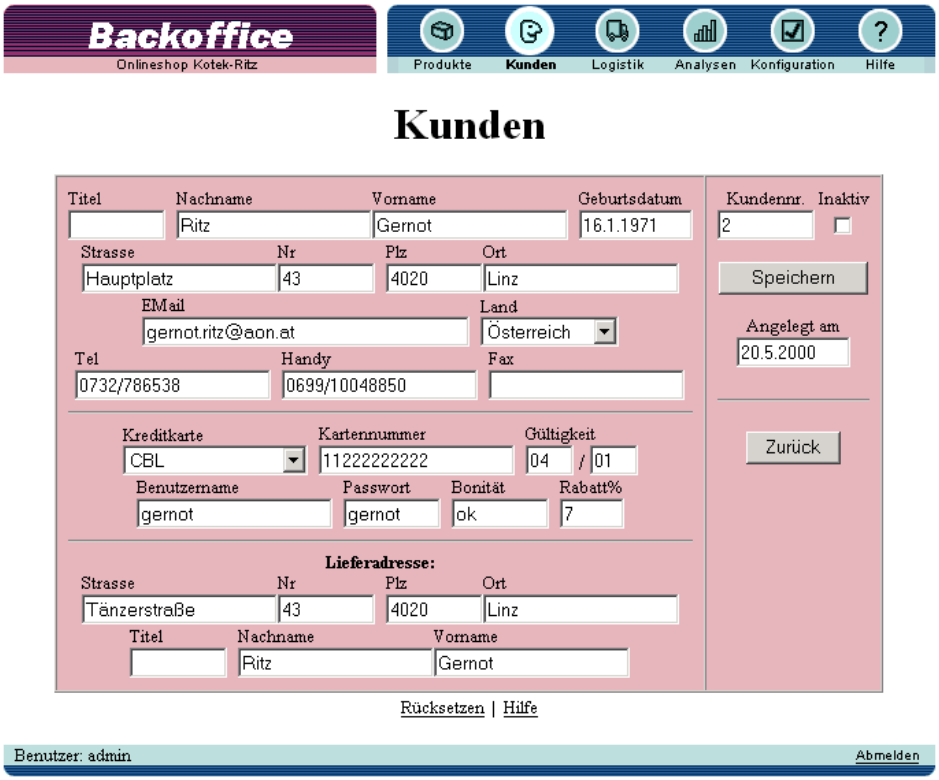



Abbildung 23: Backoffice - Kunden - Detailmaske

```
505  ' ------------------------------------------------------------------------------
506  ' # Kunden-Detaildaten besorgen:
507   IF action="detail" and knr<>"" THEN
508    sql="SELECT * FROM Kunde WHERE Kundennr='" & knr & "'"
509    runsql db,rs,sql
510     kundennr=rs("Kundennr")
511     inaktiv=""
512     IF rs("Aktiv")<>1 THEN inaktiv="on"
513     titel=rs("Titel")
514     nachname=rs("Nachname")
515     vorname=rs("Vorname")
516     strasse=rs("Strasse")
517     nr=rs("Nr")
```

```
518     plz=rs("Plz")
519     ort=rs("Ort")
520     landnr=rs("Landnr")
521     email=rs("Email")
522     tel=rs("Tel")
523     handy=rs("Handy")
524     fax=rs("Fax")
525     geburtsdatum=rs("Geburtsdatum")
526     kreditkarte=rs("Kreditkarte")
527     kartennummer=rs("Kartennummer")
528     gueltigkeit=rs("Gueltigkeit")
529      gueltigA=mid(gueltigkeit,1,2)
530      gueltigB=mid(gueltigkeit,4,2)
531     benutzername=rs("Benutzername")
532     passwort=rs("Passwort")
533     bonitaet=rs("Bonitaet")
534     rabatt=rs("Rabatt")
535     liefer_titel=rs("Liefer_Titel")
536     liefer_vorname=rs("Liefer_Vorname")
537     liefer_nachname=rs("Liefer_Nachname")
538     liefer_strasse=rs("Liefer_Strasse")
539     liefer_nr=rs("Liefer_Nr")
540     liefer_plz=rs("Liefer_Plz")
541     liefer_ort=rs("Liefer_Ort")
542     datum=rs("Datum")
543    endsql db,rs
544   END IF
545  ' ------------------------------------------------------------------------------
546  ' # Kunden-Detaildaten ausgeben:
547   IF action="detail" THEN%>
548  <input type=hidden name="action" value="aendern">
549  <script>document.kunden.action.value="aendern";</script>
550  <table border=1 cellspacing=0 cellpadding=8>
551   <tr bgcolor=#e1b6bd align=center><td>
552    <table border=0 cellspacing=0 cellpadding=0><tr>
553     <td><small>Titel</small><br><input type=text name="titel" size=8
554      value="<%=myHTML(titel)%>"></td>
555     <td width=8></td>
556     <td><small>Nachname</small><br><input type=text name="nachname" size=20
557      value="<%=myHTML(nachname)%>"></td>
558     <td><small>Vorname</small><br><input type=text name="vorname" size=20
559      value="<%=myHTML(vorname)%>"></td>
560     <td width=8></td>
561     <td><small>Geburtsdatum</small><br><input type=text name="geburtsdatum"
562      size=10 value="<%=myHTML(mydate(geburtsdatum))%>"></td>
563    </tr></table>
564    <table border=0 cellspacing=0 cellpadding=0><tr>
565     <td><small>Strasse</small><br><input type=text name="strasse" size=20
566      value="<%=myHTML(strasse)%>"></td>
567     <td><small>Nr</small><br><input type=text name="nr" size=8
568      value="<%=myHTML(nr)%>"></td>
569     <td width=8></td>
570     <td><small>Plz</small><br><input type=text name="plz" size=8
571      value="<%=myHTML(plz)%>"></td>
572     <td><small>Ort</small><br><input type=text name="ort" size=20
573      value="<%=myHTML(ort)%>"></td>
574    </tr></table>
575    <table border=0 cellspacing=0 cellpadding=0><tr>
576     <td><small>EMail</small><br><input type=text name="email" size=36
577      value="<%=myHTML(email)%>"></td>
578     <td width=8></td>
579     <td><small>Land</small><br><select name="landnr">
580      <%optland landnr%></select></td>
581    </tr></table>
582    <table border=0 cellspacing=0 cellpadding=0><tr>
583     <td><small>Tel</small><br><input type=text name="tel" size=20
584      value="<%=myHTML(tel)%>"></td>
585     <td width=8></td>
```

```
586     <td><small>Handy</small><br><input type=text name="handy" size=20
587      value="<%=myHTML(handy)%>"></td>
588     <td width=8></td>
589     <td><small>Fax</small><br><input type=text name="fax" size=20
590      value="<%=myHTML(fax)%>"></td>
591    </tr></table><hr>
592    <table border=0 cellspacing=0 cellpadding=0><tr>
593     <td><small>Kreditkarte</small><br><select name="kreditkarte">
594      <%optkarte(kreditkarte)%></select></td>
595     <td width=8></td>
596     <td><small>Kartennummer</small><br><input type=text name="kartennummer"
597      size=20 value="<%=myHTML(kartennummer)%>"></td>
598     <td width=8></td>
599     <td><small>G&uuml;ltigkeit</small><br><input type=text name="gueltigA"
600      size=2 value="<%=myHTML(gueltigA)%>"> / <input type=text name="gueltigB"
601      size=2 value="<%=myHTML(gueltigB)%>"></td>
602    </tr></table>
603    <table border=0 cellspacing=0 cellpadding=0><tr>
604     <td><small>Benutzername</small><br><input type=text name="benutzername"
605      size=20 value="<%=myHTML(benutzername)%>" readonly></td>
606     <td width=8></td>
607     <td><small>Passwort</small><br><input type=text name="passwort"
608      size=8 value="<%=myHTML(passwort)%>" readonly></td>
609     <td width=8></td>
610     <td><small>Bonit&auml;t</small><br><input type=text name="bonitaet"
611      size=8 value="<%=myHTML(bonitaet)%>"></td>
612     <td width=8></td>
613     <td><small>Rabatt%</small><br><input type=text name="rabatt"
614      size=4 value="<%=myHTML(rabatt)%>"></td>
615    </tr></table><hr>
616    <table border=0 cellspacing=0 cellpadding=0 width=100%><tr>
617     <td align=center><small><b>Lieferadresse:</b></small><br></td>
618    </tr></table>
619    <table border=0 cellspacing=0 cellpadding=0><tr>
620     <td><small>Strasse</small><br><input type=text name="liefer_strasse"
621      size=20 value="<%=myHTML(liefer_strasse)%>"></td>
622     <td><small>Nr</small><br><input type=text name="liefer_nr"
623      size=8 value="<%=myHTML(liefer_nr)%>"></td>
624     <td width=8></td>
625     <td><small>Plz</small><br><input type=text name="liefer_plz" size=8
626      value="<%=myHTML(liefer_plz)%>"></td>
627     <td><small>Ort</small><br><input type=text name="liefer_ort" size=20
628      value="<%=myHTML(liefer_ort)%>"></td>
629    </tr></table>
630    <table border=0 cellspacing=0 cellpadding=0><tr>
631     <td><small>Titel</small><br><input type=text name="liefer_titel"
632      size=8 value="<%=myHTML(liefer_titel)%>"></td>
633     <td width=8></td>
634     <td><small>Nachname</small><br><input type=text name="liefer_nachname"
635      size=20 value="<%=myHTML(liefer_nachname)%>"></td>
636     <td><small>Vorname</small><br><input type=text name="liefer_vorname"
637      size=20 value="<%=myHTML(liefer_vorname)%>"></td>
638    </tr></table>
639   </td><td valign=top>
640    <table border=0 cellspacing=0 cellpadding=0><tr>
641     <td align=center><small>Kundennr.</small></td>
642     <td align=center><small> Inaktiv</small></td>
643    </tr><tr>
644     <td align=center><input type=text name="kundennr" size=8
645      value="<%=myHTML(kundennr)%>" readonly></td>
646     <td align=center><input type=checkbox name="inaktiv" size=20
647      <%IF inaktiv<>"" THEN%> checked<%END IF%>></td>
648    </tr></table>
649    <small> </small><br>
650    <input type=submit name="aendern" value="  Speichern  "><br>
651    <small> <br>Angelegt am</small><br>
652    <input type=text name="datum" size=10
653     value="<%=myHTML(mydate(datum))%>" readonly><br>
```

```
654     <small> <hr> </small><br>
655     <input type=button name="back" value=" Zur&uuml;ck "
656      onclick="document.location.href='kunden.asp';">
657    </td></tr>
658   </table>
659   <small><a href="kunden.asp?knr=<%=kundennr%>">R&uuml;cksetzen</a>  | 
660    <a href="help/kunden.asp">Hilfe</a></small><%
661    END IF
      ' ------------------------------------------------------------------------
663   %>
664   </form>
665
666   <%endbody status%>
667
      </html>
669   <%
670   ' ========================================================================
671   %>
```

### Stored-Procedure: Write_Kunde

- Diese SQL-Prozedur **speichert** einen neuen **Kunden** bzw. **Benutzer** (`@benutzername, @passwort`) inkl. seiner personenbezogenden Daten in der Shop-Datenbank ab (`@title, @nachname, @vorname, @strasse, @nr, @plz, @ort, @landnr, @email`).

  Wird ein leerer Benutzername übergeben, werden die Personendaten des Kunden sofort abgespeichert und die neu angelegte Kundenummer retourniert. Ansonsten wird geprüft, ob dieser Benutzername bereits existiert. Ist dies der Fall, wird kein neuer Datensatz in der DB erzeugt und die Kundennr. 0 an das aufrufende ASP-Skript übergeben.

```
1     CREATE PROCEDURE Write_Kunde (
2     @titel        VARCHAR(20),
3     @nachname     VARCHAR(30),
4     @vorname      VARCHAR(30),
5     @strasse      VARCHAR(20),
6     @nr           VARCHAR(20),
7     @plz          VARCHAR(10),
8     @ort          VARCHAR(30),
9     @landnr       INT,
10    @email        VARCHAR(60),
11    @benutzername VARCHAR(20),
12    @passwort     VARCHAR(20)) AS
13    BEGIN
14    TRANSACTION
15      IF @benutzername<>''
16        IF NOT EXISTS(SELECT * FROM Kunde AS K WHERE K.Benutzername=@benutzername)
17          BEGIN /*Benutzername existiert noch nicht */
18            INSERT INTO Kunde(Titel, Nachname, Vorname, Strasse, Nr, Plz, Ort, Landnr,
              Email, Benutzername, Passwort, Rabatt, Liefer_Titel, Liefer_Nachname,
              Liefer_Vorname, Liefer_Strasse, Liefer_Nr, Liefer_Plz, Liefer_Ort,
              Liefer_Landnr, Datum, Aktiv)
19            VALUES (@Titel, @Nachname, @Vorname, @Strasse, @Nr, @Plz, @Ort, @Landnr,
              @Email, @benutzername, @passwort, 0, @Titel, @Nachname,
              @Vorname, @Strasse, @Nr, @Plz, @Ort, @Landnr, GETDATE(), 1)
20            SELECT knr =  IDENT_CURRENT('Kunde')
21          END
22        ELSE
```

```
23          BEGIN /*Benutzername existiert schon */
24            SELECT dummy =0
25            SELECT knr =0
26          END
27        ELSE
28          BEGIN /* Insert für Kunden die im Backoffice angelegt werden */
29            INSERT INTO Kunde(Titel, Nachname, Vorname, Strasse, Nr, Plz, Ort, Landnr,
              Email, Rabatt, Liefer_Titel, Liefer_Nachname, Liefer_Vorname,
              Liefer_Strasse, Liefer_Nr, Liefer_Plz, Liefer_Ort, Liefer_Landnr, Datum,
              Aktiv)
30            VALUES (@Titel, @Nachname, @Vorname, @Strasse, @Nr, @Plz, @Ort, @Landnr,
              @Email, 0, @Titel, @Nachname, @Vorname,
              @Strasse, @Nr, @Plz, @Ort, @Landnr, GETDATE(), 1)
31            SELECT knr =  IDENT_CURRENT('Kunde')
32          END
33        IF @@error=0 COMMIT
34          ELSE ROLLBACK
35      GO
```

### 5.4.4 Weitere Funktionen

Zum Abschluss der Backoffice-Beschreibung folgen jetzt noch die restlichen Menü-**Screenshots** der noch nicht besprochenen Hauptmodule, welche sich jeweils in diverse eigenständige ASP-Scripts aufspalten. Alle diese Programm-Listings befinden sich natürlich auf der beigelegten CD (siehe Kapitel 5.6). Der prinzipielle Aufbau aller dieser weiteren Backoffice-Teile unterscheidet sich aber konzeptionell nicht wesentlich von den bisher ausgeführten Modulen, sodass auf diese nicht einzeln eingegangen wird.

Eine Benutzerdokumentation (die sich über den vollen Funktionsumfang des Online-Shop-Backoffice erstreckt) ist in der Diplomarbeit von Georg Kotek (siehe [28]) inkludiert und kann bei Bedarf eingesehen werden.

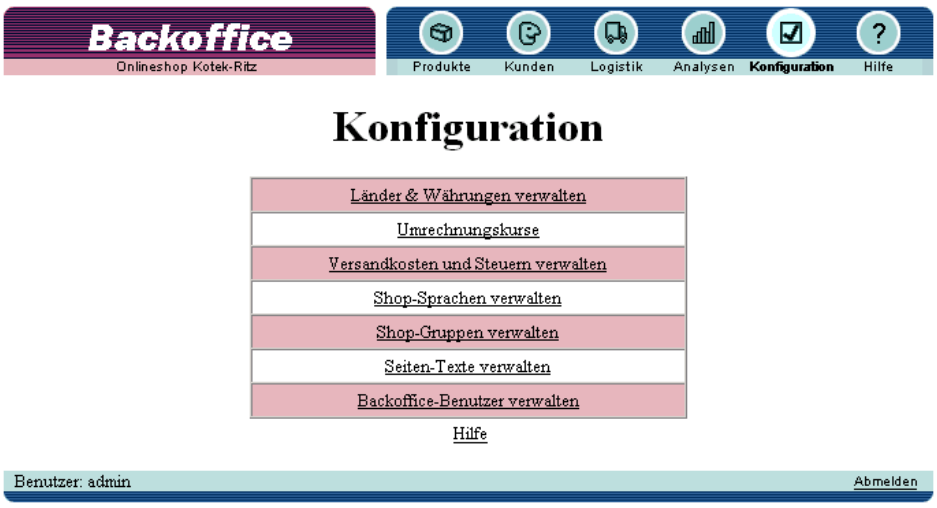



Abbildung 24: Backoffice - Konfiguration

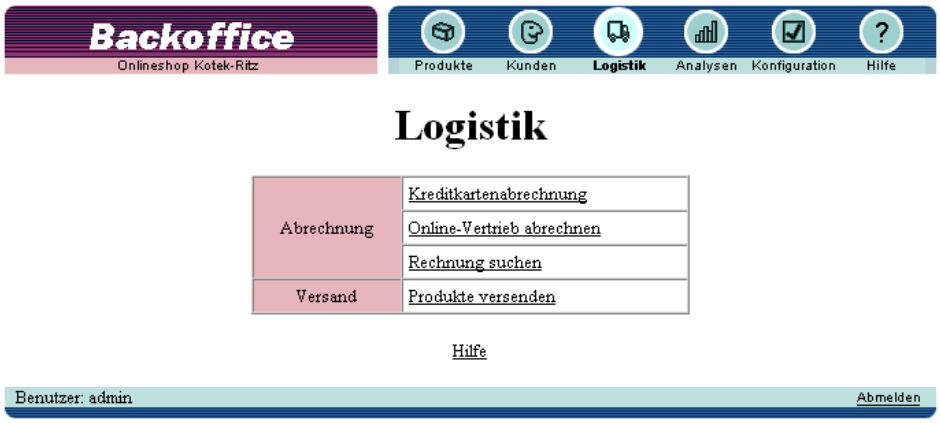



Abbildung 25: Backoffice - Logistik

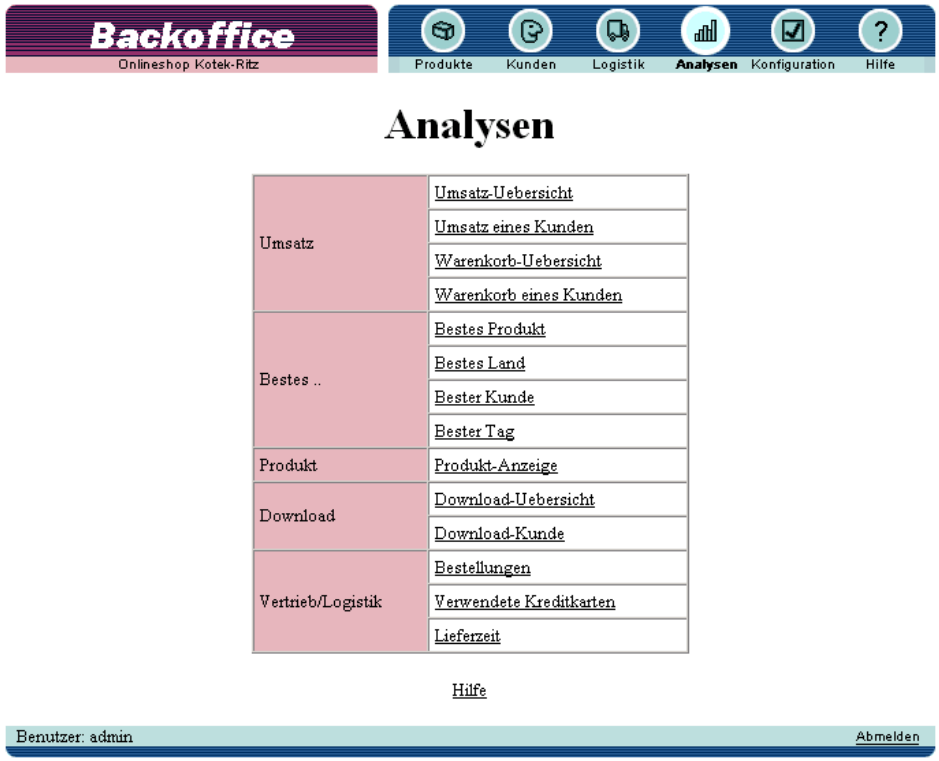



Abbildung 26: Backoffice - Analysen

## 5.5 Shop-Website

Nachdem nun durch das Kapitel 5.4 das Backoffice des Online-Shops behandelt wurde, widmet sich dieses Kapitel nun der **Website-Implementierung**, welche für die Kommunikation mit dem Online-Kunden zuständig ist. Auch diesmal werden wieder nur Ausschnitte des Gesamtsystems genauer betrachtet, um im Speziellen die gewählten Implementierungskonzepte zu beleuchten.

### 5.5.1 System

Wie schon das Backoffice besteht auch die Website (auch Frontoffice genannt) aus mehreren einzelnen Shopmodulen: z.B. Home, Produkte, Suche, Support, Download, Information, Kontakt und Warenkorb. Darum wurden wieder modulübergreifende Funktionen zusätzlich zu den bereits besprochenen Dateien `dbConnection.asp` und `dbV2.asp` (siehe 5.4.1) in einzubindende (`#include`) **ASP-Scripts** ausgelagert.

Eine neue Besonderheit der Shop-Website ist, dass sie zur Anzeige **HTML-Frames** in Anspruch nimmt. Um diese leichter zu verwalten, werden mehrere System-Scripts zur Verfügung gestellt, die mit Ausnahme des Hauptfensters (in welchem die einzelnen Shopmodule ihren Inhalt anzeigen) die restlichen Frame-Inhalte (Kopfzeile, Navigationsleiste und Fußzeile) verwalten.

**EShop/system/main.asp**

```
1   <!--#include file="../dbSys/dbV2.asp"-->
```

- Dieses Skript dient unter anderem der HTML-**Frames-Generierung** der Online-Shop-Website.
- Zu Beginn dieses Skripts wird zunächst geprüft, ob eine **Kunden-Anmeldung** vorliegt ( POST-Parameter: `"user"` & `"pwd"` ) oder ein Cookie am Client existiert, welches einen eventuell bereits früher angemeldeten Kunden identifiziert. Ist danach bekannt, ob ein dem Online-Shop-System bekannter Kunde angemeldet ist, werden diverse Kundendaten aus der DB in dafür vorgesehene ASP-Session-Variablen geladen (kundennr, sprachnr, landnr, ...).

```
    <%
3   ' ================================================================================
4   ' # Hauptseite (ruft navi.asp, head.asp und ../%page%.asp)
5   ' --------------------------------------------------------------------------------
6     Response.Expires=-1440
7   ' --------------------------------------------------------------------------------
8    user=myCond(Request.Form("user"),20)
9    pwd=myCond(Request.Form("pwd"),20)
10   IF user="" AND pwd="" AND session("benutzer")="" THEN
```

```
 user=Request.Cookies("user")
 pwd=Request.Cookies("pwd")
END IF
' ------------------------------------------------------------------------------
IF user<>"" and pwd<>"" THEN
 sql="SELECT count(*) AS 'Anzahl' FROM Kunde WHERE" & _
     " Benutzername='"&user&"' and Passwort='"&pwd&"'"
 runsql db,rs,sql
  count=rs("Anzahl")
 endsql db,rs
 IF count=1 THEN
  sql="SELECT * FROM Kunde WHERE" & _
      " Benutzername='"&user&"' and Passwort='"&pwd&"'"
  runsql db,rs,sql
   session("benutzer")=rs("Benutzername")
   session("passwort")=rs("Passwort")
   session("kundennr")=rs("Kundennr")
   session("sprachnr")=rs("Sprachnr")
   session("landnr")=rs("Landnr")
   session("sel")=0
  endsql db,rs
 END IF
END IF
```

- Falls kein Kunde am System angemeldet wurde, wird noch geprüft, ob neue für **anonyme Benutzer** vorgesehene **Sprach- und Ländereinstellungen** (POST-Parameter: `"sprachnr"` & `"landnr"`) in ASP-Session-Variablen übertragen werden sollen.

