



Near East University

Faculty of Engineering

Department of Computer Engineering

Books E-Commerce by ASP Pages

**Graduation Project
COM-400**

Student: Vehbi Okay Dilek

Supervisor: Mr. Ümit İlhan

Nicosia-2007

ACKNOWLEDGEMENTS

First of all, thanks to my mom and dad for raising me, supporting me, and pushing me towards fulfilling all of my educational goals and aspirations

Thanks to my all preltors for supporting me and pushing me to be a successful student. Especially, Mr. Ümit İlhan as my supervisor for his invaluable advice and belief in my work and myself over the course of this Graduation Project.

Finally, I thanksgiving the Infinite Power of the Universe to give me the patience to complete my tasks in my life.

ABSTRACT

In this project, an interactive client server Web technology will be inspected. There are two kinds of the Web technology. The one is Microsoft's Active Server Pages technology and the other is PHP which based on linux/unix like systems.

The technologies both are a revolutions in Web sector. The main idea is of them establishing an interactive connection between clients and servers. Earlier, the Web technology was static which means the sending information from the servers to the clients are same. Also, the servers does not care about what kinds of clients it serves to. Thus there is no any updated information could be send to the users, and it is the information depends on the clients needs.

For example, what a client thinks about your internet site? Can you know? Off course, it is not possible the old web technologies. But now the web technologies make it possible. First of all, we must get data from users, and keep them in databases to later use. However, we also need some little programms called scripts to implement some job to get result for response to the user requests. Also the scripts make any operations in more and more secure.

The knowing Scripts are VB, Java, and Perl. These are programming languages that we will use them for the jobs.

All in all, this new approach takes effect our lifes. Because we use this technologies a lot in our daily life. The implementation of the interactive pages make our Web Surving more usefull and reliable.

TABLE OF CONTENTS

ACKNOWLEDGMENT	i
ABSTRACT	ii
TABLE OF CONTENTS	iii
INTRODUCTION	iv
CHAPTER ONE: ACTIVE SERVER PAGES	1
1.1. Client Side vs. Server-Side Web Technologies	1
1.1.1. The Web's Client-Server Relationship	1
1.1.2. Which Web Servers Supports Active Server Pages?	2
CHAPTER TWO: WRITING ASP PAGES	3
2.1. ASP Code In An HTML Context	3
2.1.1. ASP delimiters <% %>	3
2.1.2. ASP Code Is "Language Neutral"	4
2.2. VBScript Statements to Execute in ASP	4
2.2.1. Enclosing Scripting Code within your ASPs	5
2.2.2. String in VBScript	6
2.2.3. An Abbreviation for Response.Write	6
2.3. Developing ASPs	7
2.3.1. Executing Your ASPs	7
2.3.2. Writing Your ASPs	8
2.3.2.1. ASP Directives, and Comments	8
2.3.3. VBScript Comments	9
2.4. VBScript Flexibility	10
CHAPTER THREE: VARIABLES	10
3.1. Declaring Variables	11
3.1.1. Letting VBScript declare variables implicitly	11
3.1.2. Syntax rules for variables	11
3.2. Data types: string, integer, and Boolean	12
3.3. Variable concatenation	13
3.4. Variables and Math	15
3.5. Variable naming conventions	15
3.5.1. Using understandable variable names	15
3.6. Organize Variables with Option Explicit	15
CHAPTER FOUR: VBScript's HOUSE of BUILT-IN FUNCTIONS	17
4.1. Built-in functions	17
4.1.1. Conversion functions	17
4.1.2. String functions	18
4.1.3. Date/time functions	18
4.1.4. Format functions	18
4.1.5. Math functions	18
4.1.6. Array functions	18
4.1.7. Conversion functions	18

CHAPTER FIVE: OPERATORS

- 5.1. Assignment operators
- 5.2. Logical operators
- 5.3. Comparison operators
- 5.4. Mathematical operators
- 5.5. Assignment operators
- 5.6. Logical operators
- 5.6.1 AND, OR
- 5.6.2. NOT
- 5.7. Comparison operators
- 5.8. Mathematical operators

CHAPTER SIX: CONDITIONAL LOGIC

- 6.1. The If statement
- 6.2. The If . . . Then . . . Else statement
- 6.3. The If . . . Then . . . ElseIf statement
- 6.4. Loops in ASP
- 6.4.1. For-Next Loop
- 6.4.2. While-wend Loop
- 6.4.3. Do-Loop

CHAPTER SEVEN: COOKIES

- 7.1. Response.Cookies and Request.cookies
- 7.2. Cookie expiration
- 7.3. Updating cookies
- 7.4. Deleting cookies
- 7.5. A word of warning about the use of cookies

CHAPTER EIGHT: DATABASES

- 8.1. Creating a database
- 8.1.1. Inside Microsoft Access
- 8.2. Database design
- 8.2.1. Object naming conventions
- 8.2.2. Creating tables in Access
- 8.2.3. Creating tables in SQL Server
- 8.3. Relational databases and referential integrity
- 8.3.1. Creating a relationship in Access
- 8.4. SQL Server views and Access queries
- 8.5. Fundamental SQL
- 8.5.1. Selecting all records from a table
- 8.5.2. Selecting all records that meet one criterion
- 8.5.3. Selecting all records that meet several criteria (using AND)
- 8.5.4. Selecting records that meet one or more of several criteria (using OR)
- 8.6. Useful SQL keywords
- 8.6.1. COUNT
- 8.6.2. SUM
- 8.6.3. TOP 8.6.4. BETWEEN
- 8.6.4. BETWEEN

8.6.5. ORDER BY	45
8.6.6. IN	47
8.6.7. GROUP BY	47
8.6.8. DISTINCT	49
8.6.9. Going on a DATE	49
8.7. Making the connection	49
8.7.1. Setting up a DSN to an Access database	50
8.7.2. Using commands	50
8.7.3. Inserting a record	51
8.7.4. Updating a record	52
8.7.5. Updating multiple records (a simple example)	52
8.7.6. Deleting a record	53
8.7.7. Deleting multiple records (simple example)	54
CHAPTER NINE:PROCESSING DATA from ONLINE FORMS	55
9.1. The Action Attribute	55
9.2. The Get and Post Methods	55
9.3. ASP Objects	56
9.3.1. The Request and Response Objects	57
9.3.2. Request Object Collections	57
9.4. Using Values Returned by a Form In Math	59
CHAPTER TEN:PROJECT ASP RUNNING SESSION EXPLANATION	60
COCLUSION	73
APPENDIX	74
REFERENCES	157

INTRODUCTION

The basic Web server performs a simple task which receives HTTP requests from client computers, generally from a browser. Web page requested, and it responds by serving up the page to the client. Active Server Pages (ASP) changed and improved the classical technology. ASPs sent scripts to the client site and able to get results from client side. Thus an interactive communication between server and the client is performed.

No all of the server systems support this technology. The best one for ASP is Microsoft Internet Information Server on Windows NT Server 4.0 and later. PWS is the Microsoft Personal Web Server is an other choice for Windows 2000 Server and the earlier ones. These are required to work with ASP pages. Also ASP has versions, such as Active Server Pages 1.0, Active Server Pages 2.0 and the version 3.0.

Writing ASP pages is not too difficult for a person that has a basic programming knowledge. Especially, a developer experienced on VB and Java, perform ASP application easily. Of course ASP is a programming language, so it has syntax rules. Therefore, carefully writing is necessary for ASP as the other languages have. However ASP servers same flexibilities such as insensitiveness that we don't have to care about capitalization of letters and so on. Also ASP is free for which code you could use such as JavaScript, VBScript or PerlScript.

Chapter One describes what is ASP and how it works, and its requirements to implement. And Chapter 2,3,4,5,6 describe the basis and rules writing ASP, declaration of variables in ASP, built in functions for VBScripting, the operators to use in, and the logical control statements.

Chapter Seven describes the cookie mechanism in ASP pages, Chapter Eighth describe everything about DataBase implementation in ASPs, Chapter Nine about getting information from the user using by the form pages, and the last one the running sessions of my applied ASP project in described in steps.

CHAPTER ONE:

ACTIVE SERVER PAGES

1.1. Client Side vs. Server-Side Web Technologies

The basic Web server performs a simple task which receives HTTP requests from client computers, generally from a browser like Mozilla or Internet Explorer, it searches through its directory structure for the Web page requested, and it responds by serving up the page to the client. The Web page is streamed to the browser as simple text containing HTML markup and perhaps JavaScript. It is the browser itself that parses and interprets the code, rendering it into the attractive and functional content that is displayed on the screen. The page might also contain JavaScript code that causes buttons to light up or prompts to appear. HTML and JavaScript are parsed and processed on the client computer by the browser which are called client-side Web technologies.

The Web environment today requires more than static HTML pages. Today's Web sites must deliver dynamically generated, customized content to the browser. They greet the visiting user by name, collect information entered in online forms, compose and send customized e-mail, conduct searches based on keywords supplied by the client, and access and modify the contents adds. The operations that provide this dynamic functionality are executed on the Web server which are called server-side applications.

In the early days of Web server-side programming, developers mostly wrote their Web server-side applications in CGI scripting languages like Perl. In 1996, Microsoft Corporation released Active Server Pages (ASP), a Web scripting technology, to run on its Internet Information Server (ISS) and Personal Web Server (PSW). ASP pages (or ASPs), contain scripted programming statements that execute on the server, providing the dynamic interactivity central to today's Web application.

1.1.1. The Web's Client-Server Relationship

The request of each HTTP request is discrete and unique connection between the client and the Web server. The server remembers nothing about the client between page requests, and it regards every HTTP request as brand-new, coming from a brand-new client. Because the server forgets everything it knows about a client after it has served up a response, the Web environment is known as a stateless protocol. This lack of a persistence of state of connectivity between the client and the server presents a special set of challenges to the Web developer, whose Web applications must remember the contents of shopping carts, login information, or personal and technical information throughout the client's browsing "session". Web server-side applications play any integral role in providing functionality, including cookies and session objects, that can make different kinds of

To run ASP on a non-ISS Web server, Chili!SOFT makes a commercial add-on application that provides support for ASP on a wide array of additional servers. In my project, I will be running ASP 2.0 on ISS 4.0 in the Windows XP Professional operating system.

ASP Version		Web Server Version		Operating System	
Active Server Pages 1.0	ISS 3.0	PWS 1.0	Windows NT Server 4.0	Windows 95	
Active Server Pages 2.0	ISS 4.0	PWS 4.0	Windows NT Server 4.0 and later	Windows 95, 98, and NT Workstation 4.0	
Active Server Pages 3.0	ISS 5.0	PWS 5.0	Windows 2000 Server	Windows 2000 Professional	
KEY: ISS		Microsoft Internet Information Server		PWS	
				Microsoft Personal Web Server	

Table 1.1. The table of supported server versions.

Active Server Pages (ASP), is a proprietary product of Microsoft Corporation. Various versions of ASP ship with various versions of the Microsoft Windows operating system and Web server applications.

1.1.2. Which Web Servers Supports Active Server Pages?

information about the client persist between connections, helping to maintain state.

CHAPTER TWO:

WRITING ASP PAGES

2.1. ASP Code In An HTML Context

Most ASP pages will look familiar to the experienced HTML developer who look a lot like HTML pages. Closer inspection, however, will reveal ASP code throughout the page, inserted within the HTML markup and content, and set off inside of special tags that look like this:

```
<% your ASP code here %>
```

This provides an optimum development environment for the Web application development team who the HTML designers can create the font-end, static HTML content, and then turn the pages over to the server-side team to add in the ASP code that will create the dynamic content.

2.1.1. ASP delimiters <% %>

All ASP code must be written within ASP delimiters, like this:

```
<% Response.Write "Hello World" %>
```

This is also suitable:

```
<%  
Response.Write "Hello World"  
%>
```

ASP will ignore the empty spaces between the opening delimiter and the beginning of your code line, and between the end of your code line and the closing delimiter. A pair of open and closed delimiters is considered to be an ASP block. These delimiters let the server know that ASP code is coming, so that it knows to parse the code properly (and not think that it's HTML, PHP, or any other language).

Also, You can close and reopen new delimiters for each line of ASP code, but it's better practice to keep related lines of ASP code in one block. For example, look at the following two code snippets, which show the same code written in one and two blocks.

Here's the one-code-block version:

```

<%
Response.Write "Vehbi"
Response.Write "Okay"
%>

```

And here's the same code divided into two code blocks:

```

<%
Response.Write "Vehbi"
%>
<%
Response.Write "Okay"
%>

```

The style is up to you, but always keep your code consistent. Remember that it's a better to group similar lines of ASP code into one code block, but it's possible to break lines into multiple code blocks. Sometimes this is actually necessary, since HTML often must be surrounded by ASP code.

2.1.2. ASP Code Is "Language Neutral"

Microsoft promotes ASP as a "language neutral" scripting environment, meaning it can be coded in a variety of programming scripts. VBScript, a scripting derivative of the Microsoft Visual Basic family of programming languages, includes Jscript (Microsoft's server-side version of JavaScript) and PerlScript. While support for VBScript and Jscript are included with the default installation of ASP, the PerlScript interpreter must be downloaded from an internet site.

```
<% LANGUAGE="VBSCRIPT" %>
```

If no language directive is included on the page, ASP assumes that we are using VBScript. When the ASP processor on the Web server processes an ASP, the embedded ASP code is executed and removed from the page. Only the HTML content, along with any dynamic information resulting from the execution of the ASP code, is returned to the client. The browser does not interpret or execute any of ASP code, and the ASP source code cannot be viewed on the client side.

2.2. VBScript Statements to Execute in ASP

When you write programming code, you are writing a series of steps that you want the program to execute in sequential order. In VBScript, each of these steps is referred to as a statement. Statements in VBScript are written one statement per line, with no character marking the end of a statement.

For example:

```
Response.Write "<H1>Welcome to My Page</H1>"
```

```
Response.Write "<P>I am delighted that you visited.</P>"
```

is the code for two statements which would be executed sequentially. You can break a long statement into multiple lines with the line continuation character the underscore(_). If you use it, insert a spacebar before it, and make it the last character on the line. For example, the two statements above could be written like this (the ampersand, the concatenation character, is explained below):

```
Response.Write "<H1>Welcome to My Page</H1> & _  
"<P>I am delighted that you visited</P>"
```

VBScript treats the above code as one line, a single statement.

2.2.1. Enclosing Scripting Code within your ASPs

As mentioned above, your ASPs will include a mix of HTML and ASP statements. The server will execute the ASP statements before the page is transmitted to the browser. Each block of ASP statements is set off inside special scripting tags that look like this:

```
<% your ASP code here %>
```

Place as many ASP statements as necessary inside each set of scripting tags. In the example below, the statement Response.Write will write output to the browser. Time and Date are VBScript functions that return the current time and date, as found on the server's clock:

```
<H2>The following is a demonstration of ASP:</H2>  
<UL>  
<%  
  Response.Write "<LI>The time is: "  
  Response.Write Time  
  Response.Write "<LI>The date is: "  
  Response.Write Date  
%>  
</UL>
```

In the example above, the text and HTML tags that are not enclosed between the <%...%> are ordinary HTML, and will be sent to the browser just as they are. The ASP statements are grouped in a sequential block between the <%...> scripting tags.

2.2.2. String in VBScript

In VBScript, string of text are enclosed in "quotation marks. "The string of text can be concatenated together with other strings, or with scripting code that resolves into strings with ampersands (&). For example, the code below would give us the same results as the previous code:

```
<H2>The following is a demonstration of ASP:</H2>
<UL>
<%
    Response.Write "<LI>The time is: " & Time
    Response.Write "<LI>The date is: " & Date
%>
</UL>
```

2.2.3. An Abbreviation for Response.Write

One of the major responsibilities of ASP, in fact of any Web serve-side technology, is to write a response to the browser is so common, VBScript includes a special abbreviation for Response.Write:

```
<%= expression_to_output %>
```

Placing an expression inside the `<%=...%>` tag achieved the same effect as using Response.Write to display it. For example:

```
<H2>The following is a demonstration of ASP:</H2>
<UL>
    <LI>The time is: <%=Time%>
    <LI>The date is: <%=Date%>
</UL>
```

Expression tags can contain anything that will evaluate to a string, but, all in all, everything inside the tag concatenated together can still only amount to one line, or one string expression. These expression code tags cannot span multiple lines.

2.3. Developing ASPs

2.3.1. Executing Your ASPs

As we create ASPs, we will want to run them so that the server will process the ASP

code, execute the ASP statements, and return an HTML response to our browsers. The server will do this only if the page is requested an HTML request. To facilitate this, the demonstration and exercise files you will work with throughout this class have already been assembled on the Windows NT 4.0 Web server we will be using. You will run your files by making HTTP requests to file in a folder set aside for you on the server. You will need access to a server, either on a remote computer, or on your own local machine. For example, navigate in your browser to `http://servername/ProjectASP/` inserting your computer's servername to see a list of links representing your my student files and folders. Clicking on a link will request your file by means of HTTP request. We will notice some file names contain the word temp, while others contain the word done. We will work in the temp files.

2.3.2. Writing Your ASPs

Because ASPs are simple text files, they can be written and edited in any text editor that support simple text files. We will use Windows Notepad. To edit an ASP, drag the file you wish to work on Notepad icon, or open a blank Notepad window and drag your file into it from the explorer Window. With the file open in Notepad, enter any markup or scripting code necessary and save it. To run the ASP, switch to the list of files displayed in the browser and click on the link for the file. If the page is already displayed in the browser, refresh it. At times, caching may require that you return to the browser list of files and request a file with a new click on the link.

An Example: "Hello Near East University!"

```
<%@ Language="VBScript" %>

<Html>
<Head>
<Title>Hello Near East University, Active Server Pages!</Title>
</Head>

<Body Bgcolor="#DBFFBF">

  <Font Size="6" Face="Comic Sans MS">
    Hello Near East University!<Br> The time now is <%=Time%> on
    <%=Date%>
  </Font>

</Body>
</Html>
```

Although ASP generated the time and date dynamically, the page source shows the result rather than the ASP statements that generated it. We see no ASP statements because,

as the statements were executed on the server, they were removed and replaced with the output, and the response was sent to the browser as plain albeit dynamically-generated HTML.

If we make an HTTP request to view a page in the browser, and the result will look something like this:

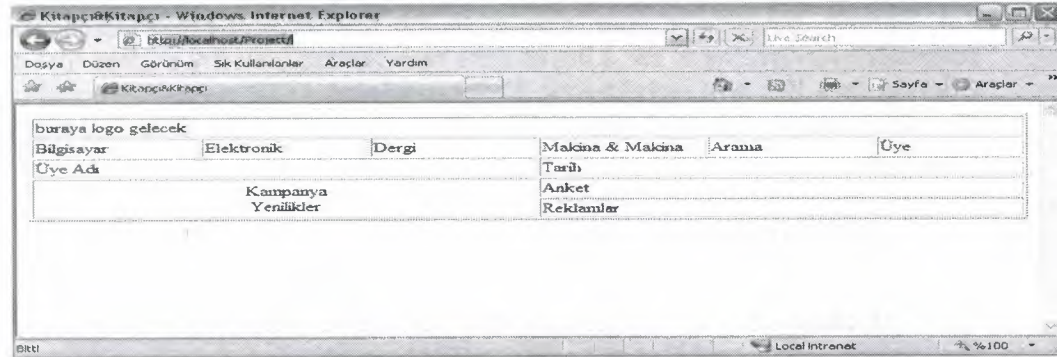


Figure 2.1. A Sample result for requesting a page in the browser.

2.3.2.1. ASP Directives, and Comments

To recap, I have shown ASP statements embedded in an HTML context in three kinds of tags:

Directive - `<%@ directive %>`

We looked at the language directive, `<% Language="VBScript"%>`

Statements To Execute - `<% ASP_statements_to_execute %>`

A scripting tag like this can include multiple statements, each on a separate, complete line.

Expressions - `<% expression %>`

Expression tags are a kind of short-cut for Response.Write. Insert a string, or code that evaluates to a string, and the expression tag will write the content to the browser. An expression tag can only contain one expression, and cannot span multiple lines.

2.3.3. VBScript Comments

Within ASP tags, VBScript can be commented by placing a single quote at the beginning of each commented line:


```

<!--This is an HTML comment ; and will be returned to the browser -->
<H2>The following is a demonstration of ASP:</H2>
<UL>
<%
'This is a VBScript comment, and will removed
'before the response is returned to
'the browser
    Response.Write "<LI>The time is: " & Time
    Response.Write "<LI>The date is: " & Date
%>
</UL>

```

2.4. VBScript Flexibility

VBScript is not a case-sensitive language, and VBScript statements can be written in all lowercase, or in any combination of upper and lowercase. It is common practice to code in VBScript by capitalizing keywords, or "intercapitalizing" keywords formed by combining multiple words. Whichever style you choose, good programming practices call for consistency. It is the best to maintain the same style of capitalization throughout our ASPs.

CHAPTER THREE:

VARIABLES

The variable is one of the most important aspects of a programming language. A variable is a sort of placeholder for any value that's likely to change. For example, on a login page where many different users will log in to a web application. The user's password will be different for each user, so instead of referencing a literal (meaning actual) value, we store the values the user enters as variables, and then check these variables against the information stored in a "login details" database to see if they are correct.

Variables capture the essence of what I mean by dynamic—a single page can be used by hundreds of site visitors. A variable's value can change, so we don't have to worry about the actual values while you're coding—unless you'll be manually specifying the values of a variable, which is rarely the case. The values will usually be retrieved from your site visitors and used immediately, or simply stored separately in a database to be called, changed, or deleted when needed.

Now that we understand what variables are, and how to create them, and what they look like in code. Even though VBScript is case-insensitive, try to stay in the habit of keeping the case of our code consistent. Because consistency is key to all good coding, because it makes the code easier to read and therefore easier to debug when we come up with errors.

For example:

This is easy to read. This is easy to read. This is easy to read. This is hardEr to read.
IS hArDEr to ReAd. tHIS is HaRdEr to rEad.

3.1. Declaring Variables

How to actually create variables in ASP VBScript.

- Create a new dynamic ASP VBScript page,
- Switch to Code in our editor view and we enter the following code after opening <body> tag

```
<%  
Dim myUsername  
myUsername = "Vehbi"  
%>
```

In Code view, the code that declares the variable myUsername. In the first line, we're using the Dim keyword to explicitly declare the variable myUsername (as opposed to declaring it implicitly, which will be discussed shortly). In the second line, we're giving the variable its value.

3.1.1. Letting VBScript declare variables implicitly

We can also give a variable its value and allow VBScript to automatically declare the variable implicitly, as follows:

```
<%  
myUsername = "Vehbi"  
%>
```

Compare the preceding implicit declaration with the following explicit one:

```
<%  
Dim myUsername  
myUsername = "Okay"  
%>
```

3.1.2. Syntax rules for variables

As with all languages, there are some rules that we can't break when declaring variables in VBScript. Here are the ones we need to know when declaring the name of our variable: We will not use spaces, periods, or hyphens. My Variable, My.Variable, and My-Variable are all unacceptable.

Variables must begin with a letter, not a number. 2myVariable is unacceptable. Reserved words can't be used as names for your variables, because, well, they're reserved.

For example, we can't use Dim as the name of your variable because that stands for something in VBScript already.

3.2. Data types: string, integer, and Boolean

Every variable has a specific data type. This is important to know, because each data type has its own function, and you have to know when to use one data type rather than another. This section will cover three of the most important data types: string, integer, and Boolean. The following includes these as well as list some of the other data types we may come across in your programming future:

It's time-saving to let VBScript declare the variable implicitly, but it's better to declare a variable explicitly before giving it its value, because there's an ASP directive, <% Option Explicit %>, that can be added at the top of the page that will force you to declare variables explicitly. Most editors doesn't use this directive, so you won't have any problems if you don't explicitly declare our variables, but it's good to know.

String: used for any alphanumeric character that can be up to about 2 billion characters in length,

Integer: used for an integer ranging from -32,768 to 32,767

Boolean: used to hold the values True (-1) or False (0)

Empty: used for a variable that hasn't been initialized. An empty numerical variable will have a value of 0, and an empty string variable will have a zero-length string ("")

Null: used when a variable has no valid data

Byte: used for an integer in the range of 0 to 255

Currency: used for a number between -922,337,203,685,477.5808 and 922,337,203,685,477.5807

Long: used for an integer ranging from -2,147,483,648 to 2,147,483,647

Single: used for a number ranging from -3.402823E38 to -1.401298E-45 for negative values, and 1.401298E-45 to 3.402823E38 for positive values

Double: used for a number ranging from -1.79769313486232E308 to -4.94065645841247E-324 for negative values, and 4.94065645841247E-324 to 1.79769313486232E308 for positive values

Date (time): used to express the date and/or time; uses a number that represents a date between January 1, 100 and December 31, 9999

Again, the three most important data types, which you'll be looking at in detail here, are string, integer, and Boolean.

A data type is also referred to as a variable type. Its importance is that a variable will always act according to its type. You won't get the same results if you do the same thing with a string that you do with an integer. Strings contain alphanumeric characters, and integers contains numerical digits. We'll need to remember that strings must always be surrounded by quotation marks, but integers never do. Booleans hold the values True or False. The numerical value for True is -1 and the numerical value for False is 0.

