Table of Contents

What is ASP?

- ASP stands for **A**ctive **S**erver **P**ages
- ASP is a program that runs inside **IIS**
- IIS stands for **I**nternet **I**nformation **S**ervices
- IIS comes as a free component with **Windows 2000**
- IIS is also a part of the **Windows NT 4.0 Option Pack**
- The Option Pack can be **downloaded** from Microsoft
- **PWS** is a smaller - but fully functional - version of IIS
- PWS can be found on your **Windows 95/98 CD**

ASP Compatibility

- ASP is a Microsoft Technology
- To run IIS you must have Windows NT 4.0 or later
- To run PWS you must have Windows 95 or later
- ChiliASP is a technology that runs ASP without Windows OS
- InstantASP is another technology that runs ASP without Windows

What is an ASP File?

- An ASP file is just the same as an HTML file
- An ASP file can contain text, HTML, XML, and scripts
- Scripts in an ASP file are executed on the server
- An ASP file has the file extension ".asp"

How Does ASP Differ from HTML?

- When a browser requests an HTML file, the server returns the file
- When a browser requests an ASP file, IIS passes the request to the ASP engine. The ASP engine reads the ASP file, line by line, and executes the scripts in the file. Finally, the ASP file is returned to the browser as plain HTML

What can ASP do for you?

- Dynamically edit, change or add any content of a Web page
- Respond to user queries or data submitted from HTML forms
- Access any data or databases and return the results to a browser
- Customize a Web page to make it more useful for individual users
- The advantages of using ASP instead of CGI and Perl, are those of simplicity and speed
- Provides security since your ASP code can not be viewed from the browser
- Since ASP files are returned as plain HTML, they can be viewed in any browser
- Clever ASP programming can minimize the network traffic

How to Run ASP on your own PC

You can run ASP on your own PC without an external server. To do that, you must install Microsoft's Personal Web Server (PWS) or Internet Information Server (IIS) on your PC.

If you are serious about using ASP, you should have at least Windows 98, Second Edition.

If you are really serious about using ASP, you should go for Windows 2000.

How to install PWS and run ASP on Windows 95

Personal Web Server (PWS) is not shipped with Windows 95 !!

To run ASP on Windows 95, you will have to download "Windows NT 4.0 Option Pack" from Microsoft.

How to install PWS and run ASP on Windows NT

Personal Web Server (PWS) is not shipped with Windows NT !!

To run ASP on Windows NT, you will have to download "Windows NT 4.0 Option Pack" from Microsoft.

How to install PWS and run ASP on Windows 98

1. Open the **Add-ons** folder on your Windows98 CD, find the **PWS** folder and run the **setup.exe** file.
2. An **Inetpub folder** will be created on your harddrive. Open it and find the **wwwroot** folder.
3. **Create a new folder**, like "MyWeb", under wwwroot.
4. **Use a text editor** to write some ASP code, save the file as "test1.asp" in the "MyWeb" folder.
5. Make sure your Web server is running - The installation program has added a new icon on your task bar (this is the PWS symbol). Click on the icon and press the Start button in the window that appears.
6. **Open your browser** and type in "http://localhost/MyWeb/test1.asp", to view your first ASP page.

How to install PWS and run ASP on Windows ME

Personal Web Server (PWS) is not included with Windows Me !!

How to install IIS and run ASP on Windows 2000

1. From your **Start Button**, go to **Settings**, and **Control Panel**
2. In the Control Panel window select **Add/Remove Programs**
3. In the Add/Remove window select **Add/Remove Windows Components**
4. In the Wizard window check **Internet Information Services**, **click OK**
5. An **Inetpub folder** will be created on your harddrive
6. Open the Inetpub folder, and find a folder named **wwwroot**
7. **Create a new folder**, like "MyWeb", under wwwroot.
8. **Use a text editor** to write some ASP code, save the file as "test1.asp" in the "MyWeb" folder
9. Make sure your Web server is running - The installation program has added a new icon on your task bar (this is the IIS symbol). Click on the icon and press the Start button in the window that appears.
10. **Open your browser** and type in "http://localhost/MyWeb/test1.asp", to view your first ASP page

How to install IIS and run ASP on Windows XP Professional

Note: You cannot run ASP on Windows XP Home Edition.

1. Insert the Windows XP Professional CD-Rom into your CD-Rom Drive
2. From your **Start Button**, go to **Settings**, and **Control Panel**
3. In the Control Panel window select **Add/Remove Programs**
4. In the Add/Remove window select **Add/Remove Windows Components**
5. In the Wizard window check **Internet Information Services**, **click OK**
6. An **Inetpub folder** will be created on your harddrive
7. Open the Inetpub folder, and find a folder named **wwwroot**
8. **Create a new folder**, like "MyWeb", under wwwroot.
9. **Use a text editor** to write some ASP code, save the file as "test1.asp" in the "MyWeb" folder
10. Make sure your Web server is running - The installation program has added a new icon on your task bar (this is the IIS symbol). Click on the icon and press the Start button in the window that appears.
11. **Open your browser** and type in "http://localhost/MyWeb/test1.asp", to view your first ASP page

ASP Syntax

You cannot view the ASP source code by selecting "View source" in a browser, you will only see the output from the ASP file, which is plain HTML. This is because the scripts are executed on the server before the result is sent back to the browser.

In our ASP tutorial, every example displays the hidden ASP source code. This will make it easier for you to understand how it works.

Example:

```
<html>
<body>

<%
response.write("Hello World!")
%>

</body>
</html>
```

Example:

```
<html>
<body>
<%
response.write("<h2>You can use HTML tags to format the text!</h2>")
%>

<%
response.write("<p style='color:#0000ff'>This text is styled with the style
attribute!</p>")
%>
</body>
</html>
```

The Basic Syntax Rule

An ASP file normally contains HTML tags, just like an HTML file. However, an ASP file can also contain **server scripts**, surrounded by the delimiters **<%** and **%>**. Server scripts are **executed on the server,** and can contain any expressions, statements, procedures, or operators valid for the scripting language you prefer to use.

The Response Object

The **Write** method of the ASP **Response Object** is used to send content to the browser. For example, the following statement sends the text "Hello World" to the browser:

```
<%
response.write("Hello World!")
%>
```

VBScript

You may use different scripting languages in ASP files. However, the default scripting language is VBScript:

```
<html>
<body>
<%
response.write("Hello World!")
%>
</body>
</html>
```

The example above writes "Hello World!" into the body of the document.

JavaScript

To set JavaScript as the default scripting language for a particular page you must insert a language specification at the top of the page:

```
<%@ language="javascript"%>
<html>
<body>
<%
Response.Write("Hello World!")
%>
</body>
</html>
```

Note: Unlike VBScript - JavaScript is case sensitive. You will have to write your ASP code with uppercase letters and lowercase letters when the language requires it.

Other Scripting Languages

ASP is shipped with VBScript and JScript (Microsoft's implementation of JavaScript). If you want to script in another language, like PERL, REXX, or Python, you will have to install script engines for them.

Important: Because the scripts are executed on the server, the browser that displays the ASP file does not need to support scripting at all!

ASP Variables

A variable is used to store information.

If the variable is declared outside a procedure it can be changed by any script in the ASP file. If the variable is declared inside a procedure, it is created and destroyed every time the procedure is executed.

Examples

Declare a variable

Variables are used to store information. This example demonstrates how to declare a variable, assign a value to it, and use the value in a text.

```
<html>
<body>

<%
dim name
name="Donald Duck"
response.write("My name is: " & name)
%>

</body>
</html>
```

Declare an array

Arrays are used to store a series of related data items. This example demonstrates how to declare an array that stores names.

```
<html>
<body>

<%
Dim famname(5),i
famname(0) = "Jan Egil"
famname(1) = "Tove"
famname(2) = "Hege"
famname(3) = "Stale"
famname(4) = "Kai Jim"
famname(5) = "Borge"

For i = 0 to 5
     response.write(famname(i) & "<br />")
Next
%>

</body>
</html>
```

Loop through the HTML headers

How to loop through the six headers in HTML.

```
<html>
<body>

<%
dim i
for i=1 to 6
   response.write("<h" & i & ">Header " & i & "</h" & i & ">")
next
%>

</body>
</html>
```

Time-based greeting using VBScript

This example will display a different message to the user depending on the time on the server.

```
<html>
<body>
<%
dim h
h=hour(now())

response.write("<p>" & now())
response.write(" (Norwegian Time) </p>")
If h<12 then
   response.write("Good Morning!")
else
   response.write("Good day!")
end if
%>
</body>
</html>
```

Time-based greeting using JavaScript

This example is the same as the one above, but the syntax is different.

```
<%@ language="javascript" %>
<html>
<body>
<%
var d=new Date()
var h=d.getHours()

Response.Write("<p>")
Response.Write(d + " (Norwegian Time)")
Response.Write("</p>")
if (h<12)
   {
   Response.Write("Good Morning!")
   }
else
   {
   Response.Write("Good day!")
   }
%>
</body>
</html>
```

Lifetime of Variables

A variable declared outside a procedure can be accessed and changed by any script in the ASP file.

A variable declared inside a procedure is created and destroyed every time the procedure is executed. No scripts outside the procedure can access or change the variable.

To declare variables accessible to more than one ASP file, declare them as session variables or application variables.

Session Variables

Session variables are used to store information about ONE single user, and are available to all pages in one application. Typically information stored in session variables are name, id, and preferences.

Application Variables

Application variables are also available to all pages in one application. Application variables are used to store information about ALL users in a specific application.

ASP Procedures

In ASP you can call a JavaScript procedure from a VBScript and vice versa.

Examples

Call a procedure using VBScript

How to call a VBScript procedure from ASP.

```
<html>

<head>
<%
sub vbproc(num1,num2)
response.write(num1*num2)
end sub
%>
</head>

<body>
<p>
You can call a procedure like this:
</p>
<p>
Result: <%call vbproc(3,4)%>
</p>
<p>
Or, like this:
</p>
<p>
Result: <%vbproc 3,4%>
</p>
</body>

</html>
```

Call a procedure using JavaScript

How to call a JavaScript procedure from ASP.

```
<%@ language="javascript" %>
<html>
<head>
<%
function jsproc(num1,num2)
{
Response.Write(num1*num2)
}
%>
</head>

<body>
<p>
Result: <%jsproc(3,4)%>
</p>
</body>

</html>
```

Call procedures using VBScript

How to call both a JavaScript procedure and a VBScript procedure in an ASP file.

```
<html>
<head>
<%
sub vbproc(num1,num2)
Response.Write(num1*num2)
end sub
%>
<script  language="javascript" runat="server">
function jsproc(num1,num2)
{
Response.Write(num1*num2)
}
</script>
</head>

<body>
<p>Result: <%call vbproc(3,4)%></p>
<p>Result: <%call jsproc(3,4)%></p>
</body>

</html>
```

Procedures

The ASP source code can contain procedures and functions:

```
<html>
<head>
<%
sub vbproc(num1,num2)
response.write(num1*num2)
end sub
%>
</head>
<body>
<p>Result: <%call vbproc(3,4)%></p>
</body>
</html>
```

Insert the <%@ language="*language*" %> line above the <html> tag to write procedures or functions in another scripting language than default:

```
<%@ language="javascript" %>
<html>
<head>
<%
function jsproc(num1,num2)
{
Response.Write(num1*num2)
}
%>
</head>
<body>
<p>Result: <%jsproc(3,4)%></p>
</body>
</html>
```

Differences between VBScript and JavaScript

When calling a VBScript or a JavaScript procedure from an ASP file written in VBScript, you can use the "call" keyword followed by the procedure name. If a procedure requires parameters, the parameter list must be enclosed in parentheses when using the "call" keyword. If you omit the "call" keyword, the parameter list must not be enclosed in parentheses. If the procedure has no parameters, the parentheses are optional.

When calling a JavaScript or a VBScript procedure from an ASP file written in JavaScript, always use parentheses after the procedure name.

ASP Forms and User Input

The Request.QueryString and Request.Form commands may be used to retrieve information from forms, like user input.

Examples

A form with method="get"

How to interact with the user, with the Request.QueryString command.

```
<html>
<body>
<form action="demo_reqquery.asp" method="get">
Your name: <input type="text" name="fname" size="20">
<input type="submit" value="Submit">
</form>
<%
dim fname
fname=Request.QueryString("fname")
If fname<>"" Then
      Response.Write("Hello " & fname & "!<br />")
      Response.Write("How are you today?")
End If
%>
</body>
</html>
```

A form with method="post"

How to interact with the user, with the Request.Form command.

```
<html>
<body>
<form action="demo_simpleform.asp" method="post">
Your name: <input type="text" name="fname" size="20">
<input type="submit" value="Submit">
</form>
<%
dim fname
fname=Request.Form("fname")
If fname<>"" Then
      Response.Write("Hello " & fname & "!<br />")
      Response.Write("How are you today?")
End If
%>
</body>
</html>
```

A form with radio buttons

How to interact with the user, through radio buttons, with the Request.Form command.

```
<html>
<%
dim cars
cars=Request.Form("cars")
%>
<body>
<form action="demo_radiob.asp" method="post">
<p>Please select your favorite car:</p>

<input type="radio" name="cars"
<%if cars="Volvo" then Response.Write("checked")%>
value="Volvo">Volvo</input>
<br />
<input type="radio" name="cars"
<%if cars="Saab" then Response.Write("checked")%>
value="Saab">Saab</input>
<br />
<input type="radio" name="cars"
<%if cars="BMW" then Response.Write("checked")%>
value="BMW">BMW</input>
<br /><br />
<input type="submit" value="Submit" />
</form>
<%
if cars<>"" then
   Response.Write("<p>Your favorite car is: " & cars & "</p>")
end if
%>
</body>
</html>
```

User Input

The Request object may be used to retrieve user information from forms:

```
<form method="get" action="simpleform.asp">
First Name: <input type="text" name="fname">
<br />
Last Name: <input type="text" name="lname">
<br /><br />
<input type="submit" value="Submit">
</form>
```

User input can be retrieved in two ways: With Request.QueryString or Request.Form.

Request.QueryString

The Request.QueryString command is used to collect values in a form with method="get". Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send.

If a user typed "Bill" and "Gates" in the form example above, the URL sent to the server would look like this:

```
http://www.w3schools.com/simpleform.asp?fname=Bill&lname=Gates
```

Assume that the ASP file "simpleform.asp" contains the following script:

```
<body>
Welcome
<%
response.write(request.querystring("fname"))
response.write(" " & request.querystring("lname"))
%>
</body>
```

The browser will display the following in the body of the document:

```
Welcome Bill Gates
```

Request.Form

The Request.Form command is used to collect values in a form with method="post". Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.

If a user typed "Bill" and "Gates" in the form example above, the URL sent to the server would look like this:

```
http://www.w3schools.com/simpleform.asp
```

Assume that the ASP file "simpleform.asp" contains the following script:

```
<body>
Welcome
<%
response.write(request.form("fname"))
response.write(" " & request.form("lname"))
%>
</body>
```

The browser will display the following in the body of the document:

```
Welcome Bill Gates
```

Form Validation

User input should be validated on the browser whenever possible (by client scripts). Browser validation is faster and you reduce the server load.

You should consider using server validation if the user input will be inserted into a database. A good way to validate a form on the server is to post the form to itself, instead of jumping to a different page. The user will then get the error messages on the same page as the form. This makes it easier to discover the error.

ASP Cookies

A cookie is often used to identify a user.

Examples

Welcome cookie

How to create a Welcome cookie.

```
<%
dim numvisits
response.cookies("NumVisits").Expires=date+365
numvisits=request.cookies("NumVisits")
%>
<html>
<body>
<%
if numvisits="" then
   response.cookies("NumVisits")=1%>
   Welcome! This is the first time you are visiting this Web page.
<%
else
   response.cookies("NumVisits")=numvisits+1
   response.write("You have visited this ")
   response.write("Web page " & numvisits)
   if numvisits=1 then
     response.write " time before!"
   else
     response.write " times before!"
   end if
end if
%>
</body>
</html>
```

What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests for a page with a browser, it will send the cookie too. With ASP, you can both create and retrieve cookie values.

How to Create a Cookie

The "Response.Cookies" command is used to create cookies.

Note: The Response.Cookies command must appear BEFORE the <html> tag.

In the example below, we will create a cookie named "firstname" and assign the value "Alex" to it:

```
<%
Response.Cookies("firstname")="Alex"
%>
```

It is also possible to assign properties to a cookie, like setting a date when the cookie should expire:

```
<%
Response.Cookies("firstname")="Alex"
Response.Cookies("firstname").Expires=#May 10,2002#
%>
```

How to Retrieve a Cookie Value

The "Request.Cookies" command is used to retrieve a cookie value.

In the example below, we retrieve the value of the cookie named "firstname" and display it on a page:

```
<%
fname=Request.Cookies("firstname")
response.write("Firstname=" & fname)
%>
```

Output:

```
Firstname=Alex
```

A Cookie with Keys

If a cookie contains a collection of multiple values, we say that the cookie has Keys.

In the example below, we will create a cookie collection named "user". The "user" cookie has Keys that contains information about a user:

```
<%
Response.Cookies("user")("firstname")="John"
Response.Cookies("user")("lastname")="Smith"
Response.Cookies("user")("country")="Norway"
Response.Cookies("user")("age")="25"
%>
```

Read all Cookies

Look at the following code:

```
<%
Response.Cookies("firstname")="Alex"
Response.Cookies("user")("firstname")="John"
Response.Cookies("user")("lastname")="Smith"
Response.Cookies("user")("country")="Norway"
Response.Cookies("user")("age")="25"
%>
```

Assume that your server has sent all the cookies above to a user.

Now we want to read all the cookies sent to a user. The example below shows how to do it (note that the code below checks if a cookie has Keys with the HasKeys property):

```
<html>
<body>
<%
dim x,y
for each x in Request.Cookies
  response.write("<p>")
  if Request.Cookies(x).HasKeys then
    for each y in Request.Cookies(x)
      response.write(x & ":" & y & "=" & Request.Cookies(x)(y))
      response.write("<br />")
    next
  else
    Response.Write(x & "=" & Request.Cookies(x) & "<br />")
  end if
  response.write "</p>"
next
%>
</body>
</html>
```

Output:

```
firstname=Alex

user:firstname=John
user:lastname=Smith
user: country=Norway
user: age=25
```

What if a Browser Does NOT Support Cookies?

If your application deals with browsers that do not support cookies, you will have to use other methods to pass information from one page to another in your application. There are two ways of doing this:

1. Add parameters to a URL

You can add parameters to a URL:

```
<a href="welcome.asp?fname=John&lname=Smith">
Go to Welcome Page</a>
```

And retrieve the values in the "welcome.asp" file like this:

```
<%
fname=Request.querystring("fname")
lname=Request.querystring("lname")
response.write("<p>Hello " & fname & " " & lname & "!</p>")
response.write("<p>Welcome to my Web site!</p>")
%>
```

2. Use a form

You can use a form. The form passes the user input to "welcome.asp" when the user clicks on the Submit button:

```
<form method="post" action="welcome.asp">
First Name:  <input type="text" name="fname" value="">
Last Name: <input type="text" name="lname" value="">
<input type="submit" value="Submit">
</form>
```

Retrieve the values in the "welcome.asp" file like this:

```
<%
fname=Request.form("fname")
lname=Request.form("lname")
response.write("<p>Hello " & fname & " " & lname & "!</p>")
response.write("<p>Welcome to my Web site!</p>")
%>
```

ASP Session Object

The Session object is used to store information about, or change settings for a user session. Variables stored in the Session object hold information about one single user, and are available to all pages in one application.

The Session object

When you are working with an application, you open it, do some changes and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are and what you do because the HTTP address doesn't maintain state.

ASP solves this problem by creating a unique cookie for each user. The cookie is sent to the client and it contains information that identifies the user. This interface is called the Session object.

The Session object is used to store information about, or change settings for a user session. Variables stored in the Session object hold information about one single user, and are available to all pages in one application. Common information stored in session variables are name, id, and preferences. The server creates a new Session object for each new user, and destroys the Session object when the session expires.