  Durch den GET-Parameter `"sel"` kann weiters noch das im **Hauptfenster** darzustellende Modul selektiert werden. Die aktuell gültige **Shopmodul-ID** wird dann für spätere Seitenaufrufe in einer ASP-Session-Variable gespeichert und in einen gültigen Web-**Seitennamen** aufgelöst. Besonders erwähnenswert ist in diesem Zusammenhang die **ID -2**, welche intern zum **Abmelden** eines Benutzers dient.

```
' ------------------------------------------------------------------------------
IF session("benutzer")="" THEN
 sprachnr=Request.Form("sprachnr")
 IF sprachnr="" THEN sprachnr=session("sprachnr")
 IF sprachnr="" THEN sprachnr=GTlangnr
 session("sprachnr")=sprachnr

 landnr=Request.Form("landnr")
 IF landnr="" THEN landnr=session("landnr")
 IF landnr="" THEN landnr=GTlandnr
 session("landnr")=landnr
END IF
' ------------------------------------------------------------------------------
sel=Request.QueryString("sel")
IF sel="" THEN sel=session("sel")
IF sel="" THEN sel=0
session("sel")=sel

SELECT CASE sel
 CASE "-3"
  page="benutzer"
 CASE "-2"
  session("benutzer")=""
  session("password")=""
  session("kundennr")=0
```

```
59      session("sel")=0
60      page="home"
61    CASE "-1"
62      page="anmelden"
63    CASE 0
64      page="home"
65    CASE 1
66      page="produkte"
67    CASE 2
68      page="suche"
69    CASE 3
70      page="support"
71    CASE 4
72      page="download"
73    CASE 5
74      page="info"
75    CASE 6
76      page="kontakt"
77    CASE 7
78      page="warenkorb"
79    CASE ELSE
80      session("sel")=0
81      page="home"
82   END SELECT
83  ' ------------------------------------------------------------------------
84      Response.Cookies("user")=session("benutzer")
85      Response.Cookies("user").Expires=DateAdd("m",1,Now())
86      Response.Cookies("pwd")=session("passwort")
87      Response.Cookies("pwd").Expires=DateAdd("m",1,Now())
88  ' ------------------------------------------------------------------------
89  %>
```

- Nun folgt die **Frame-Definition**s-Ausgabe an den Web-Client. Dadurch ist es dem Browser möglich die Frame-Inhalte nach folgendem Schema vom Server abzurufen:

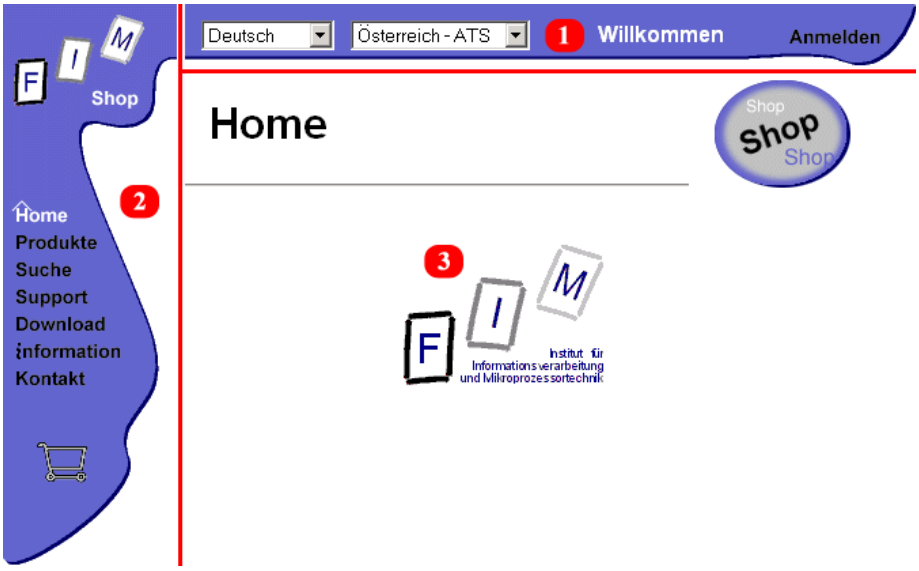



Abbildung 27: Shop-Website - Main-Frames

( 1 ) ..... `head.asp`
( 2 ) ..... `navi.asp`
( 3 ) ..... Shop-Websitemodul (default: `home.asp`)

```
<HTML>
<META http-equiv="Content-Script-Type" content="text/javascript">
<HEAD>
  <title>EShop</title>
</HEAD>

<FRAMESET COLS="134,*" BORDER=0>
  <FRAME NAME="navi" SRC="navi.asp" SCROLLING="no" MARGINHEIGHT=0 MARGINWIDTH=0
FRAMEBORDER=0>
  <FRAMESET ROWS="50,*" BORDER=0>
    <FRAME NAME="head" SRC="head.asp" SCROLLING="no" MARGINHEIGHT=0 MARGINWIDTH=0
FRAMEBORDER=0>
    <FRAME NAME="info" SRC="../<%=page%>.asp" MARGINHEIGHT=0 MARGINWIDTH=0
FRAMEBORDER=0>
  </FRAMESET>
</FRAMESET>

</HTML>
<%
' ============================================================================
%>
```

### EShop/system/head.asp

```
<!--#include file="../dbSys/dbV2.asp"-->
```

- Nun wird die Shop-Website-**Kopfzeile** durch dieses Skript generiert. Dabei passt sie sich automatisch an die gewählte Sprache (in den DB-Tabellen `seite` und `seite_text` wird dazu die interne `seitennr` 1 verwendet) an.

```
<%
' ============================================================================
' # Kopfzeile
' ----------------------------------------------------------------------------
  Response.Expires=-1440
  IF session("sprachnr")="" THEN Response.Redirect("Error.asp?sys=1")
' ----------------------------------------------------------------------------
 sel=session("sel")
 sprachnr=session("sprachnr")
 landnr=session("landnr")
 benutzer=session("benutzer")
' ----------------------------------------------------------------------------
 seitennr=1
 txt_willkommen=GetText(1)
 IF benutzer<>"" THEN txt_willkommen=txt_willkommen&" "&myHTML(benutzer)
 txt_meinedaten=GetText(2)
' ----------------------------------------------------------------------------
%>
```

- Das Aussehen verändert sich auch abhängig vom selektierten Land und vom Anmeldestatus eines Kunden.


Abbildung 28: Shop-Website - Kopfzeile

```
<HTML>
<META http-equiv="Content-Script-Type" content="text/javascript">
<HEAD>
```

```
23      <title>Head</title>
24    </HEAD>
25
26    <SCRIPT>
27     var item = new Array();
28     item[0]=new Image(); item[0].src="<%=sprachnr%>/anmelden1.gif";
29     item[1]=new Image(); item[1].src="<%=sprachnr%>/anmelden0.gif";
30     item[2]=new Image(); item[2].src="<%=sprachnr%>/abmelden1.gif";
31     item[3]=new Image(); item[3].src="<%=sprachnr%>/abmelden0.gif";
32
33     function change(x,s) {document.images[x].src=item[s].src;}
34    </SCRIPT>
35
36    <BODY background="<%=sprachnr%>/zeile.gif">
37    <FORM METHOD="POST" ACTION="main.asp" name="head" TARGET="aspshop">
38    <TABLE border=0 cellspacing=0 cellpadding=0 width=100%><TR>
39
40    <% IF benutzer="" THEN %>
41     <TD>   <select name="sprachnr"
42      onchange='document.head.submit();'><%
43    ' ------------------------------------------------------------------------------
44     sql="SELECT Sprachnr,Sprache FROM Sprache ORDER BY Sprache"
45     runsql db,rs,sql
46      DO UNTIL rs.EOF
47       s=""
48       IF sprachnr<>"" AND IsNumeric(optnr) THEN
49        IF CInt(sprachnr)=rs(0) THEN s=" selected"
50       END IF
51       %><option value="<%=rs(0)%>"<%
52       %><%=s&">"&myHTML(rs(1))%></option><%
53       rs.MoveNext
54      LOOP
55     endsql db,rs
56    ' ------------------------------------------------------------------------------
57    %></select>  
58    <select name="landnr" onchange='document.head.submit();'><%
59    ' ------------------------------------------------------------------------------
60     sql="SELECT Landnr,Land,Waehrung " & _
61         "FROM Land L, Waehrung W WHERE L.Waehrungsnr=W.Waehrungsnr " & _
62         "ORDER BY Land"
63     runsql db,rs,sql
64      DO UNTIL rs.EOF
65       s=""
66       IF landnr<>"" AND IsNumeric(landnr) THEN
67        IF CInt(landnr)=rs(0) THEN s=" selected"
68       END IF
69       %><option value="<%=rs(0)%>"<%
70       %><%=s&">"&myHTML(rs(1))&" - "&myHTML(rs(2))%></option><%
71       rs.MoveNext
72      LOOP
73     endsql db,rs
74    ' ------------------------------------------------------------------------------
75    %></select>    </TD>
76    <% ELSE %>
77     <TD>    
78      <input type=button value="<%=txt_meinedaten%>"
79       onclick="window.parent.document.location.href='main.asp?sel=-3';">
80     </TD>
81    <% END IF %>
82
83     <TD align=center><strong><font face="arial">
84      <font color="#ffffff"><%=txt_willkommen%></font>
85     </font></strong></TD>
86
87    <% IF benutzer="" THEN %><TD align=right><A HREF="main.asp?sel=-1"
88      onmouseover="change(0,0);"
89      onmouseout="change(0,<%IF sel="-1" THEN%>0<%ELSE%>1<%END IF%>);"
90      target="_parent"><IMG
```

```
91      src="<%=sprachnr%>/anmelden<%IF sel="-1" OR sel="-3" THEN
92      %>1<%ELSE%>0<%END IF%>.gif"
93      border="0" alt=""></A></TD>
94    <% ELSE %><TD align=right><A HREF="main.asp?sel=-2"
95      onmouseover="change(0,2);"
96      onmouseout="change(0,3);"
97      target="_parent"><IMG
98       src="<%=sprachnr%>/abmelden0.gif"
99       border="0" alt=""></A></TD>
100   <% END IF %>
101
102   </TR></TABLE>
103   </FORM>
104   </BODY>
105   </HTML>
106   <%
107   ' ================================================================================
108   %>
```

### EShop/system/navi.asp

- Um zwischen den einzelnen Shopmodulen wechseln zu können, wird durch dieses Skript eine **Navigationsleiste** ausgegeben. Die einzelnen Menüpunkte sind dabei im GIF-Format abgelegt. Ausgehend von `EShop/system` sind diese Grafiken sprachabhängig innerhalb eines mit der `sprachnr` betitelten Verzeichnisses abgelegt.


Abbildung 29: Shop-Website - Navigationsleiste

```
1     <%
2     ' ================================================================================
3     ' # Navigator
4     ' --------------------------------------------------------------------------------
5       Response.Expires=-1440
6       IF session("sprachnr")="" THEN Response.Redirect("Error.asp?sys=1")
7     ' --------------------------------------------------------------------------------
8      sel=session("sel")
9      sprachnr=session("sprachnr")
```

```
10  ' ------------------------------------------------------------------------
11  %>
12  <HTML>
13  <META http-equiv="Content-Script-Type" content="text/javascript">
14  <HEAD>
15    <title>Navi</title>
16  </HEAD>
17
18  <SCRIPT>
19   var item = new Array();
20   item[0]=new Image(); item[0].src="<%=sprachnr%>/home1.gif";
21   item[1]=new Image(); item[1].src="<%=sprachnr%>/home0.gif";
22   item[2]=new Image(); item[2].src="<%=sprachnr%>/produkte1.gif";
23   item[3]=new Image(); item[3].src="<%=sprachnr%>/produkte0.gif";
24   item[4]=new Image(); item[4].src="<%=sprachnr%>/suche1.gif";
25   item[5]=new Image(); item[5].src="<%=sprachnr%>/suche0.gif";
26   item[6]=new Image(); item[6].src="<%=sprachnr%>/support1.gif";
27   item[7]=new Image(); item[7].src="<%=sprachnr%>/support0.gif";
28   item[8]=new Image(); item[8].src="<%=sprachnr%>/download1.gif";
29   item[9]=new Image(); item[9].src="<%=sprachnr%>/download0.gif";
30   item[10]=new Image(); item[10].src="<%=sprachnr%>/information1.gif";
31   item[11]=new Image(); item[11].src="<%=sprachnr%>/information0.gif";
32   item[12]=new Image(); item[12].src="<%=sprachnr%>/kontakt1.gif";
33   item[13]=new Image(); item[13].src="<%=sprachnr%>/kontakt0.gif";
34   item[14]=new Image(); item[14].src="<%=sprachnr%>/warenkorb1.gif";
35   item[15]=new Image(); item[15].src="<%=sprachnr%>/warenkorb0.gif";
36   function change(x,s) {document.images[x].src=item[s].src;}
37  </SCRIPT>
38
39  <BODY bgcolor="#6666cc">
40  <TABLE border=0 cellspacing=0 cellpadding=0 width=101% bgcolor="#FFFFFF">
41  <TR height=144><TD><IMG src="<%=sprachnr%>/titel.gif" border="0" alt=""></TD></TR>
42  <TR height=25><TD valign="top"><A HREF="main.asp?sel=0"
43   onmouseover="change(1,0);"
44   onmouseout="change(1,<%IF sel=0 THEN%>0<%ELSE%>1<%END IF%>);"
45   target="_parent"><IMG
46    src="<%=sprachnr%>/home<%IF sel=0 THEN%>1<%ELSE%>0<%END IF%>.gif"
47    border="0" alt=""></A></TD></TR>
48  <TR height=20><TD valign="top"><A HREF="main.asp?sel=1"
49   onmouseover="change(2,2);"
50   onmouseout="change(2,<%IF sel=1 THEN%>2<%ELSE%>3<%END IF%>);"
51   target="_parent"><IMG
52    src="<%=sprachnr%>/produkte<%IF sel=1 THEN%>1<%ELSE%>0<%END IF%>.gif"
53    border="0" alt=""></A></TD></TR>
54  <TR height=20><TD valign="top"><A HREF="main.asp?sel=2"
55   onmouseover="change(3,4);"
56   onmouseout="change(3,<%IF sel=2 THEN%>4<%ELSE%>5<%END IF%>);"
57   target="_parent"><IMG
58    src="<%=sprachnr%>/suche<%IF sel=2 THEN%>1<%ELSE%>0<%END IF%>.gif"
59    border="0" alt=""></A></TD></TR>
60  <TR height=20><TD valign="top"><A HREF="main.asp?sel=3"
61   onmouseover="change(4,6);"
62   onmouseout="change(4,<%IF sel=3 THEN%>6<%ELSE%>7<%END IF%>);"
63   target="_parent"><IMG
64    src="<%=sprachnr%>/support<%IF sel=3 THEN%>1<%ELSE%>0<%END IF%>.gif"
65    border="0" alt=""></A></TD></TR>
66  <TR height=19><TD valign="top"><A HREF="main.asp?sel=4"
67   onmouseover="change(5,8);"
68   onmouseout="change(5,<%IF sel=4 THEN%>8<%ELSE%>9<%END IF%>);"
69   target="_parent"><IMG
70    src="<%=sprachnr%>/download<%IF sel=4 THEN%>1<%ELSE%>0<%END IF%>.gif"
71    border="0" alt=""></A></TD></TR>
72  <TR height=21><TD valign="top"><A HREF="main.asp?sel=5"
73   onmouseover="change(6,10);"
74   onmouseout="change(6,<%IF sel=5 THEN%>10<%ELSE%>11<%END IF%>);"
75   target="_parent"><IMG
76    src="<%=sprachnr%>/information<%IF sel=5 THEN%>1<%ELSE%>0<%END IF%>.gif"
77    border="0" alt=""></A></TD></TR>
```

```
78  <TR height=12><TD valign="top"><A HREF="main.asp?sel=6"
79   onmouseover="change(7,12);"
80   onmouseout="change(7,<%IF sel=6 THEN%>12<%ELSE%>13<%END IF%>);"
81   target="_parent"><IMG
82    src="<%=sprachnr%>/kontakt<%IF sel=6 THEN%>1<%ELSE%>0<%END IF%>.gif"
83    border="0" alt=""></A></TD></TR>
84  <TR height=131><TD valign="top"><A HREF="main.asp?sel=7"
85   onmouseover="change(8,14);"
86   onmouseout="change(8,<%IF sel=7 THEN%>14<%ELSE%>15<%END IF%>);"
87   target="_parent"><IMG
88    src="<%=sprachnr%>/warenkorb<%IF sel=7 THEN%>1<%ELSE%>0<%END IF%>.gif"
89    border="0" alt=""></A></TD></TR>
90  </TABLE>
91  <TABLE border=0 cellspacing=0 cellpadding=0
92   width=101% height=101% bgcolor="#FFFFFF">
93  <TR><TD> </TD></TR>
94  </TABLE>
95
96  </BODY>
97  </HTML>
98  <%
99  ' ================================================================================
100 %>
```

### EShop/default.asp

- Das **Startskript** der Online-Shop-Website: Es generiert 2 übereinanderliegende Frames, wobei im oberen das `main.asp` - Skript aufgerufen wird und im unteren eine Copyright-Zeile ( `definfo.asp` ) dargestellt wird.

(c) Georg Kotek und Gernot Ritz

Abbildung 30: Shop-Website - Copyright-Zeile

```
1   <HTML>
2   <META http-equiv="Content-Script-Type" content="text/javascript">
3   <HEAD>
4     <title>ASPShop</title>
5   </HEAD>
6
7   <FRAMESET ROWS="*,32" BORDER=0>
8     <FRAME NAME="aspshop" SRC="system/main.asp" SCROLLING="no" MARGINHEIGHT=0
9   MARGINWIDTH=0 FRAMEBORDER=0>
10    <FRAME NAME="aspinfo" SRC="definfo.asp" SCROLLING="no" MARGINHEIGHT=0
11  MARGINWIDTH=0 FRAMEBORDER=0>
12  </FRAMESET>
13
14  </HTML>
```

Nach Eingabe der Web-Adresse des Online-Shops werden nun automatisch folgende ASP-Scripts durch den **Web-Browser** aufgerufen:

| | | |
|---|---|---|
| `default.asp` → | `main.asp` → `definfo.asp` | `navi.asp` `head.asp` `home.asp` |

Danach kann der Kunde leicht mittels der links befindlichen Navigationsleiste zwischen den einzelnen Shopmodulen (Home, Produkte, Suche, Support, Download, Information, Kontakt und Warenkorb) wechseln.

Der beste Online-Shop macht aber nur dann Sinn, wenn man sich als Kunde auch am **System** anmelden kann. Dazu gibt es am rechten oberen Rand den „Anmelden"-Button, welcher zum **Anmeldeformular** der Shop-Website wechselt. Daraufhin kann sich ein bereits registrierter Benutzer leicht per Eingabe von **Benutzernamen und Passwort** einloggen (Groß- und Kleinschreibung ist dabei unrelevant, z.B.: gernot / gernot ) und dann mittels Warenkorb einkaufen.

Dabei sieht der **Ablauf** für den Benutzer folgendermaßen aus:

| | | | | |
|---|---|---|---|---|
| (1) **Anmeldeformular** | → | *Anmelden* ⇒ Eingabe akzeptiert | → | (2) **Home** |
| | → | *Anmelden* ⇒ Eingabe inkorrekt | → | (1) |
| | → | *Neu anmelden* | → | (3) **Neu anmelden** |

**EShop/anmelden.asp**

```
1   <!--#include file="dbSys/dbV2.asp"-->
```

- Dieses Skript gibt nur das **Anmeldeformular** zum Online-Shop aus. Die eigentliche Anmeldung (nach Drücken des „Anmelden"-Buttons) wird durch das `main.asp` - Skript erledigt.

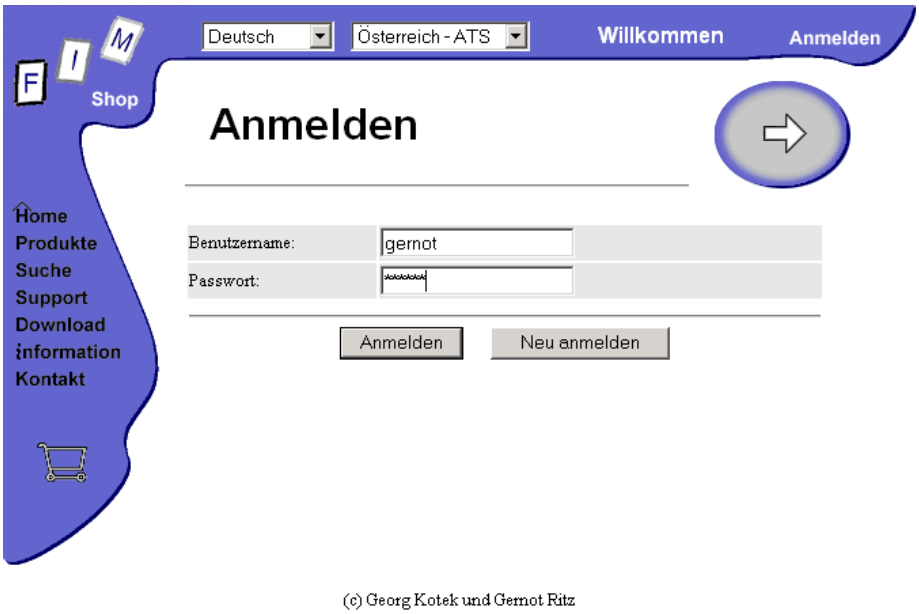



Abbildung 31: Shop-Website - Anmeldung

```
2   <%
3   ' ================================================================================
4   ' # Login-Seite
5   '   file: anmelden.asp
6   ' --------------------------------------------------------------------------------
```

```
7       Response.Expires=-1440
8       IF session("sprachnr")="" THEN Response.Redirect("System/Error.asp")
9     ' ---------------------------------------------------------------------------
10    ' # Sprachabhängige Texte lesen:
11     sprachnr=session("sprachnr")
12     seitennr=2
13     txt_titel=GetText(1)
14     txt_benutzer=GetText(2)
15     txt_passwort=GetText(3)
16     txt_anmelden=GetText(4)
17     txt_neu=GetText(5)
18    ' ===========================================================================
19    %>
20    <html>
21    <head>
22     <title>Anmelden</title>
23    </head>
24
25    <body>
26    <form METHOD="POST" ACTION="system/main.asp" name="anmelden" target="aspshop">
27    <table width=100% border=0 cellspacing=0 cellpadding=0>
28     <tr>
29      <td><br><h1><strong><font face="arial">
30        <%=txt_titel%></font></strong></h1><hr></td>
31      <td width=120 align=right><img src="system/anmelden.gif" border=0></td>
32      <td width=32> </td>
33     </tr>
34    </table><br>
35
36    <table width=90% border=0 cellspacing=2 cellpadding=1>
37     <tr>
38      <td bgcolor=#e4e4e4 width=30%><small><%=txt_benutzer%>:</small></td>
39      <td bgcolor=#e4e4e4 widht=70%><input name="user" type=text size=20></td>
40     </tr>
41     <tr>
42      <td bgcolor=#e4e4e4 width=30%><small><%=txt_passwort%>:</small></td>
43      <td bgcolor=#e4e4e4 widht=70%><input name="pwd" type=password size=20></td>
44     </tr>
45    </table>
46
47    <table width=90% border=0 cellspacing=2 cellpadding=1>
48     <tr><td align=center><hr>
49      <input type=submit value="<%=txt_anmelden%>">    
50      <input type=button value="<%=txt_neu%>"
51       onclick="window.parent.document.location.href='system/main.asp?sel=-3';">
52     </td></tr>
53    </table>
54
55    </form>
56    </body>
57    </html>
58    <%
59    ' ===========================================================================
60    %>
```

### 5.5.2 Produkte

Repräsentativ für die Implementierungstechnik aller Shopmodule wird in diesem Kapitel nur das **Produkt-Skript** näher beschrieben, um das Zusammenspiel mit den bereits besprochenen System-Dateien genauer zu illustrieren.