For example:

```
<body></body> tags:
```

```
<%
```

```
myString = "Hi, my name is Okay I am "myInteger = 26
```

```
myBoolean = True
```

```
' notice how only strings are surrounded by quotes.
```

```
Response.Write myString & myInteger & " That is " & myBoolean
```

```
%>
```

3.3. Variable concatenation

Concatenation is a fancy word for "joining things together." It doesn't mean adding things together mathematically, it simply means putting things together side by side. In VBScript, the ampersand (&) is the operator that concatenates strings. As an example, I try concatenating two strings into one variable. I Create the following file and save it as

my_name.asp. Entering the following code within the <body> tag while in Code view in a editor.

```
<%  
myFName = "Okay"  
myLName = "Dilek"  
Response.Write myFName & " " & myLName  
%>
```

As opposed to some other programming languages, with ASP VBScript, you don't specify the data type. The VBScript processor on the server picks it up on its own. When setting the value of a variable, if we surround a number with quotes, the ASP processor will see it as a string and not an integer—which is OK if that's what the scenario calls for, such as numbers appearing amidst text. Numbers need to be of an integer data type if we'll be performing mathematical computations with them. After the ASP is code is processed with the preceding code, we've concatenated the values to be displayed as one by using the & operator. We can also recreate a single variable to represent two variables concatenated. I modify the last example as follows, and we should get the same result as before:

```
<%  
myFName = "Okay"  
myLName = "Dilek"  
myFullName = myFName & " " & myLName  
Response.Write myFullName  
%>
```

The difference is in the code itself. By creating a new variable, we've made it simpler for yourself to display the full name—we don't have to concatenate both variables each time you want to display a user's full name. We can concatenate variable and literal values too. For example, the following will also give you the same result as before:

```
<%  
myFName = "Okay"  
Response.Write myFName & " Dilek"  
%>
```

3.4. Variables and Math

VBScript is quite flexible when it comes to treating scalar values as strings or numbers. Also, we can declare a numeric variable, use it as a string, and then use it as part of a math statement:

```
Dim strFirstName, intAge, intNewAge  
strFirstName = "Vehbi Okay"
```



```

intAge = 26
Response.Write strFirstName & " is " & intAge & "years old."
intNewAge = intAge + 1
Response.Write strFristName & "will be" & intAge & "next year."
Response.Write strFristName & "will be" & intNewAge & "next year."

```

In other words, intNewAge is behaving like a number, then a string then a number again. Variables that can do this are known as type variant. But, when you are performing math with scalar variables, at least one of the variables involved must be an actual numeric variable for the math to work. For example:

```

intFirstVal = 5
intSecondVal = 6
intEquation = intFirstVal + intSecondVal
Response.Write intEquation

```

In the next example, intEquation would return 11. But we see that it also return 11 in the code below, even though intSecondVal = "6" indicates a string assignment:

```

intFristVal = 5
intSecondVal = "6"
intEquation = intFristVal + intSecondVal
Response.Write intEquation

```

However if both values were assigned as string as in:

```

intFirstVal = 5
intSecondVal = "6"
intEquation = intFirstVal + intSecondVal
Response.Write intEquation

```

However, if both values were assigned as string, as in:

```

intFirstVal = "5"
intSecondVal = "6"
intEquation = intEquation

```

VBScript would have no clue that the goal here is to perform math. Instead, the two values would be concatenated as strings, and the result 56 would be returned.

3.5. Variable naming conventions

This section is about how to name your variables. We can actually name your variables whatever you like, as long as you don't break any of the syntax rules, such as

using a number as the first character of our variable. The other rules are mentioned in the previous section, "Syntax rules for variables," if we need a refresher.

Although we can name your variables whatever you like, there are some conventions we should follow to make your code more readable and consistent.

For example, if a variable is a string, we might add an `str` prefix, making it `str_myvariable` or even `strmyvariable`—but the key here is adding a prefix so that we know what data type the variable is. Sometimes, coders will add initial prefixes according to their names or companies, so that their code stands out. For example, Frontpage will often add the prefix `FP` to their variables, making them `FP_RedirectPage` or `FP_ConnectionString`, for example. I might add `OE` (my initials) to my variables. Prefixes make your variables more understandable, and can add a personal touch.

3.5.1. Using understandable variable names

This is one of the most important aspects of variables, and for our coding in general. How we name our variables is key to the readability of your code. If we want to create a variable that will store a password, don't call it `p`. We may remember what it means for the next five minutes, but when you (or other developers) come back to update the code in six months, there's no way we'll remember what `p` stands for. Use `pwd` or `user_password` instead. By the way, `password` is a reserved keyword in some languages, such as SQL. The only time you might consider using a variable name like `p` is if we want to deliberately hide the variable name for security reasons—for example, if the variable name becomes exposed to a user of our site through a querystring.

3.6. Organize Variables with Option Explicit

VBScript allow us to declare variables at will throughout your page without using the `Dim` keywords shown above. This makes VBScript quite flexible. However, this flexibility can have a negative consequence: if, for example, a variable name is used but misspelled later in the page, VBScript will not complain although our code will not work properly. To keep this from happening, use the `Option Explicit` command at the top of our ASP file, then use the `Dim` keyword to declare our variables up front, as in:

```
<%Language="VBScript"%>
<%
Option Explicit
Dim strFirstName, strLastName, intAge, intHeight
```

Then, if VBScript runs across an undeclared variable (which could also be a misspelled variable), it will raise an error similar to the following:

Error: "Microsoft VBScript runtime error '800a01f4'

Variable is undefined: 'strFirstName'

/default-temp.asp, line 11”

If we use it, option Explicit must be the first line code in the page, and should be placed immediately after your language directive.

CHAPTER FOUR:

VBScript's HOUSE of BUILT-IN FUNCTIONS

VBScript has a number of built-in functions that are at our disposal. They simplify things for us tremendously and give you the ability to manipulate variables as desired. There are different categories for built-in functions, as follows:

4.1. Built-in functions

4.1.1. Conversion functions:

It is used to convert a value from one data type to another (such as from a string to an integer).

4.1.2. String functions:

They perform actions on variables of the string data type. These functions allow you to do some cool things with strings. Let's first look at the string functions Len() and Trim(). The Len() function returns the number of characters in a string.

For example:

```
<%  
myString = "I am a undergraduate student."  
Response.Write myString  
Response.Write "<br />There are " & Len(myString) &  
" characters in your string variable."  
%>
```

It is important that to keep in mind that even empty spaces count as characters—because they are. The Trim() function gets rid of empty characters that may appear on the left and right sides of a string. This function is often useful when we're allowing web users to send text through a form (such as an e-mail submission form), during which additional empty spaces can sometimes get accidentally added to the text. When this happens, these spaces can cause problems in your database. The Trim() function is used to prevent this from happening.

For example:

```
<%  
myString = " I am a student. "  
Response.Write "." & myString & "."
```

```
Response.Write "<br />." & Trim(myString) & "."
'periods are concatenated with string so we can see the difference.
%>
```

There are also functions to get rid of empty characters on the left or right alone: LTrim() and RTrim(). They're used in the same way. Another function, Left(), allows you to remove a specified number of characters from the left side of a string. It accepts the number as an attribute after the string. For example, Left(string, 5). We can notice the empty spaces between the periods and the text in the line.

For example:

```
<%
myString = "I am a student. I finish my school. I know that I was not
successful in math. Not only me, but I am good at designing."
Response.Write Left(myString, 10)
%>
```

4.1.3. Date/time functions:

They deal with the display of the date and/or time.

4.1.4. Format functions:

They are used for formatting data according to how you want it to be displayed.

4.1.5. Math functions:

They perform mathematical operations.

4.1.6. Array functions:

They are used to manipulate arrays. An array is like a list in which each item is referenced by a name and an index number (its location in the list).

4.1.7. Conversion functions

This type of function converts a variable from one data type to another. For example, if we want to convert a number that's a string to an integer, we use the CInt() function. The variable is placed between the empty parentheses. Commenting your code allows you to provide useful information about your code, and it's important to do it while you're coding, while everything is fresh in our mind.

For example:

```
<body></body> tags:
<%
str_myAge = "26"
int_myAge = CInt(str_myAge)
'convert alphanumeric 26 to integer 26.
Response.Write int_myAge
%>
```

Now, when we print to the screen, we'll still see the result 26. That's because we made no changes to the value, but you changed the data type. The difference is that the variable can now be manipulated in a way that the string can't—that is, mathematical functions can now be performed on it. We can play with it to see for yourself. An other function TypeName(), that will display the data type of the new variable, so we're sure the operation is correct.

```
<%
str_myAge = "26"
int_myAge = CInt(str_myAge)
'convert alphanumeric 26 to integer 26.
Response.Write TypeName(int_myAge)
'see data type after conversion.%>
```


CHAPTER FIVE:

OPERATORS

Operators allow us to manipulate data stored in variables within our code. Operators have four main classes:

5.1. Assignment operators:

Store the value to the right of the operator inside the variable to its left.

5.2. Logical operators:

Join the expression on the left of the operator to the expression on the right in a conditional statement.

5.3. Comparison operators:

Compare two arguments and check whether a specified condition is met.

5.4. Mathematical operators:

Perform a mathematical operation between the values on the left and right of the operator.

5.5. Assignment operators:

There's only one assignment operator, and that's the equal sign (=). It simply stores the value to the right of the operator inside the variable to the left.

For example:

```
<%  
Dim myGreeting  
myGreeting = "Hello"  
%>
```

The preceding assignment operator simply stores the string Hello inside the variable myGreeting.

5.6. Logical operators:

There are three logical operators. Logical operators simply join together or manipulate Boolean (true or false) expressions in conditional statements.

AND Logical combination OR Logical separation NOT Logical negation Let's have a look at each of these operators and see how they join and manipulate expressions.

5.6.1 AND, OR

The AND operator combines two expressions, making a result true only when both expressions are true. The OR operator separates the two expressions, making a result evaluate to true if either expression is true.

For example:

```
<%  
number1 = 4-1  
number2 = 6-2  
If number1 AND number2 = 3 Then  
Response.Write "Correct"  
Else  
Response.Write "False"  
End If  
%>
```

The preceding example checks whether both the variables number1 and number2 equal 3. Since they do not both equal 3, False is printed out. Now if we use the same code and change the AND to OR:

```
<%  
number1 = 4-1  
number2 = 6-2  
If number1 OR number2 = 3 Then  
Response.Write "Correct"  
Else  
Response.Write "False"  
End If  
%>
```

In the preceding example, when AND is replaced with OR, the code checks whether either variable number1 or number2 equals 3 (notice that number1 equals 3 but number2 does not). If either of them do, the word Correct prints to the screen.

5.6.2. NOT

The NOT operator simply negates the statement to the right of the operator. If an expression is True, NOT will change it to False, and if the expression is False, NOT will make it True. The following two code blocks give a good indication of how the NOT operator works.

For example:

```
<%  
number1 = 4-1  
If number1 = 3 Then  
Response.Write "Correct"  
Else  
Response.Write "False"  
End If  
%>
```

Since $4 - 1 = 3$, Correct will print out. Now add the NOT operator, as shown in the following code:

```
<%  
number1 = 4-1  
If NOT(number1 = 3) Then  
Response.Write "Correct"  
Else  
Response.Write "False"  
End If  
%>
```

This time, False is returned—number1 still equals 3, of course, but the NOT operator changes the code to become the opposite of what it would otherwise be.

5.7. Comparison operators

Comparison operators compare two arguments and check whether a specified condition is met. The following table summarizes each comparison operator and its meaning.

- = Equal to
- < Less than
- > Greater than
- <= Less than or equal to
- >= Greater than or equal to
- <> Not equal to

For example:consider the following code:

```
<%  
x = 1  
y = 3  
If x < y Then  
'Execute this code  
End If  
%>
```

This code checks whether the value of the variable x is less than the value of the variable y. If so, any code placed between the lines If x , y Then and End If will execute until the End If statement is reached.

5.8. Mathematical operators

Mathematical operators perform a mathematical operation between the data on the left and the right of the operator. The following table lists each mathematical operator and its purpose.

- ^ Exponentiate
- * Multiply
- / Divide
- \ Integer divide
- MOD Modulus
- + Plus
- Minus
- & Concatenate

Here's a simple example of a mathematical function using the plus (+) operator:

```
<%  
x = 54  
y = 67  
z = x + y  
Response.Write z  
%>
```

In the preceding code, the value of the variable z printed out. The value of z is the equivalent of the sum of x and y. While the divide operator (/) divides the left-hand number by the right-hand number, returning the result plus the remainder, the integer divide operator (\) returns only the quotient, and modulus (MOD) returns only the remainder. For example, 5 divided by 3 is 1 with a remainder of 2. The integer divide operator will return 1, while the modulus operator will return 2.

```
<%  
number1 = 5/3 'Divide  
number2 = 5\3 'Integer Divide  
number3 = 5 MOD 3 ' Modulus  
Response.Write number1 & "<br />" & number2 & "<br />" & number3  
%>
```

Operators are used all the time.

CHAPTER SIX:

CONDITIONAL LOGIC

Like variables, conditional logic is one of the most important aspects of the language, because it allows we to control the flow of our application according to certain criteria—in particular, according to how web users interact with your code. Conditional logic (also known as conditional statements) allows you to use code that says things like “If this happens, then I want you to do this, but if it doesn’t, do thatinstead.” Let’s take a look at how you can control the interaction you receive with conditional logic.

6.1. The If statement

The If statement is the mother of all control. It says, “If this is true, then do this; otherwise, do that.

For example:

```
<%  
myExpectedOutput = "AA"  
If myExpectedOutput = "AA" Then  
Response.Write "That's what I thought."  
End If  
>%
```

In the preceding code snippet, the If statement checks to see whether the myExpectedOutput variable equals your desired value; and when it does, you can specify accordingly what action you want to take place when you receive an expected output. In the previous example, the action taking place is the printing of a related message to the user. It is good to keep in mind that when comparing two strings, every character in each string (including spaces) must match for them to be considered equal. You can also set the conditions around embedded HTML by using multiple code blocks.

For example:

```
<%  
myExpectedOutput = "AA"  
If myExpectedOutput = "AA" Then  
%>  
That's what I thought.  
<%  
End If  
>%
```



```

<%
userRating = 9
If userRating = 4 Then
Response.Write "Are you serious?"
ElseIf UserRating = 6 Then
Response.Write "Come on, you can do better than that."
ElseIf userRating = 9 Then

```

For example:

The If...Then...ElseIf statement allows you to provide two or more possible code continuations. This statement is preferable to using multiple If...Then...Else statements which can result in inefficient and sloppy code. The ElseIf statement is beneficial in that allows you to combine multiple If...Then...Else statements in one.

6.3. The If . . . Then . . . ElseIf statement

In the preceding code, the If statement checks to see whether the myExpectedOutput variable equals your desired value. If it doesn't, the Else statement causes the comment Hey, think again, to be output. Again, you can set these conditions around embedded HTML using multiple code blocks, as in the previous example.

```

<%
myExpectedOutput = "This doesn't deserve AA."
If MyExpectedOutput = "AA" Then
Response.Write "That's what I thought."
Else
Response.Write "Think again."
End If
%>

```

For example:

You can also add the Else keyword to the statement to introduce another output the expression doesn't meet the expected criterion.

6.2. The If . . . Then . . . Else statement

This will return the same results as the previous code. However, if the condition is not true the HTML won't display. This is often used when you want to check if a user is logged in. To do this, you could use an If statement to verify whether a cookie you previously set exists. If it does, you can allow a user access to a page; if it isn't, you can simply redirect the user to another page to prevent access.

```

Response.Write "Close, but I deserve better."
ElseIf userRating = 10 Then
Response.Write "That's good!"
End If
%>

```

The preceding code can be used to allow a web user to enter a rating into a form, and then be returned the appropriate response.

For example:

```

<%
userRating = Request.Form("userRating")
If userRating = 4 Then
Response.Write "Are you serious?"
ElseIf UserRating = 6 Then
Response.Write "C'mon... you can do better than that."
ElseIf userRating = 9 Then
Response.Write "Close, but I deserve better."
ElseIf userRating = 10 Then
Response.Write "That's good!"
End If
%>

```

You can use an Else keyword for your last check in an ElseIf statement. It gets processed only when none of the other conditions match. Be aware that if you use it, you must put it at the end of the statement, and never before an ElseIf keyword. The following example shows the correct use of the Else statement in such a case:

For example:

```

<%
userRating = Request.Form("userRating")
If userRating = 4 Then
Response.Write "Are you serious?"
ElseIf UserRating = 6 Then
Response.Write "Come on you can do better than that."
ElseIf userRating = 9 Then
Response.Write "Close, but I deserve better."
Else
Response.Write "I'm looking for a 10 rating."
End If
%>

```

6.4. Loops in ASP

Loops are so important for programming. Even they are even first importance programming. We use them to control of a program flow, they supply to do a process again and again when the condition is true. Of course, they do not statements until the infinite. it goes to the infinity we called this kinds of loops as "endless loop"

6.4.1. For-Next Loop

If want a program to do a job more than once, we determine the loops issues by number(integer) variable.

For example:

```
<%  
for counter = 1 to 4 step 1  
statements...  
Next  
%>
```

Here 3 times the statements will be executed, and next supply exit of out the loop. The counter variable increments by the integer after step, so the require statements will be executed. The important case is to keep in mind that the next is used with a variable after i

6.4.2. While-wend Loop

Sometimes the loop we want to use in our programming cannot be static, namely, value from the upper side of the program or a value received from a user as that we may use it for a loop. In this case, we may want a loop that we control it by the values as described above. We do that using while loop, because of when we use it, we must increment the control value for the loop.

For example:

```
While counter<=5  
Response.Write days(counter)  
counter = counter + 1  
Wend
```

Here we see that the while loop ended by Wend. The loop will run while the control value less than 5 and equals to 5. The increment statement is different from as used in the For loop. However, for computer it means that take the counter value simultaneously, and add 1 to it and write the value as the new value of the counter.

6.4.3. Do-Loop

Sometimes, it may be wanted a case in a loop that becomes in it. In this case, we use do-loop. The format of the loop is almost same with the others loops. Simply, if the control value is true than continue executing the loop.

For example:

```
Do while Control_Value  
Statements  
...  
...  
[Exit do]  
...  
...  
Loop
```

When the loop is in execution, if an other Control_Value is suitable the loop could be ended by Exit_do. It is hard to see a difference between Do-Loop and While-Wend, but a small one is present that in Do-While loop the Control_value is being checked when entering the lopp, and the other one is the Control_Value is being checked before the entering the loop.

CHAPTER SEVEN:

COOKIES

Whether you're an experienced web developer or simply an Internet enthusiast, you've no doubt heard about cookies. Cookies aren't our favorite snack food in this context—rather, they're simply a term used to describe another great feature, which is not only native to ASP, but usable with almost any web scripting language. (Netscape created the original cookie.)

Cookies are simple text files that you can create and store on the client machine. They allow you to store simple data such as usernames, dates, etc. Values of cookies can be created by the web application, or they can accept parameters submitted by the user. However, they must be used with care. How do you actually make use of them? Cookies are comparable to the forms and URL parameters discussed earlier—they allow you to maintain data across pages. When cookies are created, they're automatically stored in a specific folder in the user's hard drive, which can't be changed—in Windows 2000, you can find this folder in C:/Documents and Settings/Administrator/Cookies. You'll probably find dozens already there. Each cookie contains specific data that was stored by various web applications while you were using the Net.

7.1. Response.Cookies and Request.Cookies

The ASP statement `Response.Cookies` is all you need to create a cookie. You just need to name the cookie and give it a value—for example, `Response.Cookies("MyCookie") = "Hello ASP"`. In this example, the cookie is called `MyCookie` and it has a value of `Hello World`. Once the code executes, the cookie will be created. The preceding example uses a specific value, but remember that you can also store information from the user as a value. For example, if a user submits a form with a text field called `username`, you can store the value of that text field as the value of a cookie, like so:

```
Response.Cookies("MyCookie") = Request.Form("username")
```

This way, you can store the form data on the user's computer and retrieve it any time (as long as it hasn't been deleted). At this point, you're probably asking, "Now that I know how to create a cookie, how do I retrieve it?" `Request.Cookies` will retrieve any cookie you created, identifying it by name. For example, if you wanted to retrieve the cookie named `MyCookie` that was mentioned earlier, you would do it like so:

```
Request.Cookies("MyCookie")
```

How easy is that? Then, if you want to output it onscreen, you use the following code:

Response.Write Request.Cookies("MyCookie").

For example:

```
<%  
Response.Cookies("username") = "okay"  
%>  
<%=Request.Cookies("username") %>
```

We should see the following results when you load the page in your browser:

okay

You created a cookie named username and gave it a value of okay—then you displayed the cookie. Now let's create some cookies based on what a user might submit in a form. You'll use another dummy login form.

```
<form name='form1' method='Post' action='login_create_cookies.asp'>  
username: <input type="text" name="username"><br>  
password: <input type="text" name="password"><br>  
<input type="submit" name="submit" value="submit">  
</form>  
<!-- login_create_cookies.asp -->  
<%  
Response.Cookies("userid") = Request.Form("username")  
Response.Cookies("pwd") = Request.Form("password")  
%>  
userid cookie: <b><%=Request.Cookies("userid") %></b><br>  
pwd cookie: <b><%=Request.Cookies("pwd") %></b>
```

After submitting the data, when you view the login_create_cookies.asp page, you should see the values you entered. You've seen something similar before, but this time you didn't display the values of the text entered in the text fields by simply using Request.Form—rather, you stuffed those values into two cookies and then retrieved the values from each. Now that these cookies have been created, you can retrieve the data from any page without having to prompt the user again (until the cookies are deleted). This comes in handy if you wanted to, say, display the username across several pages. Many web developers use a prefix of ck when naming cookies.

For example, ckUserID, ckPassword, etc.

7.2. Cookie expiration

If you don't specify a cookie's expiration date, it will delete itself from the user's system once the user shuts down his or her browser. This is fine if you only want the cookie

information for a single session, but you may want the information to be maintained for days, weeks, or even years. For example, let's say you want a cookie named `userid` to expire after seven days. You would do this with the following code:

```
Response.Cookies("userid").Expires = Date + 7
```

Likewise, you can add expiration dates to your first cookie example in the `my_first_cookie.asp` page like so:

```
<%  
Response.Cookies("username") = "okay"  
Response.Cookies("username").Expires = Date + 7  
%>  
<%=Request.Cookies("username") %>
```

You can increase the number of days as you require.

7.3. Updating cookies

You can't create two cookies on the same computer with the same name, but you can update cookies simply by rewriting their values.

For example:

```
<%  
Response.Cookies("ckRememberMe") = True  
Response.Cookies("ckRememberMe") = False  
%>  
<%=Request.Cookies("ckRememberMe") %>
```

If you view the page, you should see the following results:

False

In the preceding code, you created a cookie called `ckRememberMe`, gave it a value of `True`, and then rewrote the value to `False`. Keep in mind that even when you set expiration dates for your cookies, users are still likely to manually delete them from their computers every so often.

7.4. Deleting cookies

You can delete cookies simply by giving them an empty string value.

For example:

```

<!-- delete_cookie.asp -->
<%
Response.Cookies("ckRememberMe") = True
Response.Cookies("ckRememberMe") = ""
%>
<%=Request.Cookies("ckRememberMe") %>

```

Nothing should display on the screen, since the cookie doesn't exist. ASP is cool in that it doesn't throw an error when trying to retrieve a cookie that doesn't exist. Cookies are cool and fun. They have many uses and can help web developers in many various situations—don't be afraid to use cookies if you see a good use for them in your web applications. You may also want to explain to your users how you use cookies to track their interaction on your site in a privacy notice or disclaimer.

7.5. A word of warning about the use of cookies

When using cookies, you should be careful not to use them to store any sensitive information, such as usernames and passwords, or—heaven forbid—credit card details! Cookies aren't dangerous per se—but it's the potential of how they can be used that makes them a security risk. Since they're merely unencrypted text stored in a file on a user's computer, it's a possibility that some unscrupulous web users could access them (as long as they have the cookies' names), and use them for untoward purposes, which might include sharing the cookie information with companies that users don't want to be contacted by. However, as long as you're wise in your use of cookies, and heed these warnings, you should be fine—besides, cookies are beneficial to any web developer. They enable persistent logins and personalization. They really give you an edge when you know when and how to use them. What's more, they don't take up server space like sessions, which are stored on the server.

CHAPTER EIGHT:

DATABASES

Includes; Creating tables Understanding relational databases and referential integrity Creating relationships Using views and queries Understanding fundamental SQL and using basic SQL commands Setting up DSNs to connect to an Access or SQL Server database Connecting to a database from Frontpage Building recordsets in Frontpage using the Simple and the Advanced Recordset builders Using the Command dialog to insert, update, and delete records in a database.

8.1. Creating a database

First, we need to establish the difference between creating a database and creating the database content. When you create your database, you're creating the container in which your data can reside. No data will be created at this point. Only after you've created the container can you proceed to creating the content inside it.

8.1.1. Inside Microsoft Access

Microsoft Access offers up myriad wizards and templates for you to choose from to get your job done. Creating a database is as simple as clicking the mouse a few times (and pressing a couple of keys too, unless you want to create a database called db1.mdb, which is Access's default name for the first database it's asked to create). When you open Access, you're presented with a blank screen and a sidebar titled Getting Started. This sidebar contains a few useful elements, including a search facility to search the Help system both on and offline, a list of recently opened Access databases, and a link to create a new database. To create a new database, click the Create a new file link in the sidebar. Alternatively, you can select File ► New, or click the New icon on the icon bar below the menu. The sidebar turns into the New File panel, where you can now make your next choice. The New File panel offers you a few options to choose from, but you're only concerned with creating a new blank database at this point, so click the first option, Blank database. Then select a location to save the database on your computer, such as a database folder in the root of the C: drive, C:\databases\, and give it a name. The database name used for this book is mysampledatabase.mdb—Of course! it's a sample. Before moving on to adding the content of your database, however, let's first cover the creation.

8.2. Database design

There's far more to good database design than just following a good naming convention.

8.2.1. Object naming conventions

SQL Server (and the SQL language itself) has a number of reserved keywords that must be avoided at all costs when creating names for your database objects and their properties. For a complete list, launch Books Online, go to the Index tab, enter keywords reserved for SQL Server, and click Display. Using a good naming convention can help you to understand what's going on when you come to use your database later on, or when you return to a project you wrote months ago.