When does a Session Start?

A session starts when:

- A new user requests an ASP file, and the Global.asa file includes a Session_OnStart procedure
- A value is stored in a Session variable
- A user requests an ASP file, and the Global.asa file uses the <object> tag to instantiate an object with session scope

---

When does a Session End?

A session ends if a user has not requested or refreshed a page in the application for a specified period. By default, this is 20 minutes.

If you want to set a timeout interval that is shorter or longer than the default, you can set the **Timeout** property.

The example below sets a timeout interval of 5 minutes:

```
<%
Session.Timeout=5
%>
```

To end a session immediately, you may use the **Abandon** method:

```
<%
Session.Abandon
%>
```

Note: The main problem with sessions is WHEN they should end. We do not know if the user's last request was the final one or not. So we do not know how long we should keep the session "alive". Waiting too long uses up resources on the server. But if the session is deleted too fast you risk that the user is coming back and the server has deleted all the information, so the user has to start all over again. Finding the right timeout interval can be difficult.

Tip: If you are using session variables, store SMALL amounts of data in them.

Store and Retrieve Session Variables

The most important thing about the Session object is that you can store variables in it.

The example below will set the Session variable *username* to "Donald Duck" and the Session variable *age* to "50":

```
<%
Session("username")="Donald Duck"
Session("age")=50
%>
```

When the value is stored in a session variable it can be reached from ANY page in the ASP application:

```
Welcome <%Response.Write(Session("username"))%>
```

The line above returns: "Welcome Donald Duck".

You can also store user preferences in the Session object, and then access that preference to choose what page to return to the user.

The example below specifies a text-only version of the page if the user has a low screen resolution:

```
<%If Session("screenres")="low" Then%>
  This is the text version of the page
<%Else%>
  This is the multimedia version of the page
<%End If%>
```

Remove Session Variables

The Contents collection contains all session variables.

It is possible to remove a session variable with the Remove method.

The example below removes the session variable "sale" if the value of the session variable "age" is lower than 18:

```
<%
If Session.Contents("age")<18 then
  Session.Contents.Remove("sale")
End If
%>
```

To remove all variables in a session, use the RemoveAll method:

```
<%
Session.Contents.RemoveAll()
%>
```

Loop Through the Contents Collection

The Contents collection contains all session variables. You can loop through the Contents collection, to see what's stored in it:

```
<%
Session("username")="Donald Duck"
Session("age")=50
dim i
For Each i in Session.Contents
  Response.Write(i & "<br />")
Next
%>
```

Result:

```
username
age
```

If you do not know the number of items in the Contents collection, you can use the Count property:

```
<%
dim i
dim j
j=Session.Contents.Count
Response.Write("Session variables: " & j)
For i=1 to j
  Response.Write(Session.Contents(i) & "<br />")
Next
%>
```

Result:

```
Session variables: 2
Donald Duck
50
```

Loop Through the StaticObjects Collection

You can loop through the StaticObjects collection, to see the values of all objects stored in the Session object:

```
<%
dim i
For Each i in Session.StaticObjects
  Response.Write(i & "<br />")
Next
%>
```

ASP Application Object

A group of ASP files that work together to perform some purpose is called an application. The Application object in ASP is used to tie these files together.

Application Object

An application on the Web may be a group of ASP files. The ASP files work together to perform some purpose. The Application object in ASP is used to tie these files together.

The Application object is used to store and access variables from any page, just like the Session object. The difference is that ALL users share one Application object, while with Sessions there is one Session object for EACH user.

The Application object should hold information that will be used by many pages in the application (like database connection information). This means that you can access the information from any page. It also means that you can change the information in one place and the changes will automatically be reflected on all pages.

Store and Retrieve Application Variables

Application variables can be accessed and changed by any page in the application.

You can create Application variables in "Global.asa" like this:

```
<script language="vbscript" runat="server">

Sub Application_OnStart
application("vartime")=""
application("users")=1
End Sub

</script>
```

In the example above we have created two Application variables: "vartime" and "users".

You can access the value of an Application variable like this:

```
There are
<%
Response.Write(Application("users"))
%>
active connections.
```

Loop Through the Contents Collection

The Contents collection contains all application variables. You can loop through the Contents collection, to see what's stored in it:

```
<%
dim i
For Each i in Application.Contents
  Response.Write(i & "<br />")
Next
%>
```

If you do not know the number of items in the Contents collection, you can use the Count property:

```
<%
dim i
dim j
j=Application.Contents.Count
For i=1 to j
  Response.Write(Application.Contents(i) & "<br />")
Next
%>
```

Loop Through the StaticObjects Collection

You can loop through the StaticObjects collection, to see the values of all objects stored in the Application object:

```
<%
dim i
For Each i in Application.StaticObjects
  Response.Write(i & "<br />")
Next
%>
```

Lock and Unlock

You can lock an application with the "Lock" method. When an application is locked, the users cannot change the Application variables (other than the one currently accessing it). You can unlock an application with the "Unlock" method. This method removes the lock from the Application variable:

```
<%
Application.Lock
  'do some application object operations
Application.Unlock
%>
```

ASP Including Files

The #include directive is used to create functions, headers, footers, or elements that will be reused on multiple pages.

The #include Directive

You can insert the content of one ASP file into another ASP file before the server executes it, with the #include directive. The #include directive is used to create functions, headers, footers, or elements that will be reused on multiple pages.

How to Use the #include Directive

Here is a file called "mypage.asp":

```
<html>
<body>
<h3>Words of Wisdom:</h3>
<p><!--#include file="wisdom.inc"--></p>
<h3>The time is:</h3>
<p><!--#include file="time.inc"--></p>
</body>
</html>
```

Here is the "wisdom.inc" file:

```
"One should never increase, beyond what is necessary,
the number of entities required to explain anything."
```

Here is the "time.inc" file:

```
<%
Response.Write(Time)
%>
```

If you look at the source code in a browser, it will look something like this:

```
<html>
<body>
<h3>Words of Wisdom:</h3>
<p>"One should never increase, beyond what is necessary,
the number of entities required to explain anything."</p>
<h3>The time is:</h3>
<p>11:33:42 AM</p>
</body>
</html>
```

Syntax for Including Files

To include a file in an ASP page, place the #include directive inside comment tags:

```
<!--#include virtual="somefilename"-->
or
<!--#include file ="somefilename"-->
```

The Virtual Keyword

Use the virtual keyword to indicate a path beginning with a virtual directory.

If a file named "header.inc" resides in a virtual directory named /html, the following line would insert the contents of "header.inc":

```
<!-- #include virtual ="/html/header.inc" -->
```

The File Keyword

Use the file keyword to indicate a relative path. A relative path begins with the directory that contains the including file.

If you have a file in the html directory, and the file "header.inc" resides in html\headers, the following line would insert "header.inc" in your file:

```
<!-- #include file ="headers\header.inc" -->
```

Note that the path to the included file (headers\header.inc) is relative to the including file. If the file containing this #include statement is not in the html directory, the statement will not work.

You can also use the file keyword with the syntax (..\) to include a file from a higher-level directory.

Tips and Notes

In the sections above we have used the file extension ".inc" for included files. Notice that if a user tries to browse an INC file directly, its content will be displayed. If your included file contains confidential information or information you do not want any users to see, it is better to use an ASP extension. The source code in an ASP file will not be visible after the interpretation. An included file can also include other files, and one ASP file can include the same file more than once.

Important: Included files are processed and inserted before the scripts are executed.

The following script will not work because ASP executes the #include directive before it assigns a value to the variable:

```
<%
fname="header.inc"
%>
<!--#include file="<%=fname%>"-->
```

You cannot open or close a script delimiter in an INC file. This script will not work:

```
<%
For i = 1 To n
  <!--#include file="count.inc"-->
Next
%>
```

But this script will work:

```
<% For i = 1 to n %>
<!--#include file="count.inc" -->
<% Next %>
```

ASP The Global.asa file

The Global.asa file is an optional file that can contain declarations of objects, variables, and methods that can be accessed by every page in an ASP application.

The Global.asa file

The Global.asa file is an optional file that can contain declarations of objects, variables, and methods that can be accessed by every page in an ASP application. All valid browser scripts (JavaScript, VBScript, JScript, PerlScript, etc.) can be used within Global.asa.

The Global.asa file can contain only the following:

- Application events
- Session events
- <object> declarations
- TypeLibrary declarations

Note: The Global.asa file must be stored in the root directory of the ASP application, and each application can only have one Global.asa file.

Events in Global.asa

In Global.asa you can tell the application and session objects what to do when the application/session starts and what to do when the application/session ends. The code for this is placed in event handlers. The Global.asa file can contain four types of events:

Application_OnStart - This event occurs when the FIRST user calls the first page from an ASP application. This event occurs after the Web server is restarted or after the Global.asa file is edited. The "Session_OnStart" event occurs immediately after this event.

Session_OnStart - This event occurs EVERY time a NEW user requests his or hers first page in the ASP application.

Session_OnEnd - This event occurs EVERY time a user ends a session. A user ends a session after a page has not been requested by the user for a specified time (by default this is 20 minutes).

Application_OnEnd - This event occurs after the LAST user has ended the session. Typically, this event occurs when a Web server stops. This procedure is used to clean up settings after the Application stops, like delete records or write information to text files.

A Global.asa file could look something like this:

```
<script language="vbscript" runat="server">
sub Application_OnStart
''''some code
end sub
sub Application_OnEnd
''''some code
end sub
sub Session_OnStart
''''some code
end sub
sub Session_OnEnd
''''some code
end sub
</script>
```

Note: We cannot use the ASP script delimiters (<% and %>) to insert scripts in the Global.asa file, we will have to put the subroutines inside the HTML <script> tag.

<object> Declarations

It is possible to create objects with session or application scope in Global.asa by using the <object> tag.

Note: The <object> tag should be outside the <script> tag!

Syntax

```
<object runat="server" scope="scope" id="id"
{progid="progID"|classid="classID"}>
....
</object>
```

Parameter	Description
scope	Sets the scope of the object (either Session or Application)
id	Specifies a unique id for the object
ProgID	An id associated with a class id. The format for ProgID is [Vendor.]Component[.Version] Either ProgID or ClassID must be specified.
ClassID	Specifies a unique id for a COM class object. Either ProgID or ClassID must be specified.

Examples

The first example creates an object of session scope named "MyAd" by using the ProgID parameter:

```
<object runat="server" scope="session" id="MyAd"
progid="MSWC.AdRotator">
</object>
```

The second example creates an object of application scope named "MyConnection" by using the ClassID parameter:

```
<object runat="server" scope="application" id="MyConnection"
classid="Clsid:8AD3067A-B3FC-11CF-A560-00A0C9081C21">
</object>
```

The objects declared in the Global.asa file can be used by any script in the application:

GLOBAL.ASA:

```
<object runat="server" scope="session" id="MyAd"
progid="MSWC.AdRotator">
</object>
```

You could reference the object "MyAd" from any page in the ASP application:

SOME .ASP FILE:

```
<%=MyAd.GetAdvertisement("/banners/adrot.txt")%>
```

TypeLibrary Declarations

A TypeLibrary is a container for the contents of a DLL file corresponding to a COM object. By including a call to the TypeLibrary in the Global.asa file, the constants of the COM object can be accessed, and errors can be better reported by the ASP code. If your Web application relies on COM objects that have declared data types in type libraries, you can declare the type libraries in Global.asa.

Syntax

```
<!--METADATA TYPE="TypeLib"
file="filename"
uuid="typelibraryuuid"
version="versionnumber"
lcid="localeid"
-->
```

Parameter	**Description**
file	Specifies an absolute path to a type library. Either the file parameter or the uuid parameter is required
uuid	Specifies a unique identifier for the type library. Either the file parameter or the uuid parameter is required
version	Optional. Used for selecting version. If the requested version is not found, then the most recent version is used
localeid	Optional. The locale identifier to be used for the type library

Error Values

The server can return one of the following error messages:

Error Code	**Description**
ASP 0222	Invalid type library specification
ASP 0223	Type library not found
ASP 0224	Type library cannot be loaded
ASP 0225	Type library cannot be wrapped

Note: METADATA tags can appear anywhere in the Global.asa file (both inside and outside <script> tags). However, it is recommended that METADATA tags appear near the top of the Global.asa file.

Restrictions

Restrictions on what you can include in the Global.asa file:

- You can not display text that is written in the Global.asa file. This file can't display information
- You can not use the #include directive in Global.asa
- You can only use Server and Application objects in the Application_OnStart and Application_OnEnd subroutines. In the Session_OnEnd subroutine, you can use Server, Application, and Session objects. In the Session_OnStart subroutine you can use any built-in object

---

How to use the Subroutines

Global.asa is often used to initialize variables.

The example below shows how to detect the exact time a visitor first arrives on a Web site. The time is stored in a Session variable named "started", and the value of the "start" variable can be accessed from any ASP page in the application:

```
<script language="vbscript" runat="server">
sub Session_OnStart
Session("started")=now()
end sub
</script>
```

Global.asa can also be used to control page access.

The example below shows how to redirect every new visitor to another page, in this case to a page called "newpage.asp":

```
<script language="vbscript" runat="server">
sub Session_OnStart
Response.Redirect("newpage.asp")
end sub
</script>
```

And you can include functions in the Global.asa file.

In the example below the Application_OnStart subroutine occurs when the Web server starts. Then the Application_OnStart subroutine calls another subroutine named "getcustomers". The "getcustomers" subroutine opens a database and retrieves a record set from the "customers" table. The record set is assigned to an array, where it can be accessed from any ASP page without querying the database:

```
<script language="vbscript" runat="server">
sub Application_OnStart
getcustomers
end sub
sub getcustomers
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"
set rs=conn.execute("select name from customers")
Application("customers")=rs.GetRows
rs.Close
conn.Close
end sub
</script>
```

Global.asa Example

In this example we will create a Global.asa file that counts the number of current visitors.

- The Application_OnStart sets the Application variable "visitors" to 0 when the server starts
- The Session_OnStart subroutine adds one to the variable "visitors" every time a new visitor arrives
- The Session_OnEnd subroutine subtracts one from "visitors" each time this subroutine is triggered

The Global.asa file:

```
<script language="vbscript" runat="server">
Sub Application_OnStart
Application("visitors")=0
End Sub
Sub Session_OnStart
Application.Lock
Application("visitors")=Application("visitors")+1
Application.UnLock
End Sub
Sub Session_OnEnd
Application.Lock
Application("visitors")=Application("visitors")-1
Application.UnLock
End Sub
</script>
```

To display the number of current visitors in an ASP file:

```
<html>
<head>
</head>
<body>
<p>
There are <%response.write(Application("visitors"))%>
online now!
</p>
</body>
</html>
```

ASP Response Object

The ASP Response object is used to send output to the user from the server.

Examples

Write text with ASP

This example demonstrates how to write text with ASP.

```
<html>
<body>

<%
response.write("Hello World!")
%>

</body>
</html>
```

Format text with HTML tags in ASP

This example demonstrates how to combine text and HTML tags with ASP.

```
<html>
<body>
<%
response.write("<h2>You can use HTML tags to format the text!</h2>")
%>

<%
response.write("<p style='color:#0000ff'>This text is styled with the style
attribute!</p>")
%>
</body>
</html>
```

Redirect the user to a different URL

This example demonstrates how to redirect the user to a different URL.

```
<%
if Request.Form("select")<>"" then
     Response.Redirect(Request.Form("select"))
end if
%>

<html>
<body>

<form action="demo_redirect.asp" method="post">

<input type="radio" name="select"
value="demo_server.asp">
Server Example<br>

<input type="radio" name="select"
value="demo_text.asp">
Text Example<br><br>
<input type="submit" value="Go!">

</form>

</body>
</html>
```

Show a random link

This example demonstrates a link, each time you load the page, it will display one of two links: W3Schools.com! OR Refsnesdata.no! There is a 50% chance for each of them.

```
<html>
<body>

<%
randomize()
r=rnd()
if r>0.5 then
  response.write("<a href='http://www.w3schools.com'>W3Schools.com!</a>")
else
  response.write("<a href='http://www.refsnesdata.no'>Refsnesdata.no!</a>")
end if
%>

<p>
This example demonstrates a link, each time you load the page, it will display
one of two links: W3Schools.com! OR Refsnesdata.no! There is a 50% chance for
each of them.
</p>

</body>
</html>
```

Controlling the buffer

This example demonstrates how you can control the buffer.

```
<%
Response.Buffer=true
%>
<html>
<body>
<p>
This text will be sent to your browser when my response buffer is flushed.
</p>
<%
Response.Flush
%>
</body>
</html>
```

Clear the buffer

This example demonstrates how you can clear the buffer.

```
<%
Response.Buffer=true
%>
<html>
<body>
<p>This is some text I want to send to the user.</p>
<p>No, I changed my mind. I want to clear the text.</p>
<%
Response.Clear
%>
</body>
</html>
```

End a script in the middle of processing and return the result

This example demonstrates how to end a script in the middle of processing.

```
<html>
<body>
<p>I am writing some text. This text will never be<br>
<%
Response.End
%>
finished! It's too late to write more!</p>
</body>
</html>
```

Set how many minutes a page will be cached in a browser before it expires

This example demonstrates how to specify how many minutes a page will be cached in a browser before it expires.

```
<%Response.Expires=-1%>
<html>
<body>
<p>This page will be refreshed with each access!</p>
</body>
</html>
```

Set a date/time when a page cached in a browser will expire

This example demonstrates how to specify a date/time a page cached in a browser will expire.

```
<%
Response.ExpiresAbsolute=#May 05,2001 05:30:30#
%>
<html>
<body>
<p>This page will expire on May 05, 2001 05:30:30!</p>
</body>
</html>
```

Check if the user is still connected to the server

This example demonstrates how to check if a user is disconnected from the server.

```
<html>
<body>

<%
If Response.IsClientConnected=true then
Response.Write("The user is still connected!")
else
Response.Write("The user is not connected!")
end if
%>

</body>
</html>
```

Set the type of content

This example demonstrates how to specify the type of content.

```
<%
Response.ContentType="text/html"
%>
<html>
<body>

<p>This is some text</p>

</body>
</html>
```

Set the name of the character set

This example demonstrates how to specify the name of the character set.

```
<%
Response.Charset="ISO8859-1"
%>
<html>
<body>

<p>This is some text</p>

</body>
</html>
```

Response Object

The ASP Response object is used to send output to the user from the server. Its collections, properties, and methods are described below:

Collections

Collection	Description
Cookies	Sets a cookie value. If the cookie does not exist, it will be created, and take the value that is specified

Properties

Property	Description
Buffer	Specifies whether to buffer the page output or not
CacheControl	Sets whether a proxy server can cache the output generated by ASP or not
Charset	Appends the name of a character-set to the content-type header in the Response object
ContentType	Sets the HTTP content type for the Response object
Expires	Sets how long (in minutes) a page will be cached on a browser before it expires
ExpiresAbsolute	Sets a date and time when a page cached on a browser will expire
IsClientConnected	Indicates if the client has disconnected from the server
Pics	Appends a value to the PICS label response header
Status	Specifies the value of the status line returned by the server

Methods

Method	Description
AddHeader	Adds a new HTTP header and a value to the HTTP response
AppendToLog	Adds a string to the end of the server log entry
BinaryWrite	Writes data directly to the output without any character conversion
Clear	Clears any buffered HTML output
End	Stops processing a script, and returns the current result
Flush	Sends buffered HTML output immediately
Redirect	Redirects the user to a different URL
Write	Writes a specified string to the output

The Cookies Collection

The Cookies collection is used to set or get cookie values. If the cookie does not exist, it will be created, and take the value that is specified.

Note: The Response.Cookies command must appear before the <html> tag.