Für den Kunden übernimmt das Produkt-Shopmodul die Darstellung des **Online-Produktkatalog**es und bietet ihm die Möglichkeit, ausgewählte Artikel in seinen eigenen **Warenkorb** abzulegen. Dabei werden dem Benutzer verschiedene Seiten-Ansichten zur Verfügung gestellt:

- alle Produktgruppen
- die Produkte einer Gruppe
- ein ausgewählter Artikel

Das Webinterface hält sich dabei an folgenden groben **Ablauf (** → Seiten-**Modi )**:

| | | | | |
|---|---|---|---|---|
| (1) **Produktgruppen** | → | *Gruppe aus Liste* | → | (2) |
| … | | | | … |
| (2) **Produkte einer Gruppe** | → | *Produkt aus Liste* | → | (3) |
| | → | *Pfeil zurück* | → | (1) |
| … | | | | … |
| (3) **Ausgewähltes Produkt** | → | *In den Warenkorb* | → | (4) |
| | → | *Pfeil zurück* | → | (2) |
| … | | | | … |
| (4) **Produkt im Warenkorb** | → | *Pfeil zurück* | → | (2) |

**EShop/produkte.asp**

```
1   <!--#include file="dbSys/dbV2.asp"-->
```

- Am Skriptbeginn werden zunächst diverse **GET-Parameter** (`gr…Gruppennr`, `nr...Artikelnr`, `fn...Funktion: "" | "warenkorb"`) von eventuell vorangegangenen Aufrufen eingelesen.
- Zusätzlich werden einige sprachabhängige Seitentexte und die Währung zum aktuell eingestellten Land aus der DB geladen.

```
    <%
3   ' ================================================================================
4   ' # Produkte-Seite
5   '   file: produkte.asp
6   ' --------------------------------------------------------------------------------
7     Response.Expires=-1440
8     IF session("sprachnr")="" THEN Response.Redirect("System/Error.asp")
9   ' --------------------------------------------------------------------------------
10  ' # Get-Werte lesen:
11    gr=Request.QueryString("gr")
12    nr=Request.QueryString("nr")
13    fn=Request.QueryString("fn")
14  ' --------------------------------------------------------------------------------
15  ' # Spracheabhängige Texte lesen:
16    sprachnr=session("sprachnr")
17    seitennr=21
18    txt_titel=GetText(1)
19    txt_artnr=GetText(8)
20    txt_preis=GetText(9)
21    txt_exlusive=GetText(10)
22    txt_bestellt=GetText(16)
```

```
23  ' ------------------------------------------------------------------------------
24  ' # Aktuelle Währung auslesen:
25   sql="SELECT Waehrung" & _
26       " FROM Land L, Waehrung W WHERE L.Waehrungsnr=W.Waehrungsnr AND" & _
27       " L.Landnr='"&session("landnr")&"'"
28   runsql db,rs,sql
29    waehrung=rs("Waehrung")
30   endsql db,rs
31  ' ------------------------------------------------------------------------------
32  %>
33  <html>
34  <head>
35   <title>Shop</title>
36  </head>
37
38  <body link=#000000 vlink=#000000 alink=#000000>
39
```

- Der folgende Programmabschnitt dient zur Generierung der **Produktgruppen-Seite**. Die erzeugten Links zu den Gruppen benutzen dabei den URL-Anhang "`?gr=...`", um auf die jeweiligen Produkte zu verweisen.


Abbildung 32: Shop-Website - Produktgruppen

```
40  <%
41  ' ==============================================================================
42  ' # Gruppenübersicht aller Produkte
43  ' ------------------------------------------------------------------------------
44   IF gr="" AND nr="" AND fn="" THEN
45  ' ------------------------------------------------------------------------------
46  %>
47  <table width=100% border=0 cellspacing=0 cellpadding=0>
48   <tr>
49    <td><br><h1><strong><font face="arial">
50      <%=txt_titel%></font></strong></h1><hr></td>
51    <td width=120 align=right><img src="system/shop.gif" border=0></td>
52    <td width=32> </td>
53   </tr>
54  </table>
55  <br>
```

```
<table width=90% border=0 cellspacing=2 cellpadding=1>
<%
' ------------------------------------------------------------------------------
' # Produktgruppen ausgeben:
 sql="SELECT * FROM Produktgruppe_Text WHERE" & _
     " Sprachnr='"&session("sprachnr")&"' AND" & _
     " Gruppenname<>'?' ORDER BY Gruppenname"
 runsql db,rs,sql
  DO UNTIL rs.EOF
   %><tr><td width=5% bgcolor=#e4e4e4 align=center>
   <img src="system/select.gif" border=0></td>
   <td width=95% bgcolor=#e4e4e4><a href="produkte.asp?gr=<%=rs("Gruppennr")%>">
   <small><%=myHTML(rs("Gruppenname"))%></small></a></td></tr><%
   rs.MoveNext
  LOOP
 endsql db,rs
' ------------------------------------------------------------------------------
%>
</table>
<%
' ------------------------------------------------------------------------------
 END IF
' ==============================================================================
```

- Nun folgt der Scriptcode zur Ausgabe aller **Produkte der zuvor selektierten Gruppe**. Alle generierten Links benutzen nun den erweiterten URL-Anhang "`?gr=...&nr=...`", um in der Folge die einzelnen Artikel adressieren zu können.

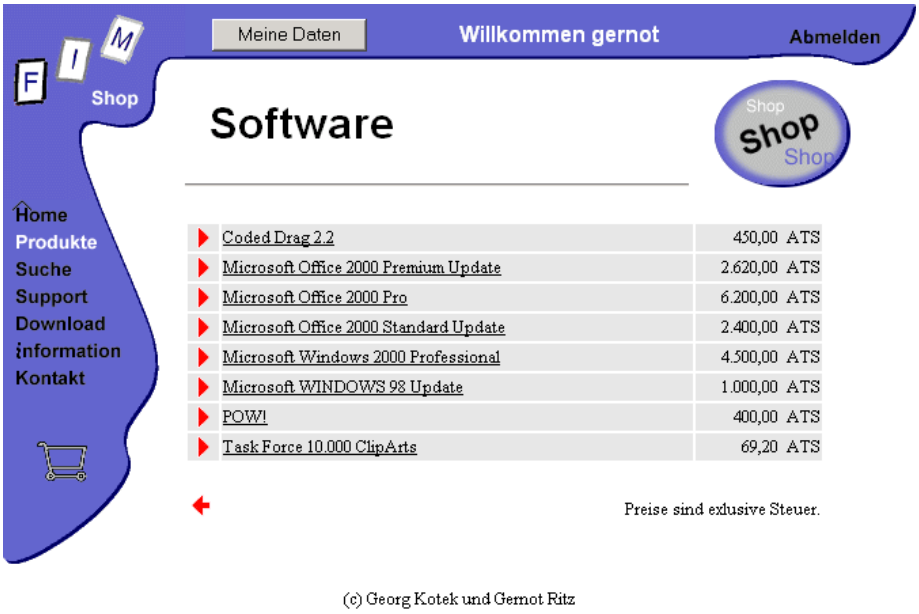



Abbildung 33: Shop-Website - Produkte einer Gruppe

```
' ==============================================================================
' # Produkte einer Gruppe
' ------------------------------------------------------------------------------
 IF gr<>"" AND nr="" AND fn="" THEN
' ------------------------------------------------------------------------------
' # Produktgruppenname:
 sql="SELECT * FROM Produktgruppe_Text WHERE" & _
     " Sprachnr='"&session("sprachnr")&"' AND Gruppennr='"&gr&"'"
```

```
 runsql db,rs,sql
  txt_titel=myHTML(rs("Gruppenname"))
 endsql db,rs
' ------------------------------------------------------------------------------
%>
<table width=100% border=0 cellspacing=0 cellpadding=0>
 <tr>
  <td><br><h1><strong><font face="arial">
    <%=txt_titel%></font></strong></h1><hr></td>
  <td width=120 align=right><img src="system/shop.gif" border=0></td>
  <td width=32> </td>
 </tr>
</table>
<br>
<table width=90% border=0 cellspacing=2 cellpadding=1>
<%
' ------------------------------------------------------------------------------
' # Produkte ausgeben:
 sql="SELECT * FROM Produkt P, Produkt_Text T, Preis V WHERE" & _
     " Gruppennr='"&gr&"' AND Landnr='"&session("landnr")&"' AND" & _
     " P.Artikelnr=T.Artikelnr AND P.Artikelnr=V.Artikelnr AND" & _
     " Sprachnr='"&session("sprachnr")&"' AND Aktiv=1" & _
     " ORDER BY Artikelbezeichnung"
 runsql db,rs,sql
  DO UNTIL rs.EOF
   %><tr><td width=5% bgcolor=#e4e4e4 align=center>
    <img src="system/select.gif" border=0></td>
   <td width=75% bgcolor=#e4e4e4>
    <a href="produkte.asp?gr=<%=gr%>&nr=<%=rs("Artikelnr")%>">
    <small><%=myHTML(rs("Artikelbezeichnung"))%></small></a></td>
   <td width=20% bgcolor=#e4e4e4 align=right><small>
    <%=formatnumber(rs("Preis"))%>  <%=waehrung%></small></td></tr><%
   rs.MoveNext
  LOOP
 endsql db,rs
' ------------------------------------------------------------------------------
%>
</table>
<br>
<table width=90% border=0 cellspacing=2 cellpadding=1>
 <tr>
  <td align=left>
   <a href="produkte.asp"><img src="system/return.gif" border=0"></a>
  </td><td align=right>
   <small><%=txt_exlusive%></small>
  </td>
 </tr>
</table>
<%
' ------------------------------------------------------------------------------
 END IF
' ==============================================================================
```

- Nachdem nun die verfügbaren Produktgruppen und die jeweils dazugehörigen Produktlisten dargestellt werden können, folgt nun der Programmabschnitt, welcher einen einzelnen **selektierten Artikel** ausgibt. Der Link „in den Warenkorb“ bekommt dabei den nochmals erweiterten URL-Anhang "`?gr=...&nr=...&fn=warenkorb`".
- Zusätzlich wird noch kurz vor der Ausgabe der Seite die SQL-Prozedur `Produktanzeige Log Write` aufgerufen, welche über die dargestellten

Artikel Buch führt. Dadurch ist es z.B. möglich, mit Hilfe des Backoffice-Analysemoduls wertvolle Rückschlüsse auf das Kundeninteresse an einzelnen Produkten ziehen zu können.

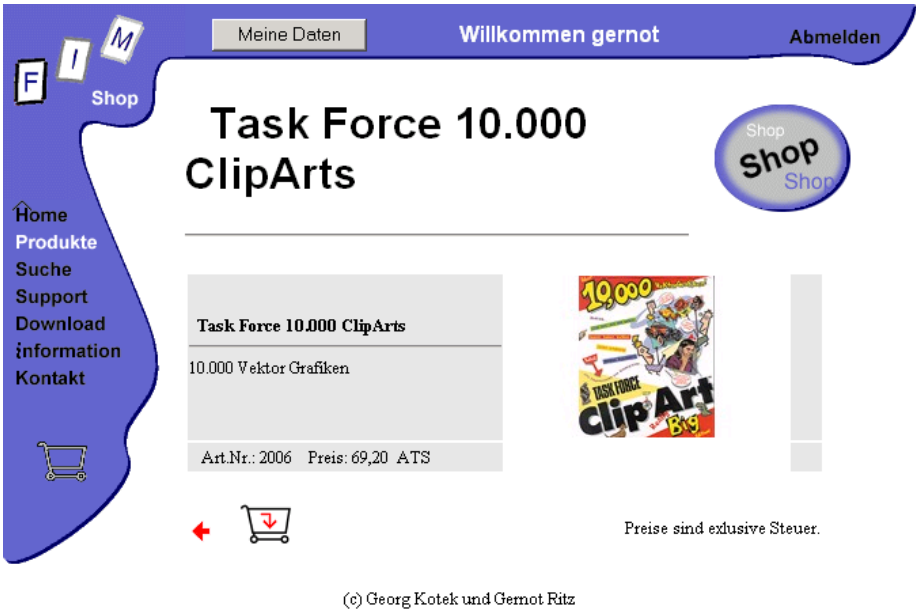



Abbildung 34: Shop-Website - Ausgewähltes Produkt

```
143  ' ================================================================================
144  ' # Ausgewähltes Produkt
145  ' --------------------------------------------------------------------------------
146   IF gr<>"" AND nr<>"" AND fn="" THEN
147  ' --------------------------------------------------------------------------------
148  ' # Produktdaten holen:
149   sql="SELECT * FROM Produkt P, Produkt_Text T, Preis V WHERE" & _
150       " P.Artikelnr='"&nr&"' AND Landnr='"&session("landnr")&"' AND" & _
151       " P.Artikelnr=T.Artikelnr AND P.Artikelnr=V.Artikelnr AND" & _
152       " Sprachnr='"&session("sprachnr")&"'"
153   runsql db,rs,sql
154    txt_titel=myHTML(rs("Artikelbezeichnung"))
155    txt_beschreibung=myHTML(rs("Artikelbeschreibung"))
156    num_artnr=rs("Artikelnr")
157    num_preis=formatnumber(rs("Preis"))
158    lnk_hersteller=rs("Herstellerlink") & ""
159    lnk_bild=rs("Bild") & ""
160    IF lnk_bild="" THEN lnk_bild="noimage.gif"
161    IF lnk_hersteller<>"" THEN
162     txt_beschreibung=txt_beschreibung&"<br> <br><a href='http://"& _
163                      lnk_hersteller&"' target='_blank'>"&lnk_hersteller&"</a>"
164    END IF
165   endsql db,rs
166   ' # Produktanzeige_Log schreiben:
167   IF session("benutzer")<>"" THEN
168    runproc db,RS,command
169    command.CommandText = "Produktanzeige_Log_Write"
170    command.Parameters.Append command.CreateParameter _
171     ("@artikelnr", 200, 1, 20, nr)
172    command.Parameters.Append command.CreateParameter _
173     ("@kundennr", 3, 1, 4, session("kundennr"))
174    SET RS=command.Execute()
175    CheckForError
176    endproc db,RS
```

```
 ELSE
  runproc db,RS,command
  command.CommandText = "Produktanzeige_Log_Write"
  command.Parameters.Append command.CreateParameter _
   ("@artikelnr", 200, 1, 20, nr)
  command.Parameters.Append command.CreateParameter _
   ("@kundennr", 3, 1, 4, unknownuser)
  SET RS=command.Execute()
  CheckForError
  endproc db,RS
 END IF
' ------------------------------------------------------------------------------
%>
<table width=100% border=0 cellspacing=0 cellpadding=0>
 <tr>
  <td><br><h1><strong><font face="arial">
    <%=txt_titel%></font></strong></h1><hr></td>
  <td width=120 align=right><img src="system/shop.gif" border=0></td>
  <td width=32> </td>
 </tr>
</table>
<br>
<table width=90% border=0 cellspacing=2 cellpadding=1>
 <tr>
  <td width=50% bgcolor=#e4e4e4><small><b>  <%=txt_titel%></b><hr>
   <%=txt_beschreibung%><br> </small></td>
  <td width=45% align=center><img src="images/<%=lnk_bild%>" border=0></td>
  <td width=5% bgcolor=#e4e4e4> </td>
 </tr><tr>
  <td width=50% bgcolor=#e4e4e4><small>  
   <%=txt_artnr%>: <%=num_artnr%>   
   <%=txt_preis%>: <%=num_preis%>  <%=waehrung%></small></td>
  <td width=45%> </td><td width=5% bgcolor=#e4e4e4> </td>
 </tr>
</table>
<br>
<table width=90% border=0 cellspacing=2 cellpadding=1>
 <tr>
  <td align=left>
   <a href="produkte.asp?gr=<%=gr%>"><img src="system/return.gif" border=0></a>
      
<% IF session ("benutzer")<>"" THEN%>
<a href="produkte.asp?gr=<%=gr%>&nr=<%=nr%>&fn=warenkorb">
<%  ELSE %>
<a href="anmelden.asp">
<%  END IF %>
   <img src="system/indenkorb.gif" border=0></a>
  </td><td align=right>
   <small><%=txt_exlusive%></small>
  </td>
 </tr>
</table>
<%
' ------------------------------------------------------------------------------
 END IF
' ==============================================================================
```

- Der nun folgende Abschnitt dient zum Ablegen des zuvor gewählten Produktes im Kunden-Warenkorb. Dazu benutzt das ASP-Skript die SQL-Prozedur `Warenkorb_Write`.
- Anschließend wird noch eine kurze Warenkorb-Bestätigung ausgegeben:


Abbildung 35: Shop-Website - Produkt im Warenkorb

```
235 ' ========================================================================
236 ' # Ausgewähltes Produkt in den Warenkorb geben
237 ' ------------------------------------------------------------------------
238  IF gr<>"" AND nr<>"" AND fn="warenkorb" THEN
239 ' ------------------------------------------------------------------------
240 ' # Produkt in den Korb:
241  runproc db,RS,command
242  command.CommandText = "Warenkorb_Write"
243  command.Parameters.Append command.CreateParameter _
244   ("@artikelnr", 200, 1, 20, nr)
245  command.Parameters.Append command.CreateParameter _
246   ("@kundennr", 3, 1, 4, session("kundennr"))
247  command.Parameters.Append command.CreateParameter _
248   ("@anzahl", 3, 1, 4, 1)
249  SET RS=command.Execute()
250  CheckForError
251  endproc db,RS
252 ' ------------------------------------------------------------------------
253 ' # Produktdaten holen:
254  sql="SELECT * FROM Produkt P, Produkt_Text T, Preis V WHERE" & _
255      " P.Artikelnr='"&nr&"' AND Landnr='"&session("landnr")&"' AND" & _
256      " P.Artikelnr=T.Artikelnr AND P.Artikelnr=V.Artikelnr AND" & _
257      " Sprachnr='"&session("sprachnr")&"'"
258  runsql db,rs,sql
259   txt_titel=myHTML(rs("Artikelbezeichnung"))
260   num_artnr=rs("Artikelnr")
261  endsql db,rs
262 ' ------------------------------------------------------------------------
263 %>
264 <table width=100% border=0 cellspacing=0 cellpadding=0>
265  <tr>
266   <td><br><h1><strong><font face="arial">
267     <%=txt_titel%></font></strong></h1><hr></td>
268   <td width=120 align=right><img src="system/shop.gif" border=0></td>
269   <td width=32> </td>
270  </tr>
271 </table>
272 <br>
273 <table width=90% border=0 cellspacing=2 cellpadding=1>
274  <tr>
```

```
275    <td bgcolor=#e4e4e4><small> <br>
276       <%=txt_artnr%>: <%=num_artnr%>
277      - <b>1 <%=txt_titel%> <%=txt_bestellt%></b>
278     <br> </small></td>
279   </tr>
280  </table>
281  <br> 
282  <a href="produkte.asp?gr=<%=gr%>">
283   <img src="system/return.gif" border=0></a>
284  <%
285  ' ------------------------------------------------------------------------
286   END IF
287  ' ========================================================================
288  %>
289
290  </body>
291  </html>
292  <%
293  ' ========================================================================
294  %>
```

### Stored-Procedure: Produktanzeige_Log_Write

- Die nachfolgende SQL-Prozedur **zählt**, wie oft ein **Artikel** (`@artikelnr`) von einem Kunden (`@kundennr`) an einem Tag zur **Ansicht** abgerufen wurde. Dazu wird einmal am Tag pro Kunde und Artikel ein neuer Eintrag in die DB-Tabelle `"Produktanzeige_Log"` geschrieben und jeder weitere Zugriff durch einfaches Erhöhen des Wertes im DB-Feld `"Zaehler"` mitprotokolliert. Alle anonymen Benutzer werden dabei durch die Kundennummer `"unknownuser"` (wird im Kapitel 5.4.1 innerhalb der Datei `dbConnection.asp` definiert) repräsentiert.

```
   CREATE PROCEDURE Produktanzeige_Log_Write (
2  @artikelnr VARCHAR(20),
3  @kundennr  INT) AS
4
5  BEGIN
6  @id INT
7  BEGIN
8    TRAN
9      IF Exists(SELECT id
10                 FROM Produktanzeige_Log
11                 WHERE Artikelnr=@Artikelnr
12                   AND Kundennr=@Kundennr
13                   AND CONVERT(VARCHAR(10),Datum)
14                       =CONVERT(VARCHAR(10),Getdate()))
15       UPDATE Produktanzeige_Log
16         SET zaehler = zaehler + 1
17         WHERE Artikelnr=@Artikelnr
18           AND Kundennr=@Kundennr
19           AND CONVERT(VARCHAR(10),Datum)
20               =CONVERT(VARCHAR(10),Getdate())
21     ELSE
22       INSERT INTO Produktanzeige_Log
23                (Artikelnr, Zaehler, Kundennr, Datum)
24         VALUES (@artikelnr, 1, @kundennr, GETDATE())
25
26
27   IF @@error=0 COMMIT
28     ELSE ROLLBACK
```

**Stored-Procedure: Warenkorb_Write**

- Diese SQL-Prozedur **legt** einen bestimmten **Artikel** (`@artikelnr`) **in den Warenkorb** eines Kunden (`@kundennr`). Dabei kann auch die zu bestellende Stückzahl (`@anzahl`) des angegebenen Produktes bestimmt werden. Auch prüft diese Funktion, ob der gewünschte Artikel online verfügbar ist oder nur per Spedition ausgeliefert werden kann.

```
CREATE PROCEDURE Warenkorb_Write (
@artikelnr    VARCHAR(20),
@kundennr     INT,
@anzahl       INT) AS

DECLARE @online VARCHAR(50)
/* schreibt Artikelnummer, Anzahl, Kundennummer und Datum in
   den Warenkorb */
BEGIN

TRANSACTION
  SELECT @online=P.Downloadlink
    FROM Produkt AS P WHERE P.Artikelnr=@artikelnr

  IF @online IS NOT NULL
  /* Downloadlink nicht null -> Produkt online verfügbar */
    INSERT INTO Warenkorb
    (Artikelnr, Kundennr, Anzahl, Datum, Ausgewaehlt, Versand)
     VALUES (@artikelnr, @kundennr, @anzahl,
             CONVERT(DateTime,GETDATE(),104), 'J', 'O')
  ELSE
  /* Downloadlink = null -> Produkt online nicht verfügbar */
    INSERT INTO Warenkorb
    (Artikelnr, Kundennr, Anzahl, Datum, Ausgewaehlt, Versand)
    VALUES (@artikelnr, @kundennr, @anzahl,
            CONVERT(DateTime,GETDATE(),104), 'J', 'S')

  IF @@error = 0 COMMIT
    ELSE ROLLBACK
```

### 5.5.3 Warenkorb

Nachdem der Kunde mit Hilfe eines der beiden Shopmodule „Produkte“ oder „Suche“ seine gewünschten Artikel in den Warenkorb abgelegt hat, ist es mit dem Modul „Warenkorb“ möglich die aktuelle Bestellung zu modifizieren und abzusenden.

Der Benutzer wird dabei durch einen 3stufigen Warenkorb-Assistenten geschleust:

1. **Warenkorb** → Hier wird eine Übersicht der zur Zeit abgelegten Produkte angezeigt, dabei werden die Artikel pro Versandart (Online und Post/Spedition) getrennt aufgelistet. Auch kann der Benutzer noch diverse Veränderungen am Inhalt des Warenkorbs durchführen: Artikel löschen, Bestellmengen korrigieren, Versandart wechseln, einzelne Artikel vom Kauf ausschließen (für später aufheben).


Abbildung 36: Shop-Website - Warenkorb

2. **Bestellen** → Nun wird die gesamte zu tätigende Bestellung noch einmal zur Kontrolle komplett dargestellt. Der Kunde kann zu diesem Zeitpunkt noch die gewünschte Zahlungsart einstellen bevor er das Formular absendet bzw. die laufende Bestellung abbrechen.
3. **Bestätigung** → Abschließend wird eine Bestätigung ausgegeben, welche die korrekte Durchführung der Bestellung quittiert.