For example, let's say you have a few database tables, named as follows:

```
tblCustomers  
tblCustomer_Orders  
tblCustomer_Order_Details  
tblProducts
```

It doesn't take a genius to figure out that `tblCustomer_Orders` would be a good place to look if you wanted to view some customer orders. Naming things logically makes it easy to quickly assess what an object contains and what it should be used for. This is especially useful when accessing these objects from external programs such as Frontpage. In the preceding example, each table name is prefixed with `tbl`, which is short for "table." You do this so that you know it's a table and not a view, stored procedure, or any other database object. For views, I use a prefix of `view`, and for stored procedures, I use a prefix of `sp`. Confirming the password to finish creating the new user Table column names are another area that can cause serious problems later on if you aren't careful.

For example, say you have a table that contains customer orders and you want to include a column to store the date that an order was placed (i.e., the date it was added to the database). You shouldn't simply use `Date` as your column name, since `Date` is a reserved word in SQL Server. You can get around this by using `DateAdded` instead. This avoids any naming conflicts, and the column name still makes sense. Avoid the use of spaces in object and property names. The examples shown here use the underscore character in place of spaces. You can use the capitalization method instead, for which the start of each new word is capitalized (instead of each word being separated with underscores). For example, using the capitalization method, the `tblCustomer_Orders` object would look like `tblCustomerOrders`.

8.2.2. Creating tables in Access

Open the empty `mysampledatabase.mdb` database that you just created. Access 2003 had several security enhancements built in, so you'll be very likely to see the Security Warning when you open your database—just click Open to continue. Click Open to get past the security warning when opening a database in Access 2003. With the database open, select Tables in the left column of the database window, click New, and select Design View from the pop-up dialog. (You can also double-click the Create Table in Design View icon

in the Tables window). You'll be presented with the table Design view. Creating a table in Access 2003. I've previously said that you can think of a record as rows of fields, running horizontally. There are three values that you can assign to each field:

Field Name: the name used by Access to identify a given field within a table.

Data Type: a general class into which the data falls, (e.g., Text, Number, or Currency).

Description: some text that you can use to remind yourself of what a given field is intended to hold. What you enter here has no wider use within the database itself, and is entirely optional.

Click inside the Data Type box for the topmost field and click the down arrow—you'll see a list of the general data types that Access recognizes. If you select a general data type, such as Number, for example, and then move down to the Field Properties dialog and select Format (for most of the general types) or Field Size (in the case of Number data types), you'll see another down arrow appear. Click this and you'll see all of the subtypes you can assign to your data. For instance, the subtypes of the Number data type are as follows:

- Byte
- Integer
- Long Integer
- Single
- Double
- Replication ID
- Decimal

If you've done any programming before, you may well be familiar with the concept of using different data types to optimize how your data is stored and handled within your code. If not, don't worry, you can play around with the different data types and field lengths at your leisure—most of them are fairly easy to grasp—and you won't need a thorough grounding in this for the simple table you're about to create. Let's now take a look at creating the previously mentioned tblCustomers table in Access. To create the primary key, type CustomerID in the first Field Name box in Design view. Give it a data type of Autonumber. Access will now ensure that the value created here for each record will be unique. The field size will be set as Long Integer by default, so leave it like that. Now click the gray square to the left of the Field Name column to highlight the entire row. Right-click, and you'll see that the first entry in the pop-up menu that appears is Primary Key. Select this, and a small key icon will appear in the gray square to the left of the field name. This signifies that Access will now treat this field as the primary key for this table. To maintain integrity, Access allows you to assign this property to only one field in a table at any one time, although you can select multiple rows to assign the primary key to a combination of columns—but you don't need to do this here. Move down to the next field and name it CustomerName. Give it a data type of Text. You may wish to increase the Field Size (the number of individual characters) to 100 or so, from the default value of 50. In order to avoid wasting space, always aim to make a field no bigger than the maximum amount of data you expect it to have to hold. However, you should bear in mind that if you try to insert more data than a column is designed to hold, you'll get an error and the insert will fail. Name the next field PostalAddress, and give it a Field Type of Text and a size of

250. Be aware that text fields can hold a maximum of 255 characters anyway, so if you feel this field should be able to hold more characters than this, you'll need to use a Memo data type. For this example, however, use a Text type to keep things simple. Name the next field Postcode. Give it a Field Type of Text. A Field Size of 10 should be sufficient. Name the next field Country, and give it a Field Type of Text and a Field Size of 50. You save your newly created table by clicking the Save icon, or by selecting the File option from the menu, choosing Save As, and entering tblCustomers as your new table's name. You can now view your table in Datasheet view by selecting View ► Datasheet View. If you had simply tried viewing your table in Datasheet view straightaway, Access would have prompted you to save you table first. Either way, you've now created your new tblCustomers table.

8.2.3. Creating tables in SQL Server

In the left pane under Databases, select the empty database called mysampledatabase that you created earlier. In the right-hand pane, right-click the Tables icon and select New Table. The New Table dialog box will open. When you enter the field name in the Field Name column, and then move into the Data Type column and select the down arrow that will appear, you'll then see a list of all the data types that SQL Server supports. Unlike Access, SQL Server's data types don't break down into more specific subtypes; also, SQL Server has a great many more actual types than Access does. This reflects the fact that SQL Server is optimized for fast data retrieval and efficiency of storage. If you refer back to your example table, tblCustomers, you'll recall that it has a primary key field named CustomerID. The New Table dialog box, in which the CustomerID column is chosen as an Identity column.

Give this field a name of CustomerID and a data type of int. Then set the Identity box in the Columns tab to Yes. Identity is the name for SQL Server's built-in primary key data type. This defaults to an identity seed (the number from which the identity field will begin counting when the first record is created) and identity increment (the rate at which the value is increased with each new record) of 1 in each case. This means that the first record will be given a primary key of 1, and each successive record will have a primary key one greater. This is fine for your purposes, but SQL

Server does provide the ability to customize many elements of how your data should be stored. The int data type occupies 4 bytes by default. You'll find that for this reason you can't change the value in the Length column. If this were a text field, such as a char (character) type, you could set the number characters it occupies and, thus, what kind of load on the total storage space the field would exert for each record. As you saw when carrying out this process in Access, the decision about how big a field should be must be taken with care. An int data type can store up to around 2 billion values, which may seem like overkill (even if this were Amazon.com that you were designing for)—but remember that any value stored here can never be used again in the same table, and customers will come and go. In the lifetime of your database, the use of an int data type for a field of this kind is appropriate—especially since the next smallest size available is tinyint, which can store only up to 256 different values. Now you can create the other fields: CustomerName, PostalAddress, Postcode, and Country. For normal fields like this, you can permit the value

to be a null, which is literally an unknown value. It's not an empty string—that is, a character string with no content (""), which contains nothing but is still something—it's just an empty something. The use of a null explicitly stores the fact that no value exists. You can also specify a default value, which is a value that will always be entered in a field if you should no other value be provided in its place.

Create CustomerName as a char data type of length 100. You may think this is too much, but it depends on whether you would expect to be receiving commercial customers or residential. A char data type is of fixed length, which is to say that its field will always hold 100 characters, even if its customer name were simply John Smith—the rest of the field would be filled up with spaces. This may sound very inefficient, and you could have made it a varchar data type instead, meaning that each field for a given record would occupy only as many characters as are needed to store its value. However, the use of fixed-length fields allows SQL Server to search for and retrieve values much faster, since it doesn't have to check for the end of each field all the time. Make sure you uncheck the Allow Nulls column, and don't bother entering any default values, since neither of these options is appropriate for this field.

Now create a field named PostalAddress with a data type of char, and give it a length of 250 characters. Make both Postcode and Country fields of type char, with lengths of 10 and 50 characters, respectively. If this were a real example, you might want to assign Country with a default value—if you expected the majority of your customers to come from your home country, for instance. A completed SQL Server table design. Save this table design by clicking the Save icon in the top-left of the main window; name it tblCustomers. Close the dialog and click the Tables icon in the right-hand pane. Your new tblCustomers table will be listed in the resulting display.

8.3. Relational databases and referential integrity

Relational databases allow you to store lots of information in a structured and organized way, removing the need to store multiple copies of the same data. The use of primary and foreign keys is crucial to how you achieve this. For example, in your e-commerce site, suppose a customer places an order for five items. You would want to store the customer's delivery details, an overview of their order, and the details of their order (the individual items they bought).

You could store all this information in one table and have the customer's name and address entered with each record of the order, but that's a lot of unnecessary repetition. Fortunately, thanks to relational databases, you can split the information over several tables. The tblCustomers table you've just created would have just one record per customer and would store the contact and delivery details. The tblCustomer_Orders table would have one record per order and might detail the order total, the date and time the order was placed, and whether the order has been fulfilled or not. The tblCustomer_Order_Details table would hold a record of each individual item ordered (the details of the order). In this case, you would set up two relationships to show how the entries relate to each other: one would be between tblCustomers and tblCustomer_Orders, and the other would be between tblCustomer_Orders and tblCustomer_Order_Details. You do this using the value CustomerID, the primary key from the tblCustomer stable.

The `tblCustomer_Orders` and `tblCustomer_Order_Details` tables will each contain a field that stores the `CustomerID` values of the customers who placed these orders—the same `CustomerID` values you stored in `tblCustomers`. This is known as a foreign key. A foreign key is simply a primary key from one table recorded as a field within another table. Like a leash connects a dog to its master, the foreign key in the `tblCustomer_Orders` and `tblCustomer_Order_Details` tables ties the order records to the customers to whom they belong— from this, you can retrieve the details of each customer, along with their orders. So, in `tblCustomer_Orders`, you would have a column called `CustomerID` that holds the foreign key value of `CustomerID` for the relevant record in `tblCustomers`. You would also have a column called `OrderID` in the `tblCustomer_Order_Details` table that holds the foreign key value of `OrderID` for the relevant record in `tblCustomerOrders`.

Using the example of a customer buying five items from your store, there would be one record created in `tblCustomers`, one record created in `tblCustomer_Orders`, and five records created in `tblCustomer_Order_Details`. The one `tblCustomer_Orders` record would contain a primary key of its own, called `OrderID`, and a foreign key from the `tblCustomers` table— `CustomerID`. The five `tblCustomer_Order_Details` records would each contain their own primary keys called `OrderDetailsID`, for example, and also a single `OrderID` foreign key that links them to the order that they were part of. Let's see how this is done. You might want to create some dummy tables called `tblCustomer_Orders` and `tblCustomer_Order_Details`, so that you can try this next example out for yourself. You can give each one dummy fields to reflect how you think they would be structured, if you like. However, all you really need is a primary key for both `OrderID` and `OrderDetailsID`, a field called `CustomerID` in the `tblCustomer_Orders` table, and a field called `OrderID` in the `tblCustomer_Order_Details` table. These latter two fields will be the foreign key in each case.

8.3.1. Creating a relationship in Access

To create a relationship in Access, click the relationships button on the toolbar. These foreign key columns aren't Identity or Autonumber columns (which contain unique values only) because you might have to store several records with the same `CustomerID` values in the `tblCustomer_Orders` table. The Relationships toolbar button in Access 2003. You'll now be looking at a large, blank gray canvas onto which you need to add the tables you want to relate to each other. Let's follow on from the examples shown before and relate the `tblCustomers` table to a `tblCustomer_Orders` table, which would be the first of two relationships you would set up if you were to follow the examples through to their conclusion. Right-click and select Show Table from the context menu. From the dialog, select `tblCustomers` and click Add. You should also add `tblCustomer_Orders` and `tblCustomer_Order_Details`. In the list of columns in the `tblCustomers` object, left-click and drag from the `CustomerID` field to the `CustomerID` field in the `tblCustomer_Orders` object, and then release the mouse button. As you start to drag between the objects, the cursor changes to a small horizontal bar. When you've finished dragging and have dropped (released the click you started on `tblCustomers`), the Edit Relationships dialog box opens, which allows you to specify the details of the relationship. The Edit Relationships dialog box You want to set up a relationship between `tblCustomers` and `tblCustomer_Orders`. The

columns that have the related data in them will both be called CustomerID in this case. They don't have to be named the same; they just have to contain the same data and be stored using the same data type. But you should use the same name for simplicity's sake—simple is good! As shown previously, you also want to specify Cascade Delete Related Records in the referential integrity area so that if a record is deleted from my tblCustomers table, all the related records from my tblCustomer_Orders table will be deleted as well. Click Create to create the relationship, and a black line will be drawn between the two tables to signify the relationship. Notice that there's a 1 above the end of the line that ends at tblCustomers, and an infinity symbol (which looks like a sideways 8) above the other end. These visually signify the one-to-many relationship you've created between these two tables. To complete the three-table relationship, perform the same process between the tblCustomer_Orders and tblCustomer_Order_Details tables, using the OrderID field in both tables. The Relationships window showing three related tables.

8.4. SQL Server views and Access queries

SQL Server views and Access queries are the ideal way to bring together the data you need to retrieve from the tables in your database. Using views or queries in your database allows you to store some of the SQL work within your database rather than having to pass an entire statement to the database from your web page. This means that to change the data displayed on a page, you might only need to modify the code in your database, rather than editing the SQL in your web page. The need to build very complex SQL queries can rear its head quite quickly in large projects, which is why I almost always use views or queries to gather the data I want, and then use a simple select statement to grab all of those records in my web page. I may even pass filtering parameters into the view to narrow down my recordset even further. The beauty of doing it this way is that the complex SQL statement stays in the database, and the runtime performance of the queries can be increased so that your web pages will load faster.

Views and queries become increasingly useful when you need to gather data from more and more tables. Using a view or query to select records from a single table may offer only slight benefits, and only if the table is large, if it has many columns, and if you're selecting most of them. If it's a small table, then the benefits are often much smaller.

8.5. Fundamental SQL

To give you a full understanding of this creation process and the final SQL code that it produces, a quick primer in some fundamental SQL is in order.

The following examples assume that the owner of the database objects in SQL Server is dbo. Also, you'll be addressing the same imaginary table object, tblLogins, in each case. The SQL code for each example is shown for both Access and SQL Server.

8.5.1. Selecting all records from a table

This is probably the simplest SQL statement there is.

Access:

```
SELECT * FROM tblLogins
```

SQL Server:

```
SELECT * FROM dbo.tblLogins
```

This statement says "Select all the records contained in tblLogins." Here, * means all columns. This is not to be confused with the mathematical *, which looks the same but performs a very different role (i.e., multiplication).

8.5.2. Selecting all records that meet one criterion

This is almost as easy as the first example, but there are a couple of crucial rules to bear in mind. You're going to use a similar statement to the one preceding, but you only want to return those records that contain a specific value in the specified column. You use the keyword WHERE to specify this.

Access:

```
SELECT * FROM tblLogins WHERE Username = 'value'
```

SQL Server:

```
SELECT * FROM dbo.tblLogins WHERE Username = 'value'
```

This statement says "Select all the records contained in tblLogins where the value contained in the Username column matches the value specified (in place of value)." What you may need to watch out for in SQL statements is the data type of the column that stores the value you need to match. In the previous example, the column Username has a text-based data type, which means that the value you're going to compare with the value in that column needs to be enclosed in single quotes. If it had been a numeric column, single quotes do not need to be used. Access doesn't need this extra owner information, nor would it understand it if you used it. The reason is that Access doesn't use the ownership metaphor or permissions principles that SQL Server does. It's intended as a single-user database manager. SQL Server ensures that only the people with the relevant permissions can perform actions on the database objects; in Access, it's a free-for-all! To illustrate this point, the following SQL code shows you the correct way to use a numeric value in the value area of this statement.

Access:

```
SELECT * FROM tblLogins WHERE LoginID = value
```

SQL Server:

```
SELECT * FROM dbo.tblLogins WHERE LoginID = value
```

This statement says "Select all the records contained in tblLogins where the value contained in the LoginID column matches the value specified (in place of value)." The LoginID in your database is a numeric value; therefore, the comparison value needs to be numeric too. By removing the single quotes from around the value, the SQL Statement becomes valid. If you left the single quotes around the value and tried to execute the

statement, you would get a data type mismatch error because the SQL code sees that you're trying to use a non-numeric value in a comparison statement with the value contained in a numeric column. Bearing these facts in mind, let's quickly step through a couple of increasingly complex, yet still very straightforward, examples.

8.5.3. Selecting all records that meet several criteria (using AND)

The AND keyword allows you to create SQL statements that specify that more than one criteria must be met in order to return any results.

Access:

```
SELECT * FROM tblLogins WHERE Username = 'value'  
AND Password = 'anothervalue'
```

SQL Server:

```
SELECT * FROM dbo.tblLogins WHERE Username = 'value'  
AND Password = 'anothervalue'
```

This statement says "Select all the records contained in tblLogins where the value contained in the Username column matches the first value specified (in place of value), and the value contained in the Password column matches the second value specified (in place of anothervalue)."

8.5.4. Selecting records that meet one or more of several criteria (using OR)

The OR keyword allows you to create SQL statements that specify that one or more criteria of either of the specified criteria must be met in order to return any results.

Access:

```
SELECT * FROM tblLogins WHERE Username = 'value'  
OR Password = 'anothervalue'
```

SQL Server:

```
SELECT * FROM dbo.tblLogins WHERE Username = 'value'  
OR Password = 'anothervalue'
```

This statement says "Select all columns for all the records contained in tblLogins where the value contained in the Username column matches the value specified (in place of value), or the value contained in the Password column matches the value specified (in place of anothervalue)." If you were to build a slightly more complex SQL statement that utilizes the OR keyword and the AND keyword, you would be well advised to use parentheses to encapsulate the OR criteria. If you don't do this, you may get erroneous results. The following Access-based examples will explain this. To use this example in SQL Server, just add dbo. in front of the table name.

```
SELECT * FROM tblLogins
```


WHERE (Username = 'value' OR Password = 'anothervalue')
AND AccessLevel = 'yetanothervalue'

This would correctly return all columns for all records from the tblLogins table that meet the criteria of having either a matching Username value or a matching Password value, and having a matching AccessLevel value. Compare it with the following statement, which could return incorrect results:

```
SELECT * FROM tblLogins  
WHERE Username = 'value'  
OR Password = 'anothervalue' AND AccessLevel = 'yetanothervalue'
```

Here, the SQL might be saying what the first example said, or it might be interpreted as saying "Return all records from the tblLogins table that meet the criteria of having a matching Username value, and also having a matching AccessLevel value or a matching Password value." It's ambiguous. If you say the whole statement out loud with a big emphasis on the OR keyword, you'll understand the point I'm making here. Basically, the more complex your SQL statements become, the more careful you need to be in creating them.

8.6. Useful SQL keywords

There are far too many SQL keywords for me to cover in this section of the book. However, a quick glance at some of the more commonly used ones might help you to understand how to achieve the results you're after. To see all the available SQL keywords.

8.6.1. COUNT

If you want to count how many login records you've stored in your tblLogins table, for example, you could use the COUNT keyword in the following way.

Access:

```
SELECT COUNT(LoginIncluded) AS TotalLogins FROM tblLogins
```

SQL Server:

```
SELECT COUNT(LoginIncluded) AS TotalLogins FROM dbo.tblLogins
```

This actually illustrates two very useful SQL keywords in one: COUNT and AS. The COUNT(LoginIncluded) section will count how many rows there are in the tblLogins table, while the AS TotalLogins section will return the COUNT value to an alias column called TotalLogins. TotalLogins doesn't exist as a column in your database, but you're declaring it as an alias by using the AS keyword. The preceding SQL statement might be more useful if you used a selection criteria to count how many rows there are in your tblLogins table that are included—this would be signified by a True value in the LoginIncluded column, as follows:

Access:

```
SELECT COUNT(LoginIncluded) AS TotalLogins  
FROM tblLogins WHERE LoginIncluded = -1
```

SQL Server:

```
SELECT COUNT(LoginIncluded) AS TotalLogins  
FROM dbo.tblLogins WHERE LoginIncluded = 1
```

8.6.2. SUM

If you want to add up the values of several rows (for example, to find the total value of a customer's orders) to create a grand total, you could use the SUM keyword in your SQL statement, as in the following example.

Access:

```
SELECT SUM(TotalCost) AS GrandTotal  
FROM tblCustomer_Order_Details WHERE OrderID = 2
```

SQL Server:

```
SELECT SUM(TotalCost) AS GrandTotal  
FROM dbo.tblCustomer_Order_Details WHERE OrderID = 2
```

The SUM(TotalCost) section will add together all the rows in the tblLogins table that match the WHERE criteria and return a single row contained in a column called GrandTotal, using the alias keyword AS. In this example, the rows that match must have an OrderID of 2.

8.6.3. TOP

The TOP keyword is very useful if you want to specify that only a certain number of rows should be returned to your recordset. Obtaining exactly the amount of data you require is good for performance reasons. For example, if you have a table that contains hundreds of thousands of rows of DVD and CD products, and you know that over 10,000 of those rows will match your specific WHERE criteria, but you only want to view a sample of these records, then the TOP keyword can arrange that for you. All you need to decide is how many records you want to retrieve. Let's say that you want 100. Following is the code that you would use.

Access:

```
SELECT TOP 100 * FROM tblProducts WHERE ProductCategory = 'DVD'
```

SQL Server:

```
SELECT TOP 100 * FROM dbo.tblProducts WHERE ProductCategory = 'DVD'
```

The process this statement goes through is fairly straightforward. It filters the tblProducts table with the criteria you specified (ProductCategory must equal DVD); then, from that

result set, it selects the first (TOP) 100 records. Notice the * immediately after the 100 figure in the statement. This signifies which columns of the table you want returned. You could have specified one or more column names here to narrow down even further the data that's returned—but in this case, all of the columns were desired, so * was used. This is all well and good, but what if you don't want the top 100 records? Say you wanted to view the last 100 DVDs added into your product database between two dates. To do that, you could either specify the dates in your SQL statement using the BETWEEN keyword, or you could use the ORDER BY keyword. You'll look at both of these in turn.

8.6.4. BETWEEN

The BETWEEN statement has a lot of uses, and just a couple of simple stipulations on how it's used. Let's look at an example of BETWEEN in use, and then break it down to see what's going on.

Access:

```
SELECT TOP 100 * FROM tblProducts  
WHERE ProductCategory = 'DVD'  
AND DateAdded BETWEEN #01 August 2005# AND #31 August 2005#
```

SQL Server:

```
SELECT TOP 100 * FROM dbo.tblProducts  
WHERE ProductCategory = 'DVD'  
AND DateAdded BETWEEN '01 August 2005' AND '31 August 2005'
```

In the preceding code, you want to select all columns of data for the TOP 100 records that are returned that match the criteria specified—in this case, the ProductCategory is DVD, and the DateAdded column must contain a date that's between the dates of the 1st and the 31st of August 2005, inclusive. There are a couple of things that you need to keep an eye out for when using BETWEEN:

It must specify inclusive values.

It must be compared to the same data type.

In the preceding example, DateAdded has a Date data type, so the first and second values must be Date values also.

8.6.5. ORDER BY

ORDER BY is a powerful keyword that you can use to order the way your records are returned to you. For example, you might want them to be in the order they were entered into the database, using the primary key value, which is usually a sequential number for each row. You might also want to specify multiple columns by which to order your data. The way in which data is returned to you is dependent on the order in which you specify the columns in the ORDER BY statement. A couple of examples will help to illustrate this powerful keyword.

Access:


```

SELECT TOP 100 * FROM tblProducts
WHERE ProductCategory = 'DVD'
AND DateAdded BETWEEN #01 August 2005# AND #31 August 2005#
ORDER BY ProductCategory, ProductPrice

```

SQL Server:

```

SELECT TOP 100 * FROM dbo.tblProducts
WHERE ProductCategory = 'DVD'
AND DateAdded BETWEEN '01 August 2005' AND '31 August 2005'
ORDER BY ProductCategory, ProductPrice

```

Using the same example as the BETWEEN example, the ORDER BY keyword has been added so that all the results that meet the criteria specified will be ordered by the ProductCategory; and within that, by their ProductPrice. This means that the DVDs that match the SQL statements entire criteria will be returned, ordered such that the cheapest DVD from the first category will be listed first, followed by increasingly expensive DVDs from the first category, followed by the cheapest DVD from the second category, followed by increasingly expensive DVDs from the second category, and so on—until all matching records have been sorted into this order. Then the TOP 100 DVDs from that list will be retrieved. You can also specify the direction in which a sort order is used, either from the smallest value to the largest value, or from the largest to the smallest. This is achieved by using the ASC or DESC keywords (short for ascending and descending). The default sort order is ascending; you can use it in your SQL statements, but it's not necessary unless you're specifying more than one sort order. To sort the previous example so that the most expensive DVDs are listed first and the cheapest are listed last within each category but the categories remain in ascending order, you can add the ASC and DESC keywords to the statement, as in the following example.

Access:

```

SELECT TOP 100 * FROM tblProducts
WHERE ProductCategory = 'DVD'
AND DateAdded BETWEEN #01 August 2005# AND #31 August 2005#
ORDER BY ProductCategory ASC, ProductPrice DESC

```

SQL Server:

```

SELECT TOP 100 * FROM dbo.tblProducts
WHERE ProductCategory = 'DVD'
AND DateAdded BETWEEN '01 August 2005' AND '31 August 2005'
ORDER BY ProductCategory ASC, ProductPrice DESC

```

8.6.6. IN

The IN keyword is useful for those times when you might have several criteria that you want to use as a filter on your data. You could write a long WHERE clause that uses many OR statements, or you could write a much shorter WHERE clause that uses the IN

statement. The following examples illustrate this point. First, have a look at using the OR keyword.

Access:

```
SELECT ProductName, ProductCategory,  
ProductDescription, ProductPrice  
FROM tblProducts  
WHERE ProductCategory = 'Cat 1' OR ProductCategory = 'Cat 2'  
OR ProductCategory = 'Cat 3' OR ProductCategory = 'Cat 4'  
OR ProductCategory = 'Cat 5'
```

SQL Server:

```
SELECT ProductName, ProductCategory,  
ProductDescription, ProductPrice  
FROM dbo.tblProducts  
WHERE ProductCategory = 'Cat 1' OR ProductCategory = 'Cat 2'  
OR ProductCategory = 'Cat 3' OR ProductCategory = 'Cat 4'  
OR ProductCategory = 'Cat 5'
```

Now, look at how this could be written using the IN keyword. You specify the values to compare against the column within parenthesis.