Syntax

```
Response.Cookies(name)[(key)|.attribute]=value
variablename=Request.Cookies(name)[(key)|.attribute]
```

Parameter	**Description**
name	Required. The name of the cookie
value	Required for the Response.Cookies command. The value of the cookie
attribute	Optional. Specifies information about the cookie. Can be one of the following parameters: • Domain - Write-only. The cookie is sent only to requests to this domain • Expires - Write-only. The date when the cookie expires. If no date is specified, the cookie will expire when the session ends • HasKeys - Read-only. Specifies whether the cookie has keys (This is the only attribute that can be used with the Request.Cookies command) • Path - Write-only. If set, the cookie is sent only to requests to this path. If not set, the application path is used • Secure - Write-only. Indicates if the cookie is secure
key	Optional. Specifies the key to where the value is assigned

Examples

The "Response.Cookies" command is used to create a cookie or to set a cookie value:

```
<%
Response.Cookies("firstname")="Alex"
%>
```

In the code above, we have created a cookie named "firstname" and assigned the value "Alex" to it.

It is also possible to assign some attributes to a cookie, like setting a date when a cookie should expire:

```
<%
Response.Cookies("firstname")="Alex"
Response.Cookies("firstname").Expires=#May 10,2002#
%>
```

Now the cookie named "firstname" has the value of "Alex", and it will expire from the user's computer at May 10, 2002.

The "Request.Cookies" command is used to get a cookie value.

In the example below, we retrieve the value of the cookie "firstname" and display it on a page:

```
<%
fname=Request.Cookies("firstname")
response.write("Firstname=" & fname)
%>
```

Output:

```
Firstname=Alex
```

A cookie can also contain a collection of multiple values. We say that the cookie has Keys.

In the example below, we will create a cookie-collection named "user". The "user" cookie has Keys that contains information about a user:

```
<%
Response.Cookies("user")("firstname")="John"
Response.Cookies("user")("lastname")="Smith"
Response.Cookies("user")("country")="Norway"
Response.Cookies("user")("age")="25"
%>
```

The code below reads all the cookies your server has sent to a user. Note that the code checks if a cookie has Keys with the HasKeys property:

```
<html>
<body>
<%
dim x,y
for each x in Request.Cookies
  response.write("<p>")
  if Request.Cookies(x).HasKeys then
    for each y in Request.Cookies(x)
      response.write(x & ":" & y & "=" & Request.Cookies(x)(y))
      response.write("<br /")
    next
  else
    Response.Write(x & "=" & Request.Cookies(x) & "<br />")
  end if
  response.write "</p>"
next
%>
</body>
</html>
%>
```

Output:

```
firstname=Alex

user:firstname=John
user:lastname=Smith
user: country=Norway
user: age=25
```

The Buffer Property

The Buffer property specifies whether to buffer the output or not. When the output is buffered, the server will hold back the response to the browser until all of the server scripts have been processed, or until the script calls the Flush or End method.

Note: If this property is set, it should be before the <html> tag in the .asp file

Syntax

```
response.Buffer[=flag]
```

Parameter	Description
flag	A boolean value that specifies whether to buffer the page output or not. False indicates no buffering. The server will send the output as it is processed. False is default for IIS version 4.0 (and earlier). Default for IIS version 5.0 (and later) is true. True indicates buffering. The server will not send output until all of the scripts on the page have been processed, or until the Flush or End method has been called.

Examples

Example 1

In this example, there will be no output sent to the browser before the loop is finished. If buffer was set to False, then it would write a line to the browser every time it went through the loop.

```
<%response.Buffer=true%>
<html>
<body>
<%
for i=1 to 100
  response.write(i & "<br />")
next
%>
</body>
</html>
```

Example 2

```
<%response.Buffer=true%>
<html>
<body>
<p>I write some text, but I will control when
the text will be sent to the browser.</p>
<p>The text is not sent yet. I hold it back!</p>
<p>OK, let it go!</p>
<%response.Flush%>
</body>
</html>
```

Example 3

```
<%response.Buffer=true%>
<html>
<body>
<p>This is some text I want to send to the user.</p>
<p>No, I changed my mind. I want to clear the text.</p>
<%response.Clear%>
</body>
</html>
```

The CacheControl Property

The CacheControl property sets whether a proxy server can cache the output generated by ASP or not. By default, a proxy server will not keep a cache copy.

Syntax

```
response.CacheControl[=control_header]
```

Parameter	**Description**
control_header	A cache control header that can be set to "Public" or "Private". Private is default and indicates that only private caches may cache this page. Proxy servers will not cache pages with this setting. Public indicates public caches. Proxy servers will cache pages with this setting.

Examples

```
<%response.CacheControl="Public"%>

or

<%response.CacheControl="Private"%>
```

The Charset Property

The Charset property appends the name of a character-set to the content-type header in the Response object. Default character set is ISO-LATIN-1.

Note: This property will accept any string, regardless of whether it is a valid character set or not, for the name.

Syntax

```
response.Charset(charsetname)
```

Parameter	**Description**
charsetname	A string that specifies a character set for the page

Examples

If an ASP page has no Charset property set, the content-type header would be:
```
content-type:text/html
```

If we included the Charset property:

```
<%response.Charset="ISO-8859-1"%>
```

the content-type header would be:

```
content-type:text/html; charset=ISO-8859-1
```

The ContentType Property

The ContentType property sets the HTTP content type for the response object.

Syntax

```
response.ContentType[=contenttype]
```

Parameter	Description
contenttype	A string describing the content type. For a full list of content types, see your browser documentation or the HTTP specification.

Examples

If an ASP page has no ContentType property set, the default content-type header would be:

```
content-type:text/html
```

Some other common ContentType values:

```
<%response.ContentType="text/HTML"%>
<%response.ContentType="image/GIF"%>
<%response.ContentType="image/JPEG"%>
<%response.ContentType="text/plain"%>
<%response.ContentType="image/JPEG"%>
```

This example will open an Excel spreadsheet in a browser (if the user has Excel installed):

```
<%response.ContentType="application/vnd.ms-excel"%>
<html>
<body>
<table>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</table>
</body>
</html>
```

The Expires Property

The Expires property sets how long (in minutes) a page will be cached on a browser before it expires. If a user returns to the same page before it expires, the cached version is displayed.

Syntax

```
response.Expires[=number]
```

Parameter	Description
number	The time in minutes before the page expires

Examples

Example 1

The following code indicates that the page will never be cached:

```
<%response.Expires=-1%>
```

Example 2

The following code indicates that the page will expire after 1440 minutes (24 hours):

```
<%response.Expires=1440%>
```

The ExpiresAbsolute Property

The ExpiresAbsolute property sets a date and time when a cached page on a browser will expire. If a user returns to the same page before this date/time, the cached version is displayed.

Syntax

```
response.ExpiresAbsolute[=[date][time]]
```

Parameter	**Description**
date	Specifies the date on which the page will expire. If this parameter is not specified, the page will expire at the specified time on the day that the script is run.
time	Specifies the time at which the page will expire. If this parameter is not specified, the page will expire at midnight of the specified day.

Examples

The following code indicates that the page will expire at 4:00 PM on October 11, 2003:

```
<%response.ExpiresAbsolute=#October 11,2003 16:00:00#%>
```

The IsClientConnected Property

The IsClientConnected property indicates if the client has disconnected from the server.

Syntax

```
response.IsClientConnected
```

Examples

```
<%
If response.IsClientConnected=true then
  response.write("The user is still connected!")
else
  response.write("The user is not connected!")
end if
%>
```

The PICS Property

The PICS property appends a value to the PICS label response header.

Note: This property will accept any string value, regardless of whether it is a valid PICS label or not.

What is PICS?

The PICS (Platform for Internet Content Selection) rating system is used to rate the content in a web site. It looks something like this:

PICS-1.1 "http://www.rsac.org/ratingsv01.html" by "your@name.com" for "http://www.somesite.com" on "2002.10.05T02:15-0800" r (n 0 s 0 v 0 l 0)

Part	**Description**
PICS-1.1	PICS version number
"http://www.rsac.org/ratingsv01.html"	Rating organization
by "your@name.com"	Author of the label
for "http://www.somesite.com"	The URL or the document that has been rated
on "2002.10.05T02:15-0800"	Expiration date
r (n 0 s 0 v 0 l 0)	Rating

One of the most popular rating system is RSACi (Recreational Software Advisory Council on the Internet). RSACi rating system uses four categories: violence, nudity, sex, and language. A number between 0 to 4 is assigned to each category. 0 means that the page does not contain any potentially offensive content and 4 means that the page contains the highest levels of potentially offensive content.

Level	**Violence Rating**	**Nudity Rating**	**Sex Rating**	**Language Rating**
0	None of the above or sports related	None of the above	None of the above or innocent kissing; romance	None of the above
1	Injury to human being	Revealing attire	Passionate kissing	Mild expletives
2	Destruction of realistic objects	Partial nudity	Clothed sexual touching	Moderate expletives or profanity
3	Aggressive violence or death to humans	Frontal nudity	Non-explicit sexual acts	Strong language or hate speech
4	Rape or wanton, gratuitous violence	Frontal nudity (qualifying as provocative display)	Explicit sexual acts or sex crimes	Crude, vulgar language or extreme hate speech

There are two ways you can obtain rating for your site. You can either rate your side yourself or use a rating provider, like RSACi. They'll ask you fill out some questions. After filling out the questions, you will get the rating label for your site.

Microsoft IE 3.0 and above and Netscape 4.5 and above support the content ratings. You can set the ratings in IE 5, by selecting Tools and Internet Options. Select the Content tab and click the Enable. When the rating exceeds the defined levels the Content Advisor will block the site. You can set the ratings in Netscape 4.7, by selecting Help and NetWatch.

We can use the META tag or the response.PICS property to add a rating to our site.

Syntax

```
response.PICS(picslabel)
```

Parameter	Description
picslabel	A properly formatted PICS label

Examples

For an ASP file that includes:

Note: Because PICS labels contain quotes, you
must replace quotes with " & chr(34) & ".

```
<%
response.PICS("(PICS-1.1 <http://www.rsac.org/ratingv01.html>
by " & chr(34) & "your@name.com" & chr(34) &
" for " & chr(34) & "http://www.somesite.com" & chr(34) &
" on " & chr(34) & "2002.10.05T02:15-0800" & chr(34) &
" r (n 2 s 0 v 1 l 2))")
%>
```

the following header would be added:

```
PICS-label:(PICS-1.1 <http://www.rsac.org/ratingv01.html>
by "your@name.com"
for "http://www.somesite.com"
on "2002.10.05T02:15-0800"
r (n 2 s 0 v 1 l 2))
```

The Status Property

The Status property specifies the value of the status line returned by the server.

Tip: Use this property to modify the status line returned by the server.

Syntax

```
response.Status=statusdescription
```

Parameter	Description
statusdescription	A three-digit number and a description of that code, like 404 Not Found **Note:** Status values are defined in the HTTP specification.

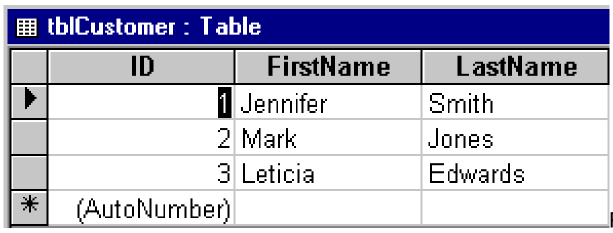
Examples

```
<%
ip=request.ServerVariables("REMOTE_ADDR")
if ip<>"194.248.333.500" then
  response.Status="401 Unauthorized"
  response.Write(response.Status)
  response.End
end if
%>
```

Step by Step: Connecting To Your Database

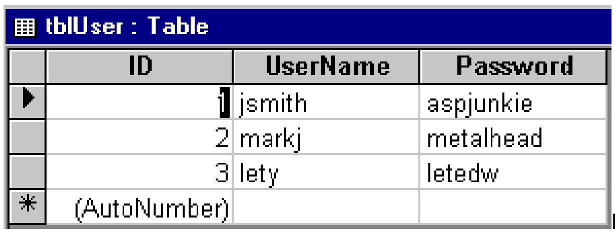
Note:
There are several ways to connect to your database. Below we show you how to create a System DSN. In the example code on the next page we'll also show you the alternate methods to connect to your database.

Create a new empty database. Build your tables as shown in Fig. 1 and Fig. 2. Save your database.


tblCustomer : Table

ID	FirstName	LastName
1	Jennifer	Smith
2	Mark	Jones
3	Leticia	Edwards
(AutoNumber)		

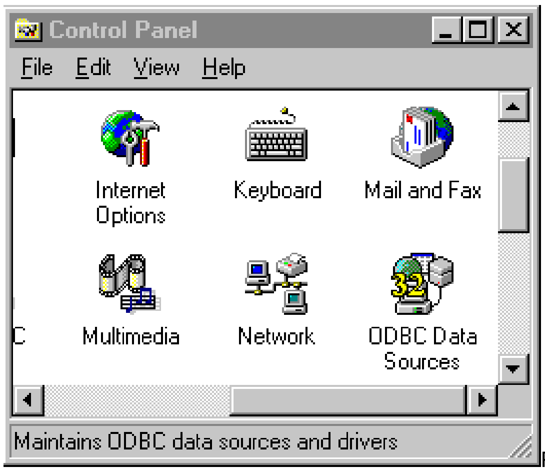
Fig. 1


tblUser : Table

ID	UserName	Password
1	jsmith	aspjunkie
2	markj	metalhead
3	lety	letedw
(AutoNumber)		

Fig. 2

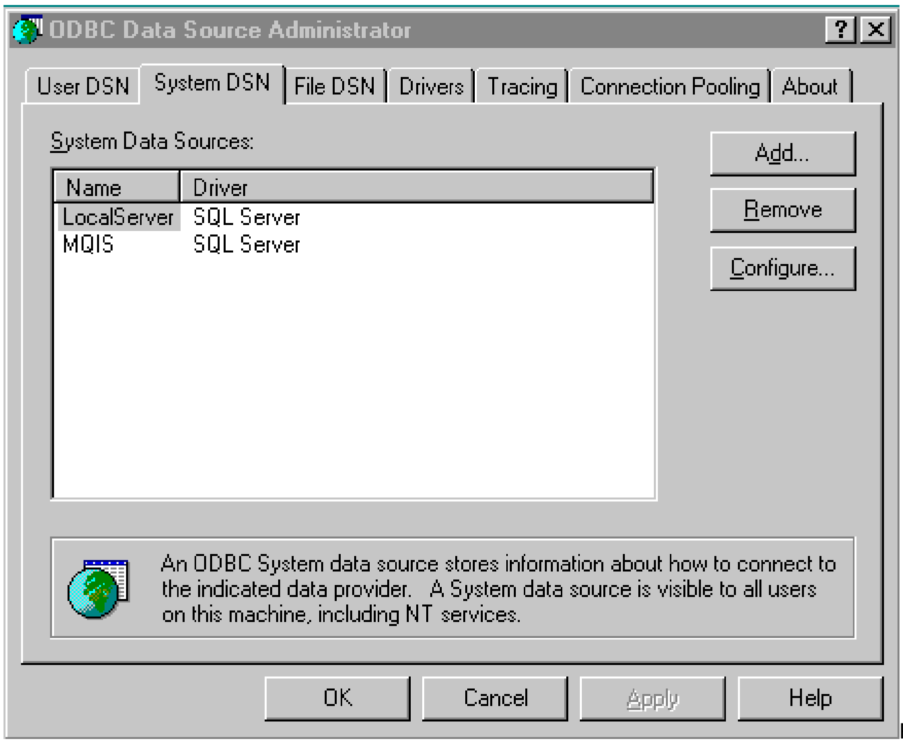
Now open up **Control Panel** and open up **ODBC Data Sources** (see Fig. 3)



Maintains ODBC data sources and drivers

Fig. 3

Select **System DSN** and then click Add. (see Fig. 4)



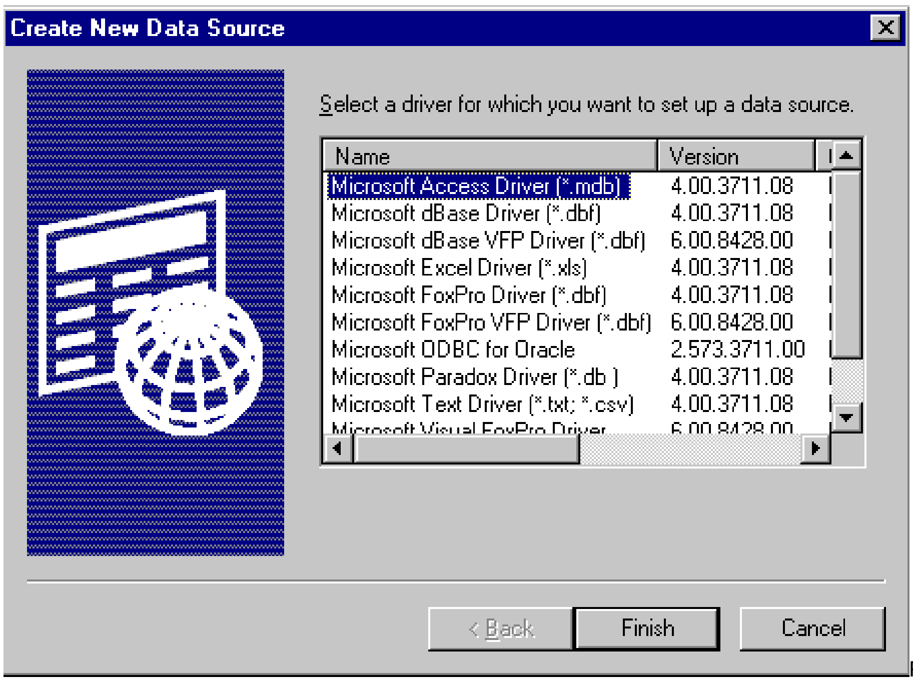
System Data Sources:

Name	Driver
LocalServer	SQL Server
MQIS	SQL Server

An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.

Fig. 4

Select the **Microsoft Access Driver** and click Finish. (see Fig. 5)

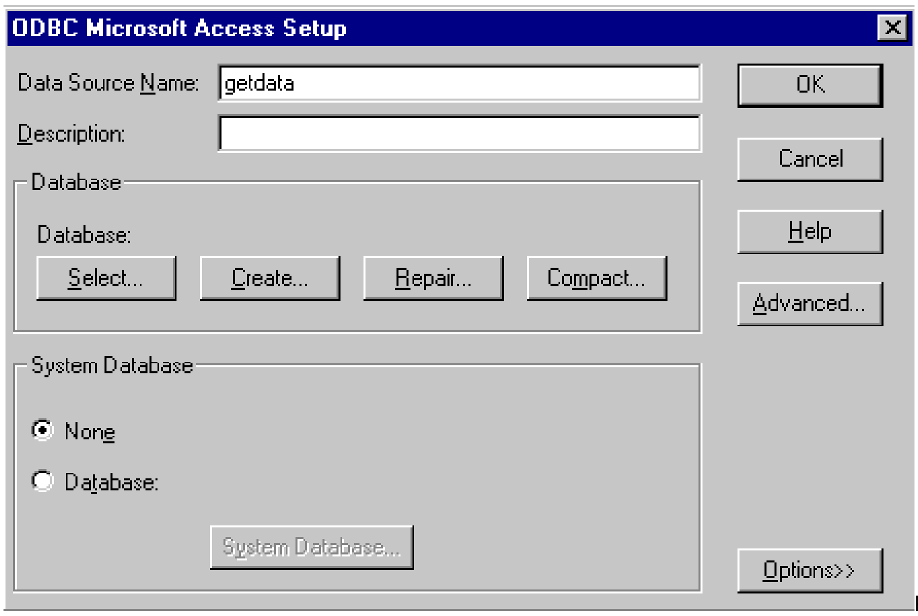


Select a driver for which you want to set up a data source.