Abschließend sei noch erwähnt, dass Produkte, welche im Online-Versand-Modus gekauft wurden, zu einem späteren Zeitpunkt vom jeweiligen Kunden mit dem Shopmodul „Download“ heruntergeladen werden können.

## 5.6 Installation

Der gesamte Online-Shop (**EShop Version 1.0**) befindet sich gezippt auf der beigelegten CD im Ordner „Projekte“. Bitte beachten sie bei einer **Installation** folgende Punkte:

- Folgende Basisprodukte werden benötigt:
  - Windows-NT4-Server basierendes OS
  - MS-IIS 4 oder höher!
  - MS-SQL-Server 7 oder höher!
- Zuerst müssen 2 Datenbanken im SQL-Server installiert werden:
  - DB „install“ erstellen und Backup aus „DB.zip“ einspielen.
  - DB „eshop_dev” erstellen und passendes Backup einspielen.
- Nun eine „Web Site“ oder ein „Virtual Directory“ namens „EShop“ im IIS erstellen:
  - Dieser Eintrag im IIS benötigt einen „anonymous access“.
  - Die benötigten Web-Files liegen im „EShop.zip“.
- Gegebenenfalls muss im File „EShop\dbSys\dbConnection.asp” noch der ADO-Connection-String angepasst werden.

# 6 Fallbeispiel: Application-Service-Providing

Bei dem nun folgenden zweiten Fallbeispiel handelt es sich um ein **Web-Portal für Sportvereine** und deren Mitglieder. Die Umsetzung wurde im Rahmen eines Projektes der Firma **Wellness-Soft** mit Unterstützung von Fabasoft und FIM erstellt. Die Basis-Idee und Grundkonzepte stammen von Frau Dipl.-Ing. Susanne Reisinger.

Ziel der Arbeit war ein System zu schaffen, welches sowohl für beliebige Sportvereine die Möglichkeit bietet im Internet (www - oder später auch - wap) präsent zu sein, wie auch für die einzelnen Mitglieder sich aktuell und clubübergreifend informieren, bzw. Anmeldungen zu diversen Veranstaltungen tätigen zu können.

## 6.1 Projektbeschreibung

Es sollte ein Web-Portal in Form eines **ASP** (Application-Service-Providing) - **Dienst**es entstehen, welches die Möglichkeit bietet die Darstellungsform leicht und effizient anpassen zu können (html, wml, ...). Auch der Funktionsumfang sollte sich ohne übermäßigen Aufwand erweitern lassen. Zur internen Verwaltung war ein ausgereiftes eCRM-System zu verwenden.

Nach einer Evaluationsphase fiel dann die Entscheidung der **System-Plattform** auf folgende Konfiguration:

- MS-Windows NT, 2000 oder XP
- MS-Internet-Infomation-Server inklusive ASP-JavaScript
- MS-SQL-Server 7 / 2000
- Fabasoft Components Version 4.0 RC1
- + ComponentsBase
- + ComponentsWeb

Der Hauptgrund für die getroffene Wahl waren die generischen Eigenschaften der Fabasoft-VApps (siehe Kapitel 4.4), welche in den Fabasoft-Components Version 4.0 Release-Candidate 1 bereits implementiert waren!

### 6.1.1 Sport-Portal

Nachdem das Web-Portal für beliebige Sportvereine und deren Klubmitglieder als auch für (vorerst) anonyme Surfer geeignet sein sollte, wurde ein System mit benutzerabhängigen **Zugriffsrechte**n **pro Sportklub** (Start → Benutzer → Experte → Administrator) angestrebt.

Jeder **Benutzer** sollte sich darüber hinaus nur einmal am **Portal anmelden** müssen, um alle seine **Klubs** und eventuell hierarchisch darunter liegende **Sektionen**, bei denen er eingetragen ist, visitieren zu können: z.B.

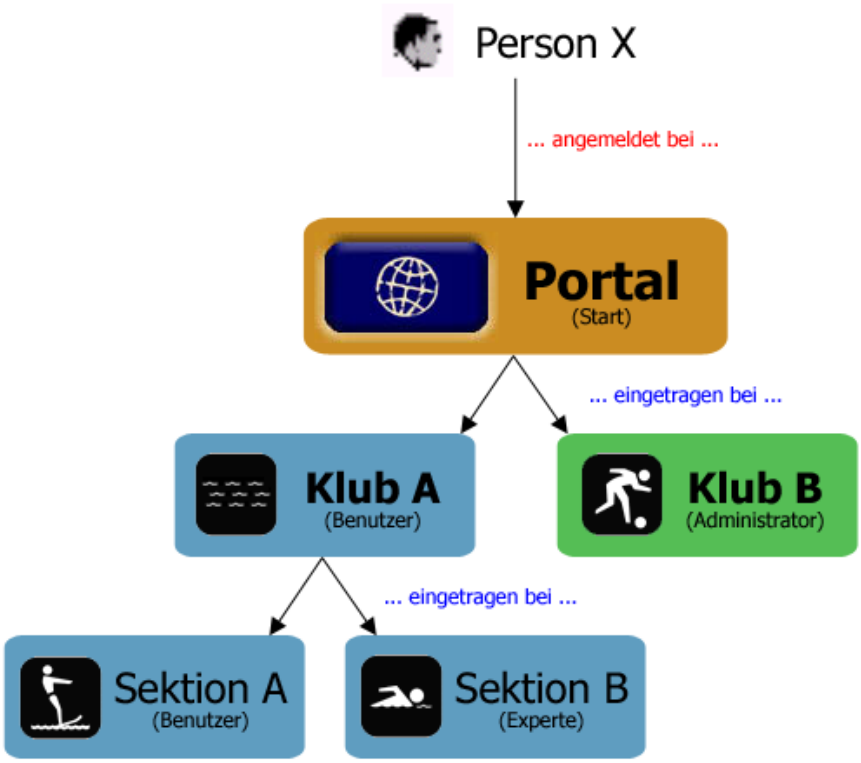



Abbildung 37: Sport-Portal - Struktur

Aufbauend auf diesem Konzept wurden in der Projektplanungsphase anschließend folgende **Features** festgelegt:

- flexibles und **sprachunabhängige**s Design
- individuell einrichtbare Homepage pro Klub
- leichte **Navigation innerhalb** der **Klubs**, bei welchen der Benutzer eingetragen ist
- **Newsboard**: Aktuelles über das Portal und die einzelnen Vereine
- **Clubverzeichnis**: Vereinssuche nach Namen, Region, Sportart → Klubdaten & Infos
- **Events**: Veranstaltungslisten der Klubs → Detailinfos & An- und Abmeldung
- **Verwaltung**: Mitglied werden (eintragen bei), Klubstammdaten, Mitgliederdaten, ...
- **FAQs** zur **Bedienung des Portals** → vollständig verfügbar auf der CD im ZIP-File:

```
"\Projekte\Wellness-Sport\Wellness-Sport.zip"
unter der Datei:  "\Wellness-Sport\faqs\faqs_COO.1.1.1.10.html"
```

### 6.1.2 Mögliche Erweiterungen

Für zukünftige Versionen des Sport-Portals wurden anschließend noch folgende erweiterte Funktionalitäten notiert:

- Spielstätten- / Turnier- / Spielergebnisverwaltung
- Forum für Vereinsmitglieder
- Abstimmungen über das Web

## 6.2 Objektmodell

Nun wurde das dazupassende Objektmodell zum Speichern aller Daten des Sport-Portales in einer eigens entwickelten **SportBase** - Fabasoft-Komponente abgelegt:

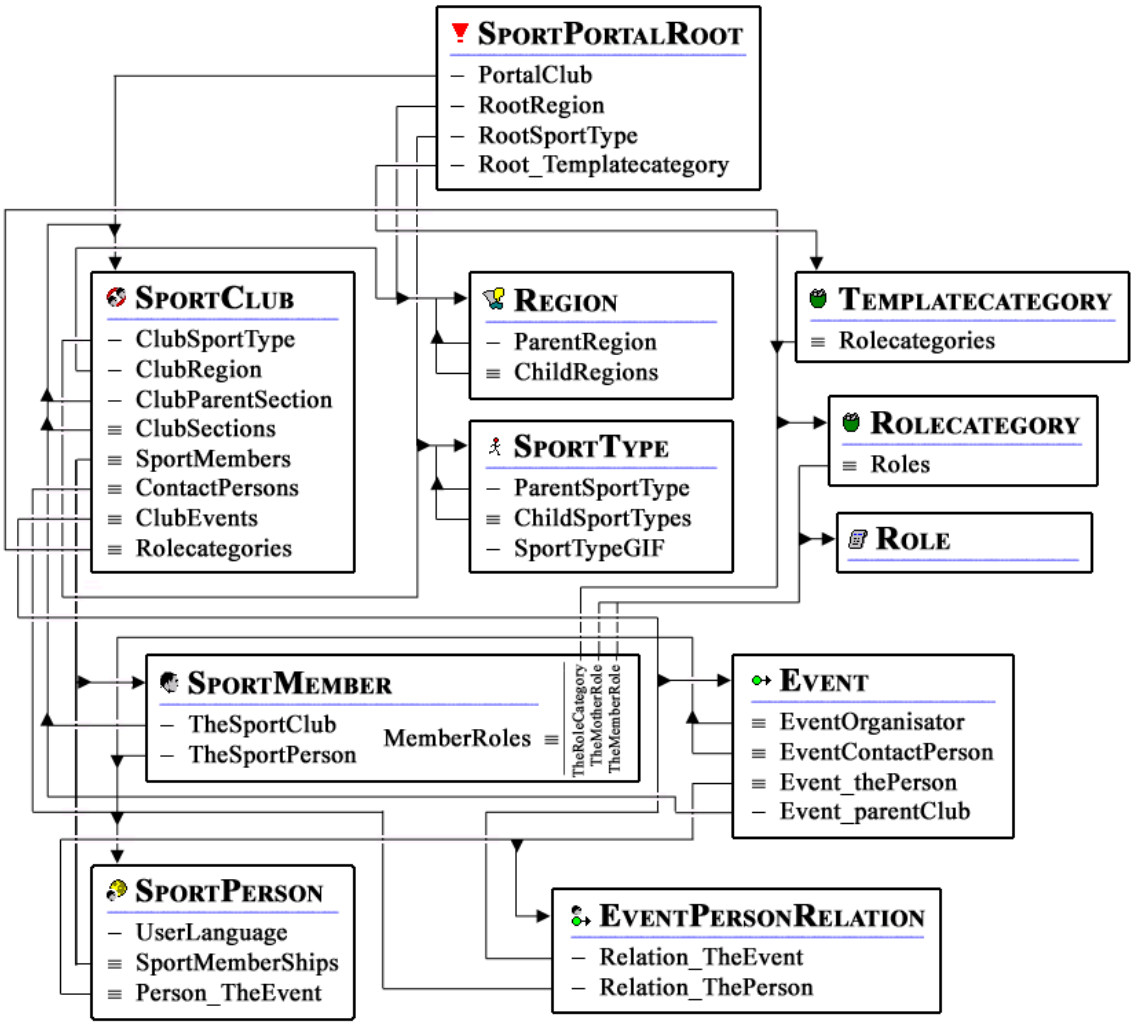



Abbildung 38: SportBase - Objektklassen inkl. Verknüpfungen und Relationen

### 6.2.1 Objektdefinitionen

Nach der **SportBase** - Zeigerübersicht[12] wird in diesem Abschnitt auf die einzelnen Objekte (Übrigens: Jedes besitzt standardmäßig auch die **Eigenschaft *Objektname*** !) eingegangen:

| **SportPortalRoot** | … basiert auf : Konfigurationsobjekt |
|---|---|
| **Eigenschaft** | **Typ** |
| Portalclub | • Objektzeiger: –> *SportClub* |
| RootRegion | • Objektzeiger: –> *Region* |
| RootSportType | • Objektzeiger: –> *SportType* |
| RootTemplatecategory | • Objektzeiger: –> *Templatecategory* |
| PortalURL | • Zeichenkette |
| VAppURL | • Zeichenkette |
| VDavURL | • Zeichenkette |

Dieses Objekt dient zur **Basiskonfiguration** des Sport-Portals. Da das **Portal** intern **ebenfalls** als **Club** angelegt sein muss, wird hier ein Zeiger auf diesen hinterlegt. Die Wurzeln der nachfolgenden Objekt-Bäume (*Root…*) werden ebenfalls hier verlinkt. → Kapitel 6.5

| **SportClub** | … basiert auf : Organisation |
|---|---|
| **Eigenschaft** | **Typ** |
| WebVisible | • Boolescher Wert |
| WebMenu | • WebMenuEntry-Liste: ≡ WebMenuCommand |
| WebAddress | • Zeichenkette |
| ClubSportType | • Objektzeiger: –> *SportType* |
| ClubRegion | • Objektzeiger: –> *Region* |
| ClubBankDate | • BankDataType: – BankName, BankCode, BankAccountNumber, BankAccountOwner |
| ClubNewsBoard | • ClubNewsEntry-Liste: ≡ ClubNewsDate, ClubNewsString |
| ClubParentSection | • Objektzeiger: –> *SportClub* |
| ClubSections | • Objektzeiger-Liste: ≡> *SportClub* |
| SportMembers | • Objektzeiger-Liste: ≡> *SportMember* |
| ContactPersons | • Objektzeiger-Liste: ≡> *SportPerson* |
| ClubEvents | • Objektzeiger-Liste: ≡> *Event* |
| Rolecategories | • Objektzeiger-Liste: ≡> *Rolecategory* |

Der *SportClub* wurde vom Fabasoft-Objekt ***Organisation*** abgeleitet, welches bereits Eigenschaften wie *Name, Kurzname, Branche, ...* weitervererbt.

Mit dem Feld *WebVisible* ist es möglich einzelne Clubs für die Clubsuche des Portals unsichtbar zu machen.

In der Liste *WebMenu* wird die Reihenfolge und die Existenz der vom **Portal** zur Verfügung gestellten **Funktionalitäten** (*newsboard, clubs, events, admin*) angegeben.

Die aktuellen **News-Einträge** werden in der *ClubNewsBoard* - Liste gesammelt.

[12] Die einzelnen Objekte werden entweder über Zeiger ( –> ) oder Zeiger-Listen ( ≡> ) miteinander verknüpft !!!

| **SportType** | … basiert auf : Basisobjekt |
|---|---|
| **Eigenschaft** | **Typ** |
| ParentSportType | Objektzeiger: -> *SportType* |
| ChildSportTypes | Objektzeiger-Liste: => *SportType* |
| SportTypeGIF | Objektzeiger: -> *Objekt mit Inhalt* |

Durch *SportType* wird ein **Objekt-Baum** aufgespannt, der als Blätter einzelne **Sportarten** trägt. Der *Objektname* beinhaltet dabei den Namen des Sports bzw. des Sportüberbegriffs.

| **Region** | … basiert auf : Basisobjekt |
|---|---|
| **Eigenschaft** | **Typ** |
| ParentRegion | Objektzeiger: -> *Region* |
| ChildRegions | Objektzeiger-Liste: => *Region* |

Auch alle diese **Objekte** bilden zusammen einen **Baum**. So kann eine geografische Einteilung der Welt in **Regionen und Unterregionen** erfolgen.

| **Templatecategory** | … basiert auf : Basisobjekt |
|---|---|
| **Eigenschaft** | **Typ** |
| Rolecategories | Objektzeiger-Liste: => *Rolecategory* |

Das Portal stellt durch diese Objekte den Klubs diverse **Vorlagen** von *Rolecategories* zur Verfügung.

| **Rolecategory** | … basiert auf : Basisobjekt |
|---|---|
| **Eigenschaft** | **Typ** |
| Roles | Objektzeiger-Liste: => *Role* |

Jeder **Club** kann hiermit eigene **Informationen** (auch pro Mitglied) in **Kategorien** einteilen.

| **Role** | … basiert auf : Basisobjekt |
|---|---|
| **Eigenschaft** | **Typ** |
| RoleValue | Währung |
| RoleString | Zeichenkette |
| RoleNumber | Gleitkommazahl |
| RoleDate | Datum und Zeit |
| RoleBoolean | Boolescher Wert |
| MemberDefault | Boolescher Wert |
| MemberEditable | Boolescher Wert |

**Club**- und / oder **Mitglieder-spezifische Daten**, wie z.B. Divisionen, Gruppen, diverse Gewandgrößen, usw. können in „**Rollen**“ abgelegt und mit verschiedenen Daten (*Role...*) befüllt werden.

| **SportPerson** | … basiert auf : Person |
|---|---|
| **Eigenschaft** | **Typ** |
| SessionHashCode | Ganze Zahl |
| Nickname | Zeichenkette |
| Nickpassword | Zeichenkette |
| SecretQuestion | Zeichenkette |
| SecretAnswer | Zeichenkette |
| WebActive | Boolescher Wert |
| WebVisible | Boolescher Wert |
| UserLanguage | Objektzeiger: -> *Sprache* |
| Job | Zeichenkette |
| resellAddress | YesNoEnum: *Nein\|Ja* |
| informationByEmail | YesNoEnum: *Nein\|Ja* |
| SportMemberships | Objektzeiger-Liste: => *SportMember* |
| Person_TheEvent | Objektzeiger-Liste: => *EventPersonRelation* |

Die *SportPerson* wurde vom Fabasoft-Objekt ***Person*** abgeleitet, welches bereits Eigenschaften wie *Titel, Anrede, Vorname, Nachname, Geboren am, Geschlecht, ...* weitervererbt.

Mit dem *Nickname* und dem korrekten *Nickpassword* wird ein zufälliger *SessionHashCode* erzeugt und man erhält **Zugang zum Sport-Portal**, außer das Feld *WebActive* ist deaktiviert.

| **SportMember** | … basiert auf : Beziehung |
|---|---|
| **Eigenschaft** | **Typ** |
| TheSportClub | Objektzeiger: -> *SportClub* |
| TheSportPerson | Objektzeiger: -> *SportPerson* |
| AccessLevelType | AccessLevelEnum: *Start\|Benutzer\|Experte\|Administrator* |
| PIN | Zeichenkette |
| JoinDate | Datum und Zeit |
| ExitDate | Datum und Zeit |
| ExitReason | Zeichenkette |
| MailType | MailEnum: *eMail\|Post* |
| Remark | Zeichenkette |
| ClubBankData | BankDataType: - BankName, BankCode, BankAccountNumber, BankAccountOwner |
| Subscriptions | SubscriptionType-Liste: ≡ Date, SubscriptionType: *Offen\|Bezahlt\|Spende*, Amount, Description |
| Earnings | EarningType-Liste: ≡ Date, EarningType: *Verdienst\|Ehrung*, Description |
| MemberRoles | MemberRolesType-Liste: ≡ The*RoleCategory* ->, TheMother*Role* ->, TheMember*Role* -> |

Durch die **Relation**sobjekte *SportMember* werden die einzelnen **Benutzer** des Portals (*SportPerson*) mit den vorhandenen **Club**s (*SportClub*) verknüpft. → Ein Objekt pro **Klubmitgliedschaft**.

Jede Person, welche somit bei einem Club eingetragen ist, besitzt durch das Feld *AccessLevelType* auch eine exakte **Benutzerberechtigung**.

Der *PIN* ist ein Club-Code, welcher zur einmaligen Benutzerwiedererkennung verwendet werden kann.

Auch werden hier Daten wie die **Monatsbeiträge**, die **Verdienste** und die **Rollen** (Kopien der Clubrollen) eines Clubmitgliedes gesammelt.

| **Event** | … basiert auf : Veranstaltung |
|---|---|
| **Eigenschaft** | **Typ** |
| EventType | EventEnum: *jeder\|Mitglieder* |
| EventOrganisator | Objektzeiger-Liste: => *SportPerson* |
| EventContactPerson | Objektzeiger-Liste: => *SportPerson* |
| CardPrice | Währung |
| MaxParticipants | Ganze Zahl |
| EventAddress | EventAddressType: – Straße, Postleitzahl, Ort, Bundesland, Land, Beschreibung |
| Event_thePerson | Objektzeiger-Liste: => *EventPersonRelation* |
| Event_parentClub | Objektzeiger: -> *SportClub* |
| EventRegistration | Boolescher Wert |
| Registrationdate | Datum und Zeit |
| Description | Zeichenkette |

Ein *Event* wurde vom Fabasoft-Objekt ***Veranstaltung*** abgeleitet, welches bereits Eigenschaften wie *Name der Veranstaltung, von, bis, Ort, ...* weitervererbt.

Jede Veranstaltung kann **offen** für jeden **oder nur** für **Mitglieder** sein. Auch ob man sich für einen Event anmelden muss oder ob bloßes Erscheinen reicht, kann festgelegt werden.

| **EventPersonRelation** | … basiert auf : Beziehung |
|---|---|
| **Eigenschaft** | **Typ** |
| Relation_TheEvent | Objektzeiger: -> *Event* |
| Relation_ThePerson | Objektzeiger: -> *SportPerson* |

Durch diese **Relation**sobjekte werden die einzelnen **Benutzer** (*SportPerson*) mit der jeweiligen **Veranstaltung** (*Event*) verknüpft.

## 6.3 Hybridlösung

Die nächste grundlegende Frage bezüglich des Sport-Portals war: „Welche **Layout-Struktur** sollte gewählt werden“. Die Entscheidung fiel zugunsten einer **frame-basiert**en Web-Anwendung aus. Da die Fabasoft-VApps in der uns zur Verfügung gestandenen Version diesbezüglich jedoch noch zu wenig flexibel waren, wurde eine Hybridlösung (siehe 4.5.3) in Angriff genommen.

Das Layout wurde dazu in folgende **2 grobe Bereiche** unterteilt:

- **Rahmen** der Web-Portal-Seite, welcher unter anderem eine **Menü**zeile für das Anwählen der Portal-Hauptfeatures und eine Club-**Navigation**sleiste beinhalten soll.
- **Zentrum** (Hauptfenster) des Web-Services, in welchem dann z.B. eine VApp der selbstentwickelten **SportVApp** - Fabasoft-Komponente ihren Dienst verrichten kann.

Für die nicht VApp-basierten Teile des Sport-Portals wurde im Anschluss dann eine kleine **Sitemap** erstellt, in welcher auch das Startskript der gesamten Anwendung abgelegt wurde:

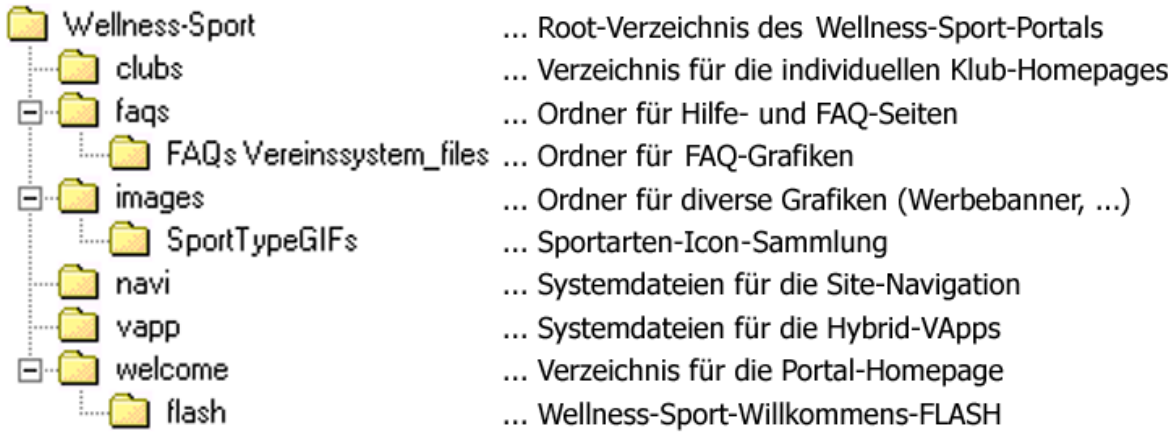



Abbildung 39: Interne Verzeichnisstruktur des Sport-Portals

**Wellness-Sport**

```
default.asp     ... Sport-Portal-Startskript
home.asp        ... Begrüßungsseite (Portal & Clubs)
top.asp         ... Logo, Menu und Werbebanner
navi.asp        ... Club-Navigator & Anmelde-Button
lang.asp        ... Sprachwahl
login.asp       ... Anmelde-Formular
config.asp      ... System-Skript
```

### 6.3.1 Portal-Oberfläche

Dieser Abschnitt beschäftigt sich nun genauer mit der **Implementierung** der ASP-Javascript-Teile des Sport-Portals, welche im Groben für den **Rahmen** der frame-basierten Web-Anwendung, also der **Web-Portal**-Oberfläche, verantwortlich ist. Die dafür benötigten Scripts befinden sich alle in dem Wellness-Sport - Ordner der bereits angeführten Sitemap. Auf der beigelegten CD findet man alle nötigen Sourcen gezippt im „Projekte“-Ordner.