Access:

```
SELECT ProductName, ProductCategory,  
ProductDescription, ProductPrice  
FROM tblProducts  
WHERE ProductCategory IN ('Cat 1','Cat 2','Cat 3','Cat 4','Cat 5')
```

SQL Server:

```
SELECT ProductName, ProductCategory,  
ProductDescription, ProductPrice  
FROM dbo.tblProducts  
WHERE ProductCategory IN ('Cat 1','Cat 2','Cat 3','Cat 4','Cat 5')
```

If the data type that you're comparing a value against is numeric, it doesn't require single quotes around it. The preceding example uses them because the ProductCategory column is a text-based column.

8.6.7. GROUP BY

The GROUP BY keyword groups identical values in a column in your result set. In this regard, it's quite similar to the ORDER BY keyword. It differs, however, in that it doesn't allow certain data types to be specified in the overall SQL statement. For example, a Memo field in Access, or its equivalent Text field in SQL Server, can't be used in a GROUP BY statement (unless they're being used as an expression of criteria to be met—for example, if you simply want all the records with a description column that's not empty). When you use GROUP BY, data will be sorted first in the order of the columns specified in

the GROUP BY statement, then in an ascending order, row by row, unless you also specify an ORDER BY statement as well. For example, in the tblProducts table, you might have many products that share the same category, and you might want to return a result set that groups together the results by category and then by price, as in the previous ORDER BY example. A valid GROUP BY statement might be created as follows.

Access:

```
SELECT TOP 100 ProductName, ProductCategory, ProductPrice  
FROM tblProducts GROUP BY ProductName, ProductCategory, ProductPrice
```

SQL Server:

```
SELECT TOP 100 ProductName, ProductCategory, ProductPrice  
FROM dbo.tblProducts  
GROUP BY ProductName, ProductCategory, ProductPrice
```

You can see that the columns that you want to be returned by this statement have to be specified. You can't use the * wildcard to return all columns because you can't include all the columns of this table in a GROUP BY clause. The following example shows the use of the GROUP BY statement in conjunction with the use of a column that contains data that can't be returned to the result set, but that can be used to filter that result set.

Access:

```
SELECT TOP 100 ProductName, ProductCategory, ProductPrice  
FROM tblProducts  
WHERE (ProductDescription IS NOT NULL)  
GROUP BY ProductName, ProductCategory, ProductPrice
```

SQL Server:

```
SELECT TOP 100 ProductName, ProductCategory, ProductPrice  
FROM dbo.tblProducts  
WHERE (ProductDescription IS NOT NULL)  
GROUP BY ProductName, ProductCategory, ProductPrice
```

You still select the same columns as before, but the results will be filtered to show only records that have a value that's not null in the ProductDescription column. The use of the NOT keyword comes into play here. It's used to reverse the logic of the statement it's contained within. In this case, you reversed the IS NULL statement by specifying IS NOT NULL, so that all records that have a ProductDescription will be returned.

8.6.8. DISTINCT

DISTINCT is used to prevent duplicate rows from being returned to the result set. If you absolutely must have a result set that contains at least one unique value on every row, then DISTINCT can be the answer. However, it isn't always necessary to use DISTINCT because, generally, you'll return the primary key value of each record in your query. If the need arises for a result set that doesn't include the primary key value, then DISTINCT will ensure that no duplicate records are returned. The following is an example of using the

DISTINCT keyword.

Access:

```
SELECT DISTINCT ProductCategory FROM tblProducts
```

SQL Server:

```
SELECT DISTINCT ProductCategory FROM dbo.tblProducts
```

This will select all the individual categories from the ProductCategory column in the tblProducts table. You may have thousands of rows of products in that table; each of them stored in, let's say, 20 categories. This example will return those 20 distinct rows to your result set.

8.6.9. Going on a DATE

Dates are notoriously awkward when it comes to databases. If you live anywhere outside of the countries that follow the US date format, you can find yourself in all sorts of trouble when storing dates in your database. In the examples shown for the keyword BETWEEN earlier in this chapter, the date was specified in a manner that leaves no ambiguity over how it should be read. However, depending on where you are in the world, 01/08/2005 could mean the first of August or the eighth of January. Access and SQL Server always try to store the date in the US format: month/day/year. If they come across a date that can't be stored this way, such as 17/08/2005, then it's converted into the month/day/year format, while keeping the date accurate. To save yourself a lot of trouble, just use unambiguous dates like 17 August 2005. When using dates in your SQL statements (such as when you perform a search for matching items that have a date that falls between two dates that you specify), you'll need to remember the following strict rules about how each database application needs dates presented to it:

Access needs the date value to be wrapped in hash (or pound) symbols; for example, #01 August 2005#.

SQL Server needs the date value to be wrapped in single quotes; for example, '01 August 2005'.

8.7. Making the connection

To be able to use data on your web pages, you need to tell your web application where to look for it. There are two ways of doing this: you can either tell Frontpage that a system DSN (data source name) has been set up on your computer and you wish to use that, or you can define a custom connection string that encapsulates all the information required to connect to the data source, including the location of the database and security information needed to access the data. In the following example, you'll concentrate on using a System DSN. There are no real advantages to be had in using one method over the other. It might simply come down to your hosts not allowing you to create a system DSN on their servers. If this were the case, you would then need to use a custom connection

string, known as a DSN-less connection. This can all be done from Frontpage, but it's just as easy to go out to your operating system and set it up from there—the dialogs that are used are exactly the same, but getting to them requires fewer clicks on your part (and far less explanation on mine).

8.7.1. Setting up a DSN to an Access database

1. Open the ODBC Data Source Administrator on your computer by going to Control Panel ► Administrative Tools ► Data Sources (ODBC).
2. Click the System DSN tab, and then click Add.
3. Select Microsoft Access Driver (*.mdb) from the list, and click Finish.
4. Now you get to enter the name of this DSN. Name it DSNmysampledatabaseAccess (or choose another name of your liking).
5. Leave the Description field empty unless you really want to enter something in there—it's not necessary for this to work.
6. Next, click Select to select the location of your database. Use the Select Database dialog to find the database and select it. Click OK, and the full path to your database will be shown above the Select button. Mine is C:\databases\mysampledatabase.mdb.
7. Click OK twice to close the two open dialog boxes.

Your DSN is set up and ready to use.

8.7.2. Using commands

At some point in your website development, you'll no doubt want to add, edit, or delete data from your database. If you need to have more than one of the same database interactions on the same page, then you'll need to use one command per database interaction. For instance, if you want to insert a new record into your database that needs to be split over two tables, you could set up two commands on a page to do this for you. The server behavior versions of these database interactions can only be included on your web page once. Perhaps the major benefit of commands over server behaviors is the amount of code produced. Server behaviors tend to produce bulky code because they need to cater to a lot of different scenarios, whereas their command counterparts produce a small amount of code while maintaining adaptability. The following examples will introduce you to the usage of each of the commands available to you, with the exception of the stored procedure command, which is beyond the scope of this book. The creation process of each of these commands begins in exactly the same way. Click the plus (+) button in either the Bindings panel or the Server Behaviors panel and select Command from the popup menu. You'll see the Command dialog. The first thing to do in creating any command is to name it, which you can do in the top-left box of the dialog. All commands are given a default name, but it's not too descriptive, so you should always change that to something that makes more sense. Your command will only be listed in, and therefore editable from, the Server Behaviors panel after creation, no matter which panel you choose to start the creation process with.

8.7.3. Inserting a record

The following example explains how to insert a record using a command. You'll be adding a new quote to the database, so you might name your command `cmdInsertQuote`. My connection is already selected for me because I've already set up a connection for this site. If yours isn't, then you'll need to set up a connection before continuing, which you can do by clicking the Define button in this dialog (or follow the steps outlined earlier in this chapter in the "Making the connection" section). From the Type drop-down list, you want to select Insert because this is an INSERT command. (Easy, isn't it?) Immediately after you select Insert from the list, the basic INSERT statement is added to the SQL window. All you're going to do is populate this statement with the relevant information. You can ignore the check box next to Return recordset named, as this is used only for stored procedures. The buttons in the lower-right of the dialog, under the Add to SQL heading, change to reflect the type of statement you're building. For the INSERT statement, only one of the two buttons is active—it's called COLUMN. In this case, you want to insert only certain pieces of information into the `tblQuotes` table. In all likelihood, those pieces of information would have been submitted from a form on the previous page. With this in mind, you need to select the columns you're inserting your data into from `tblQuotes` in the Database items window, and then click the COLUMN button. Select `AuthorID` and click COLUMN. The SQL window is updated to reflect your selection. Then, select `CategoryID` and click COLUMN. Finally, select `Quote` and click COLUMN again. The SQL window should now read as follows:

```
INSERT INTO tblQuotes (AuthorID, CategoryID, Quote) VALUES ( )
```

All you need to do is tell the command what values to insert into the specified columns. An important point to note is that you can't mix up the items between lists; they must be in the exact same order.

In other words, the order in which you specify the items in the INSERT INTO line must be the same order in which you specify the items in the VALUES line. To get values into this INSERT command from a fictitious form on the previous page, you need to request that information and use variable placeholders in the values listing, as follows. Click inside the parentheses after VALUES, and enter the following string of variable names, each separated by a comma (don't forget, text-based values need to be surrounded by single quotes): `varAuthorID,varCategoryID,'varQuote'`

Now you need to define where these variables are going to get their information. This example still assumes that you're getting your information from a fictitious form on the previous page and that the form elements were named as shown in the Request parts of the following details. Click the plus (+) button above the Variables window, and then click in the Name column and type `varAuthorID`. Tab to the Run-Time Value column and type `Request("AuthorID")`. Add another variable line for `varCategoryID` and enter a runtime value of `Request("CategoryID")`. Finally, repeat the process once more, this time with a name of `varQuote` and a runtime value of `Request("Quote")`. The completed Command dialog, ready for data to be inserted. With all that done, click OK, and your command will be applied to the page. Now, when the form on the previous page submits its information to this page, it will be inserted into the database.

8.7.4. Updating a record

This UPDATE command will update the AuthorID of a record (for instance, the one you just inserted with the INSERT command). With the Command dialog open, name this command cmdUpdateAuthorID, and select Update from the Type drop-down list. Your connection should already be selected in the Connection drop-down list. The two buttons in the Add to SQL area have the labels SET and WHERE, because that's the syntax of the UPDATE statement, as you'll see shortly. In the Database items window, expand the Tables listing, and then expand the tblQuotes table. Click to highlight the AuthorID column, and then click the SET button. Your SQL window will be updated to show the following SQL code (this example uses Access—the SQL Server version of this would be the same but with dbo. prefix on the table name):

```
UPDATE tblQuotes  
SET AuthorID  
WHERE
```

Now you need to tell the command what value AuthorID should be set to, and you also need to identify which record this statement should be applied to. You do this by typing = 'varAuthorID' after AuthorID. The SET line of the SQL window will now read as follows: SET AuthorID = varAuthorID varAuthorID, as you've probably guessed by now, is a variable that will get its runtime value from a field called AuthorID, which you'll assume has been submitted from the previous page. Click the plus (+) button above the Variables window and add varAuthorID as the variable name and Request("AuthorID") as the runtime value. The only thing left to do is to make sure that you're going to update the right record in your database. You do this by using the ID number of the record, which will also have been passed through to this page from the previous page. Click ID in the Database items window and then click WHERE. Click in the SQL window after ID and add = varQuoteID. You now need to add that variable to the variables list, so click the plus (+) button above the Variables window again, and on the new variable line, set the variable name to varQuoteID and the Run-Time Value to Request("QuoteID"). The completed Command dialog, ready for data to be updated

8.7.5. Updating multiple records (a simple example)

You'll often want to be able to update more than one user at a time, so that, for example, you can activate or deactivate several accounts at once. In the following example, suppose you have a page that displays all of your website administrators, and against each one is a check box that can be checked and unchecked to activate and deactivate that administrator. With the Command dialog open, name this command cmdUpdateMultiAdmin and select Update from the Type drop-down list (your connection should already be selected in the Connection drop-down list). Expand the Tables listing in the Database items window, expand the tblAdmin table, highlight Included, and click SET. Your SQL window should now display the following code:


```
UPDATE tblAdmin
SET Included
WHERE
```

Now type = 0 after Included. This column of the database is a binary column, so a 1 (SQL Server) or -1 (Access) in this column would mean that the record is included—a 0 means it is excluded. Now you need to specify which records to update. You'll be using the IN keyword in this case (see the "Useful SQL Keywords" section earlier in the chapter for more information on the IN keyword). In the Database items window, click ID and then click WHERE. ID will be added to the SQL window. Click in the SQL window to place the cursor after ID, and type the following:

```
IN (varID)
```

Your SQL should now look like the following:

```
UPDATE tblAdmin
SET Included = 0
WHERE ID IN (varID)
```

All you need to do now is add varID to the Variables window. Click the plus (+) button above the Variables window and enter the variable name (varID) in the Name column. In the Run-Time Value column, type Request("ID"). The completed Command dialog, ready for multiple rows of data to be updated. Now, when multiple check boxes are ticked on the page that submits to this page, a list of IDs will be delivered in a comma-delimited string, and all the records that match one of those IDs will have their Included column updated to contain a 0. When this happens, these administrators will no longer be able log in—as long as the correct logic is in place to assure that the Included column is used, of course!

8.7.6. Deleting a record

Deleting records is sometimes a necessary step for keeping your database up to date. The following example shows you how passing the ID number of a product record to a DELETE command will delete that product. With the Command dialog open, give the command a name of cmdDeleteRecord and select Delete from the Type drop-down list. Your connection should already be selected in the Connection dropdown list. The shortcut detailed in other commands doesn't work for the DELETE command, you have to specifically click the table name in the Database Objects window and click the DELETE button to add the table name to the SQL. Then, expand the tblQuotes table in the Database Objects window and click the ID column. Click WHERE to add this column to the SQL window.

Your SQL window should look like this:

```
DELETE FROM tblQuotes
WHERE ID
```

You now need to place the cursor after ID and type = varID. Add the variable to the Variables window by clicking the plus (+) button above it. In the Name column, type varID and set the Run-Time Value to Request("ID"). The completed Command dialog, ready for data to be deleted. Click OK to apply this command to the page. When this page is passed

the ID number of a record in the ProductID parameter, that record will be deleted.

8.7.7. Deleting multiple records (simple example)

Following on from the previous example, a common scenario would be the need to delete multiple items from your database. You could do this one at a time if you want, but if you need to delete lots of items then I wouldn't recommend it. What I would recommend, however, is using a command to delete multiple items, as shown in the following example. With the Command dialog open, give the command a name of cmdDeleteMultiQuotes and select Delete from the Type drop-down list. Your connection should already be selected in the Connection drop-down list. In the Database items window, select the table that you are deleting the products from—in this case, tblQuotes—and click the DELETE button. Then click the ID column and click the WHERE button. Your SQL window should now look like the following:

```
DELETE FROM tblQuotes  
WHERE ID
```

Now place the cursor after D and type: IN (varID).

Click the plus (+) button above the Variables window to add a new variable line. Click the Name column and type varID. Tab to the Run-Time Value column and type Request("ID"). The completed Command dialog, ready for multiple rows of data to be deleted. Click OK to apply this command to the page. When this page is passed the number of one or more records in the ID parameter, those records will be deleted. You can pass a comma-delimited string of numbers into this command, and it will delete each record whose ID is contained in that string.

CHAPTER NINE:

PROCESSING DATA from ONLINE FORMS

A defining character of Web server-side applications is their ability to retrieve and process data from online forms. Given the ubiquity of online forms on the Web today, and their importance to e-commerce, it makes sense that ASP includes functionality for retrieving and processing data from online forms. Before we look at how to program our ASPs, let us review some of the basics of online HTML forms.

The <Form> Tag and its Attributes

HTML online forms are built between <Form>...</Form> tags. A typical <Form>, including the opening <Form> tag.

The user will complete the form and click the Submit button, sending the data entered into the form to the server as variable name and values pairs. For example, if Ümit entered his name in the form coded above and clicked submit, we could represent the variable name and value pair sent to the server as `firstname=Ümit`, with `firstname`, the name taken from the Input fields's Name attribute, as the variable name, and the value entered in the field as the associated value.

9.1. The Action Attribute

The value assigned to the action attributes is the Web page being request when the Submit button clicked. Most likely, this will be the page that contains the code that will receive and process the data being submitted. This submission is in the form of an HTTP request which will include the variable name and value pairs sent by the form, as well as the location and name of the page requested.

A basic information for Self-Referencing" page:

On occasion we may see a <Form> tag without an Action attribute. This <Form> is requesting the current page, which is the default value of the Action attribute. In the other words, the ASP is requesting itself; it is a self-referencing page.

9.2. The Get and Post Methods

The Method attribute of the <Form> tag determines the method by which the data entered into the form is encoded and sent to the server to be processed. The Method tag has two has two possible values:

Method="Get": The variable name and value pairs from the form will be concatenated onto the end of the URL as a Query String. A typical URL containing a Query String might look like:

`http://www.neubookstore.com/responsepage.asp?firstname=Ümit&lastname=İlhan`

Pro: The resulting response page can be book marked in the browser.

Con: When the response page is displayed in the browser, the URL will be displayed in the address bar. Any sensitive information (i.e., credit card or T.C. identify card numbers), will be displayed as well as saved in the browser's history lists. Some servers also limit the number of characters that can be sent as a Query String.

Method="Post": The variable name and value pairs are encoded and sent to the server transparently in the HTTP headers when the form is submitted.

Pro: Sensitive information is not displayed or saved by the browser, and any number of variable name and value pairs may be included.

Con: The resulting response page cannot be bookmarked.

The default Method is Get. A <Form> tag without a Method attribute will, by default, use Method="Get".

9.3. ASP Objects

ASP expose a family of objects that we can use to retrieve information from the browser or the server, as well as to send HTML, including dynamically generated HTML, to the browser in the form of a response. An object can be through of as bundle of methods and properties, held in memory on the server, that we can manipulate to accomplish programming tasks.

Methods: Methods are actions that we can perform with or on object. I explained an given example; Response.Write which invokes the Write Method of the Response object to write HTML to the browser. Write is something the Response object can do.

Properties: Properties describe something about an object. Properties are sometimes read/write. A read/write property can tell us something about the current state of the object, but we can also assign a new value to the property to deliberately change the state of the object. In ASP, objects can also contain collections of other objects.

In my report, I will explain be working with several objects that are native to ASP: the server object, and its sub-, or child, objects. Request, Response, Session and Application. The relationship between these objects can be represented by this chart:

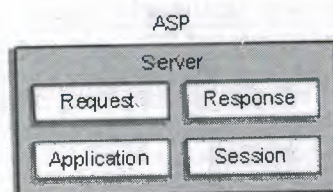


Figure 9.1 Mostly used ASP objects.

Server Object: Provides methods and properties that enable our ASPs to communicate with the Server.

Request Object: Provides methods and properties that enable our ASPs to retrieve information from the client, including information from forms.

Response Object: Provides methods and properties that enable our ASPs to send information, including HTML, to the client.

Application Object: Used to store information about your Web Application that can be made available to clients requesting pages from our site.

Session Object: Used to store information about a particular client's browsing session.

The Server, Request, Response, Application and Session objects are always available in the server memory, and always ready to use. These objects are "intrinsic" to ASP; they are ASP's native objects that are held in memory all of the time.

9.3.1. The Request and Response Objects

The Request objects provides methods, properties, and collections that requests made by the client to the server. The Response object provides methods, properties, and collections that handle the response made by the server to the client.

9.3.2. Request Object Collections

Some of the Collections of the Request object are:

Table 9.1. Some collections of Request objects table.

Collection	Description
Form	Variable name and value pairs sent by an online form, as submitted by METHOD="POST". Read-only.
QueryString	Variable name and value pairs sent in a Query String, as in those submitted from an online form using METHOD="GET". Read-only.
Cookies	Values of cookies sent from the browser. Read-only.
ServerVariables	Values of the HTTP and environment variables, including information about the client browser type and version, client operating system and version, client IP address, page the client was one when the current page was requested, etc. Read-only.

For example, to retrieve the value of a formfield with the name firstname that has been submitted via Method="Post", use this statement:

```
Request.Form(?firstname?)
```

To retrieve the value and assign it to a local variable we have named strFirstname, using like:

```
strFirstname = Request.Form(?firstname?)
```

In my two statements, the Form collection of the Request object is being accessed, and the value associated with the variable name firstname is being returned. If the data from the form had been submitted via Method="Get", the same operation could be accomplished with this statement:

```
strFirstname = Request.QueryString(?firstname?)
```

Table 9.2. Cookies and response object collection and Response object methods tables.

Collection	Description
Cookies	Contains the values being set as cookies on the client computer as part of the HTTP response. Integrating ASPs with cookies will be covered later in this class.

Response Object Methods

Method	Description
Write	Writes the specified text and/or variables to the current page
Redirect	Instructs the browser to connect to a different URL
Flush	Sends buffered output immediately
End	Stops the processing of the page and returns the current result
Clear	Erases any buffered HTML output
BinaryWrite	Sends text to the browser without character-set conversions
AppendToLog	Adds text to the Web server log entry for this request

9.4. Using Values Returned by a Form In Math

It is not unusual that our ASP would have to retrieve values from a form, and then use them in mathematical equations. In this case, it is important to remember that all values returned by a mathematical equations. In this case, it is important to remember that all values returned by a form, whether they they were entered as strings or numbers, or submitted via Post or Get, are returned to the server as strings. In this case, we may need to convert, or cast, the values to integers, or whole numbers, before performing math. VBScript provides the function `Cint(String)` for this purpose; it casts the string to an integer.

For example;

```
inty = Cint(Request.Form("numx"))
intxy = Cint(Request.Form("numxy"))
intxy_TotalQty = inty + intx
Response.Write = "Your " & intxy_total & " products are in the mail."
```

CHAPTER TEN:

ProjectASP Running Sessions Explanation

Mainpage



This is the index page of the site, there are categories and search tools for find a book As we see the user as an guest now.

Register

20.06.2007 - 08:48:49

Welcome to Registration!

NEW BOOK STORE

REGISTRATION FORM

First Name:

Last Name:

Address:

Country:

Phone: At least 11 digits.

Preferred User Name:

Password: At least 5 digits.

Confirm Password:

88ti

Internet %100

Before we log in the site. We have to register as a user to the database of the site. A registration form appears.

Search item

The screenshot shows a web browser window displaying the 'NEW BOOK STORE' website. The page has a dark header with a logo on the left and the site name in a stylized font. Below the header, there is a navigation bar with a 'Home' link. A sidebar on the left contains three search sections: 'Enter ISBN Number:' with a text input field containing '001' and a 'Search' button; 'Title:' with a text input field and a 'Search' button; and 'Author Name:' with a text input field and a 'Search' button. The main content area on the right features a 'Welcome Guest! Log in / Register' message, a note about adding items to a shopping cart, and a 'CATEGORIES' section with a list of subjects: Computer, Chemistry, Electronics, Physics, Mechanical, and Mathematics, each preceded by a right-pointing arrow. The browser's status bar at the bottom shows 'Internet' and '100%' zoom.

20.06.2007 - 08:38:34

[Home](#)

SEARCH

Enter ISBN Number:

Title:

Author Name:

NEW BOOK STORE

Welcome Guest! [Log in](#) / [Register](#)

To add shopping cart you have to log in or you can register.

CATEGORIES

- Computer
- Chemistry
- Electronics
- Physics
- Mechanical
- Mathematics

Internet 100%

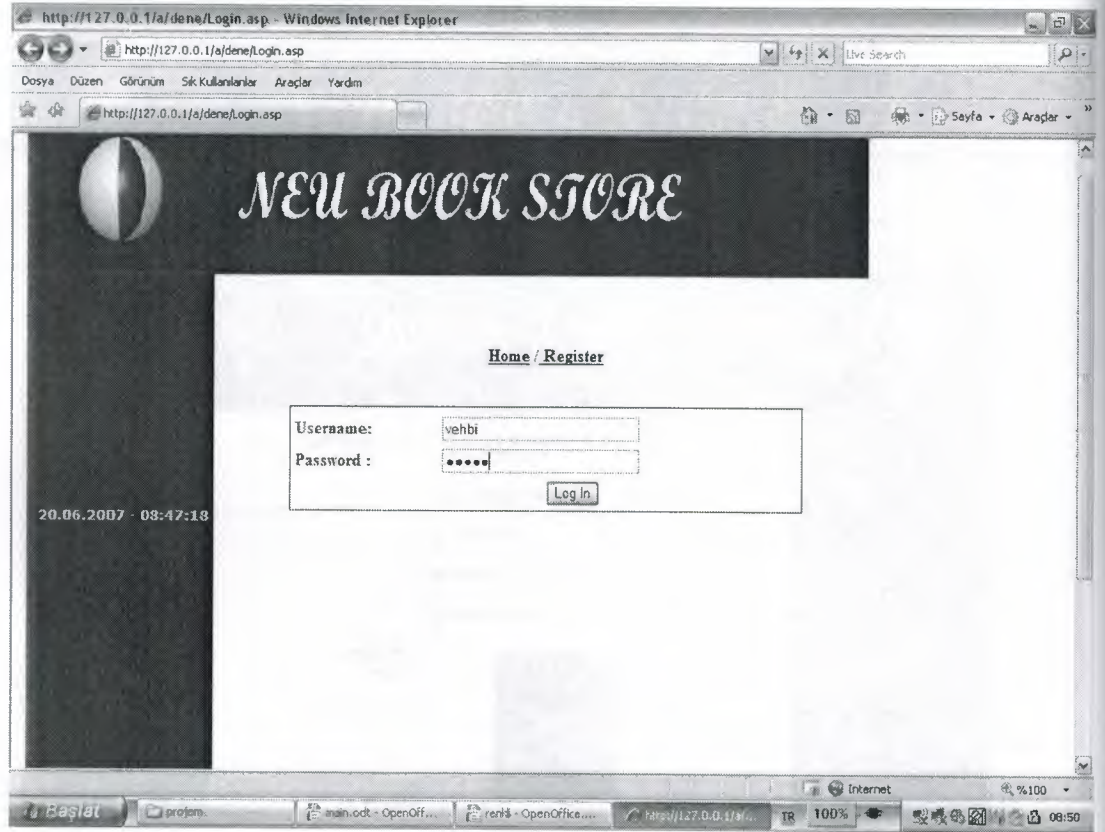
We may perform a search entering a information about a book in the related tabs. For example, a user performing a search entering isbn-number of a book.