Name	Version
Microsoft Access Driver (*.mdb)	4.00.3711.08
Microsoft dBase Driver (*.dbf)	4.00.3711.08
Microsoft dBase VFP Driver (*.dbf)	6.00.8428.00
Microsoft Excel Driver (*.xls)	4.00.3711.08
Microsoft FoxPro Driver (*.dbf)	4.00.3711.08
Microsoft FoxPro VFP Driver (*.dbf)	6.00.8428.00
Microsoft ODBC for Oracle	2.573.3711.00
Microsoft Paradox Driver (*.db)	4.00.3711.08
Microsoft Text Driver (*.txt; *.csv)	4.00.3711.08
Microsoft Visual FoxPro Driver	6.00.8428.00

Fig. 5

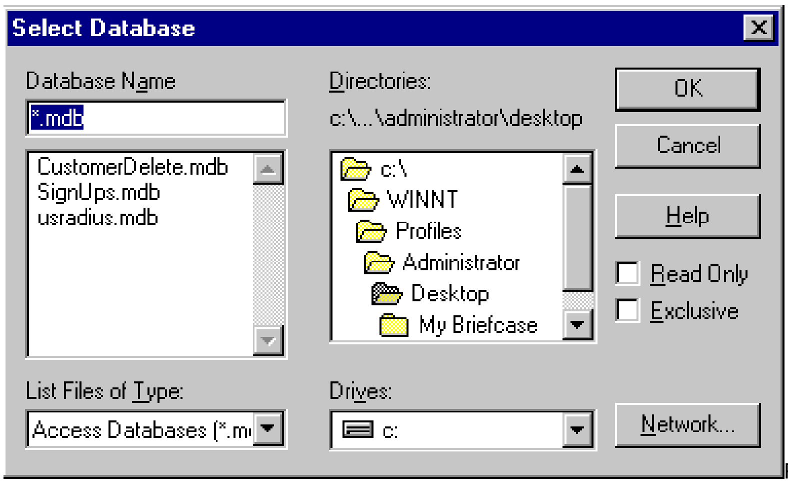
Type in your **Data Source Name (DSN)**. Then click on Select. (see Fig. 6)



Data Source Name: getdata
Description:
Database:
System Database: (●) None ( ) Database:

Fig. 6

Find and select your database and then click OK. (see Fig. 7)



Database Name: *.mdb
CustomerDelete.mdb
SignUps.mdb
usradius.mdb
Directories: c:\...\administrator\desktop
c:\
WINNT
Profiles
Administrator
Desktop
My Briefcase
List Files of Type: Access Databases (*.m
Drives: c:

Fig. 7

You're now done setting up your DSN.

Now you're ready to create your pages.

First let's create the search page. This page uses a form where one can enter the criteria to search the database. In this example the last name is entered into the form.

```
<%@ Language=VBScript %>

<html>
<body>

<form name="Search" method="Post" action="searchdb.asp">
'searchdb.asp is the page that actually queries the database.
<input type="text" name="searchStr" size="20">
<input type="submit" value="Search">
</form>

</body>
</html>
```

Name this page *form.asp*

Now we create the page that retrieves the data from the database.

```
<%@ Language=VBScript %>
<% Response.Buffer=True %>

<html>
<body>

<%
'Dim your variables
Dim searchStr, MyConn, RS, i

'grab the values entered into the form (from form.asp)
'the Request.Form() method is used to grab the value. The argument is
'the name given to the form field.
'the Replace() method replaces any single apostrophes ( ' )
'with double apostrophes ( '' ). Otherwise, you'll get a nasty error
searchStr = Replace(Request.Form("searchStr"), "'", "''")

'create the Connection object
Set MyConn=Server.CreateObject("ADODB.Connection")

'Open the connection to the database
'Note: pick whichever connection method you want to use.
'the first method uses a System DSN, like the one at the beginning
'of this article.
'the second method uses an OLE DB connection... the preferred method.
'and the third method uses a DSN-Less connection
'the first method (System DSN) is set as the default connection
MyConn.Open "getdata"
'MyConn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Databases\demo.mdb"
'MyConn.Open "Driver={Microsoft Access Driver (*.mdb)}; DBQ=C:\Databases\demo.mdb"

SQL = "SELECT * FROM tblCustomer, tblUser WHERE tblCustomer.LastName "
SQL = SQL & "LIKE '%"&searchStr&"%' AND tblCustomer.ID = tblUser.ID"

Set RS = MyConn.Execute(SQL)
```

```
'as long as a value was entered into the form...
If searchStr <> "" Then

'here we check for both Beginning of file (BOF) And End of file (EOF)
'if both are True then no records were found.
  If RS.BOF AND RS.EOF Then
    Response.Write "<center> No Records Found.</center>"

  'otherwise, create a HTML table and fill it with the search result(s)
  Else
    Response.Write "<center><table border=""1""><tr>"

    'the RS.Fields.Count counts the number of columns in the database.
    'the HTML table's headers ( <th> ) are created and the column names
    'are entered into them.
    For i= 0 to RS.Fields.Count - 1
      Response.Write "<th>" & RS(i).Name & "</th>"
    Next

  Response.Write "</tr>"

  'the headers are now completed, now the rest of the HTML table is filled
  'with the result(s) of the search.
  While Not RS.EOF
    Response.Write "<tr>"
    For i= 0 to RS.Fields.Count - 1
      Response.Write "<td>" & RS(i) & "</td>"
    Next
    Response.Write "</tr>"

    'move to the next record
    RS.MoveNext
    WEND

    'close the HTML table
    Response.Write "</table>"
  End If


'if the form was left blank then alert the user.
Else
  Response.Write "<center>No Search String Entered.</center>"

End IF


'close and destroy all objects created to free up system memory
RS.Close
MyConn.Close
Set RS = Nothing
Set MyConn = Nothing
%>

</body>
</html>
```

Name this page *searchdb.asp*

That's it! Go here to view the demo.

I've even added the pseudo-code below:

```
get the data from the form
open the connection
query database

if data was entered into the form then
    if no data is found meeting the search criteria tell the user no data found
otherwise
    build the table and show the search results

if no data had been entered into the form then tell the user no search string entered

close the connection
```

Deleting Records from More than One Table

This article is a continuation of the Step by Step: Database Tutorial. However, the concept extends to all databases. To delete a record you will first need to create a form for which you can select the record to delete.

```
<%@ Language=VBScript %>

<html>
<body>

<%
Set MyConn = Server.CreateObject("ADODB.Connection")
Set RS=Server.CreateObject("ADODB.RecordSet")
MyConn.OPEN "demo"
'demo is the name of your DSN
RS.Open "Select * From tblCustomer", MyConn

Do While Not RS.eof
%>
<table>
<tr>
<td>
<form action="deletion.asp" method="Post">
<INPUT Type=CheckBox Name="toDelete" Value="<% = RS("ID")%>"></td>
'this puts a checkbox beside each record. you then select the record to delete and the value is the
key field ID
<TD><% = RS("FirstName")%></TD>
<TD><% = RS("LastName")%></TD>
</TR>
<%
RS.movenext
Loop
%>
</table>
<%
RS.close
MyConn.close
%>
<input type="submit" value="Delete">
</form>
</table>

</body>
</html>
```

Save this page as **delete.asp**

Now create the page that actually deletes the record...

```
<%
Dim RS, MyConn, SQL, SQL2

Set MyConn=Server.CreateObject("ADODB.Connection")
MyConn.Open "demo"

SQL = "DELETE FROM tblCustomer WHERE ID IN("&request.form("toDelete")&")"
'grab the value(s) selected from the checkboxes
SQL2 = "DELETE FROM tblUser WHERE ID IN("&request.form("toDelete")&")"

Set RS = MyConn.Execute(SQL)
Set RS = MyConn.Execute(SQL2)
'both tables use ID as the key field and both corresponding records are deleted

MyConn.Close
Set MyConn = Nothing
%>

<center>Record(s) Deleted!</center>

</body>
</html>
```

Save this as **deletion.asp**

Enjoy!

Adding Records to More than One Table

This article is a continuation of the Step by Step: Database Tutorial. However, the concept extends to all databases. To add a record you will first need to create a form for which you can enter the data for the fields.

Below is a form that collects the *FirstName, LastName, UserName* and *Password*. The first two fields are part of the *tblCustomers* table and the last two fields are part of the *tblUsers* table.

```
<form name="Search" method="Post" action="addtodatabase.asp">
'addtodatabase.asp is the page that connects to the database.
FirstName <input type="text" name="txtFirstName" size="20">
LastName <input type="text" name="txtLastName" size="20">
UserName <input type="text" name="txtUserName" size="20">
Password <input type="text" name="txtPassword" size="20">
<input type="submit" name="btnSearch" value="Add">
</form>
```

Save this page as *addRec.asp*.

Now create your *addtodatabase.asp* page. (note the action method in the above form).

```
'put the below 3 lines at the top of the page above all HTML tags.
<%@ Language=VBScript %>
<% Option Explicit %>
<!--#INCLUDE VIRTUAL="adovbs.inc" -->
```

'you will need to include the **adovbs.inc**. If you get an error that the adovbs.inc file couldn't be found make sure you have the correct path or use *FILE* instead of *VIRTUAL*.

```
<%
Dim MyConn, RS, RS2, strFirstName, strLastName, strUserName, strPassword

strFirstName = Request.Form("txtFirstName")
strLastName = Request.Form("txtLastName")
strUserName = Request.Form("txtUserName")
strPassword = Request.Form("txtPassword")
'grab the form contents

Set MyConn=Server.CreateObject("ADODB.Connection")
Set RS=Server.CreateObject("ADODB.RecordSet")
```

'since you are working with the *RecordSet* you need to create an instance of the RecordSet Object

```
Set RS2=Server.CreateObject("ADODB.RecordSet")
```

'in this case, since we're adding to **two** separate tables we need to create two instances of the RecordSet Object

```
MyConn.Open "getdata"
'getdata is your DSN (data source name) you created in ODBC

RS.Open "Select * From tblCustomer", MyConn, adOpenDynamic, adLockPessimistic, adCMDText
RS2.Open "Select * From tblUser", MyConn, adOpenDynamic, adLockPessimistic, adCMDText
'open both RecordSets

RS.AddNew
RS("FirstName")= strFirstName
RS("LastName")= strLastName
RS.Update
'Update the first RecordSet

RS2.AddNew
```

```
RS2("UserName")= strUserName
RS2("Password")= strPassword
RS2.Update
'Update the second RecordSet

'Clean up
RS.Close
RS2.Close
MyConn.Close
Set RS = Nothing
Set RS2 = Nothing
Set MyConn = Nothing
%>
```

That's it!

Databases and Checkboxes

by Jim Rudnick, KKT INTERACTIVE

Having been a contractor now for almost 2 decades, it came as a surprise to me that using checkboxes in ASP forms could cause me any problems. I mean what's really to it; you allow the user to click on one or two, store that data in a dbase and then when they ask to see that page again, you put 'em back up already checked. Simple, eh? But not really -- at least for me.

The project I'm currently working on is to create a new web application for a used car lot in a major city near my own. This dealer must be one of the largest in the whole province (Canadian, eh?) and they're needs have been intensively planned for and are under current development. But the checkbox thing came up as an aside really, in that I had trouble from the start storing that data and then reading it out for their users.

Checkboxes, first of all, are just that. A little square box, created in HTML as a part of the form you are building and for the most part, they allow the user to make some/any/all/none of them 'checked' or not. That is, unlike radio buttons where the user is allowed to check only one radio button per family (like MALE (x) FEMALE( )) a series of checkboxes can be either checked or not by the user in any number of ways. Now often, some developers add a VALUE to the HTML string like below.

Here's the HTML for the checkbox...

```
<input type="checkbox" name="Drive" value="All-WheelDrive">
```

Using the above, when that part of the form is processed and added to the database, IF the checkbox is checked by the user, then the value "All-WheelDrive" is stored into the database in the proper column and row. This is a simple way to show what the user has chosen to 'check' or not to 'check,' in this case the user indicated by the checkbox that the car they were describing DID have All-WheelDrive. Simple and neat, and the column and row in the dbase would simply have the term All-Wheel Drive in that cell indicating that the car does indeed have All-Wheel Drive.

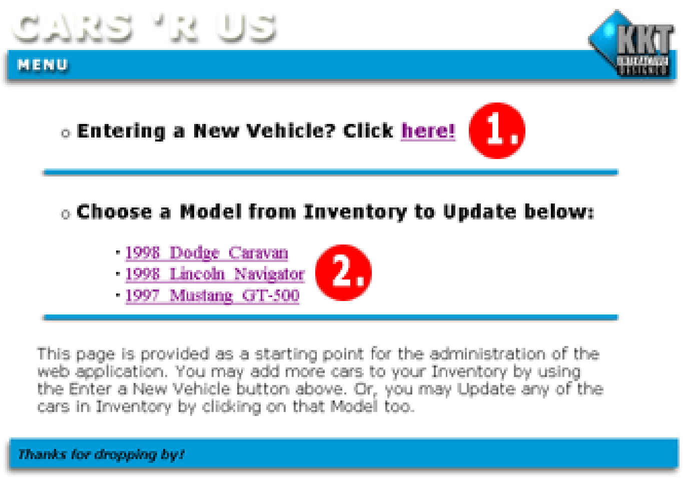
This is fairly basic and simple to plan and develop upon; the thing of it is (and I was surprised to learn this too) that there are more than 230 of these checkboxes that can be used to describe every single option and feature and item about any kind of car or van or truck or SUV or whatever. Now that's one heck of a lot of items to plan for; one heck of a lot of code to write to allow for all of these 230 various types of data and the odds are that the dbase will bloat quickly with this much content stored for thousands and thousands of vehicles that the client needs to record over the life of the web application.

Hmm...so, is there a better way? Yup, I think I found one too...

Using Access here as the simplest form of dbase to describe, I've found that using the YES/NO type which is only 1 bit rather than the TEXT type which stores up to 255 chars is very much a performance enhancer for quick database polling. YES/NO types allow the storage of a single bit -- hence either a CHECKED/UNCHECKED value can be stored in that cell and then recalled as needed if it's properly 'translated' for the user to understand.

To help the reader understand this process, I've built a series of screens here, using ASP and Access97 as the dbase to show how to store and call up that checkbox data. This is under the guise of being a CarLot Dealership web application, and will allow you to actually 'work' the screens to make the Inventory grow as needed.

To begin with, here's the first page that the whole article starts with, called 'menu.asp'


CARS 'R US

MENU

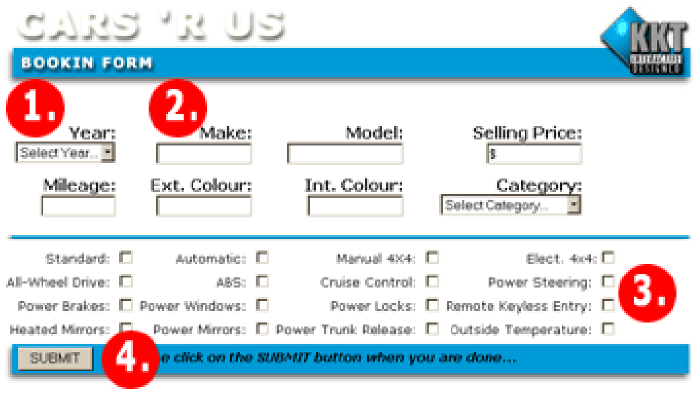
- Entering a New Vehicle? Click here! 1.

- Choose a Model from Inventory to Update below:
  - 1998 Dodge Caravan
  - 1998 Lincoln Navigator 2.
  - 1997 Mustang GT-500

This page is provided as a starting point for the administration of the web application. You may add more cars to your Inventory by using the Enter a New Vehicle button above. Or, you may Update any of the cars in Inventory by clicking on that Model too.

Thanks for dropping by!

I've numbered the areas that you will need to look at as follows. On **'menu.asp,'** Number 1 is the button that you will need to push to call up the page wherein you 'book-in' a new car into the inventory and Number 2 lists the cars that are already in inventory and by clicking on any of them, you can call them up to see their options and make any updates or edits that you might need to do. To get started, click on the Number 1 and lets book in a new car...


CARS 'R US

BOOKIN FORM

1. Year: Select Year...	2. Make:	Model:	Selling Price: $
Mileage:	Ext. Colour:	Int. Colour:	Category: Select Category...

Standard: ☐	Automatic: ☐	Manual 4X4: ☐	Elect. 4x4: ☐
All-Wheel Drive: ☐	ABS: ☐	Cruise Control: ☐	Power Steering: ☐
Power Brakes: ☐	Power Windows: ☐	Power Locks: ☐	Remote Keyless Entry: ☐
Heated Mirrors: ☐	Power Mirrors: ☐	Power Trunk Release: ☐	Outside Temperature: ☐

3.

SUBMIT 4. *e click on the SUBMIT button when you are done...*

So...looking now at the **'bookIn.asp'** page are we? You'll note that there are 4 numbers on this one as well for your study. Let's look at Number 1 first. This is a 'drop-down' or a 'pull-down' as they are called. The user simply clicks on the little triangle to select the year that the car was manufactured. You can look up the code of course by looking at the area I called CAR DATA HERE near the top of the 'bookIn.asp' page. I put in a coupe of these types of data entry just to show how it's done.

Number 2 on this page, allows the user to simply enter text into a text box for storage later in the database, something we're all familiar with, but note that trying NOT to enter data has been disallowed as I've put in a little javascript verification function at the top of the page. You must complete each of the main text and drop-down areas or you'll not be allowed to move on -- a good idea for user data entry control!

Number 3 has our checkboxes, all 16 of them in this case. Each one offers up the opportunity of allowing the user to check the box to indicate that the vehicle DOES HAVE that option. That is, if the user clicks on All-Wheel Drive then that means that this vehicle HAS that option. The user can check all of these 16 checkboxes; some of them or none of them as they are NOT validated by the javascript function that protects the main areas of the vehicle description. Here's the code segment that controls that All-Wheel Drive checkbox display for the user as a part of the form and table...

```
<TD align=right> <FONT SIZE="1" Face="Verdana">All-Wheel Drive: </font>
<INPUT TYPE="checkbox" NAME="ckAllWheelDrive" value="1"> </TD>
```

You'll note that I called my checkbox "ckAllWheelDrive" which is both self-commenting as well as it adds a shortform of the type to the beginning of the name so that just by glancing at it I know that any name that begins with 'ck' must be a checkbox name. This naming convention is rather personal I know but it's shared by many many other programmers as I've learned over the years -- most likely a good habit to pass along too! Also, note that if this checkbox IS checked, then it will receive it's value or 1 automatically. That is an unchecked box = 0 and a checked box = 1.

So once the user has chosen their options for this vehicle, checking here and checking there as they add the options that exist for that vehicle. Once they're done, they click on the Number 4 area, the SUBMIT button at the bottom left of that page. Once clicked, this automatically goes to the page called 'bookBuilder.asp' and writes the chosen data to the Access dbase.

But lets look at the specific code that first 'reads' the checkboxes, and then assigns a value for that checkbox for the proper YES/NO column in the table. Here's one of the code segments tha covers the All-Wheel Drive checkbox for example along with the var that I'm using too...

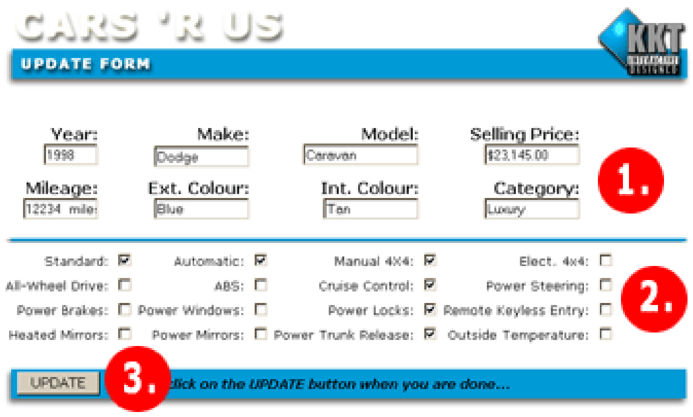
```
DIM jr1
jr1 = 0


 if request.form("ckAllWheelDrive") = "" then
    rs.Fields("AllWheelDrive") = jr1
 else
    rs.Fields("AllWheelDrive") = request.form("ckAllWheelDrive")
 end if
```

Okay, so what's this mean, right? Well first, I establish a var, called jr1 and assign it a value of 0 (zero). Next, I use an If...then to conditionally look at the value of the checkbox in question. IF that checkbox is "" or nothing, then it was NOT checked, so assign it a value of jr1 (or 0)....ELSE...that checkbox is NOT nothing, then is WAS checked, so assign it the value that it received by being checked, which is a 1 (or TRUE if you like).