**Wellness-Sport/config.asp**

- Das erste Skript ist **zum Inkludieren in alle weiteren Scripts** gedacht, da es anpassbare Funktionen zur Verfügung stellt, welche von allgemeinem Interesse sind. Auch werden gewisse konfigurierbare Variablen im Zusammenhang mit den SportVApps und dem SportBase definiert, das Web-Seiten-Caching de-aktiviert und die **ActiveX**-Schnittstelle der Fabasoft-Components eingerichtet.

```
<%@ LANGUAGE=JAVASCRIPT%>
<%
 // ================================================================================
 var coo_config = "COO.1.980.1.1270"; // SportBaseConfig
 var r_dx       = "COO.1.980.1.1220"; // VApp-Dispatcher
 var german     = "COO.1.1.1.10";
 var english    = "COO.1.1.1.9";
 var std_lang   =  german;
 // --------------------------------------------------------------------------------
 Response.Expires=-1440;
 var coort=new ActiveXObject ("Coo.Runtime"); coort.Login();
 var cootx=new ActiveXObject ("Coo.Transaction");
 // --------------------------------------------------------------------------------
 var obj_config = coort.GetObject(coo_config);
 var def_portalclub=coort.GetAttributeDefinition("SPORTBASE@1.980:PortalClub");
 var obj_portalclub=obj_config.GetAttributeValue(cootx,def_portalclub,0);
 var def_portalurl=coort.GetAttributeDefinition("SPORTBASE@1.980:PortalURL");
 var val_portalurl=obj_config.GetAttributeValue(cootx,def_portalurl,0);
 var def_vappurl=coort.GetAttributeDefinition("SPORTBASE@1.980:VAppURL");
 var val_vappurl=obj_config.GetAttributeValue(cootx,def_vappurl,0);
 var def_vdavurl=coort.GetAttributeDefinition("SPORTBASE@1.980:VDavURL");
 var val_vdavurl=obj_config.GetAttributeValue(cootx,def_vdavurl,0);

 var homeurl= val_portalurl; var vappurl= val_vappurl; var vdavurl= val_vdavurl;
 var banner = homeurl+"images/banner.gif";
 var rootcoo=obj_portalclub.GetAddress();
 // ================================================================================
```

- Nun folgen die **4 integrierten Hilfsfunktionen**:

  1. Zum Aufruf der später im Hauptfenster des **Portal**s laufenden **Programme** benötigt man eine eindeutige Script- bzw. VApp-ID. Dabei führt die erste Hilfsfunktion die **Auflösung** eines der im Webmenü (des gerade aktiven Sportklubs) eingetragenen **Applikationsnamen** (*home, newsboard, events, clubs, admin, ...*) auf die jeweilig aktive Script- bzw. VApp-ID durch.
  2. Die nächste Hilfsfunktion liefert einen **sprachbezogenen Menü-Text** abhängig vom aktuellen Applikationsnamen zurück.
  3. Stellt sicher, dass eine beliebige Variable wirklich nur einen String enthält.
  4. Codiert die Umlaute innerhalb eines Strings ins HTML-Format um.

```
 function r_ax(appname) { // >> Applications
  var x_ax;
  switch (appname) {
   case "home":          x_ax="home";              break;
   case "login":         x_ax="login";             break;
   case "logout":        x_ax="logout";            break;
   case "newsboard":     x_ax="COO.1.980.1.1249";  break;
   case "events":        x_ax="COO.1.980.1.1356";  break;
   case "clubs":         x_ax="COO.1.980.1.1344";  break;
   case "admin":         x_ax="COO.1.980.1.1538";  break;
   default:              x_ax="home";              break;
  }
  return(x_ax);
 }
 // --------------------------------------------------------------------------------
 function menu(appname,lx) { // >> Menu-Text
  var text;
  switch (lx) {
   case english:
```

```
47      switch (appname) {
48       case "home":         text="";                  break;
49       case "login":        text="Login";             break;
50       case "logout":       text="Logout";            break;
51       case "newsboard":    text="Newsboard";         break;
52       case "events":       text="Events";            break;
53       case "clubs":        text="Clubs";             break;
         case "admin":        text="Administration";    break;
55       default:             text="";                  break;
56      }
57     break;
58     default: // german
59      switch (appname) {
60       case "home":         text="";                  break;
61       case "login":        text="Anmelden";          break;
62       case "logout":       text="Abmelden";          break;
63       case "newsboard":    text="Aktuelles";         break;
64       case "events":       text="Veranstaltungen";   break;
65       case "clubs":        text="Klubs";             break;
66       case "admin":        text="Verwaltung";        break;
67       default:             text="";                  break;
68      }
69     break;
70    }
71    return(text);
72   }
73   // ================================================================================
74   function conv(value) {
75    value=value+""; if (value=="undefined") value="";
76    return(value);
77   }
78   // --------------------------------------------------------------------------------
79   function html(string) {
80    string=string.replace("Ä","&Auml;");
81    string=string.replace("Ö","&Ouml;");
82    string=string.replace("Ü","&Uuml;");
83    string=string.replace("ä","&auml;");
84    string=string.replace("ö","&ouml;");
85    string=string.replace("ü","&uuml;");
86    string=string.replace("ß","&szlig;");
87    return(string);
88   }
89   // ================================================================================
90  %>
```

**Wellness-Sport/default.asp**

```
1   <!--#include file="config.asp"-->
```

- Das **Startskript** des Wellness-Sport-Portals, welches sich primär um die HTML-**Frames-Generierung** der Website kümmert.
- Dabei passt sich der Aufbau durch Übergabe folgender **GET-Parameter** entsprechend an:
    1. vx ... Script- bzw. VApp-ID (**Portal-Programme** im Hauptfenster)
    2. lx .... aktuell selektierte Sprache
    3. ux ... angemeldete *SportPerson* oder „*guest*“
    4. hx ... bei der Anmeldung per Zufall erzeugter *SessionHashCode*

5. mx .. aktuell aktives *SportMember* (-ship)
6. ix ..... interner Index für die Club-Navigation (siehe `navi.asp`)

- Zusätzlich werden diese Parameter auch am Client als **Cookie** abgelegt, um bei einem späteren Aufruf der Website wieder beim letzten Stand fortsetzen zu können.

```
2   <%
3    // ---------------------------------------------------------------------------
4    var vx=conv(Request.QueryString("vx"));
5    var lx=conv(Request.QueryString("lx"));
6    var ux=conv(Request.QueryString("ux"));
7    var hx=conv(Request.QueryString("hx"));
8    var mx=conv(Request.QueryString("mx"));
9    var ix=conv(Request.QueryString("ix"));
10   if (vx=="") vx=conv(Request.Cookies("vx")); if (vx=="") vx="home";
11   if (lx=="") lx=conv(Request.Cookies("lx")); if (lx=="") lx=std_lang;
12   if (ux=="") ux=conv(Request.Cookies("ux")); if (ux=="") ux="guest";
13   if (hx=="") hx=conv(Request.Cookies("hx")); if (hx=="") hx="0";
14   if (mx=="") mx=conv(Request.Cookies("mx")); if (mx=="") mx="root";
15   if (ix=="") ix=conv(Request.Cookies("ix")); if (ix=="") ix="0";
16   // ---------------------------------------------------------------------------
```

- Werden die **POST-Parameter** in den Zeilen 18-20 empfangen, wird durch dieses Skript auch die eigentliche **Benutzeranmeldung** durchgeführt. Dabei wird bei erfolgreichem Login ein zufälliger *SessionHashCode* bestimmt, welcher bei jedem weiteren Zugriff auf das Portal verglichen wird und somit sichergestellt werden kann, dass der jeweilige Benutzer nur auf einem einzigen Web-Client im gesamten Internet auf seinen Account zugreifen kann.
- Auch die **Abmeldung** vom Portal kann durch die dafür vorgesehene Script-ID `"logout"` durchgeführt werden.

```
17   var err_code="";
18   var func=Request.Form("func");    // Function
19   var f0_0=Request.Form("f0_0");    // Nickname
20   var f0_1=Request.Form("f0_1");    // Password
21   if (func=="flogin" && f0_0!="") { // => LOGIN
22
23    var coort=new ActiveXObject ("Coo.Runtime");
24    coort.Login();
25    var cootx=new ActiveXObject ("Coo.Transaction");
26
27    var query="SELECT SPORTBASE@1.980:Nickname, SPORTBASE@1.980:Nickpassword ";
28     query+="FROM SPORTBASE@1.980:SportPerson WHERE ";
29     query+=".SPORTBASE@1.980:Nickname = '"    +f0_0+"' AND ";
30     query+=".SPORTBASE@1.980:Nickpassword = '"+f0_1+"' AND ";
31     query+=".SPORTBASE@1.980:WebActive";
32    try {
33     var person_lst=coort.SearchObjects3(cootx,query);
34     var person_obj=person_lst.getItem(0);
35     var npwd_def=coort.GetAttributeDefinition("SPORTBASE@1.980:Nickpassword");
36     var npwd_val=person_obj.GetAttributeValue(cootx,npwd_def,0);
37     if (npwd_val==f0_1) {
38      var person_coo=person_obj.GetAddress();
39      var hash_def=coort.GetAttributeDefinition("SPORTBASE@1.980:SessionHashCode");
40      var hashcode=(Math.round(Math.random()*32766))+1;
41      person_obj.SetAttributeValue(cootx,hash_def,0,hashcode);
```

```
42        cootx.Commit();
43           vx="home"; ux=person_coo; hx=hashcode;
44         }
45       }
46       catch (e) {vx="login"; ux="guest"; hx="0"; err_code=1}
47       lx  =Request.Form("lx");
48       mx="root"; ix="0";
49      }
50      // ---------------------------------------------------------------------------
51      if (vx=="logout") {
52       var hash_def=coort.GetAttributeDefinition("SPORTBASE@1.980:SessionHashCode");
53       var hashcode=(Math.round(Math.random()*32766))+1;
54       coort.GetObject(ux).SetAttributeValue(cootx,hash_def,0,hashcode);
55       cootx.Commit();
56       vx="home"; lx=lx; ux="guest"; hx="0"; mx="root"; ix="0";
57      }
58      // ---------------------------------------------------------------------------
59      var exp=new Date(); exp.setTime(exp.getTime()+(60000*60*24*7));
60      Response.Cookies("vx")=vx; Response.Cookies("vx").expires=exp.getVarDate();
61      Response.Cookies("lx")=lx; Response.Cookies("lx").expires=exp.getVarDate();
62      Response.Cookies("ux")=ux; Response.Cookies("ux").expires=exp.getVarDate();
63      Response.Cookies("hx")=hx; Response.Cookies("hx").expires=exp.getVarDate();
64      Response.Cookies("mx")=mx; Response.Cookies("mx").expires=exp.getVarDate();
65      Response.Cookies("ix")=ix; Response.Cookies("ix").expires=exp.getVarDate();
66      // ---------------------------------------------------------------------------
```

- Nun wird die **Frame-Definition** an den Web-Client ausgegeben. Daraufhin ist es dem Browser möglich, die Frame-Inhalte nach folgendem Schema zu laden:


Abbildung 40: Sport-Portal - Main-Frames

( 1 ) ..... `top.asp`
( 2 ) ..... `navi.asp`
( 3 ) ..... `lang.asp`
( 4 ) ..... Portal-Programme (`default: home.asp`)

( Der schmale grafische Rand um Frame ( 4 ) wird durch simple statische Frames erzeugt. )

```
67    nav="vx="+vx+"&lx="+lx+"&ux="+ux+"&hx="+hx+"&mx="+mx+"&ix="+ix;
68    if (vx.substr(0,3)=="COO")
69     url=vappurl+"?dx="+r_dx+"&ax="+vx+"&"+nav;
70    else {
71     if (vx=="login") url="login.asp?lx="+lx+"&err="+err_code;
72     else             url="home.asp?"+nav;
73    }
74    // ---------------------------------------------------------------------------
75   %>
76   <HTML>
77   <HEAD>
78    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
79    <meta http-equiv="Content-Script-Type" content="text/javascript">
80    <title>Wellness-Sport</title>
81   </HEAD>
82
83   <FRAMESET ROWS="108,*" BORDER=0>
84    <FRAME NAME="head" SRC="top.asp?<%=nav%>"
85     SCROLLING=no MARGINHEIGHT=0 MARGINWIDTH=0 FRAMEBORDER=0>
86    <FRAMESET COLS="148,24,*,32" BORDER=0>
87     <FRAMESET ROWS="*,42" BORDER=0>
88      <FRAME NAME="navi" SRC="navi.asp?<%=nav%>"
89       SCROLLING=no MARGINHEIGHT=0 MARGINWIDTH=0 FRAMEBORDER=0>
90      <FRAME NAME="lang" SRC="lang.asp?<%=nav%>"
91       SCROLLING=no MARGINHEIGHT=0 MARGINWIDTH=0 FRAMEBORDER=0>
92     </FRAMESET>
93     <FRAME SRC="navi/border_l.html"
94      SCROLLING=no MARGINHEIGHT=0 MARGINWIDTH=0 FRAMEBORDER=0>
95     <FRAMESET ROWS="16,*" BORDER=0>
96      <FRAME SRC="navi/border_t.html"
97       SCROLLING=no MARGINHEIGHT=0 MARGINWIDTH=0 FRAMEBORDER=0>
98      <FRAME NAME="main" SRC="<%=url%>"
99       MARGINHEIGHT=0 MARGINWIDTH=0 FRAMEBORDER=0>
100    </FRAMESET>
101    <FRAME SRC="navi/border_r.html"
102     SCROLLING=no MARGINHEIGHT=0 MARGINWIDTH=0 FRAMEBORDER=0>
103   </FRAMESET>
104  </FRAMESET>
105
106  <NOFRAMES>
107   <CENTER>Sorry, aber diese Seite ben&ouml;tigt Frames!</CENTER>
108  </NOFRAMES>
109  </HTML>
```

### Wellness-Sport/top.asp

```
1    <!--#include file="config.asp"-->
```

- Nun wird mittels dieses Scripts die **Menüzeile** zur Auswahl der **Portal-Programme** des Hauptfensters erzeugt. Dabei passt sie sich automatisch der aktuell eingestellten Sprache an.
- Am Beginn wird dabei anhand der empfangenen GET-Parameter festgestellt, ob bzw. welcher **Portal-Benutzer** an diesem Web-Client zur Zeit gerade **angemeldet** und welcher **Sportklub** aktuell **selektiert** ist.

```
     <%
3     // ---------------------------------------------------------------------------
4     var vx=conv(Request.QueryString("vx"));
5     var lx=conv(Request.QueryString("lx"));
6     var ux=conv(Request.QueryString("ux"));
7     var hx=conv(Request.QueryString("hx"));
```

```
8   var mx=conv(Request.QueryString("mx"));
9   var ix=conv(Request.QueryString("ix"));
10  // ------------------------------------------------------------------------
11  var obj_sportclub=coort.GetObject(rootcoo);
12  try {
13   var obj_person   =coort.GetObject(ux);
14
15   var def_webactive=coort.GetAttributeDefinition("SPORTBASE@1.980:WebActive");
16   var val_webactive=obj_person.GetAttributeValue(cootx,def_webactive,0);
17   if (val_webactive==false) throw("Not WebActive");
18
19   var cmp_hashcode =hx;
20   var def_hashcode =
       coort.GetAttributeDefinition("SPORTBASE@1.980:SessionHashCode");
21   var val_hashcode =obj_person.GetAttributeValue(cootx,def_hashcode,0);
22   if (val_hashcode!=cmp_hashcode) throw("Illegal hashcode");
23
24   var obj_membership=coort.GetObject(mx);
25   var def_membership=
       coort.GetAttributeDefinition("SPORTBASE@1.980:TheSportPerson");
26   var cmp_person    =obj_membership.GetAttributeValue(cootx,def_membership,0);
27   if (obj_person.GetAddress()!=cmp_person.GetAddress())
       throw("Illegal membership");
28
29   var def_sportclub=
       coort.GetAttributeDefinition("SPORTBASE@1.980:TheSportClub");
30   obj_sportclub=obj_membership.GetAttributeValue2(cootx,def_sportclub,0);
31  }
32  catch (e) {}
33  // ------------------------------------------------------------------------
```

- Der folgende Code-Teil ermittelt die im aktuellen ***SportClub*** - Objekt hinterlegte ***WebMenuEntry*** - **Liste** und generiert aus den darin abgelegten internen Applikationsnamen z.B. folgende HTML-Ausgabe:


Abbildung 41: Sport-Portal - Menüzeile

```
34  var def_menu=coort.GetAttributeDefinition("SPORTBASE@1.980:WebMenu");
35  var cnt_menu=obj_sportclub.GetAttributeValueCount(cootx,def_menu);
36  var lst_menu=obj_sportclub.GetAttribute3(cootx,def_menu);
37  var def_mcmd=coort.GetAttributeDefinition("SPORTBASE@1.980:WebMenuCommand");
38  // ------------------------------------------------------------------------
39 %>
40 <html>
41 <head>
42  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
43  <meta http-equiv="Content-Script-Type" content="text/javascript">
44  <title>Wellness-Sport</title>
45
46  <style type="text/css">
47   div.usel {font-family: Verdana, Helvetica, Arial, sans-serif;
48             font-size: 16px; text-decoration:none; font-weight:bold;}
49   div.sel  {font-family: Verdana, Helvetica, Arial, sans-serif; color:#2e277a;
50             font-size: 16px; text-decoration:none; font-weight:bold;}
51   a:hover  {color:#ffffff}
52  </style>
53 </head>
54
55 <body background="navi/top_0.jpg"
56  text="#f1e1c3" link="#f1e1c3" vlink="#f1e1c3" alink="#f1e1c3"
```

```
 leftmargin=0 topmargin=0 marginwidth=0 marginheight=0>
<div id="fill" style="position:absolute; left:248; top:0;">
 <img src="navi/top_2.jpg" border=0 width="334" height="106">
</div>
<div id="banner" style="position:absolute; left:0; top:0;">
 <table cellspacing=0 cellpadding=0 border=0 width=100%><tr>
  <td align=right><img src="<%=banner%>" border=0 height="60"></td>
 </tr></table>
</div>
<div id="home" style="position:absolute; left:0; top:0;">
 <a href="default.asp?vx=home&lx=<%=lx
  %>&ux=<%=ux%>&hx=<%=hx%>&mx=root&ix=<%=ix%>" target="_parent"
  ><img src="navi/top_1.jpg" border=0 width="248" height="106"></a>
</div>
<div id="menu" style="position:absolute; left:248; top:68;">
 <table cellspacing=0 cellpadding=0 border=0><tr>
<%
 // ------------------------------------------------------------------------
 for (idx=0; idx<cnt_menu; idx++) {
  obj_menu=lst_menu.getItem(idx);
  val_mcmd=obj_menu.GetAttributeValue(def_mcmd,0);
  n_vx=r_ax(val_mcmd); if (vx!=n_vx) sel="u"; else sel="";
  %>
 <td><img src="navi/top_br.jpg" border=0 width="16" height="32"></td>
 <td><a href="default.asp?vx=<%=n_vx%>&lx=<%=lx
  %>&ux=<%=ux%>&hx=<%=hx%>&mx=<%=mx%>&ix=<%=ix%>" target="_parent"
  ><div class="<%=sel%>sel"><%=menu(val_mcmd,lx)%></div></a></td>
  <%
 }
 // ------------------------------------------------------------------------
%>
 <td><img src="navi/top_br.jpg" border=0 width="16" height="32"></td>
 </tr></table>
</div>
</body>
</html>
```

### Wellness-Sport/navi.asp

```
<!--#include file="config.asp"-->
```

- Dieses Skript ist für die **Navigationsleiste** am linken Portalrand zuständig. Mit ihr kann sich ein Benutzer an- und abmelden, zwischen dem **Portal** und seinen Clubs (**Klubmitgliedschaften**) **wählen** und diverse Hilfen abrufen.
- Am Beginn wird dabei wieder festgestellt, ob bzw. welcher **Portal-Benutzer** gerade am Web-Client **angemeldet** und welcher **Sportklub selektiert** ist.

```
<%
 // ------------------------------------------------------------------------
 var vx=conv(Request.QueryString("vx"));
 var lx=conv(Request.QueryString("lx"));
 var ux=conv(Request.QueryString("ux"));
 var hx=conv(Request.QueryString("hx"));
 var mx=conv(Request.QueryString("mx"));
 var ix=conv(Request.QueryString("ix"));
 // ------------------------------------------------------------------------
 var obj_sportclub=coort.GetObject(rootcoo);
 var str_status="login"; var bln_userknown=false;
 try {
  var obj_person   =coort.GetObject(ux);

```

```
16    var def_webactive=coort.GetAttributeDefinition("SPORTBASE@1.980:WebActive");
17    var val_webactive=obj_person.GetAttributeValue(cootx,def_webactive,0);
18    if (val_webactive==false) throw("Not WebActive");
19
20    var cmp_hashcode =hx;
21    var def_hashcode =
        coort.GetAttributeDefinition("SPORTBASE@1.980:SessionHashCode");
22    var val_hashcode =obj_person.GetAttributeValue(cootx,def_hashcode,0);
23    if (val_hashcode!=cmp_hashcode) throw("Illegal hashcode");
24
25    str_status="logout"; bln_userknown=true;
26    var obj_membership=coort.GetObject(mx);
27    var def_membership=
        coort.GetAttributeDefinition("SPORTBASE@1.980:TheSportPerson");
28    var cmp_person    =obj_membership.GetAttributeValue(cootx,def_membership,0);
29    if (obj_person.GetAddress()!=cmp_person.GetAddress())
        throw("Illegal membership");
30
31    var def_sportclub=
        coort.GetAttributeDefinition("SPORTBASE@1.980:TheSportClub");
32    obj_sportclub=obj_membership.GetAttributeValue2(cootx,def_sportclub,0);
33   }
34   catch (e) {}
35   // ------------------------------------------------------------------------
```

- Nun folgt der Code-Abschnitt, welcher die im aktuellen ***SportPerson*** - Objekt hinterlegte ***SportMember*** **- Liste** durchläuft und dabei alle verketteten ***SportClub*** - Objektnamen samt der zugehörigen ***SportTypeGIF*** - URLs und COO-Adressen (siehe 4.5.2) in einen clientseitigen Javascript-Code integriert.