Search Results



This is the search result of the user. One item is found because of isbn numbers are unique.

Login



The screenshot shows a Windows Internet Explorer browser window. The address bar displays the URL `http://127.0.0.1/a/dene/Login.asp`. The page title is "NEW BOOK STORE". The main content area features a login form with the following elements:

- A header bar with the text "NEW BOOK STORE" and a logo.
- A navigation bar with links: [Home](#) / [Register](#).
- A login form with two input fields: "Username:" and "Password:". The "Username:" field contains the text "vehbi". The "Password:" field contains five dots.
- A "Log In" button located below the password field.
- A timestamp "20.06.2007 - 08:47:18" displayed on the left side of the page.

The browser's taskbar at the bottom shows several open applications, including "Başlat", "projam...", "main.odt - OpenOff...", "renli - OpenOffice...", and "http://127.0.0.1/a/...". The system clock in the bottom right corner indicates the time is 09:50.

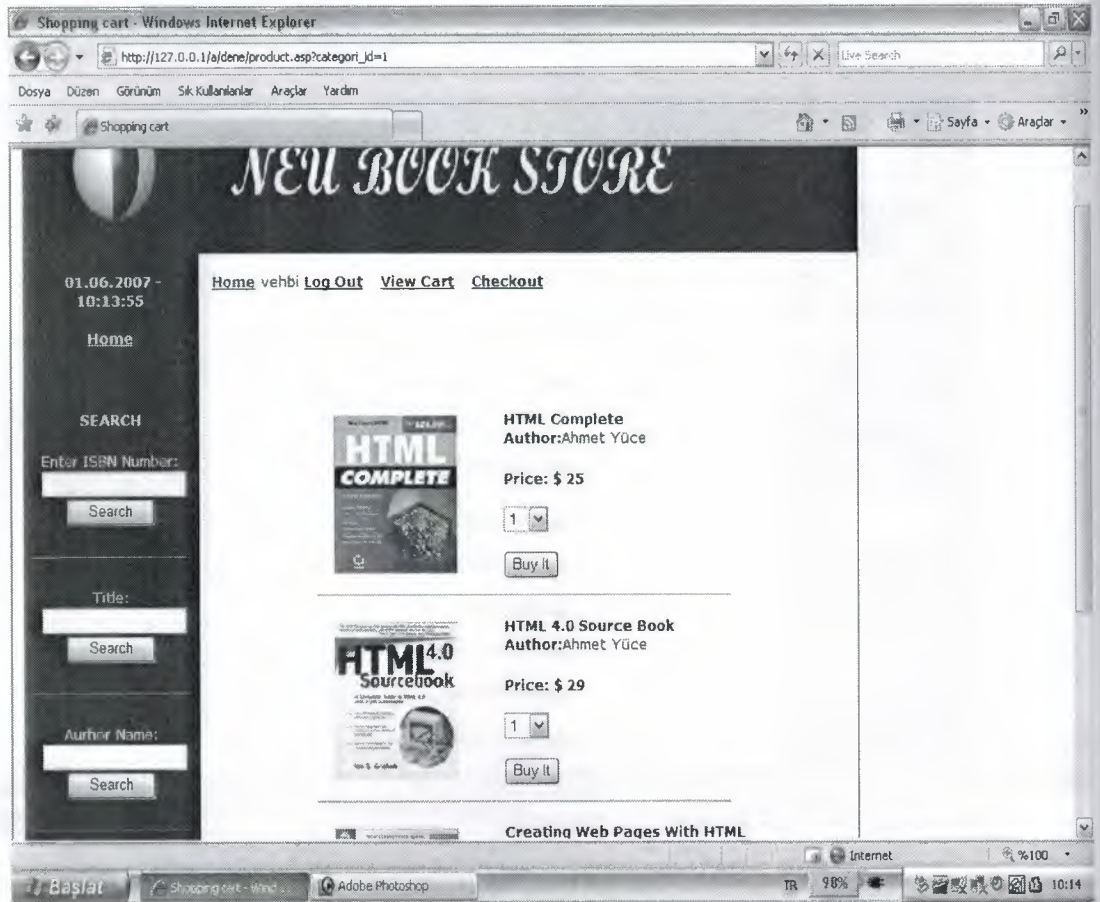
We have completed a registration. Let's log in.

Logged in



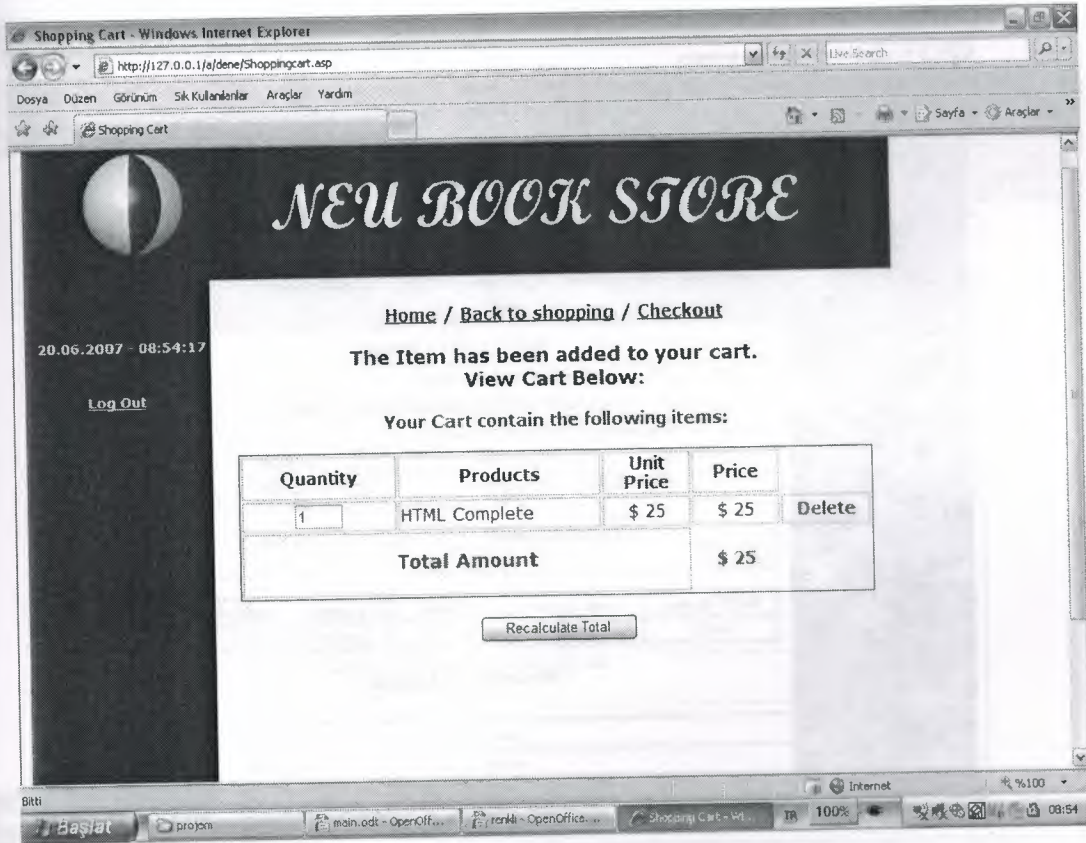
As we see the user “vehbi” login to the site as appears above.

Buying item



Let's perform a buying operation which begins selecting a book category in the main page and choose a book from the selected category and click the buy button. After clicking buy button we perform some other operations in the following sections which will be described.

Shopping Cart



When we click the buy button, we are being add the book to our shopping cart as we see above.

Checkout

Shopping Cart - Windows Internet Explorer

http://127.0.0.1/a/dene/checkout.asp

Dosya Düzen Görünüm Sık Kullanılanlar Araçlar Yardım

Shopping Cart

20.06.2007 - 08:56:03

[Home](#) / [Log Out](#)

NEU BOOK STORE

[Proceed to Payment](#)

Your Cart contain the following items:

Quantity	Products	Unit Price	Price	
2	HTML Complete	\$ 25	\$ 50	Delete
Total Amount			\$ 50	

First Name	Last Name	Address	Country	Telephone
vehbi	dilek	girne	Cyprus	6255555

[Proceed to Payment](#)

Bild

Başlat

projem

main.edt - OpenOffice...

renk - OpenOffice...

Shopping Cart - Wi...

TR 100%

Internet

08:56

In the shopping cart page, we checkout the buying operation clicking by the Proceed to Payment. Also we may check our user information at bottom of the page.

Payment

The screenshot shows a Windows Internet Explorer window with the address bar displaying 'http://127.0.0.1/aldene/payment.asp'. The page title is 'Payment'. The main content area features a dark header with a globe icon and the text 'NEU BOOK STORE'. Below the header, the date and time '20.06.2007 - 08:57:37' are displayed. The central part of the page contains a 'Payment by Credit Card' form. The form has a table-like structure with the following fields and values:

Payment by Credit Card	
Charged By:	Near East University
Cardholder's Name:	Expiration Date:(mm/dd/yyyy)
vehbi okay dilek	22/09/2010
Pick Card Type:	
Visa	
Enter Credit Card Number:	
2432505340585854	
Please fill the credit card over and input the 3 or 4 digit CSC code.	
123	
Name:	vehbi
Address:	girne
Submit Payment	

The bottom of the browser window shows the taskbar with several open applications: 'Başlat', 'projen', 'main.odt - OpenOff...', 'renköl - OpenOffice...', 'Pazarlar - Hedef...', and 'TR 100%'. The system clock in the bottom right corner shows '08:58'.

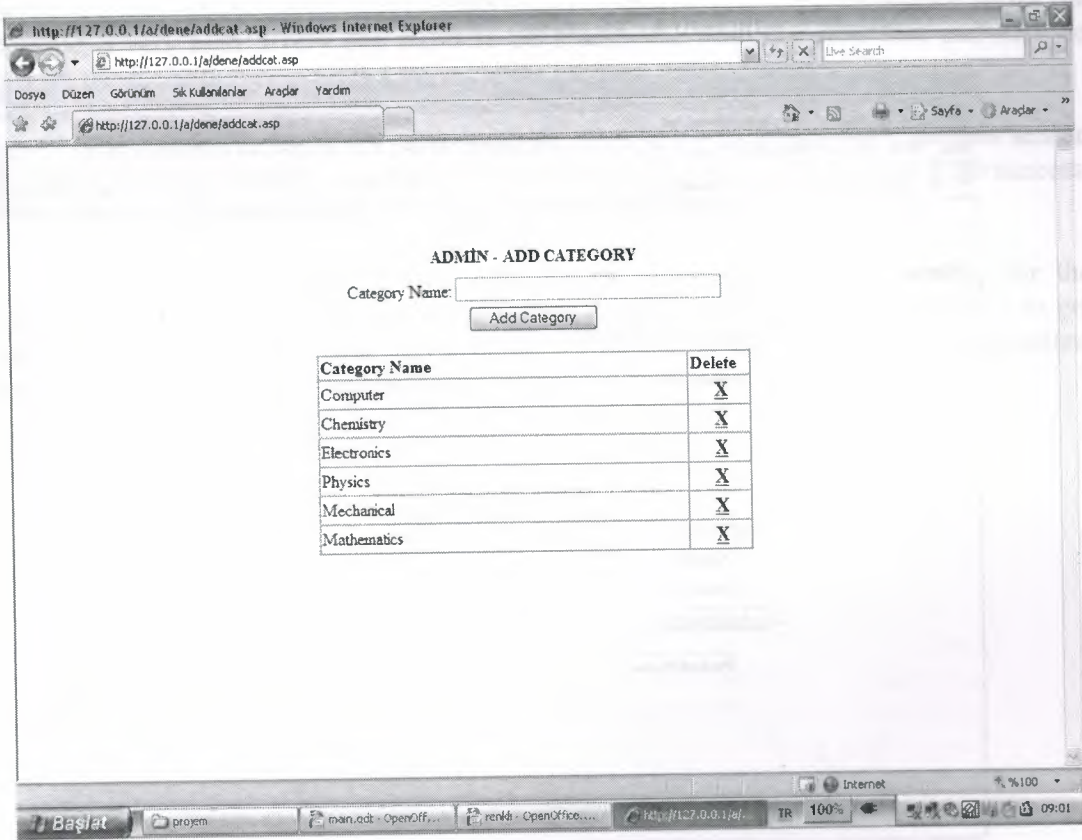
Let's complete our payment by our credit card. As we see above we must choose our credit card company for correct validation operation of our card. After entering the required information we can Submit our Payment.

Completing Purchase



Then our purchase operation is done in a secure way.

Administration Page



Adding and removing categories.

Adding Products to a Categories

http://127.0.0.1/a/dene/addproduct.asp - Windows Internet Explorer

http://127.0.0.1/a/dene/addproduct.asp

Dosya Düzen Görünüm Sık Kullanılanlar Araçlar Yardım

http://127.0.0.1/a/dene/addproduct.asp

Homepage

ADMIN - ADD PRODUCT

Product Code:

Description:

Price:

Available:

Category:

Source (image):

Quantity:

Category Id: Computer - ID No.1

Add Product

Product Name	Delete
	X

Bitli

Başlat

proje

mainnode - OpenOffice...

renkli - OpenOffice...

http://127.0.0.1/...

TR 100%

Internet

09:02

Adding products to selected category.

Conclusion

In real word ASP pages reaches the success because of the their interactivity between the client and server that is really magic. Especially, in the e-commerce sector the demand is very high. Every one wants to get a response from the other side and ASP do this. In these conditions, ASP is a good choice for Web development.

The interactive pages are performed, the request are taken and the responses sent to the client side successfully, thus the implementation of the project ASP reached the success. Especially, the database connection have been done properly.

The ASP pages were a beginning for the interactivity. We are wating for the interactive new technologies in the near future. Perhaps, next time we will be able to get information from the client side by voice, and we will not need to any validation operations because of finger prints of the clients. Who knows what is next?

APPENDIX

Index.asp

[illegible]

```

<p align="center"><font face="Verdana" color="#FFFFFF"><b><font
size="2">SEARCH<br>
</font>
</b><font size="2"><br>
<form method=post action=searchisbn.asp>
<p align="center">Enter ISBN Number:<br>
&nbsp;</font><font face="Verdana"><input name=isbn size="17"></font><font
size="2"><br>
</font>
<font face="Verdana">
<input type=submit value=" Search "></font><font size="2"> </font></font></p>
</p>
<hr width="90%" size="1"></form>
<form method=post action=searchtitle.asp>
<p align="center"><font face="Verdana" color="#FFFFFF"><font size="2">Title:<br>
&nbsp;</font><font face="Verdana"><input name=title size="17"></font><font
size="2"><br>
</font>
<font face="Verdana">
<input type=submit value=" Search "></font><font size="2"> </font></font></p>
<hr width="90%" size="1"></form>
<form method=post action=searchauthor.asp>
<p align="center"><font face="Verdana" color="#FFFFFF"><font size="2">Author
Name:<br>
&nbsp;</font><font face="Verdana"><input name=author size="17"></font><font
size="2"><br>
</font>
<font face="Verdana">
<input type=submit value=" Search "></font><font size="2"> </font></font></p>
<hr width="90%" size="1">
</form>

</p>
<p align="center">&nbsp;</p>
<p align="center">&nbsp;</p>
<p align="center">&nbsp;</p>
<p align="center"><font face="Verdana" size="2"><br>
</font>
</p>
</td>
</tr>
</table>
<div id="Layer1" style="position:absolute; width:638px; height:477px; z-index:1; left:
191px; top: 147px">
<font size="2" face="Verdana"><br>

```



```

        </td>
    </tr>
</table>
</div>

</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p><font face="Verdana" size="2"><br>
</font>
</p>
<p align="center">&nbsp;</p>
<p align="center">&nbsp;</p>
<p align="center">&nbsp;</p>
<p align="left"><font face="Verdana" size="2">&nbsp;</font> </p>
</div>
<hr width="84%" align="left" color="#800000" size="3">
<p>&nbsp;</p>
<p>&nbsp;</p>
<p><font face="Verdana" size="2">&nbsp;</font> </p>

</BODY></HTML>

```

checklogin.asp

```

oRs.Open "select*from customer",objConn
oRs.MoveFirst
oRs.Find "Username=" &strUsername&""
'redirect to login.asp if username doesnot exist'
if oRs.EOF then
Response.Redirect "login.asp"
' check password and redirect to logagain.asp if password is wrong'
else if strPassword=oRs("Password") then
Session("Username")=strUsername

'if username is admin then redirect to Administrator.asp else redirect to Getajob.asp'
if Session("Username")="Administrator" then
Response.Redirect "Administrator.asp"
else
Response.Redirect "Product.asp"
end if
Response.Redirect "Product.asp"
else
Response.Redirect "login.asp"
end if

```

```

end if
end if
'close connection
oRS.Close
Set oRS = Nothing
objConn.Close
Set objConn = Nothing
%>
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<title>NeuBookStore.com</title></HEAD>
<BODY background="images/Animation2.gif">
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<table id="AutoNumber5" style="BORDER-COLLAPSE: collapse"
bordercolor="#111111" cellspacing="0" cellpadding="0" width="80%" border="0"
align="center">
<tr>
<td width="100%">
<center>
<table id="Table3" cellspacing="0" cellpadding="0" border="0">
<tr>
<td width="5">&nbsp;</td>
<td>
<table id="Table4" cellspacing="0" cellpadding="0" width="311" bgcolor="#b7c4ff"
border="0">
<tr>
<td width="6" bgcolor="black" colspan="2" height="6" rowspan="2">
</td>
<td width="299" bgcolor="#000000" height="1"></td>
<td width="6" bgcolor="black" colspan="2" height="6" rowspan="2">
</td>
</tr>
</tr>

```

```

<td width="299" bgcolor="#000000" height="5"> </td>
</tr>
<tr>
<td width="1" bgcolor="#000000" rowspan="4"> </td>
<td width="5" bgcolor="#000000"> </td>
<td width="299" bgcolor="#FFFFCC">
<table id="Table5" cellspacing="0" cellpadding="0" width="100%" border="0">
<tr bgcolor="#FFFFCC">
<td><font face="Verdana" size=2 color="#FFFFFF"><b><font
color="#000000">Validating
your login.</font></b></font></td>
</tr>
<tr>
<td> </td>
</tr>
</table>
</td>
<td width="3" bgcolor="#000000"> </td>
<td width="3" bgcolor="#000000" rowspan="4"> </td>
</tr>
<tr>
<td width="307" bgcolor="#b1b1b1" colspan="3"> </td>
</tr>
<tr>
<td valign="top" width="307" bgcolor="#4675b8" colspan="3">
<table id="Table6" cellspacing="0" cellpadding="2" width="299" border="0">
<tr bgcolor="#4675b8">
<td width="294"><font face="Verdana" size=2 <font color="#FFFFFF"><b>Please
wait a few moments... </b></font></font></td>
</tr>
</table>
</td>
</tr>
<tr>
<td width="5" bgcolor="#b7c4ff"> </td>
</tr>
<tr>

```



```

<td width="6" bgcolor="black" colspan="2" height="6" rowspan="2">
  </td>
  <td width="299" bgcolor="#000000" height="5"> </td>
  <td width="6" bgcolor="black" colspan="2" height="6" rowspan="2">
    </td>
</tr>
<tr>
  <td width="299" bgcolor="#000000" height="1"> </td>
</tr>
</table>
</td>
</tr>
</table>
</center>
</td>
</tr>
</table>
<div align="center"></div>
<P>&nbsp;</P>

</BODY>
</HTML>

```

addcat.asp

```

<!--#include file=connection.asp-->
<head>
<meta http-equiv="Content-Language" content="tr">
</head>
<p>&nbsp;</p>
<p>&nbsp;</p>
<div align="center">
<FORM method=post action=addcat2.asp>
<table border="0" width="48%" id="table1" height="93">
<tr>
<td height="31">
<p align="center"><b>ADMIN - ADD CATEGORY</b></td>
</tr>
<tr>
<td>
<p align="center">Category Name:

```

```


</td>
</tr>
<tr>
<td height="22">
<p align="center">
<input type=submit value="Add Category" Category name="Add Category">
<%
set RS = server.createobject("adodb.recordset")
sql="select * from cat"
rs.open sql,Objconn,1,3
%>
</p>
<div align="center">
<table border="1" width="89%" id="table2" cellspacing="0">
<tr>
<td><b>Category Name</b></td>
<td width="56"><b>Delete</b></td>
</tr>
<%while not rs.eof%>
<tr>
<td><%=rs("name")%></td>
<td width="56">
<p align="center">&nbsp;<font color="#FF0000"><span style="font-size: 15pt; font-
weight: 700"><a href="delcat.asp?id=<%=rs("c_id")%>">X</a></span></font></td>
</tr>
<%
rs.movenext
wend
%>
</table>
</div>
</td>
</tr>
</table>
</FORM>
</div>

```

addcat2.asp

```

<!--#include file=connection.asp-->
<%
catname=request.form("catname")
set RS = server.createobject("adodb.recordset")

```

```

sql="select * from cat"
rs.open sql,Objconn,1,3
rs.addnew
rs("name")=catname
rs.update
%>
<meta http-equiv=refresh content=0;url=admin.asp>

```

addcustomer.asp

```

<%@ Language=VBScript %>
<%option Explicit%>

```

```

<!-- #include file="adovbs.inc" -->
<!-- #include file="connection.asp" -->
<%

```

```

'Collect information from jobseekers registration form'
'variables to store information from form'

```

```

dim
strFirstName,strLastName,strGender,strBirth,MaritalStatus,strphone,strmobilephone,strOp
hone,strwebsite,strplace
dim
strnationality,stradd1,stradd2,strpostalcode,strprovince,strcountry,stremai, strPassword,strU
sername,strQuestion,strAnswer
strFirstName=Request.Form("txtFirstName")
strLastName=Request.Form("txtLastName")
stradd1=Request.Form("txtadd")
strcountry=Request.Form("country")
strphone=Request.Form("txtPhone")
strUsername=Request.Form("txtusername")
strPassword=Request.Form("txtPassword")

```

```

'open connection
dim oRs
Set oRs=Server.CreateObject("ADODB.Recordset")
Set oRs.ActiveConnection=objConn
oRs.CursorType=adOpenDynamic
oRs.LockType=adLockOptimistic
oRs.Open "select*from customer",objConn
oRs.MoveFirst
' if username already exist redirect to registeragain.asp'
oRs.Find "Username='"&strUsername&'"
if oRs.EOF then

```



```

oRs.MoveFirst
'record data in database'
oRs.AddNew
oRs("Firstname")=strFirstName
oRs("Lastname")=strLastName
oRs("Address")=stradd1
oRs("Country")=strcountry
oRs("Phone")=strphone
oRs("Username")=strUsername
oRs("Password")=strPassword
oRs.update
response.Redirect "login.asp"
else
Response.Redirect "registerAgain.asp"
end if

```

```

'close connection
oRS.Close
Set oRS = Nothing
objConn.Close
Set objConn = Nothing
%>

```

```

<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</HEAD>
<BODY>

```

```

<P>&nbsp;</P>

```

```

</BODY>
</HTML>

```

addproduct.asp

```

<!--#include file=connection.asp-->

```

```

<head>
<meta http-equiv="Content-Language" content="tr">
</head>
<p>&nbsp;</p>
<p><a href="admin.asp">Homepage</a></p>
<div align="center">
<FORM method=post action=addproduct2.asp>
<table border="0" width="600" id="table1" height="93">

```



```

<td align="left" width="484">
<input name=source size="38"></td>
</tr>
<tr>
<td align="right" width="106">
Quantity:
&nbsp;
</td>
<td align="left" width="484">
<input name=quan size="38"></td>
</tr>
<tr>
<td align="right" width="106">
Cateory Id: &nbsp;
</td>
<td align="left" width="484">
<input name=catid size="9">
<%
set RS = server.createobject("adodb.recordset")
sql="select * from cat"
rs.open sql,Objconn,1,3
%> <select>
<%while not rs.eof
%>
<option><%=rs("name")%> - ID No:<%=rs("c_id")%> </option>
<%
rs.movenext
wend
%>
</select>
</td>
</tr>
<tr>
<td height="22" width="594" colspan="2">
<p align="left">
<input type=submit value="Add Product" Category name="Add Product">
</p>
<div align="center">
<table border="1" width="89%" id="table2" cellpadding="0">
<tr>
<td><b>Product Name</b></td>
<td width="56"><b>Delete</b></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>