Storing that value is simple for a dbase, it's a simple (in Access) thing to view same so do take alook at how these YES/NO columns are structured in the accompanying dbase when you've a moment. You'll note that Access doesn't actually store the 1 s and 0 s, it actually has a little checkbox that is checked or unchecked all at a 1 bit size! Cool, eh?

So far, so good. Clicking on that SUBMIT button took you to 'bookBuilder.asp' which processed the addition of this new vehicle to the dbase, and then took you back to 'menu.asp,' which is where you should be now. So now, let's look at updating some data on a one of our vehicles in the CarLot Inventory, shall we?

We're back now on the main menu, on 'menu.asp' page, and note Number 2 in the lower area. Here, I've listed the Inventoried vehicles that should now have your own entry on the list, but for purposes of this article, lets choose one that I entered and know is there for sure. Click on the 1999 Dodge Caravan please and then watch the next page, 'bookEdit.asp' come up for you to study.



Year: 1998	Make: Dodge	Model: Caravan	Selling Price: $23,145.00
Mileage: 12234 mile:	Ext. Colour: Blue	Int. Colour: Tan	Category: Luxury

Standard: ✓	Automatic: ✓	Manual 4X4: ✓	Elect. 4x4: ☐
All-Wheel Drive: ☐	ABS: ☐	Cruise Control: ✓	Power Steering: ☐
Power Brakes: ☐	Power Windows: ☐	Power Locks: ✓	Remote Keyless Entry: ☐
Heated Mirrors: ☐	Power Mirrors: ☐	Power Trunk Release: ✓	Outside Temperature: ☐

click on the UPDATE button when you are done...

Again, this page has some numbered areas for your attention. Number 1 is the main data about the vehicle, like the year and make and model etc. Number 2 is the area that has the checkboxes which we'll look at first. Note for me, that the checkbox by the 'All-Wheel Drive' option is un-checked. This is because in the dbase, under the matching column, the value stored for this vehicle was a 0, so when it was called by this page, the value was transferred over and the resulting checkbox is unchecked. Here's that code segment from 'bookEdit.asp' ...

```
<TD align=right><FONT SIZE="1" Face="Verdana">All-Wheel Drive: </font>
 <%
    If rs.Fields("AllWheelDrive") = TRUE then
 %>
<INPUT TYPE="checkbox" NAME="ckAllWheelDrive" value="1" CHECKED>
 <%
    ELSE
 %>
<INPUT TYPE="checkbox" NAME="ckAllWheelDrive" value="1">
 <%
    end IF
 %>
```

Notice, that again I qualify what's found in the dbase column. IF the contents of that field called rs.Fields("AllWheelDrive") which is a YES/NO are TRUE (or 1) then I write that checkbox in it's place and add the extra CHECKED HTML argument that means that this one MUST be shown in it's 'checked' form....ELSE, the contents of that YES/NO are not TRUE, then just show the checkbox without being checked but again keep the value = 1 in there in case it's checked this time. Simple and neat!

Okay, so let's check that All-Wheel Drive checkbox, shall we? Note that the little box is now checked, and once you've done that, click below on the Number 3 area called UPDATE. This button transfers control to the page that will write this newly updated material to the dbase, called 'bookEditBuilder.asp.' Here, once again, we read each of the input fields for data, changed or not, and store the currently displayed data in the proper cells in the dbase. Here's the code segment that I used this time to store that data...

```
 if request.form("ckAllWheelDrive") = "1" then
    rs.Fields("AllWheelDrive") = TRUE
 else
    rs.Fields("AllWheelDrive") = "0"
 end if
```

Once that checkbox and all the other data has been written to the dbase, you are again transferred back to the main menu page for further use if needs be. Other items that could/should be written to this app include in perhaps the ability to click a button and DELETE a vehicle from the Inventory, the addition of perhaps a list view of the current Inventory, a complete series of checkboxes for all of those 200+ options...the list can be endless.

But hopefully, you've seen what I've done via the article, and by studying the accompanying asp files and the dbase too, you can figure out what you have to do to use checkboxes as a regular part of web applications!

Creating a Guestbook in ASP

by Chris Perry

ASP Guestbook, of course a simple database can be constructed using following:
****
Database name: guests.mdb
Table name: guestbook
Field names:
name
email
url
message
****

This is my first guestbook. The code is original - and I hope simple and straightforward. The first file is called "guestbook.asp" and is the form the user sees to enter his/her message to the guestbook. If you examine this file, you will see that this form is sent to another ASP page (or script) called "guestbookThankYou.asp".

I have used a simple javascript script for the button which - when clicked - sends the user to the "guests.asp" page - which displays the guestbook entries. I have tested this script and it works great -

* NOTE: Of course, you may need to modify the path for the database, place database in your \db directory.

guestbook.asp

```
<html>
<head>
<title>ASP Guestbook</title>
</head>
<body>

<font face="ARIAL" size="2" color="BLACK"><b>
Welcome to my guestbook...</b></font></p>
<FORM METHOD="POST" ACTION="guestbookThankYou.asp">
Name: <INPUT NAME="name" size="33"><BR>
Email: <INPUT NAME="email" size="33"><BR>
URL: <INPUT NAME="url" size="33"><BR>
Message: <textarea rows="6" name="message" cols="30">
</textarea><BR>
<INPUT TYPE="SUBMIT" VALUE="SEND">
<INPUT TYPE="BUTTON" VALUE="VIEW GUESTBOOK" onClick="location.href='guests.asp';">
</form>
</body>
</html>
```

guestbookThankYou.asp

```
<html>
<head>
</head>
<body>
<p>Thank you for signing my guestbook...</p>
<FORM METHOD="POST" ACTION="guests.asp">
<%
Set cn = Server.CreateObject("ADODB.Connection")
openStr = "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
Server.MapPath("\db\guests.mdb")
cn.Open openStr
SQL = "SELECT * FROM guestbook"
Set record = Server.CreateObject("ADODB.Recordset")
record.Open sql, cn, 2, 2
record.AddNew
record("name") = Request.Form("name")
record("email") = Request.Form("email")
record("url") = Request.Form("url")
record("message") = Request.Form("message")
record("todaysDate") = Now()
record.Update
record.Close
Set record = Nothing
cn.Close
Set cn = Nothing

%>
<INPUT TYPE="BUTTON" VALUE="VIEW GUESTBOOK" onClick="location.href='guests.asp';">
</form>
</body>
</html>
```

guests.asp

```
<html>
<head>
</head>
<body>
<p>Guestlist entries....</p>
<%
openStr = "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
Server.MapPath("\db\guests.mdb")
Set cn = Server.CreateObject("ADODB.Connection")
cn.Open openStr, UserID, Password
SQL = "SELECT * FROM guestbook ORDER BY 'ID' DESC"
Set record = Server.CreateObject("ADODB.Recordset")
record.Open sql, cn
record.MoveFirst
Do While Not record.EOF
Response.Write"<B>Date</B>: " & record("todaysDate").Value & "<br>"
Response.Write"<B>Name</B>: " & record("name").Value & "<br>"
emailValue=record("email").Value
Response.Write"<B>Email: </B>" & _
"<A HREF='mailto:" & emailValue & "'>" & emailValue & "</A><BR>"
urlValue=record("url").Value
Response.Write"<B>URL: </B><A HREF='" & _
urlValue & "' target='blank'>" & urlValue & "</A><BR>"
Response.Write"<B>Message</B>: " & _
record("message").Value & "<br>"
Response.Write"<HR width='80%'>"
record.MoveNext
Loop
record.Close
Set record = Nothing
cn.Close
Set cn = Nothing
%>
</body>
</html>
```

Show The Size Of A Directory Using ASP

This is an example of using the FileSystemObject to determine the total size of all files in a directory and any sub directories in it. This is handy because sometimes you might want to know just how big your web site is. Replace "somedirectory" with your directory name. If you know the physical path you can replace Server.MapPath("/somedirectory") with the actual path like so. "E:\Inetpub\virtuals\somesite\somedirectory"

Here's the catch.
If you are using NT the directories you run this on must have the correct permissions. Basically the directories and all subdirectories within it all need "read" permissions from the anonymous account or you will get a permission denied error.

```
<%
Set MyFileSize = Server.CreateObject ("Scripting.FileSystemObject")
MyPath = Server.MapPath("/somedirectory")
Set MyFolder = MyFileSize.GetFolder(MyPath)
%>

<p><% =MyPath %> is <% =MyFolder.Size %> bytes</p>
```

The results might look something like this.

```
E:\Inetpub\virtuals\somesite\somedirectory is 47331 bytes
```

Show Directory Contents Using ASP

This is an example of using the FileSystemObject to list the files in a directory. Replace "somedirectory" with your directory name.

Replace "somedirectory" with your directory name. If you know the physical path you can replace **Server.MapPath("/somedirectory")** with the actual path like so.
"E:\Inetpub\virtuals\somesite\somedirectory"

Here's the catch.
If you are using NT the directory you run this on must have the correct permissions. Basically the directory needs "read" permissions from the anonymous account or you will get a permission denied error.

```
<%
Set MyDirectory=Server.CreateObject("Scripting.FileSystemObject")
Set MyFiles=MyDirectory.GetFolder(Server.MapPath("/somedirectory"))
For each filefound in MyFiles.files
%>

<% =filefound.Name %>
<br>

<%
Next
%>
```

The results might look something like this.

```
global.asa
search.htm
dir.asp
check_box.asp
simple-listbox-web-code.asp
radio_buttons.asp
size.asp
```

Searching Your Database

If you have a database you'll want to be able to do searches against the records you have in it. With this script you can base your search on either a single field or all your table fields.

First, create the search form:

```
<html>
<body>

<form method="Post" action="searchdb.asp">
<input type="text" name="searchvalue"><br>
<input type="checkbox" name="all" value=1>Search All Fields<br>
<input type="submit" value="Search">
</form>

</body>
</html>
```

That's it for the form. Name this default.htm or whatever.

Now create the results page:

```
<%@Language=VBScript%>
<% Response.Buffer = True %>

<%
'dim your variables
Dim MyConn, SQL, RS, srchval, i

'grab the contents of the form fields
srchval = Replace(Request.Form("searchvalue"), "'", "''")
srchall = Request.Form("all")

'create the Connection object and open the connection to the database
Set MyConn=Server.CreateObject("ADODB.Connection")
'note there are 3 separate MyConn.Open strings: one for dsn-less, one for OLEDB, and one for a
DSN. just pick the one you want to use and comment out the others.
MyConn.Open "Driver={Microsoft Access Driver (*.mdb)}; DBQ=C:\Databases\demo.mdb"
'MyConn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Databases\demo.mdb"
'MyConn.Open DSN_Name

'check to see if the checkbox was checked. if it is then search all fields...
If srchall = 1 Then
SQL = "Select * From tblCustomer Where FirstName Like '%"&srchval&"%'"
SQL = SQL & " Or LastName Like '%"&srchval&"%'"
'otherwise search only the LastName field
Else
SQL = "Select * From tblCustomer Where LastName Like '%"&srchval&"%'"
End If

'execute the SQL statement
Set RS=MyConn.Execute(SQL)

'as long as a value was entered into the search form...
If srchval <> "" Then
'if both begin of file and end of file are true then you know no records were found
If RS.BOF And RS.EOF Then
Response.Write "No Records Found."
```

```
'otherwise...
Else
'build the HTML table to display the records.
Response.Write "<center><table border=""1""><tr>"
'determine the number of table fields (columns) and display the field names
For i = 0 To RS.Fields.Count - 1
Response.Write "<th>" & RS(i).Name & "</th>"
Next
Response.Write "</tr>"
'now loop through the result set of the query and display the results
While Not RS.EOF
Response.Write "<tr>"
For i = 0 To RS.Fields.Count - 1
Response.Write "<td>" & RS(i).Value & "</td>"
Next
Response.Write "</tr>"
'move to the next record
RS.MoveNext
Wend
'now close the HTML table
Response.Write "</table>"
End If
'if no value was typed into the form then redirect the user back to the form
Else
Response.Redirect "default.htm"
End If

'close and destroy your connection and recordset objects
RS.Close
MyConn.Close
Set RS = Nothing
Set MyConn = Nothing
%>
```

That's it!

Password Protecting Your Web Page(s)

Updated: You can now have more than one username and password account set up. Whereas before you could only have one account.

You may have a page or pages that you don't want anyone else to be able to view, such as an administrative page. By using a form, a session variable, a query, and 2 lines of code added to each page you want protected, you'll be all set.

First, create a new table and name it tblLogin. Then create two fields, one named UserName and the other Password. Give both fields a value that you'll use when logging into your password protected page.

When you enter the valid UserName and Password a Session variable will be changed to True allowing you to go freely into any other pages you may have password protected. This Session variable will be checked each time a page you set to be password protected is called.

Next, you create the form that you'll login with...

```
<HTML>
<BODY>

<form name="Login" method="Post" action="login.asp">
<input type="text" name="username" size="20"> UserName<br>
<input type="password" name="password" size="20"> Password<br>
<input type="submit" name="btnLogin" value="Login">
</form>

</BODY>
</HTML>
```

Name this page **main.htm** or whatever you want to name it.

In Part 2 we'll create the query that checks to see if the UserName and Password you typed into the form matches that with the values in the database.

Now we create the query that will check the values you typed into the form with the values in the database.

```
<%@ Language=VBScript %>
<% Response.Buffer = True %>

<HTML>
<BODY>

<%
UserName = Request.Form("username")
Password = Request.Form("password")
'grab the form contents

Set MyConn=Server.CreateObject("ADODB.Connection")
MyConn.Open "your connection string here"

SQL = "Select UserName, Password From tblLogin " _
& "Where UserName = '"&UserName&"' And Password = '"&Password&"'"
```

```
Set RS = MyConn.Execute(SQL)

If Not RS.EOF Then
Session("allow") = True
'if there is a match then show the page
%>
```

Put the contents of your page here.

```
<%
Else
Response.Redirect "http://www.yourdomain.com/main.htm"
RS.Close
MyConn.Close
Set RS = Nothing
Set MyConn = Nothing
End If
%>
'if there was no match then make the visitor try again to login.

</BODY>
</HTML>
```

Name this page **login.asp**

Finally, add the following 2 lines at the top of any other page you want protected. You do not need to add this to the 2 pages you've just created.

```
<% Response.Buffer=True %>
<% If session("allow") = False Then Response.Redirect "main.asp" %>
```

If you didn't name your page main.asp besure to change the above URL to whatever you did name it.
If the session variable equals **False** then the visitor will be redirected to the login page.

Special thanks go out to **Ian Watt** for contributing to the script.

Enjoy!

Log In System

This application is basically an improvement to our Password Protection script.

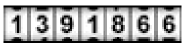
Features include:

- Multiple accounts.
- Different "Clearance Levels". Each page you want protected is assigned a level (1 - 3). Level 3 being the highest level, which is normally reserved for the site administrator (you).
- Accounts can expire.
- To protect a page simply add an include file (Level1.inc, Level2.inc, or Level3.inc) at the top of the page. How much simpler could it be?!

Drop the files into a folder and it's ready to go!

Page Counter

By:
Subhendu Mohapatra
Freelance Web Designer
www.plus2net.com
smo@plus2net.com



Counters are popular among the web developers. They are used to count the number of visitors (or say hits) to a particular page. We can program a attractive graphical counter like this one at the top or a hidden one whose reading can be sent to the owner by email. A judicious use of counter in different pages will help the designer to improve the site. If more visitors are visiting help section then some more details are to be provided in pages for the visitors. Which is the most popular section in your site? How much you should charge for an advertisement in your page? Answer to all this and many more is to use counters.

A few lines of code in ASP will do this for us. This code can be kept inside the page of our interest or can be stored in a separate file (say counter.asp) which we can execute from our main page. We require a text file to store our counter readings (say counter.txt). Our code will read the value from this text file and will update this text file with new incremented value. So a write permission is required for this directory to update the text file. First create counter.txt file and store any initial value in this.

First let us create some variables and we will use the MapPath method to converts a virtual path into a physical path.

```
<%

Dim  strPath, File, file2, x,liCount, j ,resfile, d, i ,fl

strPath = "counter.txt"

strPath = Server.MapPath(strPath)
```

We will use file system object to read and write our counter.txt file.

```
Set File = Server.CreateObject("Scripting.FileSystemObject")

set file2=File.OpenTextFile(strPath)
```

This will create instance of the file system object and a file object to access the file represented by strPath.

```
x=file2.readline
```

x stores the old value of the counter

```
set resfile = File.createTextFile (strPath)

resfile.writeline(x+1)
```

This will open the counter file for writing and the value x+1 stored in the file. We can display the value by Response.Write (x+1). But we will try to display them in a very attractive way. For this we have to create some attractive digits from 0 to 9 and store them in different files like digi0.gif, digi1.gif, digi2.gif, digi3.gif to digi9.gif .

This can be created by using any graphic software or can be downloaded from some sites. One of my favorite is digitmania.com but you can get some more through your search . We have to count the number of digits our counter present reading is having and to do this we have to convert our numeric value to String

```
liCount=Cstr(x+1)

j = Len(liCount)
```

We have to decide how many digits our counter will have and we will store the difference between this and the value of j in another variable d. The value in d will display the number of zeros we have to display to the left of our counter value. For example to display a value of 435 in a 9 digit counter we have to place 6 zeros to the left of 435 and our counter should display 000000435.

```
d=9 - j

For i=1 to d step 1

      Response.Write ("<IMG SRC='digi0.gif'>")

Next
```

This will display digi0.gif (digit 0) number of times equal to the value stored in d. Now this is time to display our counter value.

```
For i=1 to j step 1

  fl="digi" + Mid(liCount, i , 1) + ".gif"

  Response.Write ("<IMG SRC=") & (fl) & ">"

 Next
```

This will select the digit from our counter value and display them from left to right. Finally our counter will look like this

If you are good in graphics then you can design many attractive digits matching your background and text color you used.

```
<%

Dim strPath, File, file2, x,liCount, j ,resfile, d, i ,fl

strPath = "counter.txt"

strPath = Server.MapPath(strPath)

Set File = Server.CreateObject("Scripting.FileSystemObject")
```

```
set file2=File.OpenTextFile(strPath)

x=file2.readline

set resfile = File.createTextFile (strPath)

resfile.writeline(x+1)

liCount=Cstr(x+1)

j = Len(liCount)

d=9 - j

%>

<%

        For i=1 to d step 1

         Response.Write ("<IMG SRC='digi0.gif'>")

        Next

  For  i=1 to j step 1

   fl="digi" + Mid(liCount, i , 1) + ".gif"

   Response.Write ("<IMG SRC=") & (fl) & ">"

 Next

%>
```

Forms - Populating radio buttons with info from a database

Below is an example of populating radio buttons from a field in your database.

This is just an example.. when doing this for real you would do a query to determine the existing values for all the form fields... for this example we will populate the variable ourselves with one of 3 possible values.. "Extra Cheese" , "Pepperoni" , or "Sausage".

```
<%
Topping1 = "Pepperoni"
%>

<form>
<div align="center"><center><table border="0">
<tr>
<td colspan="2" align="center">Topping 1</td>
</tr>
<tr>
<td><input type="radio" value="Extra Cheese" name="Topping1"
<% If Topping1 = "Extra Cheese" Then %><% Response.Write(" checked") %><% End
If %>></td>
<td>Extra Cheese</td>
</tr>
<tr>
<td><input type="radio" value="Pepperoni" name="Topping1"
<% If Topping1 = "Pepperoni" Then %><% Response.Write(" checked") %><% End If
%>></td>
<td>Pepperoni</td>
</tr>
<tr>
<td><input type="radio" value="Sausage" name="Topping1"
<% If Topping1 = "Sausage" Then %><% Response.Write(" checked") %><% End If
%>></td>
<td>Sausage</td>
</tr>
</table>
</center></div>
</form>
```

And this is what the output would look like if the field had the value "Pepperoni".