Abbildung 42: Sport-Portal - Navigationsleiste

```
36    if (bln_userknown) {
37     var def_memberships=
         coort.GetAttributeDefinition("SPORTBASE@1.980:SportMemberships");
38     var cnt_memberships=obj_person.GetAttributeValueCount(cootx,def_memberships);
39     var lst_memberships=obj_person.GetAttribute3(cootx,def_memberships);
40     var def_sportclub=coort.GetAttributeDefinition("SPORTBASE@1.980:TheSportClub");
41     var def_clubname=coort.GetAttributeDefinition("COOSYSTEM@1.1:objname");
42     var def_clubtype=coort.GetAttributeDefinition("SPORTBASE@1.980:ClubSportType");
43     var def_clublogo=coort.GetAttributeDefinition("SPORTBASE@1.980:SportTypeGIF");
44    }
45    // ------------------------------------------------------------------------
46   %>
47   <html>
```

```
48  <head>
49   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
50   <meta http-equiv="Content-Script-Type" content="text/javascript">
51   <title>Wellness-Sport</title>
52
53   <style type="text/css">
54    div.usel {font-family: Verdana, Helvetica, Arial, sans-serif;
55              font-size: 14px; text-decoration:none; font-weight:bold;}
56    div.sel  {font-family: Verdana, Helvetica, Arial, sans-serif; color:white;
57              font-size: 14px; text-decoration:none; font-weight:bold;}
58    .info    {font-family: Verdana, Helvetica, Arial, sans-serif;
59              font-size: 10px; text-decoration:none;}
60    a:hover  {color:#f1e1c3}
61   </style>
62
63   <script>
64    var iportal = new Array();
65    iportal[0]=new Image(); iportal[0].src="navi/navi_gu.jpg";
66    iportal[1]=new Image(); iportal[1].src="navi/navi_gs.jpg";
67
68    var isel = new Array();
69    isel[0]=new Image(); isel[0].src="navi/sel_l.gif";
70    isel[1]=new Image(); isel[1].src="navi/sel_r.gif";
71    var empty=new Image(); empty.src="navi/empty.gif";
72
73    var club = new Array(); var name = new Array(); var member = new Array();
74  <%
75   // ------------------------------------------------------------------------
76   var cnt_clubs=0; var sel_club=-1;
77   if (bln_userknown) {
78    for (idx=0; idx<cnt_memberships; idx++) {
79     var obj_member=lst_memberships.getItem(idx);
80     var coo_member=obj_member.GetAddress();
81     if (coo_member==mx) sel_club=idx;
82     var obj_club=obj_member.GetAttributeValue(cootx,def_sportclub,0);
83     var obj_type=obj_club.GetAttributeValue2(cootx,def_clubtype,0);
84     var val_name=obj_club.GetAttributeValue2(cootx,def_clubname,0);
85     var obj_logo=obj_type.GetAttributeValue2(cootx,def_clublogo,0);
86     var coo_logo=obj_logo.GetAddress();
87     Response.Write('  club['+idx+']=new Image(); club['+idx+'].src="');
88     Response.Write(vdavurl+coo_logo+"/"+coo_logo+'.gif";');
89     Response.Write(' name['+idx+']="'+val_name+'";');
90     Response.Write(' member['+idx+']="'+coo_member+'";');
91     cnt_clubs=cnt_memberships;
92    }
93   }
94   Response.Write('  var clubcnt = '+cnt_clubs+';');
95   Response.Write('  var clubsel = '+sel_club +';');
96   Response.Write('  var clublst = '+ix*1     +';');
97   // ------------------------------------------------------------------------
98  %>
99    function ViewClubs() {
100    document.club0.src=empty.src;
       document.sela0.src=empty.src; document.selb0.src=empty.src;
101    document.club1.src=empty.src;
       document.sela1.src=empty.src; document.selb1.src=empty.src;
102    document.club2.src=empty.src;
       document.sela2.src=empty.src; document.selb2.src=empty.src;
103    document.club3.src=empty.src;
       document.sela3.src=empty.src; document.selb3.src=empty.src;
104    if ((clublst+0)<clubcnt) document.club0.src=club[clublst+0].src;
105    if ((clublst+1)<clubcnt) document.club1.src=club[clublst+1].src;
106    if ((clublst+2)<clubcnt) document.club2.src=club[clublst+2].src;
107    if ((clublst+3)<clubcnt) document.club3.src=club[clublst+3].src;
108    if ((clublst+0)==clubsel)
       {document.sela0.src=isel[0].src; document.selb0.src=isel[1].src}
109    if ((clublst+1)==clubsel)
       {document.sela1.src=isel[0].src; document.selb1.src=isel[1].src}
```

```
110     if ((clublst+2)==clubsel)
        {document.sela2.src=isel[0].src; document.selb2.src=isel[1].src}
111     if ((clublst+3)==clubsel)
112     {document.sela3.src=isel[0].src; document.selb3.src=isel[1].src}
113    }
114    function ViewClub(idx) {
115     clubinfo.clubname.value=<%if (sel_club==-1)
        {%>"Wellness-Sport"<%} else {%>name[<%=sel_club%>]<%}%>;
116     if ((idx>=0) && ((clublst+idx)<clubcnt))
          clubinfo.clubname.value=name[clublst+idx];
117     if (idx==-1) clubinfo.clubname.value="Wellness-Sport";
118    }
119    function StartClub(idx) {
120     if ((clublst+idx)<clubcnt)
121      parent.location.href=
           "default.asp?vx=home&lx=<%=lx%>&ux=<%=ux%>&hx=<%=hx%>&mx="+
           member[clublst+idx]+"&ix="+clublst;
122    }
123    function ScrollUp() {
124     if (clublst+1<clubcnt) {clublst++; ViewClubs();}
125    }
126    function ScrollDown() {
127     if (clublst>0) {clublst--; ViewClubs();}
128    }
129   </script>
130  </head>
131
132  <body bgcolor="#cc8f22"
133   text="#cc8f22" link="#cc8f22" vlink="#cc8f22" alink="#cc8f22"
134   leftmargin=0 topmargin=0 marginwidth=0 marginheight=0
135   onload="ViewClubs(); ViewClub();">
136  <form name="clubinfo">
137  <table cellspacing=0 cellpadding=0 border=0 width=100%>
138   <tr><td height=380></td></tr>
139   <tr><td><center><input
       class="info" type="text" name="clubname" value="" size=16 readonly
      ></center></td></tr>
140  </table>
141  </form>
142  <div id="Button" style="position:absolute; left:0; top:10;">
143   <img src="navi/navi_bu.jpg" border=0 width="138" height="38">
144  </div>
145  <div id="Login" style="position:absolute; left:0; top:20;">
146   <table cellspacing=0 cellpadding=0 border=0>
147    <tr><td width=138 align=center>
148     <a href="default.asp?vx=<%=str_status%>&lx=<%=lx
       %>&ux=<%=ux%>&hx=<%=hx%>&mx=<%=mx%>&ix=<%=ix%>" target="_parent"
149      ><div class="usel"><%=menu(str_status,lx)%></div></a>
150    </td></tr>
151   </table>
152  </div>
153  <div id="Portal" style="position:absolute; left:20; top:58;">
154   <a href="default.asp?vx=home&lx=<%=lx
      %>&ux=<%=ux%>&hx=<%=hx%>&mx=root&ix=<%=ix%>" target="_parent"
155  <% var sel_portal="s"; if (sel_club!=-1) { sel_portal="u";%>
156    onmouseover="document.portal.src=iportal[1].src; ViewClub(-1);"
       onmouseout="document.portal.src=iportal[0].src; ViewClub(-2);"
157  <% } %>
158    ><img name="portal" src="navi/navi_g<%=sel_portal%>.jpg" border=0
        width="106" height="64"></a>
159  </div>
160  <div id="Window" style="position:absolute; left:15; top:122;">
161   <table cellspacing=0 cellpadding=0 border=0 background="navi/navi_ls.jpg"
162    height=256 width=118>
163    <tr><td></td></tr>
164   </table>
165  </div>
166  <div id="List" style="position:absolute; left:15; top:134;">
```

```
167   <table cellspacing=0 cellpadding=0 border=0>
168    <tr>
169     <td><img src="navi/empty.gif" width=26 height=12></td>
170     <td align="center"><%if (cnt_clubs>1) { %><a href="#"
         onclick="ScrollDown();"><img name="up" src="navi/sel_u.gif"
         width=17 height=12 border=0></a><a href="#"
         onclick="ScrollUp();"><img name="up" src="navi/sel_d.gif"
         width=17 height=13 border=0></a><% } %></td>
171     <td><img src="navi/empty.gif" width=26 height=12></td>
172    </tr>
173    <tr>
174     <td><img name="sela0" src="navi/empty.gif" width=26 height=48></td>
175     <td><a href="#" onclick="StartClub(0);" onmouseover="ViewClub(0);"
         onmouseout="ViewClub(-2);"><img name="club0" src="navi/empty.gif"
         width=64 height=48 border=0></a></td>
176     <td><img name="selb0" src="navi/empty.gif" width=26 height=48></td>
177    </tr>
178    <tr height=6><td></td><td></td></tr>
179    <tr>
180     <td><img name="sela1" src="navi/empty.gif" width=26 height=48></td>
181     <td><a href="#" onclick="StartClub(1);" onmouseover="ViewClub(1);"
         onmouseout="ViewClub(-2);"><img name="club1" src="navi/empty.gif"
         width=64 height=48 border=0></a></td>
182     <td><img name="selb1" src="navi/empty.gif" width=26 height=48></td>
183    </tr>
184    <tr height=6><td></td><td></td></tr>
185    <tr>
186     <td><img name="sela2" src="navi/empty.gif" width=26 height=48></td>
187     <td><a href="#" onclick="StartClub(2);" onmouseover="ViewClub(2);"
         onmouseout="ViewClub(-2);"><img name="club2" src="navi/empty.gif"
         width=64 height=48 border=0></a></td>
188     <td><img name="selb2" src="navi/empty.gif" width=26 height=48></td>
190    </tr>
191    <tr height=6><td></td><td></td></tr>
192    <tr>
193     <td><img name="sela3" src="navi/empty.gif" width=26 height=48></td>
194     <td><a href="#" onclick="StartClub(3);" onmouseover="ViewClub(3);"
         onmouseout="ViewClub(-2);"><img name="club3" src="navi/empty.gif"
         width=64 height=48 border=0></a></td>
195     <td><img name="selb3" src="navi/empty.gif" width=26 height=48></td>
196    </tr>
197   </table>
198  </div>
199  <div id="FAQs" style="position:absolute; left:0; top:413;">
200   <map name="FAQsMap">
      <area shape=rect coords="31,12,43 ,27"
       href="faqs/help_<%=  lx%>.html" target="main">
201   <area shape=rect coords="50,12,90 ,27"
       href="faqs/faqs_<%=  lx%>.html" target="main">
202   <area shape=rect coords="99,12,117,27"
       href="faqs/mailto_<%=lx%>.html" target="main">
203  </map>
204  <img src="navi/navi_faqs.gif" usemap="#FAQsMap" border=0
      width="138" height="38">
205  </div>
206  </body>
207  </html>
```

### Wellness-Sport/lang.asp

```
1   <!--#include file="config.asp"-->
```

- Das letzte zum Rahmen des Portals gehörige Skript gibt grafische Buttons zur **Sprachselektion** aus:

Abbildung 43: Sport-Portal - Sprachwahl

```
<%
 // ---------------------------------------------------------------------------
 var vx=conv(Request.QueryString("vx"));
 var lx=conv(Request.QueryString("lx"));
 var ux=conv(Request.QueryString("ux"));
 var hx=conv(Request.QueryString("hx"));
 var mx=conv(Request.QueryString("mx"));
 var ix=conv(Request.QueryString("ix"));
 // ---------------------------------------------------------------------------
%>
<html>
<head>
 <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
 <title>Wellness-Sport</title>
</head>

<body bgcolor="#cc8f22"
 leftmargin=0 topmargin=0 marginwidth=0 marginheight=0>
 <table cellspacing=0 cellpadding=0 border=0 width=100%>
  <tr><td align=center>
   <a href="default.asp?vx=<%=vx%>&lx=<%=german
   %>&ux=<%=ux%>&hx=<%=hx%>&mx=<%=mx%>&ix=<%=ix%>" target="_parent">
    <img src="navi/lang_ge.gif" border=0 width="32" height="36"></a>
  </td><td align=center>
   <a href="default.asp?vx=<%=vx%>&lx=<%=english
   %>&ux=<%=ux%>&hx=<%=hx%>&mx=<%=mx%>&ix=<%=ix%>" target="_parent">
    <img src="navi/lang_en.gif" border=0 width="32" height="36"></a>
  </td></tr>
 </table>
</body>
</html>
```

### Wellness-Sport/home.asp

```
<!--#include file="config.asp"-->
```

- Das erste Skript, welches seinen Dienst im Hauptfenster des Portals verrichtet, begrüßt den Benutzer (sprachabhängig) am **Portal oder** im ausgewählten **Club**.
- Am Beginn wird wieder festgestellt, ob bzw. welcher **Portal-Benutzer** gerade am Web-Client **angemeldet** und welcher **Sportklub** aktiv **selektiert** ist.

```
<%
 // ---------------------------------------------------------------------------
 var vx=conv(Request.QueryString("vx"));
 var lx=conv(Request.QueryString("lx"));
 var ux=conv(Request.QueryString("ux"));
 var hx=conv(Request.QueryString("hx"));
 var mx=conv(Request.QueryString("mx"));
 var ix=conv(Request.QueryString("ix"));
 // ---------------------------------------------------------------------------
 text001="Willkommen";
 text002="zu ihrer pers&ouml;nlichen Wellness-Sport-Seite!"
 text003="im Klub ..."
  if (lx==english) {
  text001="Welcome";
```

```
16      text002="to your personal Wellness-Sport-Page!";
17      text003="to the club ...";
18     }
19     // ------------------------------------------------------------------------
20     var obj_sportclub=coort.GetObject(rootcoo);
21     var bln_userknown=false;
22     try {
23      var obj_person   =coort.GetObject(ux);
24
25      var def_webactive=coort.GetAttributeDefinition("SPORTBASE@1.980:WebActive");
26      var val_webactive=obj_person.GetAttributeValue(cootx,def_webactive,0);
27      if (val_webactive==false) throw("Not WebActive");
28
29      var cmp_hashcode =hx;
30      var def_hashcode =
          coort.GetAttributeDefinition("SPORTBASE@1.980:SessionHashCode");
31      var val_hashcode =obj_person.GetAttributeValue(cootx,def_hashcode,0);
32      if (val_hashcode!=cmp_hashcode) throw("Illegal hashcode");
33
34      bln_userknown=true;
35      var obj_membership=coort.GetObject(mx);
36      var def_membership=
          coort.GetAttributeDefinition("SPORTBASE@1.980:TheSportPerson");
37      var cmp_person    =obj_membership.GetAttributeValue(cootx,def_membership,0);
38      if (obj_person.GetAddress()!=cmp_person.GetAddress())
          throw("Illegal membership");
39
40      var def_sportclub=
          coort.GetAttributeDefinition("SPORTBASE@1.980:TheSportClub");
41      obj_sportclub=obj_membership.GetAttributeValue2(cootx,def_sportclub,0);
42     }
43     catch (e) {}
44     // ------------------------------------------------------------------------
45     var def_clubname =coort.GetAttributeDefinition("COOSYSTEM@1.1:objname");
46     var val_clubname =obj_sportclub.GetAttributeValue2(cootx,def_clubname,0);
47     var coo_sportclub=obj_sportclub.GetAddress();
48     // ------------------------------------------------------------------------
```

- Ist kein Club aktiv, sprich das **Portal** selbst ist **selektiert**, wird je nach Anmeldestatus eine Begrüßung mit dem Benutzernamen (Abbildung 40) ausgegeben oder auf eine **Portal-Startseite** mit einer eingebundenen Flash-Animation verwiesen.

```
49     if (coo_sportclub==rootcoo) {
50      if (bln_userknown) {
51       var def_nickname=coort.GetAttributeDefinition("SPORTBASE@1.980:Nickname");
52       var val_nickname=obj_person.GetAttributeValue(cootx,def_nickname,0);
53    %>
54    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
       "DTD/xhtml1-transitional.dtd">
55    <head>
56    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
57    <title>Login</title><link rel="stylesheet" type="text/css"
       href="<%=homeurl%>vapp/vapp.css" />
58    </head>
59    <body marginwidth="0" marginheight="0" classname="FscBody">
60    <div class="FscCaption"><%=text001%></div>
61    <table class="FscDescription"><tr><td><%=text002%></td></tr></table>
62    <table class="FscData0"><tbody>
63      <tr><td align=center width="100%" class="FscScalarDataCaption"
        ><b><%=text001+" <big>"+html(val_nickname)%></big> !</b></td></tr>
64      <tr><td align=center><br><br><small>Powered by</small><br><img
        src="navi/home.gif"></td></tr>
65    </tbody></table>
66    </body>
```

```
</html>
<%
  } else Response.Redirect(homeurl+"welcome/home.html");
```

- Hat aber der Benutzer einen seiner **Club**s **selektiert**, wird, falls dieser eine eigene Club-Homepage besitzt (***WebAddress***), im Hauptfenster des Portals auf diesen URL weitergeleitet. Im anderen Fall wird eine generische **Club-Begrüßung** ausgegeben.

```
 } else {
  var bln_generic=true;
  try {
   var def_webaddr=coort.GetAttributeDefinition("SPORTBASE@1.980:WebAddress");
   var val_webaddr=obj_sportclub.GetAttributeValue(cootx,def_webaddr,0);
   Response.Redirect(homeurl+"clubs/"+val_webaddr);
   var bln_generic=false;
  } catch (e) {}
 // ------------------------------------------------------------------------
  if (bln_generic) {
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "DTD/xhtml1-transitional.dtd">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Login</title><link rel="stylesheet" type="text/css"
 href="<%=homeurl%>vapp/vapp.css" />
</head>
<body marginwidth="0" marginheight="0" classname="FscBody">
<div class="FscCaption"><%=text001%></div>
<table class="FscDescription"><tr><td><%=text003%></td></tr></table>
<table class="FscData0"><tbody>
  <tr><td align=center width="100%" class="FscScalarDataCaption
  "><b><big><%=html(val_clubname)%></big> !</b></td></tr>
</tbody></table>
</body>
</html>
<%
  }
 }
 // ------------------------------------------------------------------------
%>
```

### 6.3.2 Anmeldung

**Wellness-Sport/login.asp**

```
<!--#include file="config.asp"-->
```

- Nun folgt der letzte ASP-Javascript-Teil des Sport-Portals. Dieses Skript stellt das **Anmeldeformular** (**Login**) für die Portal-Benutzer im Hauptfenster bereit. Die eigentliche Prüfung der eingegebenen Daten wird danach durch das `"default.asp"` - Skript durchgeführt (siehe Kapitel 6.3.1).


Abbildung 44: Sport-Portal - Anmeldung

```
2   <%
3    // ------------------------------------------------------------------------
4    var lx=conv(Request.QueryString("lx"))
5    var err_code=conv(Request.QueryString("err"));
6    // ------------------------------------------------------------------------
7    text001="Anmeldung";
8    text002="Bitte melden Sie sich am Portal an!"
9    text003="Anmelden";
10   text004="Neuen Benutzer anlegen";
11   text010="Benutzername";
12   text011="Kennwort";
13   text020="Benutzer nicht bekannt!";
14   if (lx==english) {
15
16    text002="Please login to the portal!";
17    text003="Login";
18    text004="Register new user";
19    text010="Nickname";
20    text011="Password";
21    text020="User not known!";
22   }
23   // ------------------------------------------------------------------------
24  %>
25  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "DTD/xhtml1-transitional.dtd">
26  <head>
27  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
28  <title>Login</title><link rel="stylesheet" type="text/css"
     href="<%=homeurl%>vapp/vapp.css" />
29  </head>
30  <body marginwidth="0" marginheight="0" classname="FscBody">
31  <div class="FscCaption"><%=text001%></div>
32  <table class="FscDescription"><tr><td><%=text002%></td></tr></table>
33  <form method="post" action="default.asp" name="f" target="_parent">
34  <input type="hidden" name="func" value="flogin">
35  <input type="hidden" name="lx"  value="<%=lx%>">
36  <table class="FscBranches"><tr>
37   <td align="left"><span class="FscBranch"><a href="<%=vappurl
      %>?dx=<%=r_dx%>&ax=COO.1.980.1.1575&vx=login&lx=<%=lx
```

```
      %>&ux=guest&hx=0&mx=root&ix=0" title="<%=text004%> Alt+1"
      accesskey="0"> <%=text004%> </a></span></td>
38    <td align="right"><span class="FscBranch"><a href="#"
      onclick="document.f.submit();" title="<%=text003%> Alt+0"
      accesskey="0"> <%=text003%> <img
      src="<%=homeurl%>vapp/next.gif" border="0" /></a></span></td>
39    </tr></table>
40    <table class="FscData0"><tbody>
41      <tr><td width="22%" class="FscScalarDataCaption"><%=text010%></td><td
        width="78%" class="FscDataValues"><input type="text" name="f0_0"
        maxlength="64" size="64" style="width:250" value="" /></td></tr>
42      <tr><td width="22%" class="FscScalarDataCaption"><%=text011%></td><td
        width="78%" class="FscDataValues"><input type="password" name="f0_1"
        maxlength="64" size="64" style="width:200" value="" /></td></tr>
43    <% if (err_code==1) { %>
44      <tr><td></td><td><span class="FscError"><%=text020%></span></td></tr>
45    <% } %>
46    </tbody></table>
47    </form>
48    </body>
49    </html>
```

## 6.4 VApps

Der nun folgende Abschnitt ist den VApps des Sport-Portals gewidmet, welche ihren Dienst im Zentrum (Hauptfenster) der Web-Site verrichten. Alle hierfür entwickelten COO-Objekte (siehe 4.5.2) wurden in der **SportVApp** - Fabasoft-Komponente zusammengefasst.

Neben einem HTML-*Anwendungsformat* & einer *Anwendungssteuerung* (VApp-Dispatcher) wurden noch folgende VApps-***Anwendungen*** zum Starten der einzelnen Einträge in der Portal-Menüzeile eingerichtet:

**VApps**

| | |
|---|---|
| COO.1.980.1.1575 | ... **Neuen Benutzer** anlegen |
| COO.1.980.1.1538 | ... **Verwaltung** |
| COO.1.980.1.1356 | ... **Veranstaltungssuche** |
| COO.1.980.1.1344 | ... **Klubsuche** |
| COO.1.980.1.1249 | ... **Aktuelles** |

Jede einzelne dieser Anwendungen spaltet sich, nachdem sie aufgerufen wurde, in **zahlreiche *Anwendungssichten***, ***Sichten*** und ***Skriptkomponentenobjekte*** weiter auf. Möchte man auf alle involvierten Objekte dieser fünf vollständig implementierten VApps näher eingehen, würde der Rahmen dieser Arbeit sehr schnell gesprengt werden. Darum liegt das weitere **Hauptaugenmerk** auf der genauen Erläuterung einer VApp-*Anwendung* und der dazu in Verbindung stehenden Objekte. Die dabei erkennbaren **Konzepte** beim Einsatz dieser VApp finden sich dann in allen weiteren zugehörigen VApps wieder. Die komplette Wellness-Sport - Softwarelösung ist auf der beigelegten CD im „Projekte“-Ordner zum weiterem Studium enthalten.

### 6.4.1 Sicherheit

Nachdem die Benutzer des Portals unterschiedliche Berechtigungsstufen besitzen, müssen diese auch am **Anfang aller VApps kontrolliert** und weitergereicht werden, da für die Fabasoft-Components alle **SportPerson**s der Wellness-Sport - Site gleichberechtigt wären.