```



```
<td width="56">
```

```
<p align="center">&nbsp;<font color="#FF0000"><span style="font-size: 15pt; font-weight: 700">X</a></span></font></td>
</tr>
```

```
</table>
</div>
</td>
</tr>
</table>
</FORM>
</div>
```

addproduct2.asp

```
<!--#include file=connection.asp-->
<%
set RS = server.createObject("adodb.recordset")
sql="select * from product"
rs.open sql,Objconn,1,3
rs.addnew
rs("description")=request.form("desc")
rs("price")=request.form("price")
rs("available")=request.form("ava")
rs("category")=request.form("cat")
rs("source")=request.form("source")
rs("quantity")=request.form("quan")
rs("c_id")=request.form("catid")
rs("productid")=request.form("pid")
rs.update
%>
<meta http-equiv=refresh content=0;url=admin.asp>
```

admin.asp

```
<html>

<head>
<meta http-equiv="Content-Language" content="tr">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<title>BOOK STORE</title>
</head>
<body>
```

```

<div align="center">
    &nbsp;&nbsp;&nbsp;</div>
<p>&nbsp;&nbsp;&nbsp;</p>
<p>&nbsp;&nbsp;&nbsp;</p>
<p align="center">&nbsp;&nbsp;&nbsp;</p>
<p align="center"><a href="addcat.asp">Add Categories</a></p>
<p align="center">&nbsp;&nbsp;&nbsp;</p>
<p align="center"><a href="addproduct.asp">Add Product</a></p>
<p align="center">&nbsp;&nbsp;&nbsp;</p>
</body>
</html>

```

author.asp

```

<!-- #include file="adovbs.inc" -->
<!-- #include file="connection.asp" -->
<%

```

```

isbn=request.form("isbn")
'codes to display products on different pages
DIM mySQL, objRS,b
mySQL = "SELECT * FROM Product where ProductID like '"&isbn&'"
Set objRS = Server.CreateObject("ADODB.Recordset")
objRS.CursorType = 1
objRS.Open mySQL, objConn
DIM intPageRecords, intRecords, intRecordCount, intCurrentPage
DIM intNumberOfPages, intDisplayPage
intPageRecords = Request.QueryString("page")
IF intPageRecords = "" THEN intPageRecords = 1 : intRecords = 1
intRecords = intPageRecords
intPageRecords = ((intPageRecords - 1) * 4) + 1
intRecordCount = 0

```

```

%>

```

```

<html>
<head>
<title>Shopping cart</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

```

```

<body bgcolor="#FFFFFF" text="#000000">

<div id="Layer1" style="position:absolute; width:504px; height:139px; z-index:1; left:

```

```

228px; top: 321px; background-color: #FFFFFF; layer-background-color: #FFFFFF;
border: 1px none #000000">
<p align="left">&nbsp;</p>
<%
IF NOT objRS.EOF THEN
objRS.Move (intPageRecords - 1)
DO WHILE intRecordCount < 3 and NOT objRS.EOF
%>
<table width="75%" border="1" align="center" bgcolor="#FFFFFF"
bordercolor="#FFFFFF">
<form method="post" action="Shoppingcart.asp">
<tr>
<td width="66%">
<div align="center">
" width="115"
height="135"></div>
</td>
<td width="34%"><b><%=objRs("description")%> </b><br>
<p><b>Price:&nbsp; $&nbsp; <%=objRs("Price")%></b><br>
<b>
<input type="hidden" name="txtproductID" value="<%=objRs("ProductID")%>">
<br>
<select name="txtQuantity">
<option value="1">1</option>
<option value="2">2</option>
<option value="3">3</option>
<option value="4">4</option>
<option value="5">5</option>
<option value="6">6</option>
<option value="7">7</option>
<option value="8">8</option>
<option value="9">9</option>
<option value="10">10</option>
</select>
<br>
<br>
<input type="submit" name="Submit" value="Buy It">
</b></p>
</td>
</tr>
<tr>
<td colspan="2">
<hr>
</td>
</tr>

```



```

</table>
<table border="1" width="78%" id="table2" cellspacing="0" cellpadding="0"
bordercolor="#6B0321">
<tr>
<td bgcolor="#6B0321"><font color="#FFFFFF">ds fsdfsdf sd fsd fsdf sdf
sdfs</font></td>
</tr>
</table>
</body>
</html>

```

cat.asp

```

<html>
<head>
<meta http-equiv="Content-Language" content="tr">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<title>BOOK STORE</title>
</head>
<body>
<div align="center">
<table border="0" width="45%" id="table1" height="18">
<tr>
<td>
<p align="left"><b>CATEGORIES</b></p>
<p align="center">
<!--#include file=connection.asp-->
<%
set RS = server.createobject("adodb.recordset")
sql="select * from cat"
rs.open sql,Objconn,1,3
while not rs.eof
%>
</p>
<table border="0" width="41%" id="table2" cellspacing="0" cellpadding="0"
height="37">
<tr>
<td width="33"></td>
<td style="border-bottom-style: solid; border-bottom-width: 1px">
<a href=product.asp?categori_id=<%=rs("c_id")%>>
<span style="text-decoration: none"><%=rs("name")%></span></a> </td>
</tr><%

```

```
rs.movenext
wend
%>
```

```
</table>&nbsp;
</p>
<hr>
</td>
</tr>
</table>
</div>
```

```
</body>
```

```
</html>
```

```
Ccard_inc.asp
```

```
<%
Const MASTERCARD = 0
Const VISA = 1
Const AMEX = 2
Const DISCOVER = 3
'*****
*
'This function validates any credit card number. NOTE: This function does
'not authorize a credit card credit.
'*****
*
Function ValidateCreditCard(CardType, CardNumber)
Dim bValidCard

bValidCard = False
If IsNumeric(CardNumber) Then
'Response.Write "Is Numeric" & "<BR>"

Select Case CLng(CardType)
Case MASTERCARD
If Len(CardNumber) = 16 Then
If Left(CardNumber, 2) = "51" Or Left(CardNumber, 2) = "52" Or Left(CardNumber, 2) =
"53" Or Left(CardNumber, 2) = "54" Or Left(CardNumber, 2) = "55" Then
'Response.Write "Is MASTERCARD" & "<BR>"
If CBool(Mod10Validation(CardNumber)) Then
bValidCard = True
End If
```



```

End If
End If
Case VISA
If Len(CardNumber) = 13 or Len(CardNumber) = 16 Then
If Left(CardNumber, 1) = "4" Then
'Response.Write "Is VISA" & "<BR>"
If CBool(Mod10Validation(CardNumber)) Then
bValidCard = True
End If
End If
End If
Case AMEX
If Len(CardNumber) = 15 Then
If Left(CardNumber, 2) = "34" Or Left(CardNumber, 2) = "37" Then
'Response.Write "Is AMEX" & "<BR>"
If CBool(Mod10Validation(CardNumber)) Then
bValidCard = True
End If
End If
End If
Case DISCOVER
If Len(CardNumber) = 16 Then
If Left(CardNumber, 4) = "6011" Then
'Response.Write "Is DISCOVER" & "<BR>"
If CBool(Mod10Validation(CardNumber)) Then
bValidCard = True
End If
End If
End If
End Select
End If

```

```

ValidateCreditCard = bValidCard
End Function

```

```

Function Mod10Validation(CardNumber)
Dim arrDoubledDigits
Dim arrUnaffectedDigits
Dim lSum
Dim sDoubledDigits
Dim sUnaffectedDigits
Dim iDoubledDigit
Dim iDigit1
Dim iDigit2
Dim i

```

lSum = 0

'Double every other digit moving from right to left starting with the second digit from the right.

For i = (Len(CardNumber)-1) To 1 step -2

iDoubledDigit = Mid(CardNumber, i, 1) * 2

'Response.Write iDoubledDigit & "
"

If Len(iDoubledDigit) > 1 Then

iDigit1 = Left(CStr(iDoubledDigit), 1)

iDigit2 = Right(CStr(iDoubledDigit), 1)

Else

iDigit1 = Left(CStr(iDoubledDigit), 1)

iDigit2 = ""

End If

If Len(iDigit2) = 0 Then

sDoubledDigits = sDoubledDigits & iDigit1 & ","

Else

sDoubledDigits = sDoubledDigits & iDigit1 & "," & iDigit2 & ","

End If

Next

sDoubledDigits = Left(sDoubledDigits, Len(sDoubledDigits) - 1)

For i = Len(CardNumber) To 1 step -2

sUnaffectedDigits = sUnaffectedDigits & Mid(CardNumber, i, 1) & ","

Next

sUnaffectedDigits = Left(sUnaffectedDigits, Len(sUnaffectedDigits) - 1)

'Create an array of all the digits that were doubled.

arrDoubledDigits = Split(sDoubledDigits, ",")

'Create an array of all the digits that were unaffected.

arrUnaffectedDigits = Split(sUnaffectedDigits, ",")

'Sum up all the digits

For i = 0 to UBound(arrDoubledDigits)

lSum = lSum + arrDoubledDigits(i)

Next

For i = 0 to UBound(arrUnaffectedDigits)

lSum = lSum + arrUnaffectedDigits(i)

Next

'Response.Write sDoubledDigits & "
"

```
'Response.Write sUnaffectedDigits & "<BR>"
'Response.Write "lSum:" & lSum
```

```
If CLng(lSum) > 0 Then
'If the sum passes the mod ten test then return true, otherwise return false.
If CLng(lSum mod 10) = 0 Then
    Mod10Validation = True
Else
    Mod10Validation = False
End If
Else
    Mod10Validation = False
End If
End Function
%>
```

checkout.asp

```
<%@ Language=VBScript %>
<%option Explicit%>
<!-- #include file="adovbs.inc" -->
<!-- #include file="connection.asp" -->
<%
    "session invalid"
    if session("Username")="" then
        Response.Redirect "Login.asp"
    end if
%>
<%
'Return to previous page'
dim strPreviouspage
strPreviouspage=Request.ServerVariables("HTTP_REFERER")
%>

<%
'*****'
'View Cart'
'*****'

dim objRs
Set objRs=Server.CreateObject("ADODB.Recordset")
Set objRs.ActiveConnection=objConn
objRs.CursorType=adOpenDynamic
objRs.LockType=adLockOptimistic
'Connect to cart table'
```



```

objRs.Open "select*from Cart where Username='" & session("Username") &
""

%>
<html>
<head>
<title>Shopping Cart</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body bgcolor="#FFFFFF" text="#000000">
<div align="left">
<table width="786" border="0">
<tr>
<td width="207" valign="middle">
<h2><font color="#510028"><b>Shopping Cart</b></font></h2>
</td>
<td width="339" valign="middle" align="center">
<div align="center">&nbsp;</div>
</td>
<td width="218" align="center" valign="middle">
<div align="right"><font color="#510028" size="1"><b><%=Date%>,<br>
<%=Time%> </b></font></div>
</td>
</tr>
</table>
<p><font color="#000000"><b><a href="index.asp"><font
color="#000000">Home</font></a>
</b><a href="product.asp"><font color="#000000"> Product</font></a></b>
<% if Session("Username")="" then%>
<b>Welcome Guest!</b></font>
<%else %>
<%=Session("Username")%>
<%end if%>
<font size=-1>
<% if Session("Username")="" then%>
<a href="Login.asp"><b><font color="#000000">Log in</font></b></a> </b><a
href="Registration.asp"><font color="#000000"> Register</font></a>
</b>
<%else%>
<a href="Logout.asp"><b><font color="#000000">Log Out</font></b></a><br>
<%end if%>
</font></p>
</div>
<div id="Layer1" style="position:absolute; width:606px; height:139px; z-index:1; left:

```

```

163px; top: 120px">
<div align="center"><b><font face="Times New Roman, Times, serif" size="3"><a
href="checkOut.asp" style="TEXT-DECORATION: none">
</a></font></b><font color="#510028"><b></b></font></div>
<div align="center">
<p><br>
<font color="#000000">
<% ' check record set if EOF Display No item in Cart '%>
<% if objRs.EOF then %>
<b>No item in cart</b>
<%else%>
<%
'count number of item and display all'

        dim numItems
        numItems=objRs.RecordCount
        if numItems= 0 then
%>
        <b>No item in cart</b>
        <% else%>
        </font><font color="#510028"><b><font color="#510028"><b><font size="3"
face="Times New Roman, Times, serif"><a href="payment.asp" style="text-
decoration:none"><font face="Times New Roman, Times, serif" size="3"><br>
        <font color="#804040">Proceed to
Payment</font></font></a></font></b></font><font color="#804040">
        </font></b></font></p>
        <p><font color="#000000"><br>
        <b><font face="Times New Roman, Times, serif" size="3"></font></b> </font>
        </p>
    </div>
    <p align="center"><font color="#000000"><b>Your Cart contain the following
items:</b>
    <br>
    </font>
    <form method="post" action="updateCart.asp">
    <div align="center">
    <table width="100%" border="1" bordercolor="#800000" bgcolor="#FFFFFF"
bordercolorlight="#800000" bordercolordark="#800000">
    <tr bordercolor="#FFCC66">
    <td width="26%">
    <div align="center"><b>Quantity</b></div>
    </td>
    <td width="34%">
    <div align="center"><b>Products</b></div>
    </td>

```

```

<td width="15%">
  <div align="center"><b>Unit Price</b></div>
</td>
<td width="15%">
  <div align="center"><b>Price</b></div>
</td>
<td bordercolor="#FFFFFF" width="10%">&nbsp;</td>
<td bordercolor="#FFFFFF" width="10%">&nbsp;</td>
</tr>
<% objRs.MoveFirst%>
<%
'Function to calculate total amount Purchased'

dim totalAmount
  totalAmount=0

  do while not objRS.EOF

    dim currentAmount,qty
    qty=objRs("Quantity")
    currentAmount=objRS("Price")
    totalAmount=totalAmount + currentAmount
    Session("totalAmount")=totalAmount

%>
<% 'display all information in table using shortcut direct from database'%>
<tr bordercolor="#FFCC66">
  <td width="26%">
    <div align="center"><%=objRs("Quantity")%></div>
  </td>
  <td width="34%">
    <div align="left"></div>
    <%=objRs("Description")%></td>
  <td width="15%">
    <div align="center">$ <%=objRs("Uprice")%></div>
  </td>
  <td width="15%">
    <div align="center">$ <%=objRs("Price")%> </div>
  </td>
  <td bordercolor="#FFCC99" colspan="2">
    <div align="center"> <a href="delete.asp?ID=<%=objRS("CartID")%>&link=del"
onClick="confirm(' DELETE Item from Cart ?')>"style="TEXT-DECORATION:
none"><b>
      <font face="Times New Roman, Times, serif" size="3"
color="#804040">Delete</font></b></a><font color="#804040">
</font>

```



```

        </div>
    </td>
</tr>
<% objRs.MoveNext
loop
%>
<tr valign="bottom" bordercolor="#FFCC66">
<td colspan="3" height="59" valign="bottom">
<div align="center">
<h3 align="center"><font color="#CC0000"><b>Total Amount</b></font></h3>
</div>
<div align="center"> </div>
</td>
<td height="59" valign="middle" bordercolor="#FFFFFF" width="15%">
<div align="center"><font color="#CC0000"><b>$
<%Response.Write totalAmount%>
</b> </font></div>
</td>
<td height="59" valign="middle" bordercolor="#FFFFFF" width="10%">&nbsp;</td>
<td height="59" valign="middle" bordercolor="#FFFFFF" width="10%">&nbsp;</td>
</table>
</div>
<div align="center"><%

dim oxrs
Set oxrs=Server.CreateObject("ADODB.Recordset")
Set oxrs.ActiveConnection=objConn
oxrs.CursorType=adOpenDynamic
oxrs.LockType=adLockOptimistic
oxrs.Open "select*from customer where Username='" & session("Username") & "'"
%>

<br>
<table width="100%" border="1" bordercolor="#800000">
<tr bgcolor="#FFFFFF">
<td width="21%">
<div align="center"><b>First Name</b></div>
</td>
<td width="22%">
<div align="center"><b>Last Name</b></div>
</td>
<td width="19%">
<div align="center"><b>Address</b></div>
</td>
<td width="14%">

```

```

<div align="center"><b>Country</b></div>
</td>
<td>
<div align="center"></div>
<div align="center"></div>
<div align="center"><b>Telephone</b></div>
</td>
</tr>
<tr bgcolor="#FFFFFF">
<td width="21%">
<div align="center"><font color="#804040"><%=oxrs("FirstName")%></font></div>
</td>
<td width="22%">
<div align="center"><font color="#804040"><%=oxrs("Lastname")%></font></div>
</td>
<td width="19%">
<div align="center"><font color="#804040"><%=oxrs("Address")%></font></div>
</td>
<td width="14%">
<div align="center"><font color="#804040"><%=oxrs("Country")%></font></div>
</td>
<td>
<div align="center"><font color="#804040"><%=oxrs("Phone")%></font></div>
</td>
</tr>
</table>
<font color="#510028"><b><font size="3" face="Times New Roman, Times, serif"><a
href="payment.asp" style="text-decoration:none"><br>
<font color="#804040">Proceed to Payment</font></a></font></b></font></div>
</form>
<% end if%> <% end if%>
<p>&nbsp;</p><p align="center"><font color="#000000"><br>
</font>
<h3 align="center"><font color="#510028"><b><font color="#000000">
</font></b></font></h3>
</div>
<p>&nbsp;</p>
</body>
</html>

```

confirm.asp

```

<%@ Language=VBScript %>
<%option Explicit%>
<!-- #include file="adovbs.inc" -->

```

```

<!-- #include file="connection.asp" -->
<%
"session invalid"
if session("Username")="" then
Response.Redirect "Login.asp"
end if
%>

<html>
<head>
<title>Shopping Cart</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body bgcolor="#FFFFFF" text="#000000">
<div align="left">
<table width="786" border="0">
<tr>
<td width="207" valign="middle">
<h2><font color="#510028"><b>Shopping Cart</b></font></h2>
</td>
<td width="339" valign="middle" align="center">
<div align="center">&nbsp;</div>
</td>
<td width="218" align="center" valign="middle">
<div align="right"><font color="#510028" size="1"><b><%=Date%>,<br>
<%=Time%> </b></font></div>
</td>
</tr>
</table>
<p><font color="#000000"><b><a href="index.asp"><font
color="#000000">Home</font></a>
</b></font></p>
<p><b><a href="product.asp"><font color="#000000"> Product</font></a></b>
<% if Session("Username")="" then%>
<b>Welcome Guest!</b></font>
<%else %>
<%=Session("Username")%>
<%end if%>
<font size=-1>
<% if Session("Username")="" then%>
<a href="Login.asp"><b><font color="#000000">Log in</font></b></a> <b><a
href="Registration.asp"><font color="#000000"> Register</font></a>
</b>
<%else%>
<a href="Logout.asp"><b><font color="#000000">Log Out</font></b></a>

```



```

<%end if%>
<br>
</font></p>
</div>
<div id="Layer1" style="position:absolute; width:606px; height:139px; z-index:1; left:
86px; top: 110px">
<div align="center"><b><font face="Times New Roman, Times, serif" size="3"><a
href="checkOut.asp" style="TEXT-DECORATION: none">
  </a></font></b><font color="#000000"><br>
  </font> </div>
<h3 align="center"><font color="#800000"><b><font face="Times New Roman, Times,
serif">Your
  order has been processed.</font></b></font></h3>
<div align="center">
  <font face="Times New Roman, Times, serif" color="#800000"><b><br>
  Thank you for your Purchase.</b></font></div>
<p>&nbsp;</p>
<p align="center"><font color="#000000"><br>
  </font>
<h3 align="center"><font color="#510028"><b><font color="#000000">
</font></b></font></h3>
</div>
<font size=-1> </font>
<p>&nbsp;</p>
</body>
</html>

```

connection.asp

```

<%
dim objConn
dim sConnect
set objConn=Server.CreateObject("ADODB.connection")
sConnect="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
Server.MapPath("books.mdb")
objConn.ConnectionString=sConnect
objConn.open

%>

```

delcat.asp

```

<!-- #include file="connection.asp" -->

```

```
<%
```

```
id=request.querystring("id")
```

```
mySQL = "SELECT * FROM cat "  
Set objRS = Server.CreateObject("ADODB.Recordset")  
objRS.CursorType = 1  
objRS.Open mySQL, objConn
```

```
del="SELECT * FROM cat where c_id"&id  
objConn.execute(del)
```

```
%>
```

```
<meta http-equiv=refresh content=0,url=addcat.asp>
```

```
delete.asp
```

```
<%@ Language=VBScript %>  
<%option Explicit%>  
<!-- #include file="adovbs.inc" -->  
<!-- #include file="connection.asp" -->  
<%
```

```
if Session("UserName")="" then  
Response.Redirect "Login.asp"  
end if  
%>
```

```
<%  
dim objRs  
Set objRs=Server.CreateObject("ADODB.Recordset")  
Set objRs.ActiveConnection=objConn  
objRs.CursorType=adOpenDynamic  
objRs.LockType=adLockOptimistic  
objRs.Open "select*from Cart",objConn  
dim strSQL  
if Request.QueryString("link")="del" then  
strSQL="DELETE from Cart WHERE CartID="&Request.QueryString("ID")&"  
objConn.Execute(strSQL)  
Response.Redirect "ViewCart.asp"  
end if  
%>
```

```
<HTML><HEAD><TITLE>XYZ SuperStore</TITLE>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
```

```
</script>
</HEAD>
<BODY>
```

```
</BODY></HTML>
```

login.asp

```
<%@ Language=VBScript %>
<%option Explicit%>
<%
```

```
%>
```

```
<HTML><HEAD><TITLE></TITLE>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</HEAD>
<BODY aLink=#006633 link=#003399 bgColor=#ffffff vLink=#660066>
<table width="786" border="0" bordercolorlight="#409FFF" bordercolordark="#409FFF">
<tr>
<td width="207" align="left" valign="middle">
<font
color="#510028"><b><br>
</b></font><font color="#FFFFFF">
<% if Session("Username")="" then%>&nbsp;
<%else %>
<%=Session("Username")%>
<%end if%>
</font> <font size=-1>
<% if Session("Username")="" then%>
</font><font size=-1>
<br>
<%else%>
<br>
<%end if%>
</font></td>
<td width="339" align="center" valign="middle">
<div align="center">&nbsp;</div>
</td>
<td width="218" align="center" valign="middle"> <font color="#510028">
```



```

In">
</b> </div>
</td>
</tr>
</table>
</form>
<p>&nbsp;</p>
<p>&nbsp;</p>
</div>
<p>&nbsp;</p>
</BODY></HTML>

```

logout.asp

```

<%@ Language=VBScript %>
<%option Explicit%>
<!-- #include file="adovbs.inc" -->
<!-- #include file="connection.asp" -->
<%

```

```

'delete item from cart
dim objRs
Set objRs=Server.CreateObject("ADODB.Recordset")
Set objRs.ActiveConnection=objConn
objRs.CursorType=adOpenDynamic
objRs.LockType=adLockOptimistic
objRs.open "delete*from Cart where Username='"& session("Username") & "'"

```

```

'abandon session
Session.Abandon

```

```

%>
<%

```

```

dim strDate
strDate = Now() %>

```

```

<HTML>
<HEAD>
<TITLE></TITLE>
<META content="text/html; charset=iso-8859-1" http-equiv=Content-Type>
<META name="description" content="shopping cart" >
<meta http-equiv="refresh" content="5; url=index.asp">
</HEAD>
<BODY bgcolor="#FFFFFF" link="#003399">
<DIV align=left> </DIV>
<p align="center">&nbsp;</p>

```

```

<p>&nbsp;</p>
<div id="Layer1" style="position:absolute; width:596px; height:334px; z-index:1; left:
121px; top: 25px">
<h3 align="center">
<h2><font face="Times New Roman, Times, serif"><b><font color="#336600">You
have logged out </font></b></font></h2>
</h3>
<div align="center">
<h3><font face="Times New Roman, Times, serif"><b><font color="#336600">Thank
you for your visit.Please come again</font></b></font></h3>
<font face="Times New Roman, Times, serif"><b><font color="#336600">In a few
moment you will be redirected to our Home page</font></b></font> </div>
<p>&nbsp;</p>
<p align="center">&nbsp;</p>
<p align="center">&nbsp;</p>
<p align="center">&nbsp;</p>
<p align="center">&nbsp;</p>
<p align="center">&nbsp;</p>
</div>
<p>&nbsp;</p>
<div align="center"><font face="Times New Roman, Times, serif" size="-1">
</font><font face="Arial, Helvetica, sans-serif"><br>
</font> </div>
<div align="center"><font face="Arial, Helvetica, sans-serif"><br>
</font> </div>
<p>&nbsp;</p>
<P>

</BODY></HTML>

```

ok.asp

```

<HTML><HEAD><TITLE>Neu Book Store</TITLE>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">

</HEAD>
<BODY aLink=#006633 link=#003399 bgColor=#ffffff vLink=#660066>
<table width="829" border="0" bordercolordark="#409FFF" bordercolor="#409FFF"
height="129" background="ust2006.jpg">
<tr>
<td width="571" valign="bottom">
<div align="center">

```



```

&nbsp;</div>
<font color="#FFFFFF" size="2" face="Verdana"><b>
<p>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</b></font>
</td>
</tr>
</table>
<table width="18%" border="0" bordercolor="#CCE6FF" height="427"
bgcolor="#CCE6FF" bordercolorlight="#CCE6FF" cellspacing="0" cellpadding="0">
<tr bgcolor="#CCE6FF" bordercolor="#CCE6FF">
<td height="387" bgcolor="#6B0321">
<p align="center">
<font color="#FFFFFF" size="2" face="Verdana"><b>
<br>
<br>
</font>
<font color="#FFFFFF" face="Verdana" style="font-size: 9pt">&nbsp;<%=Date%> -
<%=Time%> </font>
<font color="#FFFFFF" size="2" face="Verdana"> <br>
</font>
</b><font face="Verdana" size="2"><a href="index.asp"><br>
<b><font color="#FFFFFF">Home</font></b></a></font></p>
<p align="center">&nbsp;<p>
<p align="center"><font color="#FFFFFF" face="Verdana" size="2"><br>
&nbsp;</font></p>
<p align="center">&nbsp;</p>
<p align="center"><font face="Verdana" size="2">&nbsp;</p>

</font>

<form method=post action=search.asp>

<p align="center"><font face="Verdana" color="#FFFFFF"><b><font
size="2">SEARCH<br>
</font>
</b><font size="2"><br>

Enter ISBN Number:<br>
&nbsp;</font><input name=isbn size="17"><font size="2"><br>
</font>
<input type=submit value=" Search "><font size="2"> </font></font></p>
</form>

</p>
<p align="center">&nbsp;</p>
<p align="center">&nbsp;</p>