Topping 1

○ Extra Cheese

● Pepperoni

○

Forms - Populating a drop down menu with info from a database

Below is an example of a basic drop down menu populated from a database.

```
<%
' Not neccesary but a good habit
Dim DataConn
Dim CmdPopulateStates
Dim SQL
%>

<%
Set DataConn = Server.CreateObject("ADODB.Connection")
Set CmdPopulateStates = Server.CreateObject("ADODB.Recordset")
%>
```

Take away the comment on Sytem DSN version below if you want to use a system DSN instead and add a comment to the DSN-LESS connection version below it

```
<%
' DataConn.Open "DSN=System_DSN_Name"

DataConn.Open "DBQ=" & Server.Mappath("_database/zipcodes.mdb")
&  ";Driver={Microsoft Access Driver (*.mdb)};"

SQL = "SELECT DISTINCT STATE_NAME FROM STATES"
CmdPopulateStates.Open SQL, DataConn
%>

<form method="POST" action="somepage.asp">
<Select Name="STATE_NAME" size="1">
<%While Not CmdPopulateStates.EOF%>

<option value="<%= CmdPopulateStates("STATE_NAME") %>"><%=
CmdPopulateStates("STATE_NAME") %></option>

<%
CmdPopulateStates.MoveNext
Wend
%>

<%
' Not neccesary but a good habit
CmdPopulateStates.Close
Set CmdPopulateStates = Nothing
DataConn.Close
Set DataConn = Nothing
%>

</Select>
<input type="submit" value="Submit">
</form>
```

And this is what the output would look like.

Below is an example of a more advanced drop down menu populated from a database. It is intelligent and if the database record already has a value for the field it will make sure the correct value is selected in the menu.

```
<%
' Not neccesary but a good habit
Dim DataConn
Dim CmdPopulateStates
Dim SQL
Dim CURRENT_STATE_NAME
%>
```

This is just an example.. when doing this for real you would do a query to determine the existing values for all the form fields. For this example we will simply set a variable to the value for demonstration purposes.

```
<%
CURRENT_STATE_NAME = "New York"
%>
```

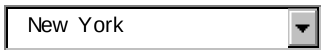
```
<%
Set DataConn = Server.CreateObject("ADODB.Connection")
Set CmdPopulateStates = Server.CreateObject("ADODB.Recordset")
%>
```

Take away the comment on Sytem DSN version below if you want to use a system DSN instead and add a comment to the DSN-LESS connection version below it

```
<%
' DataConn.Open "DSN=System_DSN_Name"

DataConn.Open "DBQ=" & Server.Mappath("_database/zipcodes.mdb")
&  ";Driver={Microsoft Access Driver (*.mdb)};"

SQL = "SELECT DISTINCT STATE_NAME FROM STATES"
CmdPopulateStates.Open SQL, DataConn
%>

<form method="POST" action="somepage.asp">
<Select Name="STATE_NAME" size="1">
<%While Not CmdPopulateStates.EOF%>

<option <% If CURRENT_STATE_NAME = CmdPopulateStates("STATE_NAME") Then %>
<% Response.Write(" selected ") %><% End If %>value="<%=
CmdPopulateStates("STATE_NAME") %>"><%= CmdPopulateStates("STATE_NAME")
%></option>

<%
CmdPopulateStates.MoveNext
Wend
%>
```

```
<%
' Not neccesary but a good habit
CmdPopulateStates.Close
Set CmdPopulateStates = Nothing
DataConn.Close
Set DataConn = Nothing
%>

</Select>
<input type="submit" value="Submit">
</form>
```

And this is what the output would look like.



Forms - Populating a check box with info from a database

Below is an example of populating a check box for use with a Boolean (Yes/No) field in your database.

This is just an example.. when doing this for real you would do a query to determine the existing values for all the form fields... for this example we will populate the variable ourselves with either a "True" or "False" value. If the value passed to the save page is not "True" then it is "False" and therefore you update the database accordingly.

```
<%
FRONTPAGE_ACCESS = "True"
%>

<form method="POST" action="somepage.asp">
<div align="center"><center><table border="0">
<tr>
<td>FRONTPAGE_ACCESS</td>
<td><input type="checkbox" name="FRONTPAGE_ACCESS" value="True"
<% If FRONTPAGE_ACCESS = "True" Then %><% Response.Write (" checked")%><%
End If %>></td>
</tr>
</table>
</center></div>
</form>
```

And this is what the output would look like if the boolean field was "True".

FRONTPAGE_ACCESS ☑

Below is an example of populating a check box for use with a text field in your database where you have manually inserted text like"Yes" or "No".

This is just an example.. when doing this for real you would do a query to determine the existing values for all the form fields... for this example we will populate the variable ourselves with eithor a "Yes" or "No" value. On the save page if the field passed doesn't = "Yes" then it is the opposite so in this case you assign the database the "No" value on the save page

```
<%
CGI_ACCESS = "No"
%>

<form method="POST" action="somepage.asp">
<div align="center"><center><table border="0">
<tr>
<td>CGI_ACCESS</td>
<td><input type="checkbox" name="CGI_ACCESS" value="Yes"
<% If CGI_ACCESS = "Yes" Then %><% Response.Write (" checked")%><% End If
%>></td>
</tr>
</table>
```

```
</center></div>
</form>
```

And this is what the output would look like if the text field has the text "No" in it

CGI_ACCESS ☐

Defining the Problem

Shhhh. As we peer through the window, we can see a figure hunched over a computer across the room. There is a steady drumming noise, like the rat-tat-tat of a machine gun. As our eyes adjust to the dim light, we can see the blur of fingers dancing across the keyboard.

Finally, the typist leans back, yawns, and stretches. Then she grabs a pack of cigarettes off the desk, and heads out of the far door. This is our chance. We slip into the room. There is the bookshelf of web design books, arranged by the color of their spines. Printouts of HTML code and e-mails, and a stack of CDs clutter the desk. Dilbert cartoons adorn the side of the monitor. On the wall, a large whiteboard is covered with boxes and arrows and scribbled lists: "Need to fix: broken links, search, left navigation..." A URL in bold lettering is splashed across the top, followed by a colon, and in all caps, the word 'RELAUNCH'. Next to the whiteboard is a large calendar with a big red circle; some sort of deadline next week.

We look, finally, at the monitor. The cursor blinks in an e-mail window, and we can read the last few paragraphs of a long, rambling rant. The sentences are sharp, bitter, and weary. I print out the e-mail – that's all the evidence we need: another confirmed sighting of the increasingly stressed, beleaguered, web professional.

Back at the office, you read over the e-mail. It's a sad, familiar story. "If only I had a dollar for every developer I saw suffering like this," you say.

"If only I had a dime for every broken link on her site," I reply. We get to work on our report.

Web Sites Are Hard Work

I'll start with a statement:

The care and feeding of the average web site has grown (and keeps growing) far beyond the capability of most web professionals to handle on their own.

This is not necessarily a bad thing – it means that there is demand for the work that we do. While it's not exactly job security, it is nice to be needed. However, the days of a lone webmaster wrangling all aspects of a web site are long behind us.

Even if you think you have been doing a pretty good job of managing your web site up till now, web sites will continue to be challenging work, at least for the foreseeable future. Let's explore the reasons why.

Tidal Wave of New Content

The amount of content that your group or company wants to make available on the Web will continue to grow. Old content doesn't die; it just goes into the archives. And the spread of technology means that the tools to create content – music, video, and plain old-fashioned text – have become ubiquitous. Like it or not, people have always had the will to communicate and now they have the means. Your job is to get it up on the Web.

"But my web site is small", you say. And here's where I respond with one of my aphorisms: inside every small web site is a huge web site struggling to get out. There is probably a lot more content that your company or group wants to get onto the Web, if it could.

Some of you may be skeptical and unwilling to take my little truism to heart, so let's do some math. How many authors do you have? How many items (news, press releases, short tidbits, calendar events, articles, etc.) does each author create or change per month?

- 3 authors @ 5 items a month: 15 updates a month; 180 per year.
- 5 authors @ 10 items a month: 50 updates a month; 600 per year.
- 100 authors @ 12 items a month: 1200 updates a month; 14,400 per year.

If we assume that we will continue to have more authors, posting on average the same amount of content, the curve of the graph will continue to go up, year on year.

It's not uncommon for sites created just three or four years ago to have more than fifty thousand pages. University web sites, with their sprawl of student and faculty publications, easily range upwards of one hundred thousand pages. Today's businesses are incredibly information intense. Corporate intranets can often have page counts in the millions.

There is no doubt that the amount of content will continue to grow. The only real question is whether you will be able to keep up with the load, the way you work now. If not, it's time to start thinking strategically and develop a plan to co-ordinate the production of the site.

An Ever-Changing World

Time doesn't stand still. A number of external factors make developing web sites more and more challenging. As we wait for final approval on the press release copy, technology presses on. Browser upgrades, plug-in incompatibilities, new wireless platforms, and the latest W3C recommendations come raining down. It's difficult to keep up with it all – let alone implement best practices for your web site.

When web sites carry information that is more current than other media, it can change the flow of information in an organization. Previously, some web sites may simply have duplicated print publications in an online form. Now, those web sites drive the production of print documents. One site may now export the company's print product catalogue; another may produce the documents that an integrated voice response system reads to telephone callers.

Although this book is very web-focused, content management should not be applied to web sites alone. There are many paths and destinations for content, and your content management system should be able to support the paths and platforms you choose. Technology changes the way an organization creates and manages content – it has also expanded the scope of the web development team, and raised the bar for the web professional.

Sometimes change forces its way onto your site. A search engine upgrade or adding a personalization feature to the site may have a wide impact. For instance, converting every HTML file on the site to a Java Server Page (JSP) could cause a few late nights.

As your company moves into new endeavors, it may run across new legal requirements in disclosing information about their products. A contract with a government agency may impose stricter accessibility requirements on your web site. Financial regulations may require certain types of disclaimers to be added to product pages. In manufacturing shops, ISO-9000-compliance requires you to be able to roll back your product sheets to earlier revisions.

Your organization may begin expanding its reach around the world and your web sites need to reflect this new diversity of language and culture. Being able to manage web sites in different languages, different currencies, and different cultures will certainly impact how you develop and plan a web site.

Increased Expectations

There is always a demand for new features or changes. Managers want to add press releases. Designers want to refresh the look and feel of the site. Usability experts want to fix the navigation. The legal section is talking about compliance to new accessibility laws. The backend geeks are suggesting an application server and new database for reliability. Finally, the CEO wants the logo to "do a rollover and match my favorite tie" (we're trying hard to ignore him).

Users, too, are becoming accustomed to features they've seen elsewhere – on sites like Yahoo (www.yahoo.com) or Amazon (www.amazon.com), for instance. They have higher expectations, in terms of clear site navigation and a comprehensive search facility to find what they are looking for. They want to see what is new, what is popular, and what is relevant to them.

While the expectations rise and rise, the time frames get shorter and shorter. If we're going to keep up, we're going to have to utilize the best tools and processes available to us.

Success can raise expectations as well. A web site that launches and gets an enthusiastic response from its target audience can often propel an organization to create more web sites. You may find yourself creating multiple sites, with the requirement that they share content and stay in sync. The technical challenges have increased substantially, and the organization still expects the same or better rollout of the new sites.

The truth is, maintaining and growing web sites is hard work. It's not getting any easier, and web professionals and the organizations they work for are feeling the pain.

The Litany of Pain

Let's take a look at the main problems that plague web sites and, by extension, their authors, web professionals, and users. The effects are often cumulative and may follow the course we'll outline here.

Bottlenecks

First, there are bottlenecks caused because each new page of the web site has to be touched by someone. Say, the page is manually coded into HTML. Templates help, a little, but the number of new pages being asked for guarantees that the HTML coder runs the risk of becoming a bottleneck, falling further and further behind schedule.

Unintended Responsibilities

These bottlenecks create an unintended and undesirable social side-effect. As content contributors wait for you to update their content, they begin to understand that they don't control the site; you, as the site manager, do. They begin to perceive the web site as your sole responsibility and not theirs. Once they begin to identify the site as "Somebody Else's Problem", they become less inclined to contribute to it, or help fix it. Suddenly you, as the site manager, are not only responsible for the technical development of the site, but for its content.

This has disastrous consequences. If you are micro-editing/correcting the content of the site, you'll never get through the backlog of page production. You'll never have the breathing space to think strategically and tackle the more interesting, compelling problems. Worst of all, your authors will feel isolated from the site, and you'll lose the interest and support of the group of people you most need in order to make it a success.

Stale Content

As the backlog of HTML pages grows, the content that's already up on the site gets old and "stale". Updates to the home page become less frequent, so people stop checking it. Information never gets updated, so the data on the web site is deemed unreliable. People start bypassing the site altogether. You find HR sending out the new vacation policy to everyone in the company as a Word attachment because it's easier than waiting for a web page to be updated.

When the marketing folks want to change the design of the site, the web team will tend to dig in their heels. They're behind as it is, and don't have the time. Even simple site design changes can be painful, requiring a couple of weeks of overtime from the team. Of course, during that time, it's nearly impossible to do regular site updates. Your Perl guy is laboriously tweaking the script he wrote, changing "Register" to "Sign up" on every page of the site.

Inconsistency

The Perl guy is worried because he knows the site is inconsistent. There's been a succession of programmers, who all preferred their own tools and have their own coding styles. Different areas of the site handle navigation slightly differently and the site doesn't feel cohesive.

There's always one person on any team who can recite the history of their web site. "This," they may say, "works weirdly because we hired that one contractor before the Christmas redesign two years ago..." There were 'good' reasons at the time why short cuts were taken, and corners cut, but the team just never had the time to go back and clean up. So the site struggles on, somehow working, but underneath it's a patchwork of hacks.

The inconsistency isn't always hidden behind the scenes. A company's brand identity often suffers online. Without a web-oriented style guide and a template system that enforces it, the presentation of the company brand can be distorted by business users whose expertise is in content, not in managing and protecting the brand.

There is another reason for inconsistency. Often the site grows piecemeal, with different groups within the organization publishing subsections in an ad hoc manner. Little fiefdoms of power emerge, and there is a lack of clear leadership.

Low Quality

The inconsistency ultimately manifests itself as a low quality site. This has two separate effects:

q The site just doesn't look good or work well – reflecting poorly on your team and your organization. It's not fulfilling its potential to communicate, and this translates, in the language of the bean counters, into poor return on investment (ROI).

q The site is no fun to work on. Low quality hurts the morale of your web team. If they don't feel that they are working on something that is inherently worthwhile, then it is hard for them to care about it, and if your web team doesn't care about the quality of the site, who will?

Inefficiency and Inflexibility

All the above problems add up to a site that is difficult to maintain. Its fragility makes its owners (rightly) fearful that it will break, so they hesitate to make improvements. Time is spent fixing broken links and tweaking design, page by page, that could be better spent elsewhere.

A brief example: I once consulted with the web team of a large, high-profile web site. Posted on the wall was a chart where they tracked the number of broken links (found using third-party link-checking tools). They had a campaign to scour the site and fix these links by hand. I took this to be a **bad** sign. While they should be applauded for doing their best to fix problems on the site, it was a terribly inefficient way for the team to be spending their time. Yet without a content management system in place, this is what they had to resort to.

The Root Problems

We've looked at the reasons why web sites are hard work, and the problems behind them, but *why* do these problems arise?

The root cause of many of these symptoms is that the site uses a **handcrafted approach**, which doesn't scale to bigger teams or larger amounts of content. In the Internet Age many of us are still, curiously, using production techniques from an earlier era.

Coding pages in HTML by hand is a useful skill. In some cases, it approaches an art. However, coding *every page* in HTML by hand is a tremendously inefficient and unpleasant way to spend your working days. Manual production of web sites is a leftover from a simpler era (1994) and it doesn't make sense for a modern production site.

Those of us who've spent so much of our creative energy in building the Web may be uncomfortable with applying Industrial Age automation techniques to our web sites. After all, doesn't it cheapen and devalue the skills we've developed, and isn't there a danger that we'll be replaced by machines? The short answer is that automation hopefully will lighten your load of monotonous, repetitive work, and free you to plan and implement redesigns.

A notable legacy of this handcrafted approach is **inter-dependency between design, code, and content.** Most often this is because, early on in the web site's life, one or two people did everything. Today, this is seldom the case. While core web development teams are still small, successful developers work alongside writers, editors, designers, and production artists to create their web sites. The effort is larger, more collaborative, and more complicated.

When design, code, and content are mixed together, it becomes extremely difficult to manage or change them independently of one another. If your web team audibly groans when a redesign effort is announced, it's probably because they intuitively understand that the process will be slow, painful, and error-prone. But a redesign should not require weekends of hand coding. Although it takes more effort to keep content and design separate, the payoff during the eventual site redesign is well worth it.

Another legacy of the handcrafted approach is **lack of a production process**. There is little need for a process when you are the only one working on the site, but the instant you start working with others (developers and content authors, for instance) you need to define a process and have the tools in place to enforce that process.

Finally, **no site can thrive unless there is a clear vision of its message and audience**. While not a technical problem, lack of strategy and direction influences the quality of the technology, content, and design choices made throughout. Let me rephrase: if you don't know what you want, you're not going to get it, but you will waste a lot of time, effort, and money trying to do so.

How This Book Can Help

It's no fun to work on a web site that doesn't work. This book is about how to stop the hurting. It proposes that a philosophy, methodology, and practice known as content management will provide web professionals with hope for their projects, and the pride of shaping a web site that not only works, but is continually getting better and better.

Content management applies technology to automate the most tedious parts of the old handcrafted approach. It helps you define a system for maintaining your site designs separately from your server code, which is kept separate from the content created by your authors. It provides the means and the opportunity to make your site into what you want it to be: usable, attractive, localized, accessible, fast, and up-to-date.

Our task in these pages is to explain what content management is, and how to apply basic principles of content management to your web site. Ultimately, I hope to help you make good choices about content management.

Why Don't People Use Content Management?

In the next section, we'll define what content management is, but let's just pause to ask ourselves why more people don't have content management systems in place already.

A Leap in the Dark

Committing to a particular content management strategy is somewhat akin to getting married, in that, hopefully, the choice will work for a lifetime. An organization may be inexperienced with content management, however, so they have to commit to a substantial relationship without understanding exactly how it will impact them.

It's Hard to Change

It takes effort, courage, and money to change the way things work. Frankly, although people understand that the changes need to be made, the status quo is preferable to an uncertain future.

Planning and implementing content management is a high-level strategic activity. Selecting a content management package is perceived to be a job for CTOs or the heads of the IT department, largely because of the high cost of the tools and because these are the people such tools are marketed at. Additionally, many web professionals are so busy trying to stay afloat that they don't have the time to consider a content management system.

However, as a web professional, you'll often have the best view of the challenges and impact content management will have, so don't be afraid to speak up and help shape the decisions about content management. No matter what happens, a content management system will impact your work significantly, and it's best to get your input in at the start.

Content management affects many people in an organization. Getting all of those people moving in the same direction and working together is hard work. The impact on the organization is often not widely understood – depending on your goals, it's possible that everyone in the company will be affected.

The Market is Confusing and Immature

For those who have decided to buy a solution, there are many vendors trying to carve out a niche in this growing market. Many agencies that have written a CMS for a client are now packaging it as a product. Some of these are nowhere near 'product' quality, and those that are industry leaders are, in many cases, only there because they made the jump early.

It is difficult to compare the content management products available because of the widespread use of jargon, buzzwords, and marketing babble. While there are some basic similarities, the approach and metaphors used by one software product often do not match precisely those of another. Content management tools may only implement a portion of a full content management strategy, but market themselves as a full end-to-end solution. This confusing and complex marketplace leaves customers, quite rightly, frustrated.

There is also trepidation about picking a product that will be a winner. In uncertain economic times, in such a competitive, untested marketplace, it is hard even to know who will be around next year. We have yet to see which companies will do well and survive, and which will fade away.

Products are also distributed unevenly. Products available in Europe may not have support in the US, while support for CMS products throughout Asia is generally weak.