**SessionStart** ( *Skriptkomponentenobjekt* )

```
1   // LANGUAGE="JavaScript"
```

- Dieses Skript wertet die **GET-Parameter**, welche vom **`default.asp`** - Skript (siehe 6.3.1) mitgegeben werden, aus und stellt unter anderem fest, welche **Berechtigungsstufe** der aktuelle **Benutzer** im gerade selektierten **Club** besitzt. Die dabei gewonnenen Informationen bzw. Werte werden in VApp-Variablen (Session...) abgelegt. Auch diverse andere Daten bezüglich des Portals (PortalClub, Root...) und der zur Verfügung stehenden Zugriffsrechte (Level...) werden so in die aktuelle VApp weitergereicht.

```
2   // ---------------------------------------------------------------------------
3   // FN:  SessionStart
4   //
5   // ARG:
6   // IN:
7   // OUT: PortalClub
8   //      Root...
9   //      Level...
10  //      Session...
11  // ---------------------------------------------------------------------------
12  var coo_config="COO.1.980.1.1270";
13  // ---------------------------------------------------------------------------
14  var root  =coometh.GetParameterValue(5,"DICTIONARY");
15  var params=coometh.GetParameterValue(1,"DICTIONARY");
16  // ---------------------------------------------------------------------------
17  var obj_config       =coort.GetObject(coo_config);
18  var def_portalclub   =coort.GetAttributeDefinition("SPORTBASE@1.980:PortalClub");
19  var def_rootsporttype=
      coort.GetAttributeDefinition("SPORTBASE@1.980:RootSportType");
20  var def_rootregion   =coort.GetAttributeDefinition("SPORTBASE@1.980:RootRegion");
21  var def_rootcategory =
      coort.GetAttributeDefinition("SPORTBASE@1.980:RootTemplatecategory");
22  var obj_portalclub   =obj_config.GetAttributeValue(cootx,def_portalclub,0);
23  var obj_rootsporttype=obj_config.GetAttributeValue(cootx,def_rootsporttype,0);
24  var obj_rootregion   =obj_config.GetAttributeValue(cootx,def_rootregion,0);
25  var obj_rootcategory =obj_config.GetAttributeValue(cootx,def_rootcategory,0);
26  params.SetEntryValue("PortalClub"   ,0,obj_portalclub);
      //  (PortalClub = RootClub)
27  params.SetEntryValue("RootSportType",0,obj_rootsporttype);
28  params.SetEntryValue("RootRegion"   ,0,obj_rootregion);
29  params.SetEntryValue("RootCategory" ,0,obj_rootcategory);
30  // ---------------------------------------------------------------------------
31  var lev_guest =0; params.SetEntryValue("LevelGuest" ,0,lev_guest );
      // (UNKNOWN user)
32  var lev_start =1; params.SetEntryValue("LevelStart" ,0,lev_start );
      // (known user..)
33  var lev_user  =2; params.SetEntryValue("LevelUser"  ,0,lev_user  );
34  var lev_expert=3; params.SetEntryValue("LevelExpert",0,lev_expert);
35  var lev_admin =4; params.SetEntryValue("LevelAdmin" ,0,lev_admin );
```

```
36  // ------------------------------------------------------------------------------
37  var obj_vx        =root.GetEntryValue("vx");
38  var obj_language  =root.GetEntryValue("lx");
39  var obj_person    =root.GetEntryValue("ux");
40  var cmp_hashcode  =root.GetEntryValue("hx");
41  var obj_membership=root.GetEntryValue("mx");
42  var val_ix        =root.GetEntryValue("ix");
43  // ------------------------------------------------------------------------------
44  var obj_sportclub=obj_portalclub; // (The PortalClub has no SportMembers)
45  var val_access   =lev_guest;      // (Standard: Guest)
46  try {
47   var obj_ux=coort.GetCurrentUser();
48   obj_ux.SetUserLanguage(obj_language);
49
50   var def_webactive=coort.GetAttributeDefinition("SPORTBASE@1.980:WebActive");
51   var val_webactive=obj_person.GetAttributeValue(cootx,def_webactive,0);
52   if (val_webactive==false) throw("LevelGuest");
53
54   var def_hashcode =
       coort.GetAttributeDefinition("SPORTBASE@1.980:SessionHashCode");
55   var val_hashcode =obj_person.GetAttributeValue(cootx,def_hashcode,0);
56   if (val_hashcode!=cmp_hashcode) throw("LevelGuest");
57
58   val_access   =lev_start;
59   var def_membership=
       coort.GetAttributeDefinition("SPORTBASE@1.980:TheSportPerson");
60   var cmp_person    =obj_membership.GetAttributeValue(cootx,def_membership,0);
61   if (obj_person.GetAddress()!=cmp_person.GetAddress()) throw("PortalClub");
62
63   var def_sportclub =coort.GetAttributeDefinition("SPORTBASE@1.980:TheSportClub");
64   var def_access    =
       coort.GetAttributeDefinition("SPORTBASE@1.980:AccessLevelType");
65   obj_sportclub=obj_membership.GetAttributeValue2(cootx,def_sportclub,0);
66   val_access   =obj_membership.GetAttributeValue2(cootx,def_access,0);
67  }
68  catch (e) {}
69  var bln_portal=0; if (obj_portalclub.GetAddress()==obj_sportclub.GetAddress())
70  bln_portal=1;
71  // ------------------------------------------------------------------------------
72  params.SetEntryValue("SessionVX"         ,0,obj_vx);
73  params.SetEntryValue("SessionLanguage"   ,0,obj_language);
74  params.SetEntryValue("SessionPerson"     ,0,obj_person);
75  params.SetEntryValue("SessionHashCode"   ,0,cmp_hashcode);
76  params.SetEntryValue("SessionMembership" ,0,obj_membership);
77  params.SetEntryValue("SessionIX"         ,0,val_ix);
78
79  params.SetEntryValue("SessionClub"       ,0,obj_sportclub);
80  params.SetEntryValue("SessionAccessLevel",0,val_access);
81  params.SetEntryValue("SessionPortal"     ,0,bln_portal);
82  params.SetEntryValue("SessionActive"     ,0,1);
83  // ------------------------------------------------------------------------------
```

### 6.4.2 Beispiel - Newsboard

Beim Newsboard handelt es sich um eine **VApp**, welche die **aktuelle**n **Ereignisse** des Portals oder eines Clubs, je nachdem ob und welcher Verein auf der Navigationsleiste selektiert wurde, im Hauptfenster anzeigt. Besitzt der gerade angemeldete Benutzer die Rechte eines Experten, so kann er auch neue Einträge hinzufügen. Hat er sogar Admin-Rechte, so ist es ihm auch möglich die bestehenden News zu verändern bzw. selektiv zu löschen.

**A)**

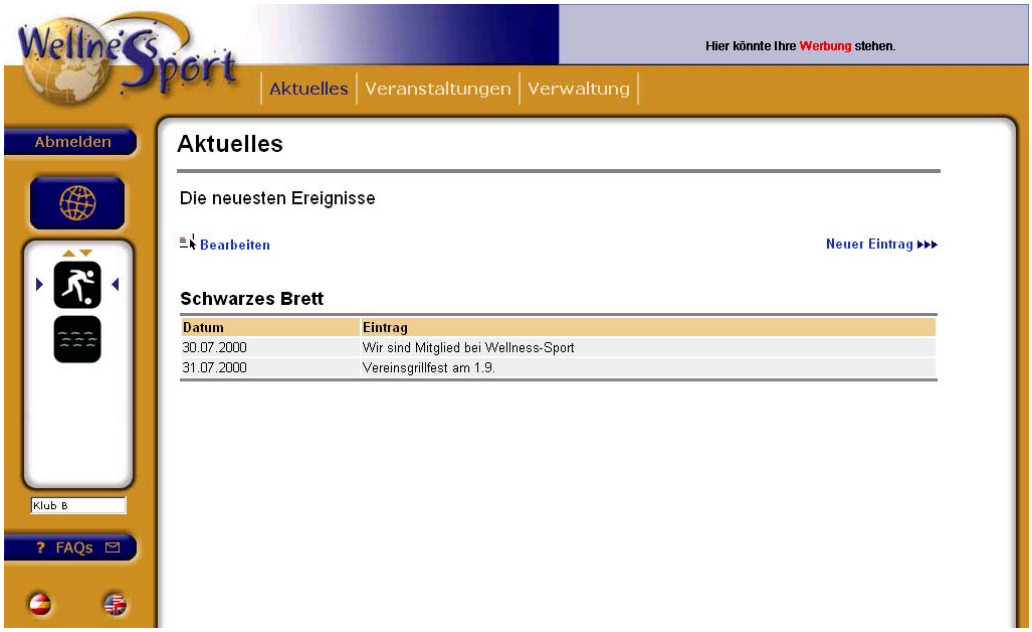
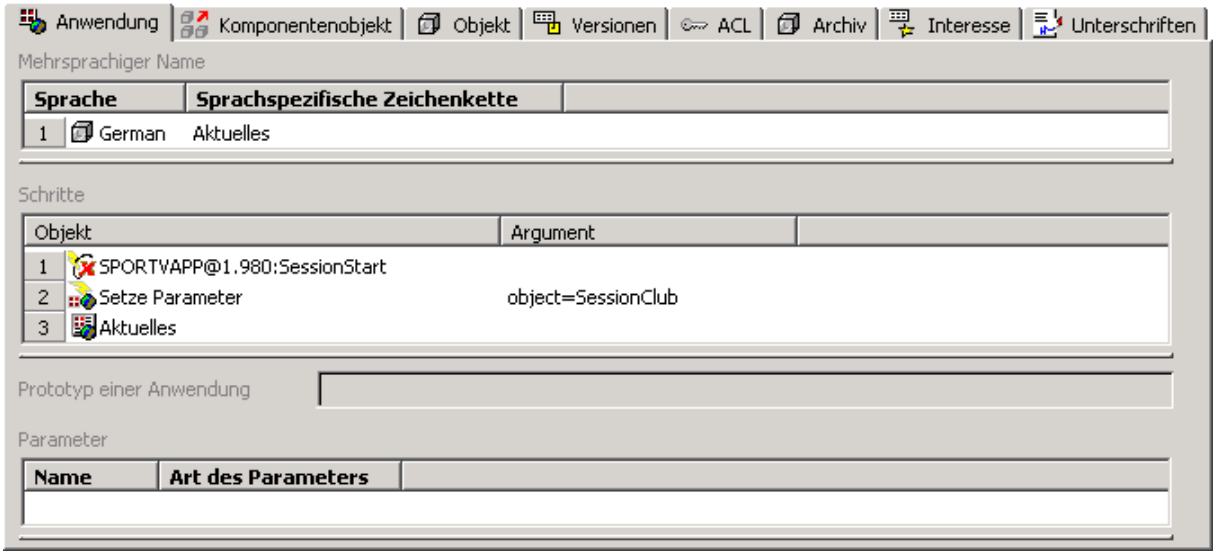



Abbildung 45: Sport-Portal - Aktuelles

Wie jede andere beginnt auch die VApp „**Aktuelles**“ (**A_Newsboard**) bei einem *Anwendungs*-Objekt, welches beliebig viele Schrittobjekte beherbergen kann. Dabei ist zu bemerken, dass der erste Schritt jeder SportVApp der Aufruf des *Skriptkomponentenobjektes* „SessionStart“ (siehe 6.4.1) sein muss. Danach wird der aktuell selektierte **Club** des jeweiligen Benutzers in die Standard-VApp-Variable „**object**“ übertragen, auf welche die darauf folgende *Anwendungssicht* „**Aktuelles**“ (**Av_Newslist**) zugreifen kann:

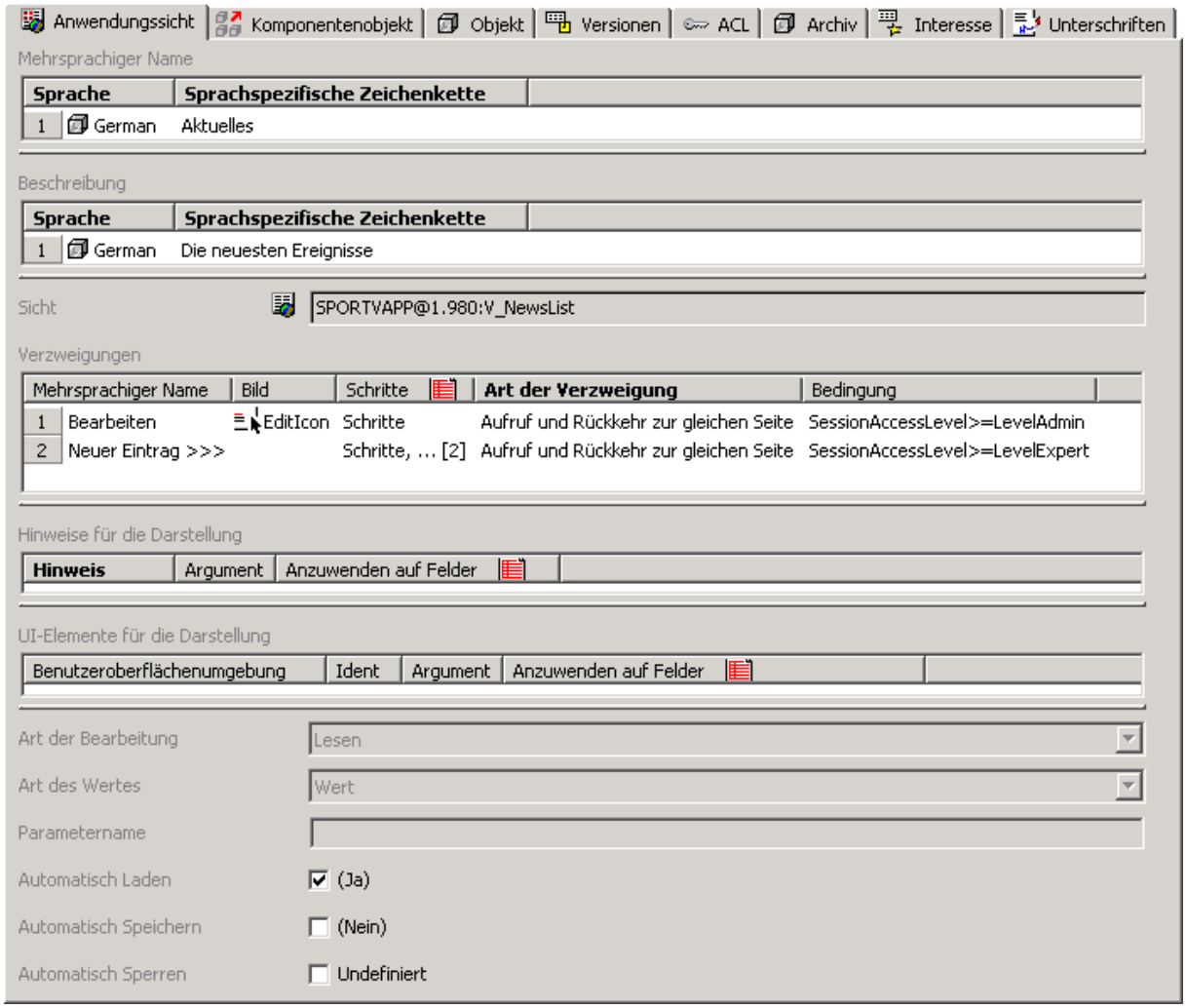
**A_Newsboard** ( *Anwendung* )



Die *Anwendungssicht* **Av_NewsList** beschreibt nun die **Darstellung** der Applikation:

- Direkt unter dem hier definierten Titel und dem Untertitel (siehe Abbildung 45) werden die angegebenen **Verzweigung**sbefehle (nur bei Erfüllung der dazugehörigen Bedingung → **Berechtigungsstufen**) ausgegeben.
- Danach werden die einzelnen Einträge des Newsboards mittels der *Sicht* **V_NewsList** an die Ausgabe angehängt.

**Av_NewsList** ( *Anwendungssicht* )




Wird zur **Laufzeit** vom Benutzer einer der **Verzweigung**sbefehle aufgerufen, so werden dann die dazugehörigen Schrittobjekte ausgeführt:

| ∇ Verzweigung | ∇ Schritte | ∇ Erklärung |
|---|---|---|
| [1] Bearbeiten | ♦ Aktuelles ändern | ♦ Wechsel zur *Anwendungssicht* **Av_NewsEdit** … siehe B) |

| | | |
|---|---|---|
| [2] Neuer Eintrag | ♦ SPORTVAPP@1.980: **AddNewsEntry_Club**<br>♦ Neuer Eintrag | ♦ Aufruf eines *Skript-komponentenobjekts*<br>♦ Wechsel zur *Anwendungssicht* **Av_NewsEntry** … siehe C) |

**AddNewsEntry_Club** ( *Skriptkomponentenobjekt* )

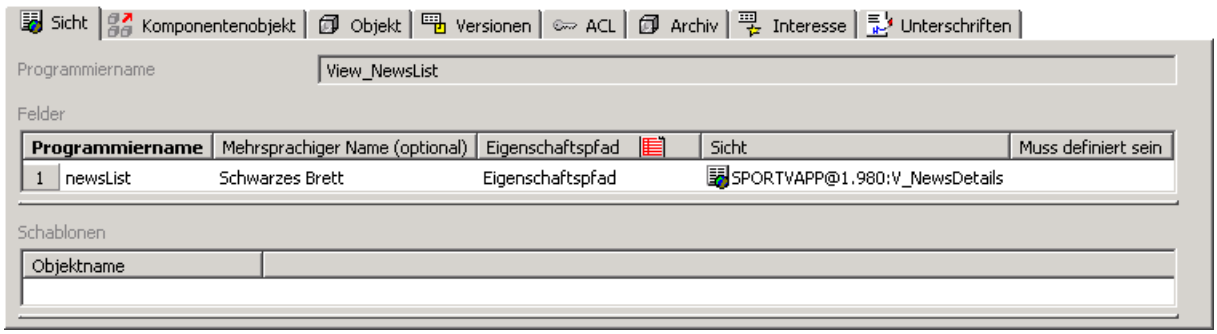
```
1   // LANGUAGE="JavaScript"
```

- Dieses Skript hängt einen neuen *ClubNewsEntry* - Eintrag am Ende des *ClubNewsBoards* des angegebenen *SportClubs* (standardmäßig der durch die VApp-Variable „object“ referenzierte) an. Die Änderung bleibt aber erst nach einem späteren Speichern / Übernehmen (commit) langfristig erhalten!

```
2   // ------------------------------------------------------------------------------
3   // FN:  AddNewsEntry_Club
4   //
5   // ARG: SportClub
6   // IN:
7   // OUT:
8   // ------------------------------------------------------------------------------
9   var params=coometh.GetParameterValue(1,"DICTIONARY");
10  var step  =coometh.GetParameterValue(9,"FSCVAPP@1.1001:Step");
11  try {var stparg=step.GetAttributeValue("FSCVAPP@1.1001:steparg");}
      catch (e) {var stparg="object";}
12  // ------------------------------------------------------------------------------
13  var club = params.GetEntryValue(stparg);
14
15  var def_news=coort.GetAttributeDefinition("SPORTBASE@1.980:ClubNewsBoard");
16  var cnt_news=club.GetAttributeValueCount(cootx,def_news); if (cnt_news>=30)
17  cnt_news=30;
18  for (idx=cnt_news; idx>0; idx--) {
19   var agg_news=club.GetAttributeValue2(cootx,def_news,idx-1);
20   club.SetAttributeValue(cootx,def_news,idx,agg_news);
21  }
22  var agg_news=coort.CreateAggregate(def_news);
23  club.SetAttributeValue(cootx,def_news,0,agg_news);
24  // ------------------------------------------------------------------------------
```

Jetzt folgt noch die bereits angesprochene *Sicht* **V_NewsList**, welche für jeden Eintrag in der newsList ( Eigenschaftspfad: „object“•→*ClubNewsBoard* ) die *Sicht* **V_NewsDetails** aufruft:

**V_NewsList** ( *Sicht* )



Die *Sicht* **V_NewsDetails** gibt nun das newsDate ( Eigenschafspfad: •→*ClubNewsDate* ) und den newsText ( Eigenschaftspfad: •→*ClubNewsString* ) eines einzelnen News-Eintrages aus:

**V_NewsDetails** ( *Sicht* )

Sicht | Komponentenobjekt | Objekt | Versionen | ACL | Archiv | Interesse | Unterschriften

Programmiername: View_NewsDetails

Felder

| | Programmiername | Mehrsprachiger Name (optional) | Eigenschaftspfad | Sicht | Muss definiert sein |
|---|---|---|---|---|---|
| 1 | newsDate | Datum | Eigenschaftspfad | | |
| 2 | newsText | Eintrag | Eigenschaftspfad | | |

Schablonen

| Objektname |
|---|
| |

**B)**

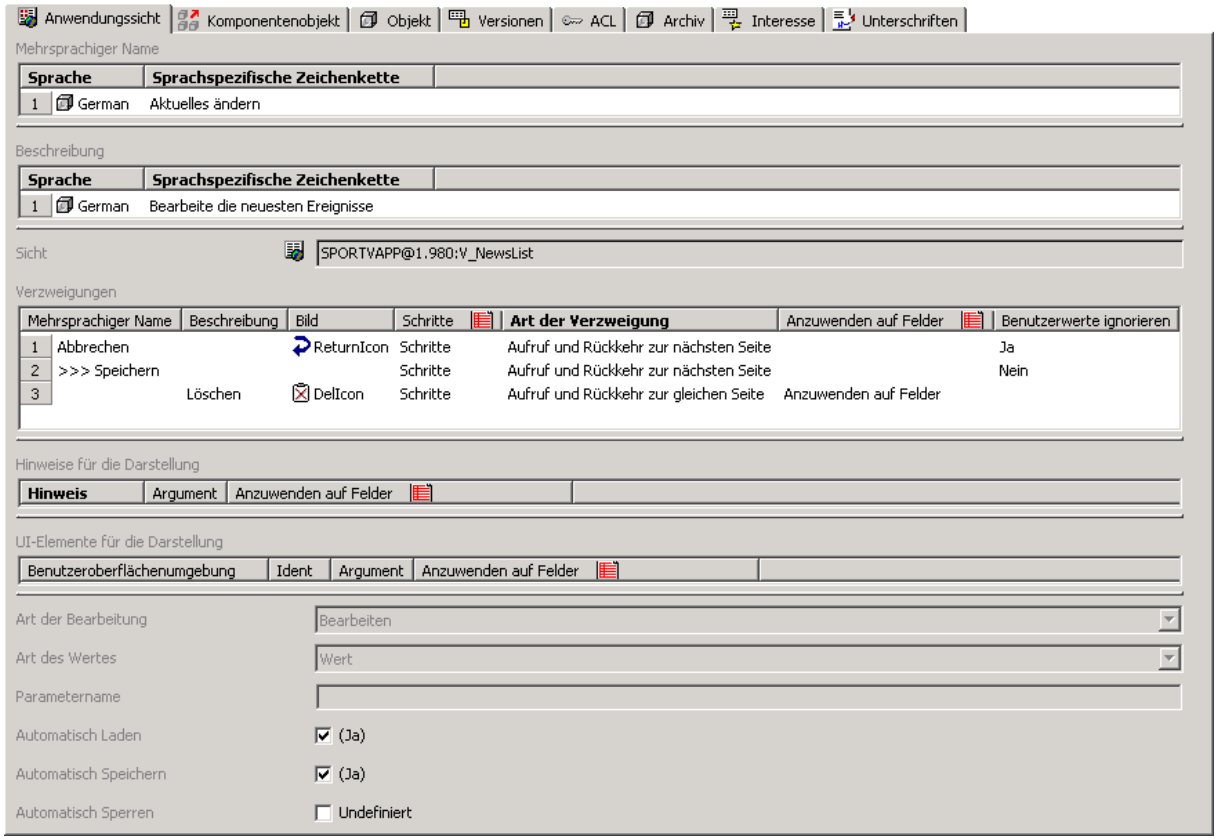

Abbildung 46: Sport-Portal - Aktuelles ändern

Die *Anwendungssicht* **Av_NewsEdit** beschreibt nun die **Darstellung** der Applikation im **Bearbeitungs-Modus**:

- Wieder werden diverse **Verzweigung**sbefehle definiert, wobei der **Löschen**-Befehl als einziger räumlich vor jeden News-Eintrag gesetzt wird (anzuwenden auf Felder: SPORTVAPP@1.980:**V_NewsDetails**•→newsDate ).

- Danach werden die einzelnen Einträge des Newsboards wieder mittels der *Sicht* **V_NewsList** an die Ausgabe angehängt. Nur durch Umsetzen des „**Automatisch Speichern**“-Wertes auf „**Ja**“ wird die Bearbeitungsmöglichkeit aktiviert.