```



```

<%
'Return to previous page'
dim strPreviouspage
strPreviouspage=Request.ServerVariables("HTTP_REFERER")
%>

<%
'*****'
'View Cart'
'*****'

dim objRs
Set objRs=Server.CreateObject("ADODB.Recordset")
Set objRs.ActiveConnection=objConn
objRs.CursorType=adOpenDynamic
objRs.LockType=adLockOptimistic
'Connect to cart table'
objRs.Open "select*from Cart where Username='"& session("Username") &

""

%>

<html>
<head>
<title>Shopping Cart</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body bgcolor="#FFFFFF" text="#000000">
<div align="left">
&nbsp;  <div align="center">
<table border="0" width="81%" id="table2" height="782">
<tr>
<td colspan="2" height="139">
</td>
</tr>
<tr>
<td width="21%" bgcolor="#6B0321">&nbsp;  </td>
<td width="77%" valign="top"><font color="#FFFFFF"><font
color="#000000"><b><a href="index.asp">
<font color="#000000">Home</font></a>
<a href="product.asp"><font color="#000000"> Product</font></a></b>
<% if Session("Username")="" then%>
<b>Welcome Guest!</b></font>
<%else %>

```



```

    <%=Session("Username")%>
    <%end if%>
    </font> <font size=-1>
    <% if Session("Username")="" then%>
    <a href="Login.asp"><b><font color="#000000">Log in</font></b></a> /<b><a
href="Registration.asp"><font color="#000000">
        Register</font></a>
    </b>
    <%else%>
    <a href="Logout.asp"><b><font color="#000000">Log
Out</font></b></a></font><br>
        <br>
    &nbsp;</td>
    </tr>
</table>
<hr width="81%" color="#800000" size="3">
</div>
<p><font size=-1>
    <br>
    <%end if%>
</font></p>
</div>
<div id="Layer1" style="position:absolute; width:576px; height:439px; z-index:1; left:
304px; top: 217px">
    <div align="center"><b><font face="Times New Roman, Times, serif" size="3"><a
href="checkOut.asp" style="TEXT-DECORATION: none">
        </a></font></b><font color="#000000"><br>
        <%
            dim oxrs
Set oxrs=Server.CreateObject("ADODB.Recordset")
Set oxrs.ActiveConnection=objConn
oxrs.CursorType=adOpenDynamic
oxrs.LockType=adLockOptimistic
oxrs.Open "select*from customer where Username="&
session("Username") & ""
        %>
    </font> </div>
    <form name=Form1 id=Form1 method=post action="TestCCard.asp">
        <input type="Hidden" name="Type" value="<%=objJrs("ProductID")%>">
    <div align="center">
    <table border="0" width="529" bordercolor="#800000" bgcolor="#800000"
cellspacing="1" >
    <tr bgcolor="#F5F5F5" bordercolor="#CCCCFF">
    <td colspan="2">

```

```

<div align="center"></div>
<div align="center">
<h3>Payment by Credit Card</h3>
</div>
</td>
</tr>
<tr bgcolor="#FEFEFE" bordercolor="#FFFFFF">
<td width="228" height="42" valign="top">
  <div align="center"><b>Charged By:</b></div>
</td>
<td width="294" height="42" valign="top">
<div align="center"><font color="#990033"><b>Near East University</b></font></div>
</td>
</tr>
<tr bgcolor="#FEFEFE" bordercolor="#FFFFFF">
<td colspan="2" width="525">
<div align="center">
<table border="0" width="100%" style="border-left-width: 0px; border-top-width: 0px;
border-bottom-width: 0px" >
<tr>
<td align="left" style="border-left-style: none; border-left-width: medium; border-right-
style: solid; border-right-width: 1px; border-top-style: none; border-top-width: medium;
border-bottom-style: none; border-bottom-width: medium" valign="top"
width="43%"><font face="Verdana"><b>
<span style="font-size: 8pt"><br>
Cardholder's Name:</span></b><span style="font-size: 8pt"><br>
<input type="text" name="txtCname" size="30"></span></font></td>
<td align="left" width="55%"><font face="Verdana"><b>
<span style="font-size: 8pt"><br>
Expiration Date:</span></b><span style="font-size: 8pt">(mm/dd/yyyy)<br>
<input type="text" name="TxtCreditCardExpireDate"></span></font></td>
</tr>
<tr>
<td align="left" style="border-left-style: solid; border-left-width: 1px; border-right-style:
solid; border-right-width: 1px; border-top-style: solid; border-top-width: 1px" valign="top"
colspan="2">&nbsp;<hr></td>
</tr>
<tr>
<td align="left" style="border-left-style: solid; border-left-width: 1px; border-right-style:
solid; border-right-width: 1px; border-bottom-style: solid; border-bottom-width: 1px"
valign="top" width="43%">
<!--#include file=testccard.asp-->
</td>

```

```

<td align="left" width="55%" style="border-left-style: solid; border-left-width: 1px"><font
face="Verdana"><b>
<span style="font-size: 8pt">Type of Card:</span></b><span style="font-size: 8pt">
<br>
<select name="Cardtype">
<option selected>Choose Card</option>
<option value="American Express">American Express</option>
<option value="Master Card">Master Card</option>
<option value="Visa">Visa</option>
</select>
</span></font></td>
</tr>
</table>
</div>
</td>
</tr>
<tr bgcolor="#FEFEFE" bordercolor="#FFFFFF">
<td colspan="2"><font size="2" face="Times New Roman, Times, serif">
Please fill the credit card over and input the 3 or 4 digit <b>CSC
code</b>.<br>
<input type="text" name="csc" size="16" maxlength="4">
</font><hr></td>
</tr>
<tr bgcolor="#FEFEFE" bordercolor="#FFFFFF">
<td colspan="2">
<div align="center"> </div>
</td>
</tr>
<tr bgcolor="#FEFEFE" bordercolor="#FFFFFF">
<td width="228">
<p><b>Name:</b></p>
</td>
<td width="294">
<p><b>
<input type="text" name="txtFname" size="30" value="<%=oxRs("firstname")%>">
</b></p>
</td>
</tr>
<tr bgcolor="#FEFEFE" bordercolor="#FFFFFF">
<td width="228">
<p><b>Address:</b></p>
</td>
<td width="294">
<p><b>
<input type="text" name="TxtAddress" size="30" value="<%=oxrs("Address")%>">

```



```

        </b></p>
    </td>
</tr>
<tr bgcolor="#FEFEFE" bordercolor="#FFFFFF">
<td colspan="2" height="23">
    <div align="center"><b><br>
    <input type="submit" name="Confirm" value="Submit Payment">
    </b></div>
</td>
</tr>
</table>
</div>

<h3 align="center"><font color="#510028"><b><font color="#000000">
</font></b></font></h3>
</div>
<p>&nbsp; </p>
</body>
</html>

```

processingorder.asp

```

<%@ Language=VBScript %>
<%option Explicit%>
<!-- #include file="adovbs.inc" -->
<!-- #include file="connection.asp" -->
<%

*****
'Recording of data in order and order detail; table
*****

'open connection

dim oxrs
Set oxrs=Server.CreateObject("ADODB.Recordset")
Set oxrs.ActiveConnection=objConn
oxrs.CursorType=adOpenDynamic
oxrs.LockType=adLockOptimistic

dim objrs
Set objrs=Server.CreateObject("ADODB.Recordset")
Set objrs.ActiveConnection=objConn
objrs.CursorType=adOpenDynamic
objrs.LockType=adLockOptimistic

```

```

dim obvrs
Set obvrs=Server.CreateObject("ADODB.Recordset")
Set obvrs.ActiveConnection=objConn
obvrs.CursorType=adOpenDynamic
obvrs.LockType=adLockOptimistic
dim ors,ko
Set ors=Server.CreateObject("ADODB.Recordset")
Set ors.ActiveConnection=objConn
ors.CursorType=adOpenDynamic
ors.LockType=adLockOptimistic
' Record details in order table, some details are taken from customer table

oxrs.Open "select*from customer where Username='" & session("Username") & "'"
objrs.Open "select* from Orders",objconn
objrs.AddNew
objrs("CustomerID")=oxrs("CustomerID")
objrs("Address")=Request.Form("txtAddress")
objrs("Country")=Request.Form("txtCountry")
objrs("OrderStatus")="UNDELIVERED"
objrs("orderDate")=now()
objrs("LatestDelivery")=now()+30

objrs("Firstname")=Request.Form("txtfname")
objrs("Lastname")=Request.Form("txtLname")
objrs.Update

'Take information from cart table and record in order detail table
obvrs.Open"select * from cart where Username='" & session("Username") & "'"
obvrs.MoveFirst
ors.Open"select * from OrderDetail",objconn
do while not obvrs.EOF
ors.AddNew
ors("ProductID")=obvrs("ProductID")
ors("Quantity")=obvrs("Quantity")
ors("OrderID")=objrs("OrderID")
ors.Update
obvrs.MoveNext
Loop
response.Redirect "confirm.asp"

%>
<html>
<head>
<title>Shopping Cart</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

```

```

<meta http-equiv="refresh" content="10; url=Confirm.asp">
</head>
<body bgcolor="#FFFFFF">
<div align="center"><br>
<br>
<table id="AutoNumber5" style="BORDER-COLLAPSE: collapse"
bordercolor="#111111" cellspacing="0" cellpadding="0" width="80%" border="0"
align="center">
<tr>
<td width="100%">
<center>
</center>
</td>
</tr>
</table>
<br>
</div>
</body>
</html>

```

product.asp

```

<!-- #include file="adovbs.inc" -->
<!-- #include file="connection.asp" -->
<%

```

```

cid=request.querystring("categori_id")

```

```

'codes to display products on different pages
DIM mySQL, objRS,b

```

```

mySQL = "SELECT * FROM Product where c_id like '"&cid&"'"
Set objRS = Server.CreateObject("ADODB.Recordset")
objRS.CursorType = 1
objRS.Open mySQL, objConn

```

```

DIM intPageRecords, intRecords, intRecordCount, intCurrentPage
DIM intNumberOfPages, intDisplayPage
intPageRecords = Request.QueryString("page")
IF intPageRecords = "" THEN intPageRecords = 1 : intRecords = 1
intRecords = intPageRecords
intPageRecords = ((intPageRecords - 1) * 4) + 1
intRecordCount = 0

```


%>

<html>

<head>

<title>Shopping cart</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

</head>

<body bgcolor="#FFFFFF" text="#000000">

<table border="0" width="76%" id="table1" cellspacing="0" cellpadding="0" height="899" bordercolor="#800000">

<tr>

<td colspan="2" height="133">

</td>

</tr>

<tr>

<td width="21%" bgcolor="#6B0321" valign="top">

<p align="center">

 <%=Date%> -
<%=Time%>

Home</p>

<p align="center">

<p align="center"><font
size="2">SEARCH

<form method=post action=searchisbn.asp>

<p align="center">Enter ISBN Number:

 <input name=isbn size="17"><font
size="2">

<input type=submit value=" Search "> </p>

</p>

<hr width="90%" size="1"></form>

<form method=post action=searchtitle.asp>

```

<p align="center"><font face="Verdana" color="#FFFFFF"><font size="2">Title:<br>
&nbsp;</font><font face="Verdana"><input name=title size="17"></font><font
size="2"><br>
</font>
<font face="Verdana">
<input type=submit value=" Search "></font><font size="2"> </font></font></p>
<hr width="90%" size="1"></form>
<form method=post action=searchauthor.asp>
<p align="center"><font face="Verdana" color="#FFFFFF"><font size="2">Author
Name:<br>
&nbsp;</font><font face="Verdana"><input name=author size="17"></font><font
size="2"><br>
</font>
<font face="Verdana">
<input type=submit value=" Search "></font><font size="2"> </font></font></p>
<hr width="90%" size="1">
</form>
</p>
<p align="center">&nbsp;</p>
<p>&nbsp;</td>
<td width="79%" valign="top" style="border-right-style: solid; border-right-width: 1px">
<font face="Verdana"><font size="2"><br>
</font><font size="2" color="#FFFFFF"><font color="#000000"><b>&nbsp;&nbsp;&nbsp;<a
href="index.asp">
Home</a></b>
<% if Session("Username")="" then%>
<b>Welcome Guest!</b></font>
<%else %>
<%=Session("Username")%>
<%end if%>
</font> <font size=-1>
<% if Session("Username")="" then%>
<a href="Login.asp"><b><font color="#000000">Log in</font></b></a> /<b><a
href="Registration.asp"><font color="#000000"> Register</font></a>
</b>
<%else%>
<a href="Logout.asp"><b><font color="#000000">Log Out</font></b></a> / <a
href="viewcart.asp"><b>
<font color="#000000">View Cart</font></b></a>
/ </font><font color="#000000"><a href="checkout.asp"><b><font color="#000000">
<font size="2">Checkout</font></b></a></font></b></a></font><font size="2"><br>
</font><font size=-1><br>
<%end if%>
<br>
</font>

```

```

</font>
<p>&nbsp;</td>
</tr>
</table>
<hr width="81%" color="#800000" size="4" align="left">
<div id="Layer1" style="position:absolute; width:591px; height:139px; z-index:1; left:
188px; top: 239px; background-color: #FFFFFF; layer-background-color: #FFFFFF;
border: 1px none #000000">
  <p align="left">&nbsp;</p>
  <font face="Verdana" size="2">
    <%
IF NOT objRS.EOF THEN
objRS.Move (intPageRecords - 1)
DO WHILE intRecordCount < 3 and NOT objRS.EOF
%>
    </font>
    <table width="75%" border="1" align="center" bgcolor="#FFFFFF"
bordercolor="#FFFFFF">
<form method="post" action="Shoppingcart.asp">
<tr>
<td width="43%">
<div align="center">
<font face="Verdana" size="2">
" width="115" height="135"></font></div>
</td>
<td width="53%"><b><font size="2" face="Verdana"><%=objRs("description")%>
<br>Author:</b><%=objRs("author")%>

</font> </b><font size="2" face="Verdana"><br>
</font>
<p><font face="Verdana"
size="2"><b>Price:&nbsp;&nbsp;&nbsp;<%=objRs("Price")%></b><br>
</font>
<b>
<font face="Verdana">
<input type="hidden" name="txtproductID" value="<%=objRs("ProductID")%>">
<font size="2">
<br>
</font>
<select name="txtQuantity">
  <option value="1">1</option>
  <option value="2">2</option>

```



```

<option value="3">3</option>
<option value="4">4</option>
<option value="5">5</option>
<option value="6">6</option>
<option value="7">7</option>
<option value="8">8</option>
<option value="9">9</option>
<option value="10">10</option>
</select><font size="2">
<br>
<br>
</font>
<input type="submit" name="Submit" value="Buy It"><font size="2">
</font></font>

</b></p>
</td>
</tr>
<tr>
<td colspan="2">
<hr width="90%">
</td>
</tr>
</form>
<%
objRS.MoveNext
intRecordCount = intRecordCount + 1
Loop
END IF%>
</table>
<p align="center"><font face="Verdana" size="2"><b>
<%count no of pages%>
<%=intPageRecords%> - <%=intPageRecords+(intRecordCount-1)%> of
<%=objRS.RecordCount%>
Products </b>
</font>
<p align="center"><font face="Verdana" size="2"><b>Scroll Through More Products
<%
intCurrentPage = Request.QueryString("page")
IF intCurrentPage = "" THEN intCurrentPage = 1
intNumberOfPages = int(objRS.RecordCount \ 3)
IF objRS.RecordCount MOD 3 <> 0 THEN intNumberOfPages = intNumberOfPages + 1
Response.Write("Pages: [")
FOR intDisplayPage = 1 TO intNumberOfPages
IF Cint(intDisplayPage) = Cint(intCurrentPage) THEN
Response.Write " <b>" & intDisplayPage & "</b> "

```



```

        isLastname=true
    LB.style.display="none"
end if
end Function

```

```

Function clearAll()
    LA.style.display="none"
    LB.style.display="none"
end function

```

```

Function isPhone()
if len(regFrm.txtPhone.value)=7 then
    isPhone=true
else
    isPhone=False
end if
end function

```

```

Function isUsername()
if len(regFrm.txtUsername.value)<5 then
    isUsername=false
else
    isUsername=True
end if
end function

```

```

Function isPasswordvalid()
If len(regFrm.txtPassword.value)<5 then
    isPasswordvalid=false
else if (regFrm.txtPassword.value) <> (regFrm.txtcPassword.value) then
    isPasswordvalid=False
else
    isPasswordvalid=True
end if
end if

```

```

end function

```

```

Function regFrm_onSubmit()
    regFrm_onSubmit=true
if not isFirstname() then
    regFrm_onSubmit=false
    MsgBox "Invalid First Name"
    regFrm.txtFirstName.Focus()

```



```

        LA.style.display=""
    end if
    if not isLastname() then
        regFrm_onSubmit=false
        MsgBox"Invalid Last Name"
        regFrm.txtLastName.Focus()
        LB.style.display=""
    end if

    if not isUsername() then
        regFrm_onSubmit=false
        MsgBox"Invalid Username!At least 5 character."
        regFrm.txtUsername.Focus()
    end if

    if not isPhone() then
        regFrm_onSubmit=false
        MsgBox"Invalid Phone number!At least 7 digits"
        regFrm.txtPhone.Focus()
    end if
    if not isPasswordvalid() then
        regFrm_onSubmit=false
        MsgBox"Invalid Password.At least 5 character."
        regFrm.txtPassword.Focus()
    end if
end Function

-->
</SCRIPT>

</HEAD>
<BODY onload=clearAll() vLink=#660066 aLink=#006633 link=#003399
bgcolor="#FFFFFF" >
<table width="786" border="0">
<tr>
<td width="207" align="left" valign="middle">
    <h2><font color="#510028"><b>Register</b></font></h2>
</td>
<td width="339" valign="middle" align="center">
    <div align="center"><font face="Arial, Helvetica, sans-serif"><b><font
color="#FFFFFF"><font color="#000000">
<% if Session("Username")="" then%>
Welcome Guest!</font></font>
<%else %>
<%=Session("Username")%>

```



```

        <div align="center">&nbsp; &nbsp;
        <input type="submit" name="btnSubmit" value="Submit">
        <input type="reset" name="btnreset" value="Reset">
        &nbsp; </div>
    </td>
</tr>
</table>
</form>
<p>&nbsp;</p>
</div>
<table width="18%" border="1" bgcolor="#800000" bordercolor="#CCE6FF"
height="506" bordercolorlight="#800000" bordercolordark="#800000">
    <tr>
        <td height="500">&nbsp;</td>
    </tr>
</table>
<p>&nbsp;</p>
</BODY></HTML>

```

registration.asp

```

<%@ Language=VBScript %>
<%option Explicit%>
<%

%>

<HTML><HEAD><TITLE>Registration</TITLE>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<SCRIPT LANGUAGE=VBscript>
<!--
Function isFirstname()
    if len(regFrm.txtFirstName.value)<1 then
        isFirstname=False

    else
        isFirstname=true
        LA.style.display="none"
    end if
end function

Function isLastname()
    if len(regFrm.txtLastName.value)<1 then
        isLastname=False

```

```

    else
        isLastname=true
        LB.style.display="none"
    end if
end Function

```

```

Function clearAll()
    LA.style.display="none"

    LB.style.display="none"

end function

```

```

Function isPhone()
    if len(regFrm.txtPhone.value)=7 then
        isPhone=true
    else
        isPhone=False
    end if
end function

```

```

Function isUsername()
    if len(regFrm.txtUsername.value)<5 then
        isUsername=false
    else
        isUsername=True
    end if
end function

```

```

Function isPasswordvalid()
    If len(regFrm.txtPassword.value)<5 then
        isPasswordvalid=false
    else if (regFrm.txtPassword.value)<>(regFrm.txtcPassword.value) then
        isPasswordvalid=False
    else
        isPasswordvalid=True
    end if
end if

end function

```

```

Function regFrm_onSubmit()
    regFrm_onSubmit=true
    if not isFirstname() then

```



```

        regFrm_onSubmit=false
        MsgBox"Invalid First Name"
        regFrm.txtFirstName.Focus()
        LA.style.display=""
    end if
    if not isLastname() then
        regFrm_onSubmit=false
        MsgBox"Invalid Last Name"
        regFrm.txtLastName.Focus()
        LB.style.display=""
    end if

    if not isUsername() then
        regFrm_onSubmit=false
        MsgBox"Invalid Username!At least 5 characters."
        regFrm.txtUsername.Focus()
    end if

    if not isPhone() then
        regFrm_onSubmit=false
        MsgBox"Invalid Phone number!At least 7 digits."
        regFrm.txtPhone.Focus()
    end if
    if not isPasswordvalid() then
        regFrm_onSubmit=false
        MsgBox"Invalid Password!At least 5 characters."
        regFrm.txtPassword.Focus()
    end if
end Function

-->
</SCRIPT>

</HEAD>
<BODY onload=clearAll() vLink=#660066 aLink=#006633 link=#003399
bgcolor="#FFFFFF" >
<table width="786" border="0" bordercolorlight="#409FFF" bordercolordark="#409FFF">
<tr>
<td width="207" align="left" valign="middle" height="41">
<h2><font color="#510028"><b>Registration</b></font></h2>
</td>
<td width="339" valign="middle" align="center" height="41">
<div align="center"><font face="Arial, Helvetica, sans-serif"><b><font
color="#FFFFFF"><font color="#000000">
<% if Session("Username")="" then%>

```

```

Welcome Guest!</font></font>
<%else %>
<%=Session("Username")%>
<%end if%>
<font size=-1>
<% if Session("Username")="" then%>
<a href="Login.asp"><font color="#000000">Log in</font></a><br>
<%else%>
<a href="Logout.asp"><font color="#000000">Log Out</font></a></font>
<%end if%>
/ <a href="index.asp"><font color="#000000">Home</font></a> / <a
href="product.asp">
<font color="#000000">Products</font></a></b></font></div>
</td>
<td width="218" align="center" valign="middle" height="41">
<div align="right"><font size="1" color="#510028"><b><%=Date%>,<br>
<%=Time%> </b></font></div>
</td>
</tr>
</table>
<table width="18%" border="0">
<tr bgcolor="#CCE6FF" bordercolor="#CCE6FF">
<td height="387" bgcolor="#510028">
<p align="center">&nbsp;</p>
<p align="center">&nbsp;</p>
<p align="center">&nbsp;</p>
<p align="center">&nbsp;</p>
<p align="center">&nbsp;</p>
</td>
</tr>
</table>
<div id="Layer1" style="position:absolute; width:613px; height:391px; z-index:1; left:
206px; top: 78px">
<form method="post" action="addcustomer.asp" name="regFrm">
<table width="98%" border="0" height="652">
<tr>
<td colspan="4">
<div align="center"><font face="Arial, Helvetica, sans-serif" size="5"><b><font
color="#510028"><br>
REGISTRATION FORM </font></b></font></div>
</td>
</tr>
<tr>
<td colspan="4">
<hr>

```



```

DIM intPageRecords, intRecords, intRecordCount, intCurrentPage
DIM intNumberOfPages, intDisplayPage
intPageRecords = Request.QueryString("page")
IF intPageRecords = "" THEN intPageRecords = 1 : intRecords = 1
intRecords = intPageRecords
intPageRecords = ((intPageRecords - 1) * 4) + 1
intRecordCount = 0

%>

<html>
<head>
<title>Shopping cart</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body bgcolor="#FFFFFF" text="#000000">

<div id="Layer1" style="position:absolute; width:504px; height:139px; z-index:1; left:
228px; top: 321px; background-color: #FFFFFF; layer-background-color: #FFFFFF;
border: 1px none #000000">
  <p align="left">&nbsp;</p>
  <%
IF NOT objRS.EOF THEN
objRS.Move (intPageRecords - 1)
DO WHILE intRecordCount < 3 and NOT objRS.EOF
%>
  <table width="75%" border="1" align="center" bgcolor="#FFFFFF"
bordercolor="#FFFFFF">
    <form method="post" action="Shoppingcart.asp">
      <tr>
        <td width="66%">
          <div align="center">
            " width="115"
height="135"></div>
          </td>
          <td width="34%"><b><%=objRs("description")%> </b><br>
            <p><b>Price:&nbsp;&nbsp;&nbsp;$&nbsp;&nbsp;&nbsp;<%=objRs("Price")%></b><br>
            <b>
              <input type="hidden" name="txtproductID" value="<%=objRs("ProductID")%>">
            <br>
            <select name="txtQuantity">
              <option value="1">1</option>
              <option value="2">2</option>

```



```

        <option value="3">3</option>
        <option value="4">4</option>
        <option value="5">5</option>
        <option value="6">6</option>
        <option value="7">7</option>
        <option value="8">8</option>
        <option value="9">9</option>
        <option value="10">10</option>
    </select>
    <br>
    <br>
    <input type="submit" name="Submit" value="Buy It">
</b></p>
</td>
</tr>
<tr>
    <td colspan="2">
        <hr>
    </td>
</tr>
</form>
<%
objRS.MoveNext
intRecordCount = intRecordCount + 1
Loop
END IF%>
</table>
<p align="center">
    <br>
    <p align="center">&nbsp;</p>
</div>
<table border="0" width="78%" id="table1" height="479" cellspacing="0"
cellpadding="0">
    <tr>
        <td rowspan="2" bgcolor="#6B0321">&nbsp;</td>
        <td height="47" width="580" style="border-right-style: solid; border-right-
width: 1px" bordercolor="#6B0321">
            <p align="center">
                <font color="#FFFFFF"><font color="#000000"><b><a href="index.asp">
                <font color="#000000">Home</font></a></b>
                <% if Session("Username")="" then%>
                <b>Welcome Guest!</b></font>
                <%else %>
                <%=Session("Username")%>
                <%end if%>