Content management vendors are on an uneven footing when it comes to advertising their products. The big vendors can afford to saturate the market with advertising (one prominent US vendor sponsors the replay at my local ice hockey arena). Smaller products – even if they have a reasonable feature set and price/performance ratio – are often lost in the noise of the market. It's hard for vendors to differentiate themselves.

Furthermore, the vendors are experimenting with all sorts of business models. Some license their products per server or per CPU; I've even seen a license based on the total CPU speed in megahertz. Others have a base server fee on top of which they charge per-seat costs. Furthermore, per-seat costs may be based on concurrent users or include a separate fee for developers and administrators.

There are open source consultancies that charge for significant professional services. Other products have a hybrid open source model – they still charge for their product, but you do have access to all the source code, and they have an explicitly open development model. Then there are application service providers, who will host a CMS for you that they have built or licensed from a vendor.

The result is that licensing costs are difficult to compare. (The one common license element I've seen from commercial vendors is a yearly maintenance and support fee that averages about 20% of the initial license fee.)

Content Management Can Be Complex

Ultimately, the practice of content management is about getting disparate parts working together. Those disparate parts are both the people involved – authors, editors, and developers – and the technology that supports them. Content management is a group activity, and requires collaboration to succeed. Putting in place a technology infrastructure that will support that collaboration is non-trivial. It requires co-ordination across disciplines.

In many organizations such inter-departmental cooperation is unheard of. And even though this is a failing of the organization and not one of content management, this is the situation into which content management may well be introduced.

Content management is also complex because it tries to prepare us for an uncertain future. There is no guarantee that we will use the same browsers and systems to view our content in the future. The compromises that need to be made between preparing for the future and still getting something that works today (and on today's budget) are not easy to make, and sorting through the various options can be complicated.

Defining Content Management

Most people encountering the term *content management* for the first time are confused by the various definitions. In some ways the expression has completely lost meaning, because it's used by so many different products to mean (often) completely different things.

It is probably best to think of content management as a broad concept that covers all aspects of publishing content with digital tools. But a 'broad concept' does not begin to approach a definition that we can understand and use.

The Activities of Content Management

Content management can be defined more precisely in terms of activity: it is appropriate to ask, "What do I need to do to manage my content?"

In its simplest form, content management does just three things:

q Asset Management: Organizing units of content

q Transformation: Presenting that content

q Publishing: Delivering the content to your audience

Personally, I call each unit of content an asset. Day in and day out, we *create* and *manage* these assets. In order to get those bits that seem worth broadcasting to a wider audience published, we submit them to a process I call asset management, which formalizes and prepares the assets for the next steps.

Don't confuse asset management as used here with Digital Asset Management (DAM), which usually refers to the cataloging and storage of multimedia – movies, audio, and high quality photographs.

Once we have some content assets available, we then make choices about how to present that content. Usually we have some sort of design, and we attempt to shoehorn our content into those design templates. Indeed, most people refer to this process as templating, but I prefer the words content transformation. "Clothes make the man", the saying goes, and I think that the right application of design to content does more than just make it look attractive; it enhances its effectiveness and impact.

Once we have created the right content, and dressed it properly, all that remains is to deliver the message. This publishing step considers our audience and makes sure that the content is available to them, in whatever formats they may need (HTML, WAP, database feeds, for example) for various devices (browsers, wireless devices, or legacy systems). This phase is primarily technical and logistical. It deals with getting the transformed content out to the intended audience. This might mean deploying static web pages, or updating a content database for an application server. The publishing phase is heavily impacted by the choices made in the previous two activities: we either enjoy the fruits of our labor or learn from our mistakes.

While the focus of this book is on web site development, content management has benefits for all possible output formats. A well-implemented content management tool can become a company's primary system for publishing to print, web, CD-ROM, and wireless platforms.

One thing to note: as you progress from asset management to transformation and finally to publishing, progressively fewer and fewer people are involved in the process, while at the same time the level of technical knowledge needed by these individuals increases.

These three activities will form the basis of the following three chapters (Chapters 2, 3, and 4), where we will explore them in more depth.

Workflow

The glue between the human processes and the technical infrastructure of content management is known as workflow.

Workflow is a predefined series of tasks that facilitates the progression of content assets through the asset management, transformation, and publishing activities.

Workflow makes human processes explicit and systematic. For example, your organization decides that authors will have their work checked by an editor, who then sends it to be approved by the legal department, after which it is published. Workflow simply codifies that process.

Content management tools have a variety of approaches to workflow. But the basics are the same: the application of *rules* on *users* and *tasks* to enforce a process.

While the basics are the same, they will be applied differently in each organization. It's important to have a workflow tool that is flexible enough to accommodate your needs. Needless to say, if you don't have a process defined, or don't understand the process, no amount of workflow tools will help you.

It's important to realize that workflow extends outside the world of content management. Content management may just be a small subset of tasks inside a much longer and more complicated workflow in your business.

A good workflow tool will automate the mundane reminders that shepherd the content assets through the entire process. It will generate a log so that we understand what is going on, what the status of each content item is, and where the (often human) bottlenecks exist.

Technical Infrastructure

Every content management tool is structured slightly differently, but the basics are the same.

Users typically add content via a browser-based interface. This interface often mirrors the look and feel of the web site, and provides rich text editing capabilities. The interface also allows users to edit existing content, look at and compare versions of content, and to approve content for publication – all the activities of asset management are handled within this interface.

There are other ways to add content. Some systems offer integration with desktop applications such as Microsoft Word or Macromedia Dreamweaver, for instance, while others provide a custom client-side application – either for a richer editing environment, or for administrative tasks. An application programming interface (API) may also allow technical users to add or manipulate content that is stored in the repository, via scripts.

That repository can take many forms: a database, a closed file system, or a mixture of both. It can even be a virtual repository – one interface to numerous backend data sources. The repository stores both the content and any associated metadata.

Metadata provides information about the content. For instance, common metadata values would be the author of the content, the date it was created, the date to publish it, and the date it expires. Metadata provides context to the content and enables the content management tool to deliver more precise searches, generate topic-based navigation, create links to related pages, and track workflow status.

A template engine applies design elements to content, in order to produce the desired output document. The templates themselves usually contain placeholders for content from the repository. More powerful schemes can allow inline code to be interpreted in the templates. While more flexible, they are also more complex.

Link management refers to how the tool tracks and maintains internal links and site navigation. Sometimes this is driven by an internally maintained glossary of unique content IDs. Other systems handle it by referring to a user-created site structure, which is also used to create navigation. The site structure is simply the hierarchy of files and folders within your web site. (This can be a logical hierarchy, or physical.) Much is made of the separation of design and content; structure is also an element of your web site that needs to be maintained.

The site is then published and deployed. Publishing can occur ahead of time, that is, pages are created statically (a more colorful term for this is 'baking'), and then deployed to the server. The dynamic approach (called 'frying') is in reverse: content is first deployed to the server (probably to a database). Then, when a request for a web page is received, the template engine is invoked to apply the design. The difference is primarily in server load and administration: a dynamic server requires more processing power, but is useful for presenting constantly changing or frequently updated content.

Workflow helps manage the flow of content through this technical structure, and makes sure that the human structure approves and is aware of what happens throughout. Version control helps people track changes to the content, providing a safety net to roll back changes. The administrator can assign users and groups to certain roles and actions: this is sometimes referred to as user access control levels (ACL).

Products that offer this technical infrastructure are commonly referred to as content management systems (CMS).

Tools or Systems?

The term "content management system" is often used generically to refer to products that offer the basic technical infrastructure: a repository, template engine, and an asset management interface. However, it is sometimes useful to draw a distinction between the tools or products and the broader context – or system – in which those tools operate.

We'll discuss such tools, and the process of buying and building them, in Chapters 5 and 6.

You may want to think of the products that are marketed as "content management systems" as content management tools (or CMS tools). Making this distinction leaves room for a broader definition of content management systems themselves as the organizational context in which the activity of content management occurs. It has three qualities:

It Occurs in the Business Environment

The surrounding environment includes the business' goals, company culture, and the decision-making processes. Content management does, in some ways, reflect the people who undertake the activity. It does not happen in a "clean room".

This has a couple of corollaries. First, content management is **a people-centric activity**. In the broadest sense, as organizations and people, we use web sites to communicate. While that communication is not the same as speaking on the phone or writing a letter, it is, ultimately, person-to-person communication. Content management is about facilitating the technical aspects of this type of communication.

The second corollary is that content management is also **a people-intensive activity**. Although the focus is often on the technology and the products, the aim is to apply that technology to help people get their jobs done faster and more efficiently.

In this book we will examine the technology, but we also talk about our experience with applying technology to real human problems. This means that you'll need to train people, and explain what the content management product you purchase or build does. You'll need to set expectations, and you may have to do some internal evangelism.

Successful content management implementations always take account of the human aspect.

It Is a Set of Processes

The processes involved marry and merge the human and the technical. While most vendors would have you believe that their technology can solve all your content management problems, the nature of the problem is such that a product on its own cannot solve the problems you may have with your people processes.

In any organization, there are agreements about how to get work done. Content management is no different – it is an agreement between people. An agreement or contract concerning how things get done is not new to business or programming. But in content management the agreement has to be more explicit and the enforcement of the process needs to be programmed into the workflow.

This workflow enforcement can also change the nature of the agreement. In many cases this can be an improvement – a simpler process can emerge, for example. In other cases, the technical enforcement of a poorly conceived workflow serves merely to calcify it.

Business processes change all the time. Once workflow is introduced, and people rely on it, it's important that the workflow reflects the current business agreements. It follows that accurate, flexible workflow tools are a must.

It Is an Infrastructure

This infrastructure supports the many mundane tasks of content management. For instance, reminding people to submit content, approving the content, and pasting it into a template. By using tools to automate as much of that as possible, we can support the most important aspects of content management and allow people to work at a higher level.

Authors can focus on what they write, rather than on waiting for a production person to make changes for them. Designers can think about site navigation, usability, and redesigns, rather than tedious, repetitive HTML coding. Developers can tweak the system for performance and improve their code, rather than getting called in to fix a broken deployment of the site.

Premeditation

Despite what the sales people will tell you, content management cannot be installed as an option, with some new piece of software. Planning is the key to successful content management, and you can't buy planning in a box. Content management is a deliberate process. This may seem obvious, but let me just make it clear that you *have* to plan.

Planning involves:

- Thinking about and identifying what your problems are
- Listing what is needed to solve these problems
- Developing a process to move forward and meet your requirements

It consists of knowledge of the issues, and a philosophy of how to handle problems. It implies some understanding of technology, and how to apply that knowledge to solve problems.

Implementing content management requires lots of choices. Do you buy a product, or build your own? How much training will be necessary? Which technology approach is the best fit for your organization?

From your plan will emerge a requirements document describing what your needs are, and a process to guide you. The actual choices of content management will then be much clearer.

ASP Chat Script

By: Kumar Shanmuganathan

INTRO

Many of you people out there will be wondering how a chat program functions, well through this article I will illustrate this to you. With the help of the new powerful web programming language Active Server Pages (ASP) and with the aid of the scripting language VBScript this can be done very easily as shown below.

OVERVIEW

In order to create a simple chat program, we will need to collect 2 pieces of data. They are both alphanumeric data. That is the "Chat name" with which the user wishes to chat with and "User Messages", these of course are the messages that the user sends to other users to chat with. Then we need to store these two pieces of data somewhere for later access. Then we will have to use these two pieces of data and display them, as the user will be entering the messages simultaneously. Finally, we will have to logout the user and display messages on the active chat window, displaying messages about user's entry and exit. Well let's get started and witness how this works.

Part 1

Firstly lets get the user's chat name and store it somewhere. This is all done in the main page, this is because, it is the first page the user accesses to enter the chat, so we will have the get the chat name from this page. To do this, create a new asp file and name it default1.asp. Now as we have created the main page lets get the chat name. In order to do this we need to declare a variable so that the user's chat name can be stored somewhere, this is done like this:

```
<%
Dim chatname
```

Now we have got a set place to store the user's chat name called 'chatname', but since active server pages utilises server-side scripting, we will have to store this chat name on the user's computer. As ASP uses VBScript and it supports the use of cookies, this can be done easily. To so this we use the 'request.cookies()' function. But wait, we have only declared a variable 'chatname', however we have not stored any information in it so it will be empty. Therefore, where will we find out the user's chat name? Well we will have to collect it from the user. We do this using a html form. As we will be collecting the user's chat name from the form and storing it in a cookie on the user's computer, we will have to ensure that the cookie has no previous data in it. Therefore we use an 'IF/ELSE' structure to check the above condition is true like this:

```
IF len(Request.Cookies("chatname")) = 0 AND len(Request.Form("chatname")) = 0

THEN

%>
```

As you can see, a 'THEN' statement is also added here. This is done so that if the cookie named 'chatname' has no value in it, then it can proceed to get a value from the html form like this:

```
<HTML>
<HEAD>
<TITLE>Welcome to Kumar's Online Chat</TITLE>
</HEAD>

<BODY>
<P>

<CENTER>Please enter Your Chat Name:

<FORM METHOD="POST" ACTION="default1.asp">
```

```
<INPUT NAME="chatname" TYPE="TEXT" SIZE=10>

<INPUT TYPE="SUBMIT" VALUE="ENTER CHATROOM NOW">

</CENTER>

</BODY>
</HTML>
```

Now as we have collected the user's desired chat name we store it in a cookie using the 'response.cookies()' function on the user's computer. This is done like this:

```
<%

ELSE

Response.Cookies("Chatname")=Request.Form("chatname")
```

Now this completes the process of collecting the user's chat name. The next process would be to send a message saying that a new user has entered the chat room. We do this as shown below:

```
APPLICATION.LOCK

Application("txt13") = Application("txt12")
Application("txt12") = Application("txt11")
Application("txt11") = Application("txt10")
Application("txt10") = Application("txt9")
Application("txt9") = Application("txt8")
Application("txt8") = Application("txt7")
Application("txt7") = Application("txt6")
Application("txt6") = Application("txt5")
Application("txt5") = Application("txt4")
Application("txt4") = Application("txt3")
Application("txt3") = Application("txt2")
Application("txt2") = Application("txt1")
Application("txt1") = "<FONT COLOR=""00FF00"">" & Request.Form("chatname") & " has entered the chat
room.</FONT>"

APPLICATION.UNLOCK

%>
```

Now, this new set of 'application()' functions may confuse you. It is basically creating lines of text messages that will be displayed on the active chat window. As you can see it starts with 1-13. But you may wonder is the '=' is for. Well, we shall leave that out for a minute. Now as you can see on the last line of the application functions (txt1) is assigned a value. This is the message stating that a new user has entered the chat room.

```
Application("txt1") = "<FONT COLOR=""00FF00"">" & Request.Form("chatname") & " has entered the chat
room.</FONT>"
```

Now then, we shall explore what the = sign is for then. Well, what is happening here is that every time a message or line of text is added it is added only in txt1, so this means that the previous txt1 line will be deleted. However, wait, with the help of the = sign we are saving the txt1 to txt2 and tx2 to txt3 and so on. This way whenever a new message is added, the previous ones up to 13 messages will not be deleted. So, this way many people can chat simultaneously without having their messages deleted. This also gives us another advantage that we can add more lines of messages. There is no limit but too many lines will look messy. I will show you how to add more lines at the end.

The last part of this asp file would be to take the user to the chat window after he has entered the chat name. So we enter the html to do this.

```
<HTML>
<HEAD><TITLE>Kumar's Online Chat</TITLE>
</HEAD>

<FRAMESET ROWS="80%, 20%" FRAMEBORDER="0" BORDER="false">

<FRAME SRC="display.asp" SCROLLING="auto">

<FRAME SRC="message.asp" SCROLLING="no">

</FRAMESET>

</HTML>

<%

END IF

%>
```

As you can see above there is an 'END IF' statement to terminate the IF statement earlier in the asp file. This is the end of the first of the four asp files that make up this chat application, which is the main one 'default1.asp'. Now let's look at the last html part of this file. As you can see it is a frames page. The two frames sources are display.asp and message.asp. This is the end of Part 1.

Part 2

In this section we will look at display.asp and the message.asp files. Let's leave out the display.asp for some time and concentrate at the message.asp file.

The message.asp file allows the user to input messages so that that he can chat with others. Now let's see how this is done. Firstly we need to again declare a variable so that the messages can be stored somewhere before it is sent to the text lines. This is done like before.

```
<%

Dim message

%>
```

Secondly, we need to create a html form as a medium for the user to enter his messages and chat. So let's do that now.

```
<HTML>
<HEAD>
<TITLE>Kumar's Online Chat Message Dialogue</TITLE>

</HEAD>

<BODY BGCOLOR="CC99FF">

<CENTER>

<FORM METHOD="POST" ACTION="message.asp">

<FONT SIZE=2>Enter your message in the text box below and click Send Message to Chat.</FONT>

<TABLE BORDER=0 CELLSPACING=0>

<TR>
<TD><INPUT NAME="message" TYPE="TEXT" SIZE=30></TD>
<TD><INPUT TYPE="Submit" Value="Send Message"></TD>
<TD> </TD>
```

```
<TD VALIGN=TOP><A HREF="logoff.asp" TARGET="_top"><IMG SRC="logoff.jpg" BORDER="NO"></A></TD>

</TABLE>
</FORM>

</CENTER>

</BODY>
</HTML>
```

As you can see above a form is created that posts messages to this same file message.asp. This means that after the user enters his message and submits it. It is sent to this same file. Then a verification check is done to check whether any data is entered into the 'message' string. This is shown below:

```
<%

If not Request.Form("message")="" THEN

APPLICATION.LOCK
Application("txt13") = Application("txt12")
Application("txt12") = Application("txt11")
Application("txt11") = Application("txt10")
Application("txt10") = Application("txt9")
Application("txt9") = Application("txt8")
Application("txt8") = Application("txt7")
Application("txt7") = Application("txt6")
Application("txt6") = Application("txt5")
Application("txt5") = Application("txt4")
Application("txt4") = Application("txt3")
Application("txt3") = Application("txt2")
Application("txt2") = Application("txt1")
Application("txt1") = "<B>" & Request.Cookies("chatname") & ":</B> " & Request.Form("message")

APPLICATION.UNLOCK

END IF

%>
```

As you can see above the validation check, checks that if the message string value is not empty then to continue and add the new message line to the lines of text using the 'application()' function. This as you can see is done like before. You may also notice that if the message value is empty when the user submits it nothing happens and no new messages are displayed. Now you may also wonder why all the = signs and the 13 text lines needed. Well, this is because once txt1 is given a new value like above, then the previous txt1 will be lost. So in order to prevent this we are constantly updated txt1 to txt2 and so on, so that every time txt1 is given a new value the previous values will not be lost. Now we shall see how these lines of messages are displayed on the active chat window. This is done using the display.asp file. Here is the code for this file:

```
<HTML>
<HEAD><TITLE>Kumar's Online Chat</TITLE>

<META HTTP-EQUIV="REFRESH" CONTENT="3; display.asp">

</HEAD>

<BODY>

<FONT SIZE=2>

<%=Application("txt13")%><BR>
<%=Application("txt12")%><BR>
<%=Application("txt11")%><BR>
<%=Application("txt10")%><BR>
<%=Application("txt9")%><BR>
```

```
<%=Application("txt8")%><BR>
<%=Application("txt7")%><BR>
<%=Application("txt6")%><BR>
<%=Application("txt5")%><BR>
<%=Application("txt4")%><BR>
<%=Application("txt3")%><BR>
<%=Application("txt2")%><BR>
<%=Application("txt1")%><BR>

</FONT>

</BODY>
</HTML>
```

As you can see above it just an ASP file displaying some lines of text. These lines of text are the lines that we were updating using the 'application()' function. Here in this file, they are just being displayed. As you can also see that this page refreshes every three seconds. This is required because, when two users are chatting they will be constantly updating the text lines. But what will be displayed is what was there when the page was opened. So, in order to display the new messages this window will need to be refreshed so that it is being constantly updated with the new text values. Now that brings us to the end of part 2.