**Av_NewsEdit** ( *Anwendungssicht* )




Wird zur **Laufzeit** vom Benutzer einer der **Verzweigung**sbefehle aufgerufen, so werden dann die dazugehörigen Schrittobjekte ausgeführt:

| ▽ Verzweigung | ▽ Schritte | ▽ Erklärung |
|---|---|---|
| [1] Abbrechen | ♦ Übernehmen false | ♦ Die durchgeführten Änderungen werden verworfen (Rollback). Danach kehrt man automatisch zur vorherigen *Anwendungssicht* zurück. |
| [2] Speichern | ♦ Übernehmen true | ♦ Die durchgeführten Änderungen werden gespeichert (Commit) Danach kehrt man automatisch zur vorherigen *Anwendungssicht* zurück. |
| [3] Löschen | ♦ SPORTVAPP@1.980: **DeleteNews_Club** | ♦ Aufruf eines *Skript-komponentenobjekts* |

**DeleteNews_Club** ( *Skriptkomponentenobjekt* )

```
1   // LANGUAGE="JavaScript"
```

- Dieses Skript löscht den *ClubNewsEntry* - Eintrag mit dem übergebenen Index im angegebenen *SportClub* (standardmäßig der durch die VApp-Variable „object“ referenzierte).

```
    // ---------------------------------------------------------------------------
3   // FN:  DeleteNews_Club
4   //
5   // ARG: SportClub
6   // IN:  sys_branchindex
7   // OUT:
8   // ---------------------------------------------------------------------------
9   var params=coometh.GetParameterValue(1,"DICTIONARY");
10  var step  =coometh.GetParameterValue(9,"FSCVAPP@1.1001:Step");
11  try {var stparg=step.GetAttributeValue("FSCVAPP@1.1001:steparg");}
      catch (e) {var stparg="object";}
12  // ---------------------------------------------------------------------------
13  var obj_club=params.GetEntryValue2(stparg);
14
15  var index=params.GetEntry3("sys_branchindex").getItem(1);
16  var news =coort.GetAttributeDefinition("SPORTBASE@1.980:ClubNewsBoard");
17  obj_club.SetAttributeValue(cootx,news,index,null);
18  // ---------------------------------------------------------------------------
```

C)

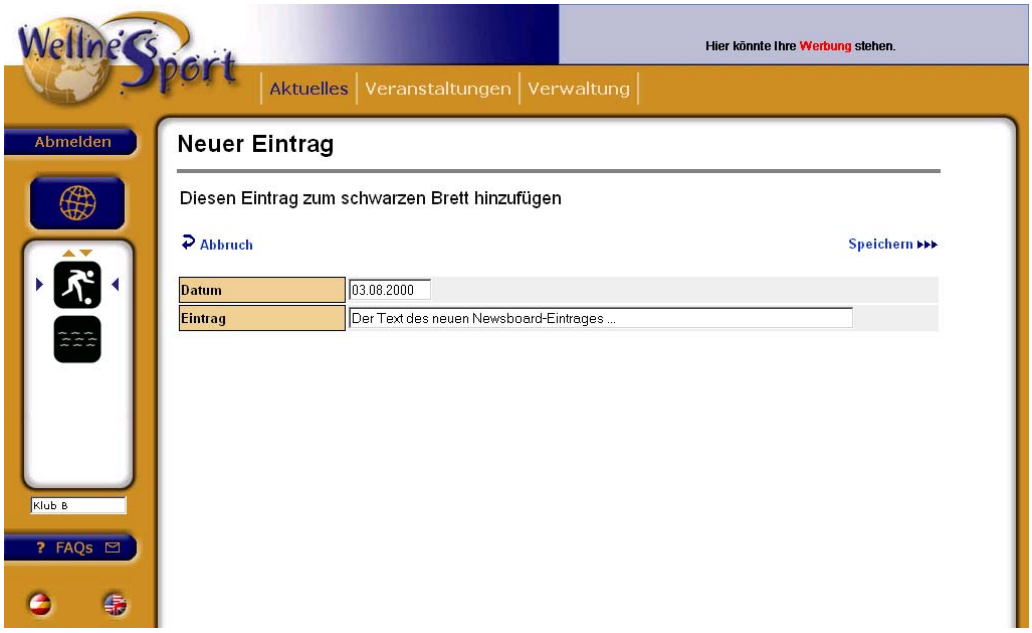


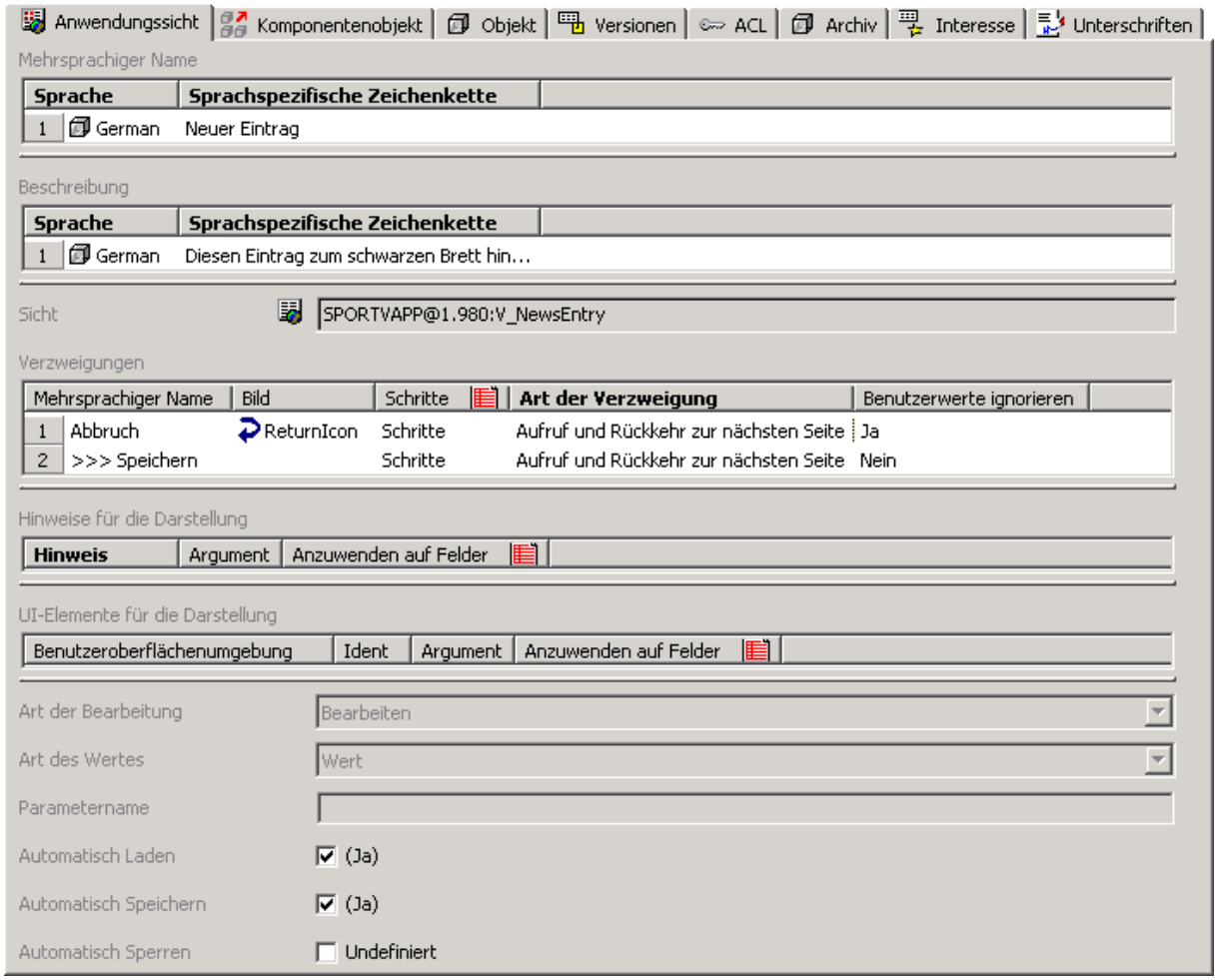

Abbildung 47: Sport-Portal - Aktuelles - Neuer Eintrag

Die *Anwendungssicht* **Av_NewsEntry** dient zur **Darstellung** der Applikation im Modus zur **Neueingabe eines News-Beitrags**:

- Diesmal werden nur zwei einfache **Verzweigung**sbefehle definiert.

- Danach wird die Eingabemaske mittels der neuen *Sicht* **V_NewsEntry** an die Ausgabe gesendet. Wieder ist das „**Automatisch Speichern**“ eingeschaltet.

**Av_NewsEntry** ( *Anwendungssicht* )



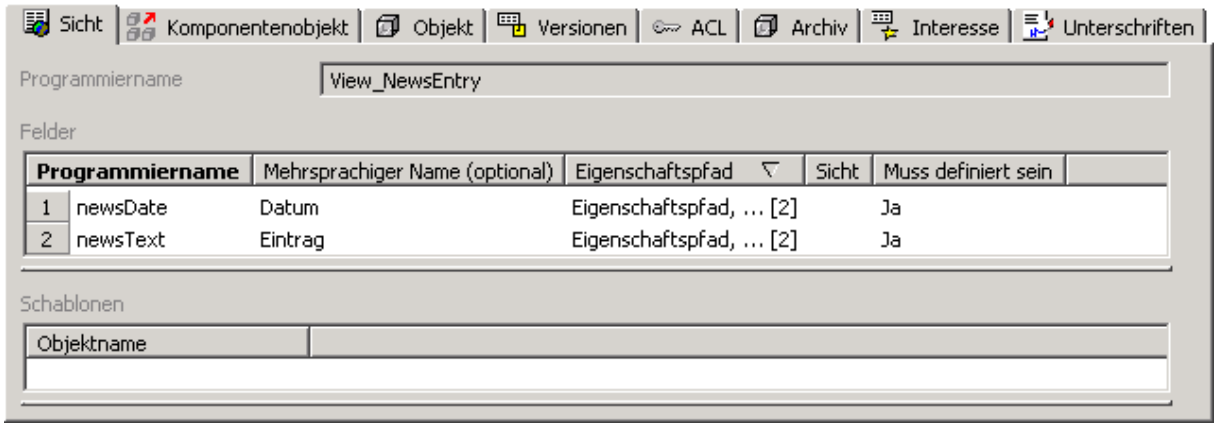

Wird zur **Laufzeit** vom Benutzer einer der **Verzweigung**sbefehle aufgerufen, so werden dann die dazugehörigen Schrittobjekte ausgeführt:

| ▽ Verzweigung | ▽ Schritte | ▽ Erklärung |
|---|---|---|
| [1] Abbrechen | ♦ Übernehmen false | ♦ Die durchgeführten Änderungen werden verworfen (Rollback). Danach kehrt man automatisch zur vorherigen *Anwendungssicht* zurück. |
| [2] Speichern | ♦ Übernehmen true | ♦ Die durchgeführten Änderungen werden gespeichert (Commit) Danach kehrt man automatisch zur vorherigen *Anwendungssicht* zurück. |

Abschließend folgt noch die *Sicht* **V_NewsEntry**, welche die Eingabemaske für einen neuen Newsboard-Eintrag ausgibt. Folgende Felder können dabei eingegeben werden:

newsDate ( Eigenschaftspfad: “object“•→*ClubNewsBoard*•→*ClubNewsDate* ) und
newsText-Feld ( Eigenschaftspfad: “object“•→*ClubNewsBoard*•→*ClubNewsDate* )

**V_NewsEntry** ( *Sicht* )




### 6.4.3 Weitere Applikationen

Zum Abschluss dieses Kapitels werden nun noch einige typische Szenarien der restlichen VApps anhand kurzer Diashows gezeigt, um einen kleinen Einblick in die wesentlichsten Features des Web-Portals zu geben:

**Die Klubsuche:**

Suchen

⇓


Klub A

⇓


Abbildung 48: Sport-Portal - Klubs

**Die Veranstaltungssuche:**


Suchen

⇓


4. Hausspiele

⇓


Abbildung 49: Sport-Portal - Veranstaltungen

**Die Verwaltungsmöglichkeiten:**

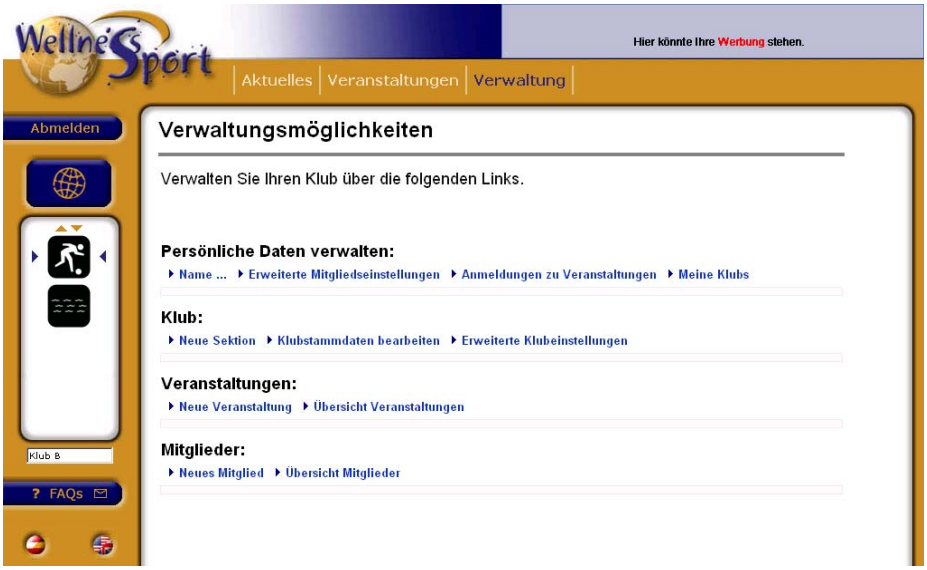



Abbildung 50: Sport-Portal - Verwaltung

## 6.5 Installation

Das gesamte Sport-Portal (**Wellness-Sport Version 1.0**) befindet sich gezippt auf der beigelegten CD im Ordner „Projekte“. Bitte beachten sie bei einer **Installation** folgende Punkte:

- Folgende Basiskonfiguration wird benötigt:
  - Windows-NT4-Server basierendes OS
  - MS-IIS 4 oder höher!
  - MS-SQL-Server 7 oder höher!
  - Fabasoft Components Version 4.0 RC1 (25.09.2000)
  - plus ComponentsBase
  - plus ComponentsWeb
- Zusätzlich noch zwei selbstentwickelte „Fabasoft Components“ hinzufügen → Components-Sport (Die benötigten coo-Files liegen im „Components-Sport.zip“):
  - SportBase
  - SportVApp
- Nun eine „Web Site“ oder ein „Virtual Directory“ namens „Wellness-Sport“ im IIS erstellen:
  - Dieser Eintrag im IIS benötigt einen „anonymous access“.
  - Die benötigten Web-Files liegen im „Wellness-Sport.zip“.
- Danach folgende Punkte durchführen:
  - Die IIS-Einträge „FSCDAV” und „FSCASP“ benötigen ebenfalls „anonymous access“.
  - Kopiere den im „Components-Sport.zip“ enthaltenen Ordner „SPORTVAPP_1_980_SportVAppDispatcher“ nach „Fabasoft/Components/Web/ASP/content/tmp/“
- Abschließend muss noch das „SPORTBASE@1.980:SportPortalConfig“ - Objekt vollständig eingerichtet werden, z.B.:

- SportPortalConfig → PortalClub: *Portal* → WebMenuCommand:
  - 1 *newsboard*
  - 2 *clubs*
  - 3 *events*
  - 4 *admin*
- SportPortalConfig → RootRegion: *World*
- SportPortalConfig → RootSportType: *Sport*
- SportPortalConfig → RootTemplatecategory: *Template*
- SportPortalConfig → PortalURL, VAppURL, VDavURL:
  - *http://Wellness-Sport/*
  - *http://Wellness-Sport/fscasp/content/bin/fscvext40.dll*
  - *http://Wellness-Sport/fscdav/*

# Zusammenfassung

Der Begriff "Electronic-Commerce" wird heutzutage von den meisten Menschen sofort mit Online-Shop-Lösungen und Shopping-Portalen assoziiert. Doch schließt EC eine weitaus größere Palette an möglichen Anwendungsgebieten ein.

So kann beinahe jede Kommunikation und Transaktion zwischen Privatpersonen, Firmen und der Regierung durch EC-Lösungen unterstützt bzw. revolutioniert werden. Dadurch sind die verschiedensten Geschäftsprozesse (z.B.: Informationsaustausch, Bestellwesen, Logistik, Zahlungsverkehr, Support, ...) wesentlich effizienter gestaltbar, wodurch Geld und Zeit gespart werden kann. Doch auch völlig neue Geschäftsfelder eröffnen sich durch den Einsatz von EC. Voll im Trend liegt dabei z.B. die Vermarktung spezieller Web-Dienste und Web-Anwendungen (wie z.B.: Free-Mailer, Kommunikationsplattformen, Wissenssammlungen, Content-Management-Systeme, ...).

Bedient werden solche EC-Systeme meist über einen Web-Browser am eigenen Rechner. Das eigentliche Programm, welches die verschiedenen Benutzereingaben aber schließlich verarbeiten soll, befindet sich oft weit entfernt auf einem fremden Web-Application-Server. Die Verbindung zwischen diesen zwei Rechnern erfolgt im Normalfall über das HTTP-Protokoll nach dem Anforderungs-Antwort-Prinzip und wird zwischenzeitlich immer wieder getrennt. Daher müssen geeignete Verfahren (z.B.: durch Synchronisierung über Session-IDs) eingesetzt werden, welche die einzelnen Web-Anwendungen zusammenhalten. Auch werden die Eingaben eines Benutzers immer erst durch einen expliziten Anstoß zum Server übertragen, was zu einer gewissen "Blindheit" des Servers führt. Darum ist eine der Hauptaufgaben in der Web-Programmierung die effiziente Unterstützung aller vorhandenen HTTP-Möglichkeiten. Alle modernen serverseitigen Web-Sprachen haben daher eigene Funktionen hierfür inkludiert. Noch einen Schritt weiter gehen objektorientierte Web-Entwicklungsumgebungen (wie z.B: Fabasoft-VApps oder ASP.net), bei denen sich die zur Verfügung gestellten Objekte selbstständig um die Kommunikation mit dem Web-Client kümmern.

Egal aber für welche webbasierte Programmiersprache man sich dann bei einer konkreten Erstellung einer EC-Lösung entscheidet, man sollte sich auf alle Fälle mit den zur Verfügung stehenden Konzepten und Standards im Zusammenhang mit der Client-Server-Programmierung im Internet auseinandergesetzt haben. Nur so kann verstanden werden, wie Web-Anwendungen intern wirklich mit den Clients kommunizieren.

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Literaturverzeichnis


[1] http://www.ecc-handel.de/ecinfos/einsteiger/1_einfue/einfueh_ec.php,
*Einführung in E-Commerce* (16.9.2002)

[2] http://www.fim.uni-linz.ac.at/Diplomarbeiten/diplomarbeit_hudak/Intro.htm
**Heidi Hudak**, *Diplomarbeit „Einführung von E-Commerce in Klein- und Mittelbetrieben - Methoden, Risken und Chancen“* (2001)

[3] **Jörg Krause**, *Praxishandbuch Electronic Commerce*, Carl Hanser Verlag (1999)

[4] **Dirk Schneider**, **Gerd Schnetkamp**, *E-Markets - B2B-Strategien im Electronic Commerce*, Dr. Th. Gabler Verlag (2000)

[5] http://selfhtml.teamone.de/,
**Stefan Münz**, *SelfHTML* (27.10.2001)

[6] **Dr. Rainer Lischka**, *E-Commerce - Techniken für den Handel im Internet* (1999)

[7] **Meinhardt Schmidt, Thomas Demmig**, *SQL*, mitp-Verlag (2001)

[8] **Henning Behme, Stefan Mintert**, *XML in der Praxis*, Addison-Wesley Verlag (2000)

[9] **Rechenberg, Pomberger**, *Informatik Handbuch*, Carl Hanser Verlag (1997)

[10] **Andrew S. Tanenbaum**, *Computer-Netzwerke*, VMI Buch AG (1992)

[11] **Ch. Lindemann, Ch. Immler, F. Harms**, *Internet intern*, Data Becker GmbH & Co. KG (1999)

[12] **Tobias Weltner**, *Active Server Pages lernen und beherrschen*, Microsoft Press (1999)

[13] **Jörg Krause**, *PHP - Grundlagen und Lösungen*, Carl Hanser Verlag (2000)

[14] **Mark Kronsbein, Thomas Weinert**, *PC Spicker PHP4*, Sybex-Verlag GmbH (2001)

[15] http://www.php.net/docs.php,
*PHP: Documentation* (12.5.2002)

[16] **Jörg Krause**, *Microsoft Site Server 3.0*, Addison Wesley Longman Verlag GmbH (1999)

[17] http://msdn.microsoft.com/library/default.asp?URL=/library/psdk/iisref/aspguide.htm, *ASP: Guide* (8.7.2002)

[18] http://www.ebxml.org/specs/index.htm, *ebXML - Specifications* (18.9.2002)

[19] http://hoohoo.ncsa.uiuc.edu/cgi/interface.html, *The CGI Specification* (30.9.2002)

[20] http://www.payer.de/cmc/cmcs0.htm, **Margarete Payer**, *Computervermittelte Kommunikation* (24.5.2002)

[21] http://www.ietf.org/rfc.html, *IETF RFC Page* (4.10.2002)

[22] http://torsten-horn.de/techdocs/, **Thorsten Horn**, *Technische Kurzdokumentationen* (12.10.2002)

[23] **Beer, Birngruber, Mössenböck, Wöß**, *Die .NET-Technologie*, dpunkt.verlag, (2002)

[24] *dot.net magazin 05.02*, Software & Support Verlag GmbH (2002)

[25] http://www.aspheute.com/kategorien/ASPdotNET.htm, *Einführung in ASP.NET Web Forms* (19.10.2002)

[26] http://www.selfphp.info/, **Damir Enseleit, Matthias Hupp**, *SelfPHP* (3.10.2002)

[27] http://www.fim.uni-linz.ac.at/Research/default.htm, **Michael Sonntag, Susanne Reisinger**, *Important Factors for E-Commerce* (14.10.2002)

[28] http://www.fim.uni-linz.ac.at/Diplomarbeiten/diplomarbeit_kotek/Diplomarbeit-Kotek.zip. **Georg Kotek**, *Diplomarbeit „Der Internet-Shop“* (2002)

# Curriculum Vitae

| | |
|---|---|
| Name | **Ritz** |
| Vorname | **Gernot** Gerhard Viktor |
| Geburtsdatum und Ort | 16. Januar 1971 in Linz a.D. |
| Anschrift | Stelzerstraße 43<br>A - 4020 Linz |
| Nationalität | Österreich |
| Kontakt | gernot.ritz@gmx.at |
| Familienstand | Verheiratet seit dem 29. September 2001 |
| Ausbildung | 4 Klassen Volksschule<br>5 Klassen Bundesrealgymnasium<br>3 Klassen Berufsschule für Einzelhandelskaufmann<br>Nov. 1991 – Lehrabschlussprüfung<br>Hochschullehrgang für die Studienberechtigungsprüfung (TNF)<br>Okt. 1993 – Studienberechtigungsprüfung<br>Studium Informatik an der Johannes Kepler Universität Linz<br>Okt. 1996 – Erste Diplomprüfung |
| Beruf | 01.08.1988 - 31.07.1991 – Lehrling bei der Fa. Niedermeyer<br>01.08.1991 - 01.01.1992 – Verkäufer bei der Fa. Niedermeyer<br>01.06.2000 - 15.01.2001 – Software Engineer bei Wellness-Soft<br>01.02.2001 - – Software Engineer bei Fa. Hexagon |
| Bundesheer | 02.01.1992 - 31.08.1992 – Präsenzdienst |
| Tätigkeiten | Ferialarbeit bei der Firma LinzNet Internet Dienstleistungen<br>EDV und grafische Arbeiten für die Firma PerformDance<br>Netzwerk und Administration der Firma Kasperek KG<br>Schuhgroßhandel Kasperek KG Internet-Shop-Projekt<br>Internetpräsenz des Österreichischen Tanzsportverbandes<br>Y2k Umstellung der Telekom in Zusammenarbeit mit S&P<br>Internetprojekt der Imperial-Finanzgruppe (Fa. Creativa)<br>Intranetprojekt LISA - Logistikapplikation (Fa. Burg)<br>DNO - Distributed Network Objects (Fa. Hexagon)<br>Stoßofen 6&7 VA Stahl - Materialverfolgung (VA Tech) |

# Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und alle den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Linz, Dezember 2002

Gernot Ritz