```



```
objRS.CursorType = 1
objRS.Open mySQL, objConn
```

```
DIM intPageRecords, intRecords, intRecordCount, intCurrentPage
DIM intNumberOfPages, intDisplayPage
intPageRecords = Request.QueryString("page")
IF intPageRecords = "" THEN intPageRecords = 1 : intRecords = 1
intRecords = intPageRecords
intPageRecords = ((intPageRecords - 1) * 4) + 1
intRecordCount = 0
```

```
%>
```

```
<html>
<head>
<title>Shopping cart</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
```

```
<body bgcolor="#FFFFFF" text="#000000">

<div id="Layer1" style="position:absolute; width:504px; height:139px; z-index:1; left:
228px; top: 321px; background-color: #FFFFFF; layer-background-color: #FFFFFF;
border: 1px none #000000">
  <p align="left">&nbsp;</p>
  <%
```

```
IF NOT objRS.EOF THEN
objRS.Move (intPageRecords - 1)
DO WHILE intRecordCount < 3 and NOT objRS.EOF
  %>
```

```
  <table width="86%" border="1" align="center" bgcolor="#FFFFFF"
bordercolor="#FFFFFF">
    <form method="post" action="Shoppingcart.asp">
      <tr>
        <td width="40%">
          <div align="center">
            " width="115"
height="135"></div>
          </td>
          <td width="56%"><b><%=objRs("description")%> </b><br><%=objRs("author")%>
          <p><b>Price:&nbsp;&nbsp;&nbsp;$&nbsp;&nbsp;&nbsp;<%=objRs("Price")%></b><br>
          <b>
            <input type="hidden" name="txtproductID" value="<%=objRs("ProductID")%>">
            <br>
            <select name="txtQuantity">
```



```

        <option value="1">1</option>
        <option value="2">2</option>
        <option value="3">3</option>
        <option value="4">4</option>
        <option value="5">5</option>
        <option value="6">6</option>
        <option value="7">7</option>
        <option value="8">8</option>
        <option value="9">9</option>
        <option value="10">10</option>
    </select>
    <br>
    <br>
    <input type="submit" name="Submit" value="Buy It">
</b></p>
</td>
</tr>
<tr>
    <td colspan="2">
        <hr>
    </td>
</tr>
</form>
<%
objRS.MoveNext
intRecordCount = intRecordCount + 1
Loop
END IF%>
</table>
<p align="center">
    <br>
    <p align="center">&nbsp;</p>
</div>
<table border="0" width="78%" id="table1" height="479" cellpadding="0" cellspacing="0">
    <tr>
        <td rowspan="2" bgcolor="#6B0321">&nbsp;</td>
        <td height="47" width="580" style="border-right-style: solid; border-right-
width: 1px" bgcolor="#6B0321">
            <p align="center">
                <font color="#FFFFFF"><font color="#000000"><b><a href="index.asp">
                <font color="#000000">Home</font></a></b>
                <% if Session("Username")="" then%>
                <b>Welcome Guest!</b></font>
                <%else %>

```



```

mysql = "SELECT * FROM Product where ProductID like '" & isbn & "'"
Set objRS = Server.CreateObject("ADODB.Recordset")
objRS.CursorType = 1
objRS.Open mysql, objConn

```

```

DIM intPageRecords, intRecords, intRecordCount, intCurrentPage
DIM intNumberOfPages, intDisplayPage
intPageRecords = Request.QueryString("page")
IF intPageRecords = "" THEN intPageRecords = 1 : intRecords = 1
intRecords = intPageRecords
intPageRecords = ((intPageRecords - 1) * 4) + 1
intRecordCount = 0

```

```
%>
```

```

<html>
<head>
<title>Shopping cart</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

```

```

<body bgcolor="#FFFFFF" text="#000000">

<div id="Layer1" style="position:absolute; width:504px; height:139px; z-index:1; left:
228px; top: 321px; background-color: #FFFFFF; layer-background-color: #FFFFFF;
border: 1px none #000000">
  <p align="left">&nbsp;</p>
  <%

```

```

IF NOT objRS.EOF THEN
objRS.Move (intPageRecords - 1)
DO WHILE intRecordCount < 3 and NOT objRS.EOF
%>

```

```

  <table width="75%" border="1" align="center" bgcolor="#FFFFFF"
bordercolor="#FFFFFF">
    <form method="post" action="Shoppingcart.asp">
      <tr>
        <td width="66%">
          <div align="center">
            " width="115"
height="135"></div>
          </td>
          <td width="34%"><b><%=objRs("description")%> </b><br>
            <p><b>Price:&nbsp;<%=objRs("Price")%></b><br>
              <b>
                <input type="hidden" name="txtproductID" value="<%=objRs("ProductID")%>">

```



```

<br>
<select name="txtQuantity">
  <option value="1">1</option>
  <option value="2">2</option>
  <option value="3">3</option>
  <option value="4">4</option>
  <option value="5">5</option>
  <option value="6">6</option>
  <option value="7">7</option>
  <option value="8">8</option>
  <option value="9">9</option>
  <option value="10">10</option>
</select>
<br>
<br>
<input type="submit" name="Submit" value="Buy It">
</b></p>
</td>
</tr>
<tr>
  <td colspan="2">
    <hr>
  </td>
</tr>
</form>
<%
objRS.MoveNext
intRecordCount = intRecordCount + 1
Loop
END IF%>
</table>
<p align="center">
  <br>
  <p align="center">&nbsp;</p>
</div>
<table border="0" width="78%" id="table1" height="479" cellpadding="0" cellspacing="0">
  <tr>
    <td rowspan="2" bgcolor="#6B0321">&nbsp;</td>
    <td height="47" width="580" style="border-right-style: solid; border-right-width: 1px" bordercolor="#6B0321">
      <p align="center">
        <font color="#FFFFFF"><font color="#000000"><b><a href="index.asp">
          <font color="#000000">Home</font></a></b>
        <% if Session("Username")="" then%>

```



```
DIM mySQL, objRS,b
```

```
mySQL = "SELECT * FROM Product where description like '%" & title & "%"  
Set objRS = Server.CreateObject("ADODB.Recordset")  
objRS.CursorType = 1  
objRS.Open mySQL, objConn
```

```
DIM intPageRecords, intRecords, intRecordCount, intCurrentPage  
DIM intNumberOfPages, intDisplayPage  
intPageRecords = Request.QueryString("page")  
IF intPageRecords = "" THEN intPageRecords = 1 : intRecords = 1  
intRecords = intPageRecords  
intPageRecords = ((intPageRecords - 1) * 4) + 1  
intRecordCount = 0
```

```
%>
```

```
<html>
```

```
<head>
```

```
<title>Shopping cart</title>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
```

```
</head>
```

```
<body bgcolor="#FFFFFF" text="#000000">
```

```

```

```
<div id="Layer1" style="position:absolute; width:504px; height:139px; z-index:1; left:  
228px; top: 321px; background-color: #FFFFFF; layer-background-color: #FFFFFF;  
border: 1px none #000000">
```

```
<p align="left">&nbsp;</p>
```

```
<%
```

```
IF NOT objRS.EOF THEN
```

```
objRS.Move (intPageRecords - 1)
```

```
DO WHILE intRecordCount < 3 and NOT objRS.EOF
```

```
%>
```

```
<table width="75%" border="1" align="center" bgcolor="#FFFFFF"
```

```
bordercolor="#FFFFFF">
```

```
<form method="post" action="Shoppingcart.asp">
```

```
<tr>
```

```
<td width="66%">
```

```
<div align="center">
```

```
" width="115"
```

```
height="135"></div>
```

```
</td>
```

```
<td width="34%"><b><%=objRs("description")%> </b><br>
```

```
<p><b>Price:&nbsp; $&nbsp; <%=objRs("Price")%></b><br>
```



```

<b>
<input type="hidden" name="txtproductID" value="<%=objRs("ProductID")%>">
<br>
<select name="txtQuantity">
  <option value="1">1</option>
  <option value="2">2</option>
  <option value="3">3</option>
  <option value="4">4</option>
  <option value="5">5</option>
  <option value="6">6</option>
  <option value="7">7</option>
  <option value="8">8</option>
  <option value="9">9</option>
  <option value="10">10</option>
</select>
<br>
<br>
<input type="submit" name="Submit" value="Buy It">
</b></p>
</td>
</tr>
<tr>
  <td colspan="2">
    <hr>
  </td>
</tr>
</form>
<%
objRS.MoveNext
intRecordCount = intRecordCount + 1
Loop
END IF%>
</table>
<p align="center">
  <br>
  <p align="center">&nbsp;</p>
</div>
<table border="0" width="78%" id="table1" height="479" cellspacing="0"
cellpadding="0">
  <tr>
    <td rowspan="2" bgcolor="#6B0321">&nbsp;</td>
    <td height="47" width="580" style="border-right-style: solid; border-right-
width: 1px" bordercolor="#6B0321">
      <p align="center">
        <font color="#FFFFFF"><font color="#000000"><b><a href="index.asp">

```



```

"session invalid"
if session("Username")="" then
Response.Redirect "Login.asp"
end if
%>
<%
'Return to previous page'
dim strPreviouspage
strPreviouspage=Request.ServerVariables("HTTP_REFERER")
%>
<%

*****
'Add to cart'
*****

dim id,sconn,cconnect,strQuantity,quant,sqlquant,strQuantity1,strqty
id = Request.Form("txtproductID")
strQuantity1=Request.Form("txtQuantity")
Set scconn=Server.CreateObject("ADODB.Recordset")
Set scconn.ActiveConnection=objConn
scconn.Open "select * from Product where ProductID='"&id&'"
quant=scconn("ProductID")

'adding to cart'

Set cconnect=Server.CreateObject("ADODB.Recordset")
Set cconnect.ActiveConnection=objConn
ccconnect.CursorType=adOpenDynamic
ccconnect.LockType=adLockOptimistic

'record in Cart table

ccconnect.Open "select*from Cart",objConn
ccconnect.Find "ProductID='"&quant&'"
if not cconnect.EOF then
dim quanp
quanp=ccconnect("Quantity")
strQuantity=quanp+Request.Form("txtQuantity")

ccconnect("ProductID")=scconn("ProductID")
ccconnect("Description")=scconn("Description")
ccconnect("Price")=scconn("Price")*strQuantity
ccconnect("Uprice")=scconn("Price")
ccconnect("Username")=Session("Username")

```



```

cconnect("Quantity")=cint (strQuantity)
cconnect.Update

else

cconnect.AddNew
cconnect("ProductID")=sconn("ProductID")
cconnect("Description")=sconn("Description")
cconnect("Price")=sconn("Price")*strQuantity1
cconnect("Uprice")=sconn("Price")
cconnect("Username")=Session("Username")
cconnect("Quantity")=cint (strQuantity1)
cconnect.Update
end if

%>

<%
'*****'
'View Cart'
'*****'

dim objRs
Set objRs=Server.CreateObject("ADODB.Recordset")
Set objRs.ActiveConnection=objConn
objRs.CursorType=adOpenDynamic
objRs.LockType=adLockOptimistic

'Connect to cart table'

objRs.Open "select*from Cart where Username='" & session("Username") & "'"
strqty=objRs("Quantity")
%>

<html>
<head>
<title>Shopping Cart</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body bgcolor="#FFFFFF" text="#000000">
<div align="left">
<table width="786" border="0">
<tr>
<td width="207" valign="middle">

```

```

<h2><font color="#510028"><b>Shopping Cart</b></font></h2>
</td>
<td width="339" valign="middle" align="center">
  <div align="center">&nbsp;</div>
</td>
<td width="218" align="center" valign="middle">
  <div align="right"><font color="#510028" size="1"><b><%=Date%>,<br>
    <%=Time%> </b></font></div>
</td>
</tr>
</table>
<p><font color="#FFFFFF"><font color="#000000">
  <%= if Session("Username")="" then%>
  <b>Welcome Guest!</b></font>
  <%=else %>
  <%=Session("Username")%>
  </font>
  <%=end if%>
  <font size=-1>
  <%= if Session("Username")="" then%>
  <a href="Login.asp"><b><font color="#000000">Log in</font></b></a> /<b><a
href="Registration.asp"><font color="#000000"> Register</font></a>
  </b>
  <%=else%>
  <a href="Logout.asp"><b><font color="#000000">Log Out</font></b></a> / <a
href="viewcart.asp"><b>
    <font color="#000000">View Cart</font></b></a><br>
  <%=end if%>
  </font></p>
<p>&nbsp;</p>
</div>
<div id="Layer1" style="position:absolute; width:606px; height:139px; z-index:1; left:
141px; top: 118px">
<div align="center">
<p><b><font size="3" face="Times New Roman, Times, serif"><a href="index.asp">
<font color="#000000">Home</font></a>
/ <a href="<%=strPreviouspage%>"><font color="#000000">Back to
shopping</font></a> </font></a href="checkout.asp" style="TEXT-DECORATION:
none"><font color="#000000">
</font>
<font size="3" face="Times New Roman, Times, serif"><font
color="#000000">Checkout</font></font></a></b>
</p>
</div>
<h3 align="center"><font color="#510028"><b><font color="#000000">The Item has

```

```

been added to your cart.</font><br>
<font color="#000000">View Cart Below:</font></b></font><br>
<font color="#000000">
<% ' check record set if EOF Display No item in Cart '%>
<% if objRs.EOF then %>
<b>No item in cart</b>
<%else%>
<%
'count number of item and display all'

dim numItems
numItems=objRs.RecordCount
if numItems= 0 then
%>
<b>No item in cart</b>
<% else%>
</font> </h3>
<p align="center"><font color="#510028"><b>Your Cart contain the following items:</b>
</font>
<form method="post" action="updateCart.asp">
<div align="center">
<table width="100%" border="1" bordercolor="#800000" bgcolor="#FFFFFF"
bordercolorlight="#800000" bordercolordark="#800000">
<tr bordercolor="#FFCC66">
<td width="26%">
<div align="center"><b>Quantity</b></div>
</td>
<td width="34%">
<div align="center"><b>Products</b></div>
</td>
<td width="15%">
<div align="center"><b>Unit Price</b></div>
</td>
<td width="15%">
<div align="center"><b>Price</b></div>
</td>
<td bordercolor="#FFFFFF" width="10%">&nbsp;</td>
<td bordercolor="#FFFFFF" width="10%">&nbsp;</td>
</tr>
<% objRs.MoveFirst%>
<%
'Function to calculate total amount Purchased'

dim totalAmount
totalAmount=0

```


do while not objRS.EOF

```
dim currentAmount,qty
qty=objRs("Quantity")
currentAmount=objRs("Price")
totalAmount=totalAmount + currentAmount
Session("totalAmount")=totalAmount
%>
<% 'display all information in table using shortcut direct from database%>
<tr bordercolor="#FFCC66">
<td width="26%">
  <div align="center">
    <input type="text" name="<%=objRs("ProductID")%>" size="2"
value="<%=objRs("Quantity")%>">
    <input type="hidden" name="txtproductID" value="<%=objRs("ProductID")%>">
  </div>
</td>
<td width="34%">
  <div align="left"></div>
  <%=objRs("Description")%></td>
<td width="15%">
  <div align="center">$ <%=objRs("UpPrice")%></div>
</td>
<td width="15%">
  <div align="center">$ <%=objRs("Price")%></div>
</td>
<td bordercolor="#FFCC99" colspan="2">
  <div align="center"> <a href="delete.asp?ID=<%=objRs("CartID")%>&link=del"
onClick="confirm(' DELETE Item from Cart ?)'style="TEXT-DECORATION:
none"><b>
    <font face="Times New Roman, Times, serif" size="3"
color="#CC0000">Delete</font><b><a><font color="#CC0000">
    </font>
  </div>
</td>
</tr>
<% objRs.MoveNext
loop
%>
<tr valign="bottom" bordercolor="#FFCC66">
<td colspan="3" height="59" valign="bottom">
<div align="center">
  <h3 align="center"><font color="#CC0000"><b>Total Amount</b></font></h3>
</div>
```

```

<div align="center"> </div>
</td>
<td height="59" valign="middle" bordercolor="#FFFFFF" width="15%">
<div align="center"><font color="#CC0000"><b>$
<%Response.Write totalAmount%>
</b> </font></div>
</td>
<td height="59" valign="middle" bordercolor="#FFFFFF" width="10%">&nbsp;</td>
<td height="59" valign="middle" bordercolor="#FFFFFF" width="10%">&nbsp;</td>
</table>
</div>
<div align="center"><br>
<input type="submit" name="Submit" value="Recalculate Total">
</div>
</form>
<% end if%> <% end if%>
<h3 align="center"><font color="#510028"><b><font color="#000000">
</font></b></font></h3>
</div>
<p>&nbsp;</p>
</body>
</html>

```

testccard.asp

```

<!-- #INCLUDE file="CCard_Inc.asp" -->
<%
    Dim m_sCardNumber
    Dim m_lCardType
    Dim m_sMessage

    m_sMessage = "&nbsp;"
    m_sCardNumber = Request.Form("fldCCardNumber")
    m_lCardType = Request.Form("fldCardType")

    If Trim(m_sCardNumber) <> "" And Trim(m_lCardType) <> "" And
    IsNumeric(m_lCardType) Then
        If CBool(ValidateCreditCard(m_lCardType, m_sCardNumber)) Then
            response.redirect("ok.asp")
        Else
            response.redirect("payment.asp")
        End If
    End If

```

```

%>
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</HEAD>
<BODY>
<form name=Form1 id=Form1 method=post action="TestCCard.asp">
<p align="left">Pick Card Type:<br>
&nbsp;<select name="fldCardType" id="fldCardType">
    <option value="0">MasterCard</option>
    <option value="1">Visa</option>
    <option value="2">American Express</option>
    <option value="3">Discover</option>
</select>
<BR>Enter Credit Card Number:<br>
&nbsp;<INPUT type="text" id=fldCCardNumber name=fldCCardNumber maxlength=16>
</p>

</BODY>
</HTML>

```

updatecart.asp

```

<%@ Language=VBScript %>
<%option Explicit%>
<!-- #include file="adovbs.inc" -->
<!-- #include file="connection.asp" -->

<%
    dim objRS
    Set objRS=Server.CreateObject("ADODB.Recordset")
    Set objRS.ActiveConnection=objConn
    objRS.CursorType=adOpenDynamic
    objRS.LockType=adLockOptimistic
    objRS.Open "Select * From Cart where Username='"&session("Username")&'"
    objRS.MoveFirst
    Do while not objRS.EOF
    Dim strQuantity
    strQuantity=Request(objRS("ProductID"))
    if not isNumeric(strQuantity) then
    Response.Redirect "Viewcart.asp"
    else if strQuantity<1 then
    Response.Redirect "Viewcart.asp"
    else if strQuantity >10 then
    Response.Redirect "Viewcart.asp"

```



```

        end if
        end if
        end if
        objRS("Quantity")=strQuantity
        objRs("price")=objRs("Uprice")*strQuantity
        objRS.Update
        objRS.MoveNext
    loop
    Response.Redirect "viewcart.asp"

'*****!

        'Add to cart'
    'adding to cart'

    %>

<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</HEAD>
<BODY>

<P>&nbsp;</P>

</BODY>
</HTML>

viewcart.asp

<%@ Language=VBScript %>
<%option Explicit%>
<!-- #include file="adovbs.inc" -->
<!-- #include file="connection.asp" -->
<%

    "session invalid"
    if session("Username")="" then
        Response.Redirect "Login.asp"
    end if

%>
<%

'Return to previous page'
    dim strPreviouspage
    strPreviouspage=Request.ServerVariables("HTTP_REFERER")

%>

```

```

<%
*****
'View Cart'
*****

dim objRs
Set objRs=Server.CreateObject("ADODB.Recordset")
Set objRs.ActiveConnection=objConn
    objRs.CursorType=adOpenDynamic
    objRs.LockType=adLockOptimistic
    'Connect to cart table'
    objRs.Open "select*from Cart where Username='"& session("Username")" &
""
' next part in body section

%>
<html>
<head>
<title>Shopping Cart</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body bgcolor="#FFFFFF" text="#000000">
<div align="left">
<table width="786" border="0">
<tr>
<td width="207" valign="middle">
<h2><font color="#510028"><b>Shopping Cart</b></font></h2>
</td>
<td width="339" valign="middle" align="center">
<div align="center">&nbsp;</div>
</td>
<td width="218" align="center" valign="middle">
<div align="right"><font color="#510028" size="1"><b><%=Date%>,<br>
<%=Time%></b></font></div>
</td>
</tr>
</table>
<p><font color="#000000"><b><a href="index.asp"><font
color="#000000">Home</font></a>
</b><%= if Session("Username")="" then%>
<b>Welcome Guest!</b></font>
<%=else %>
<%=Session("Username")%>
<%=end if%>

```

```

<font size=-1>
<% if Session("Username")="" then%>
<a href="Login.asp"><b><font color="#000000">Log in</font></b></a> /<b><a
href="Registration.asp"><font color="#000000"> Register</font></a>
</b>
<%else%>
<a href="Logout.asp"><b><font color="#000000">Log Out</font></b></a><br>
<%end if%>
</font></p>
</div>
<div id="Layer1" style="position:absolute; width:606px; height:139px; z-index:1; left:
163px; top: 120px">
<div align="center"><b><font face="Times New Roman, Times, serif" size="3"><a
href="checkOut.asp" style="TEXT-DECORATION: none">
</a></font></b><font color="#510028"><b><font color="#000000">
</font></b></font></div>
<div align="center">
<p><br>
<font color="#000000">
<% ' check record set if EOF Display No item in Cart '%>
<% if objRs.EOF then %>
<b>No item in cart</b>
<%else%>
<%
'count number of item and display all'

dim numItems
numItems=objRs.RecordCount
if numItems= 0 then
%>
<b>No item in cart</b>
<% else%>
</font></p>
<p><font color="#000000"><a href="checkout.asp"><b><font
color="#CC0000">Checkout</font></b></a><br>
<b><font face="Times New Roman, Times, serif" size="3"></font></b> </font>
</p>
</div>
<p align="center"><font color="#000000"><b>Your Cart contain the following
items:</b>
<br>
</font>
<form method="post" action="updateCart.asp">
<table width="100%" border="1" align="center" bordercolor="#FF9900"
bgcolor="#FFFFFF">

```



```

<tr bordercolor="#FFCC66">
  <td width="26%">
    <div align="center"><b>Quantity</b></div>
  </td>
  <td width="34%">
    <div align="center"><b>Products</b></div>
  </td>
  <td width="15%">
    <div align="center"><b>Unit Price</b></div>
  </td>
  <td width="15%">
    <div align="center"><b>Price</b></div>
  </td>
  <td bordercolor="#FFFFFF" width="10%">&nbsp;</td>
  <td bordercolor="#FFFFFF" width="10%">&nbsp;</td>
</tr>
<% objRs.MoveFirst%>
<%
'Function to calculate total amount Purchased'

dim totalAmount
    totalAmount=0

    do while not objRs.EOF
dim currentAmount,qty
    qty=objRs("Quantity")
    currentAmount=objRs("Price")
    totalAmount=totalAmount + currentAmount
    Session("totalAmount")=totalAmount
%>
<% 'display all information in table using shortcut direct from database'%>
<tr bordercolor="#FFCC66">
  <td width="26%">
    <div align="center">
      <input type="text" name="<%=objRs("ProductID")%>" size="2"
value="<%=objRs("Quantity")%>">
      <input type="hidden" name="txtproductID" value="<%=objRs("ProductID")%>">
    </div>
  </td>
  <td width="34%">
    <div align="left"></div>
    <%=objRs("Description")%></td>
  <td width="15%">
    <div align="center">$ <%=objRs("Uprice")%></div>
  </td>

```

```

<td width="15%">
  <div align="center">$ <%=objRs("Price")%> </div>
</td>
<td bordercolor="#FFCC99" colspan="2"><b></b>
  <div align="center"> <a href="delete.asp?ID=<%=objRS("CartID")%>&link=del"
onClick="confirm(' DELETE Item from Cart ?')""style="TEXT-DECORATION:
none"><b>
    <font face="Times New Roman, Times, serif" size="3"
color="#CC0000">Delete</font></b></a><font color="#CC0000">
</font>
  </div>
</td>
</tr>
<% objRs.MoveNext
loop
%>
<tr valign="bottom" bordercolor="#FFCC66">
<td colspan="3" height="59" valign="bottom">
  <div align="center">
    <h3 align="center"><font color="#CC0000"><b>Total Amount</b></font></h3>
  </div>
  <div align="center"> </div>
</td>
<td height="59" valign="middle" bordercolor="#FFFFFF" width="15%">
  <div align="center"><font color="#CC0000"><b>$
    <%=Response.Write totalAmount%>
  </b> </font></div>
</td>
<td height="59" valign="middle" bordercolor="#FFFFFF"
width="10%">&nbsp;</td>
<td height="59" valign="middle" bordercolor="#FFFFFF"
width="10%">&nbsp;</td>
</table>
<div align="center"><br>
  <input type="submit" name="Submit" value="Recalculate Total">
</div>
</form>
<% end if%> <% end if%>
<p align="center"><font color="#000000"><br>
  </font>
  <h3 align="center"><font color="#510028"><b><font color="#000000">
</font></b></font></h3>
</div>
<p>&nbsp;</p>
</body></html>

```

References

- [1] Y.Doç.Dr. Ahmet KAYA, ASP ile Web programlama:Ege Üniversitesi Bsm Evi Bornova-İZMİR,2002
- [2] Y.Doç.Dr.Adem KARAHOCA, ASP ile WEB Programcılığına Giriş, Beta İstanbul,Mart 2001
- [3] Erkan BALABAN, Web Tasarım Klavuzu, Pusula Yayıncılık 6.baskı İstanbul 2006
- [4] ASP in a Nutshell, Retrieved May 2005 from World Wide Web “friendsof”
- [5] Introduction to Active Server Pages for New Programmers Tutorial, from World Wide Web WastLakeInternet Training