Part 3

In this part we will now logout the user and delete the value in the chat name cookie. We will also display a message notifying other users that this user has exited the chat room. This is all done in the logoff.asp file. Firstly, to get here the user would have to click on the 'Exit Chat room' button first. Then we have to log out the user. We do this by clearing the 'chatname' cookie. But before we do this we must create a new place for the user's chat name to be stored so that a message can be displayed for notifying other users about the departure of that user. This is done like this:

```
<%

Dim exitname

exitname = Request.Cookies("Chatname")

exitname_entry = exitname & "#"

Response.Cookies("Chatname")=""

APPLICATION.LOCK
Application("txt13") = Application("txt12")
Application("txt12") = Application("txt11")
Application("txt11") = Application("txt10")
Application("txt10") = Application("txt9")
Application("txt9") = Application("txt8")
Application("txt8") = Application("txt7")
Application("txt7") = Application("txt6")
Application("txt6") = Application("txt5")
Application("txt5") = Application("txt4")
Application("txt4") = Application("txt3")
Application("txt3") = Application("txt2")
Application("txt2") = Application("txt1")
Application("txt1") = "<FONT COLOR=""red"">" & exitname & " Has left the Chat Room.</FONT>"

APPLICATION.UNLOCK

%>
```

As you can see above another variable is declared 'exitname'. This is done so that the message of the user's departure can be made with the user's chat name. This is because the cookie 'chatname' will be cleared so we will have to save the chat name somewhere else. Here I have saved it in a new cookie called 'exitname'. Now we can use this and display the user's departure message with the user's name. Once this is all done, a confirmation window is shown and then I have made it to redirect to the main page again as shown below:

```
<HTML>
<HEAD><TITLE>Thank you for Using Kumar's Chat</TITLE>

<meta http-equiv="refresh" content="5; url=default1.asp">
</HEAD>

<BODY>

<CENTER>You have just exited, Kumar's Chat. In five seconds you will be redirected to the main
page</CENTER>

</BODY>
</HTML>
```

This is end of part 3 of this article. Now I will show you how to add more lines of text for the display window so that more messages can be displayed at once. This is one of the modifications of this application. Here's how it is done.

Firstly, you open up the default1.asp file then go to the 'APPLICATION.LOCK' section of the page. Then all you have to do is add more 'Application()' functions above the thirteenth line. Like this:

```
Application("txt17") = Application("txt16")
Application("txt16") = Application("txt15")
Application("txt15") = Application("txt14")
Application("txt14") = Application("txt13")
```

I have only gone up to 17 but you can add more if you like. Now the next step would be to this same thing for the message.asp and logoff.asp file. Then the final step would be to add more application functions again to the display.asp file like this:

```
<%=Application("txt17")%>
<%=Application("txt16")%>
<%=Application("txt15")%>
<%=Application("txt14")%>
```

Again, you can add more if you like. That's it for this article.

If you want to download this code then it is available **here**.

Note: If you wish to use this application in your site, then I would be much obliged if you could just pop be an email **here** notifying me of which web site and your name. Thanks.

Part 3

In this part we will now logout the user and delete the value in the chat name cookie. We will also display a message notifying other users that this user has exited the chat room. This is all done in the logoff.asp file. Firstly, to get here the user would have to click on the 'Exit Chat room' button first. Then we have to log out the user. We do this by clearing the 'chatname' cookie. But before we do this we must create a new place for the user's chat name to be stored so that a message can be displayed for notifying other users about the departure of that user. This is done like this:

```
<%

Dim exitname

exitname = Request.Cookies("Chatname")

exitname_entry = exitname & "#"

Response.Cookies("Chatname")=""

APPLICATION.LOCK
Application("txt13") = Application("txt12")
Application("txt12") = Application("txt11")
Application("txt11") = Application("txt10")
Application("txt10") = Application("txt9")
Application("txt9") = Application("txt8")
Application("txt8") = Application("txt7")
Application("txt7") = Application("txt6")
Application("txt6") = Application("txt5")
Application("txt5") = Application("txt4")
Application("txt4") = Application("txt3")
Application("txt3") = Application("txt2")
Application("txt2") = Application("txt1")
Application("txt1") = "<FONT COLOR=""red"">" & exitname & " Has left the Chat Room.</FONT>"

APPLICATION.UNLOCK

%>
```

As you can see above another variable is declared 'exitname'. This is done so that the message of the user's departure can be made with the user's chat name. This is because the cookie 'chatname' will be cleared so we will have to save the chat name somewhere else. Here I have saved it in a new cookie called 'exitname'. Now we can use this and display the user's departure message with the user's name. Once this is all done, a confirmation window is shown and then I have made it to redirect to the main page again as shown below:

```
<HTML>
<HEAD><TITLE>Thank you for Using Kumar's Chat</TITLE>

<meta http-equiv="refresh" content="5; url=default1.asp">
</HEAD>

<BODY>

<CENTER>You have just exited, Kumar's Chat. In five seconds you will be redirected to the main
page</CENTER>

</BODY>
</HTML>
```

This is end of part 3 of this article. Now I will show you how to add more lines of text for the display window so that more messages can be displayed at once. This is one of the modifications of this application. Here's how it is done.

Firstly, you open up the default1.asp file then go to the 'APPLICATION.LOCK' section of the page. Then all you have to do is add more 'Application()' functions above the thirteenth line. Like this:

```
Application("txt17") = Application("txt16")
Application("txt16") = Application("txt15")
Application("txt15") = Application("txt14")
Application("txt14") = Application("txt13")
```

I have only gone up to 17 but you can add more if you like. Now the next step would be to this same thing for the message.asp and logoff.asp file. Then the final step would be to add more application functions again to the display.asp file like this:

```
<%=Application("txt17")%>
<%=Application("txt16")%>
<%=Application("txt15")%>
<%=Application("txt14")%>
```

Again, you can add more if you like. That's it for this article.

If you want to download this code then it is available **here**.

Note: If you wish to use this application in your site, then I would be much obliged if you could just pop be an email **here** notifying me of which web site and your name. Thanks.

Online Exam Using ASP & XML

by Mohamed Najeeb

Introduction

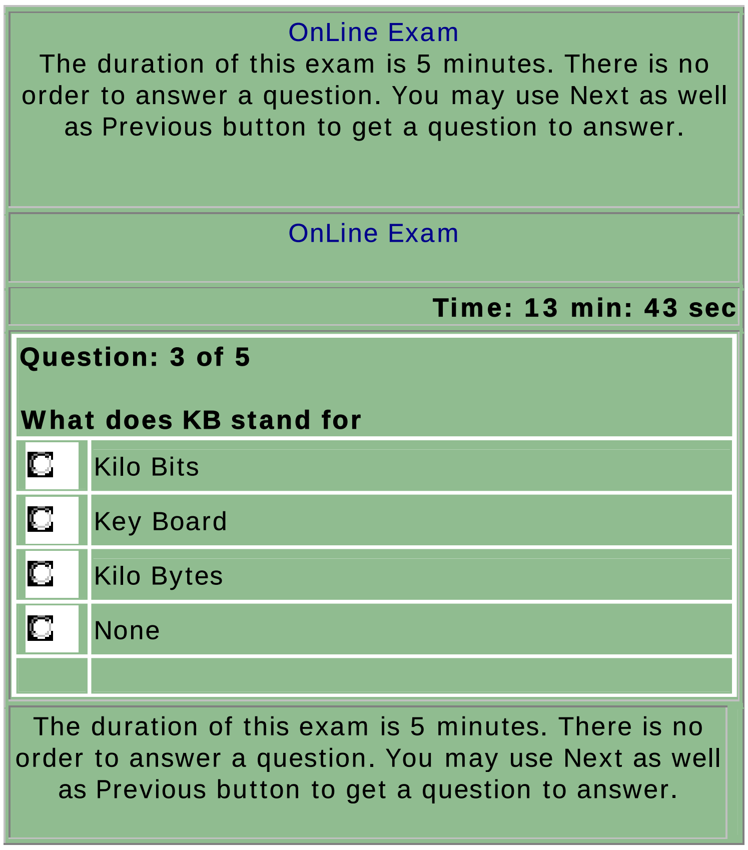
This article assumes that you are familiar with ASP, XML and HTML.

Using this OnLine Exam system, we can conduct any type of objective type examinations on line. Eventhough I implemented most of the conventional things, the reader is highly appreciated to incorporate all other features whatever so.

Let us talk on the overall functionality. The Questions are in the xml file stored in the server (it could be in the database). When the user is ready to take the exam, then the question ids list for this user will be sent to the browser from the server via Microsoft's XMLHTTP Object. Using this same XMLHTTP Object, the question contents are retrieved from the server and displayed on the page whenever the user requests for a question. On user's response to any question, the selected answer is stored in the client side itself. On end of the exam, the result will be displayed. The duration and no of questions per exam are set as 5 minutes and 5 questions respectively. But we can set them to any possible values.

How it works

The following screens are identical to the OnLine Exam system:


OnLine Exam
The duration of this exam is 5 minutes. There is no order to answer a question. You may use Next as well as Previous button to get a question to answer.

OnLine Exam

Time: 13 min: 43 sec

Question: 3 of 5

What does KB stand for

○	Kilo Bits
○	Key Board
○	Kilo Bytes
○	None

The duration of this exam is 5 minutes. There is no order to answer a question. You may use Next as well as Previous button to get a question to answer.

Once user starts the exam, the question ids list will come from the server. For every request to the server for a question, a question id in the ids list stored in the client is fetched and sent to the server. The server will return the question content, corresponding to the question id, from the xml file.

When the user selects any one answer the system will store in the answers list as well as in the selection list in the client side. Answer list is used to check no of right answers the user has selected at the end. Selection list is there so that the system will automatically select the choice which the user has already selected (for example if the user clicks the Previous button)

The exam will be over either the user clicks the Finish button or the time is over (for example 5 minutes) which comes first. On finish, the system will calculate no of right answers and display it.

The following files are used in the OnLine Exam system:

OLExam.html

```
<html>
<script>
 var objXmlHTTP,objXmlDOM;
 var aQuest; //to store question ids
 var aAnswer = new Array(); // to track the result
 var aSelected = new Array(); // to store user's response
 var count = 0; //to store the current question no
 var ansSel = 0; //to store user's selection
 var ExamDuration = 5 * 60 ; // 5 minutes
 var timerID; //to store the setInterval fun's id
 var radIndex = -1; //to store the selected radio's index

 //constructor like function
 //here XML objects are created and
 //No of questions as well as question ids list
 //are fetched from the server.
 function init(){
   objXmlHTTP = new ActiveXObject("Microsoft.XMLHTTP");
   objXmlDOM = new ActiveXObject("Microsoft.XMLDOM");
   objXmlHTTP.open("POST","OLExam.asp?Action=Start",false);
   objXmlHTTP.send("");
   temp =objXmlHTTP.ResponseText;
   aQuest = temp.split(",");

   //initialize the user's answers list
   for(i=0;i<aQuest.length; i++){
     aAnswer[i] = 0; // 0 for wrong; 1 for right answer
     aSelected[i] = -1; // to store the radio's index
   }

   if(count < aQuest.length) {
     url = "OLExam.asp?Action=NextQ&QNo=" + aQuest[count];
     objXmlHTTP.open("POST", url ,false);
     objXmlHTTP.send("");
     objXmlDOM.loadXML(objXmlHTTP.ResponseText);

     //parse the response content fetched from the server
     //and display the question
     parseQ();
   }

   //change the Start button's caption and its click event
```

```
  document.frm.btnFinish.value = "Finish the Exam";
  document.frm.btnFinish.onclick = showResult; //function

  //start the timer
  timerID = setInterval("timer()",1000);
}

function getPreQ() {
  //update the user's answers list
  checkAnswer();

  //decrement the question no - i.e. to previous Question
  count--;

  //stop the timer
  clearInterval(timerID);

  //fetch the question for the aQuest[count] id
  url = "OLExam.asp?Action=NextQ&QNo=" + aQuest[count];
  objXmlHTTP.open("POST",url ,false);
  objXmlHTTP.send("");
  objXmlDOM.loadXML(objXmlHTTP.ResponseText);

  //parse the response content fetched from the server
  //and display the question
  parseQ();

  //start the timer
  timerID = setInterval("timer()",1000);
}

function getNextQ() {
  //update the user's answers list
  checkAnswer();

  //increment the question no - i.e. to next Question
  count++;

  //stop the timer
  clearInterval(timerID);

  url = "OLExam.asp?Action=NextQ&QNo=" + aQuest[count];
  objXmlHTTP.open("POST", url ,false);
  objXmlHTTP.send("");
  objXmlDOM.loadXML(objXmlHTTP.ResponseText);

  //parse the response content fetched from the server
  //and display the question
  parseQ();

  //start the timer
  timerID = setInterval("timer()",1000);
}

function parseQ(){
  //fetch the question from theXML Object
  //format the display
  strOut  = "<table border=0 align=center width=80%>";
  strOut += "<tr><td colspan=2><b>";
  strOut += "Question No: " + (count+1) + " of ";
  strOut += aQuest.length + "</b></td></tr>";
  strOut += "<tr><td colspan=2>&nbsp;</td></tr>";
  temp = objXmlDOM.selectSingleNode("data/qtext").text;

  strOut += "<tr><td colspan=2><b>"+temp+"</b></td></tr>";
  strOut += "<tr><td colspan=2>&nbsp;</td></tr>";

  Nodes = objXmlDOM.selectNodes("data/choice");

  for(i=0;i<Nodes.length;i++){
    strOut += "<tr><td align=center width=10%>";
    strOut += "<input type=radio name=ansUsr ";
    strOut += " onClick='ansSel=" + (i+1);
    strOut += ";radIndex=" + i + "' ";
    strOut += "value=" + (i+1) + "></td><td>";
    strOut +=  Nodes.item(i).text + "</td></tr>";
  }

  //set ansNo (hidden field) to the actual answer
  temp = objXmlDOM.selectSingleNode("data/answer").text;
  document.frm.ansNo.value = temp;

  strOut += "<tr><td colspan=2>&nbsp;</td></tr>";
  strOut += "<tr><td colspan=2>";

  if(count != 0 ){
    strOut += "<input type=button value=Previous ";
    strOut += " onClick='getPreQ()'> ";
  }

  if(count < aQuest.length-1){
    strOut += " <input type=button value=Next";
    strOut += " onClick='getNextQ()'>";
  }

  strOut += "</td></tr></table>";

  //set the strOut content to <P> tag named QArea
  QArea.innerHTML = strOut;

  //set the default value to ansSel
  ansSel = 0;
  radIndex = -1;

  //check the radio if user has selected previously
  if (aSelected[count] != -1) {
    radIndex = aSelected[count];
    ansSel = radIndex + 1;
    document.frm.ansUsr[radIndex].checked = true;
  }
}

function checkAnswer(){
  //store the selected radio's index
  aSelected[count] = radIndex;

  //if the user selection matches the actual answer
  if (ansSel == document.frm.ansNo.value)
    aAnswer[count] = 1;
  else
    aAnswer[count] = 0;
}
```

```
  function showResult() {
    rights = 0;

    //stop the timer
    clearInterval(timerID);

    //update the user's answers list
    checkAnswer();

    //count no of answers
    for(i=0;i<aAnswer.length;i++){
      if(aAnswer[i] == 1)
      rights++;
    }
    strRes = "<h2 align=center><br>";

    //if all the answers are correct then greet
    if(rights == aAnswer.length)
      strRes += "<br><br>Congratulations...!";

    strRes += "<br><br>your score is " + rights;
    strRes += " out of " + aAnswer.length + "</h2>";

    document.write(strRes);
  }

  var timeCount = 0;
  function timer(){
    timeCount++;  //increment the time by one second

    //to display the time in the status bar,
    // uncomment the next line
    //window.status = "..." + timeCount + " secs" ;

    //to display the time
    temp =  "Time:   " + parseInt(timeCount/60);
    temp += "  min : " + (timeCount%60) + " sec ";
    TBlock.innerText = temp;

    //if the time is up
    if (timeCount == ExamDuration) {
      alert("Sorry, time is up");
      showResult();
    }
  }
</script>

<body>
<h2 align=center><font color=green>OnLine Exam</font></h2>

<form name=frm >
<table border=1 width=95% bgcolor=DarkSeaGreen align=center>
<tr><td align=right><b id=TBlock></b></td></tr>
<tr><td>
<p id="QArea">
<center>
<br>
Relax...! The duration of this exam is 5 minutes.
<br>
There is no order to answer a question. You may use Next as
well as Previous button to get a question to answer.
<br>
```

```
<br>
<input type=button name=btnFinish value="Start the Exam"
onClick="init()">
</center>
</p>
<input type=hidden name=ansNo>
</td></tr></table>
</form>

</body>
</html>
```

OLExam.asp

```
<%

Response.expires = 0
'create an instance of MS XMLDOM Object
'and load the QBank.xml file where all the questions are.

set obj = server.createobject("Microsoft.XMLDOM")
obj.async = false
obj.load(Server.MapPath("QBank.xml"))

'very first request from the client
if trim(request("Action")) = "Start" then
  'set no of questions per exam
  Dim NoQ,TotalQ

  NoQ = 5 'set no less than the totalquestions

  'count no of questions in the xml file
  '( or from database)
  TotalQ = obj.selectNodes("data/question").length

  Dim aQuest(),temp,isExist, strQ
  ReDim aQuest(0) 'to store the question ids

  'generate (=NoQ) question ids randomly
  while ubound(aQuest) < NoQ
    isExist = false
    temp = Int((TotalQ * Rnd) + 1)
    for i = 1 to ubound(aQuest)
      if aQuest(i) = temp then
        isExist = true
        exit for
      end if
    next
    if Not isExist then
      Redim Preserve aQuest(ubound(aQuest)+1)
      aQuest(ubound(aQuest)) = temp
      strQ = aQuest(i) & "," & strQ
    end if
  wend

  'remove the last comma ',' from strQ
  strQ = left(strQ,len(strQ)-1)

  'send the question in the strQ to the client
  response.write strQ

'all further requests - after the first request
elseif trim(request("Action")) = "NextQ" then
  'fetch the question from the XML Object
  'and form the output string
  temp = "data/question[@id=" & trim(request("QNo")) & "]"

  set Node = obj.selectSingleNode(temp)

  strXML = "<data>"
  strXML = strXML & "<qtext>"
  strXML = strXML & Node.selectSingleNode("qtext").text
  strXML = strXML & "</qtext>"
  strXML = strXML & "<answer>"
  strXML = strXML & Node.selectSingleNode("answer").text
  strXML = strXML & "</answer>"

  set Node = Node.selectNodes("choices/choice")

  for i = 0 to Node.length-1
    strXML = strXML & "<choice>"
    strXML = strXML & Node.item(i).text
    strXML = strXML & "</choice>"
  next

  strXML = strXML & "</data>"

  'send the output to the client
  Response.Write (strXML)
end if
%>
```

QBank.xml

```
<?xml version="1.0"?>
<data>
  <question id="1">
    <qtext>What does KB stand for?</qtext>
    <choices>
      <choice>Kilo Bits</choice>
      <choice>Key Board</choice>
      <choice>Kilo Bytes</choice>
      <choice>None</choice>
    </choices>
    <answer>3</answer>
  </question>
  <question id="2">
    <qtext>CPU stands for</qtext>
    <choices>
      <choice>Central Processing Unit</choice>
      <choice>Central Power Unit</choice>
      <choice>Core Processing Unit</choice>
      <choice>Core Power Unit</choice>
    </choices>
    <answer>1</answer>
  </question>
  <question id="3">
    <qtext>1 KB equals</qtext>
    <choices>
      <choice>1000 Bytes</choice>
      <choice>1024 Bytes</choice>
      <choice>1000 Bits</choice>
      <choice>1024 Bits</choice>
      <choice>Nothing</choice>
    </choices>
    <answer>2</answer>
  </question>
  <question id="4">
    <qtext>RAM is </qtext>
    <choices>
      <choice>Random Access Modifier</choice>
      <choice>Primary Memory</choice>
      <choice>Secondary Memory</choice>
      <choice>Read And Modify</choice>
    </choices>
    <answer>2</answer>
  </question>
  <question id="5">
    <qtext>Hard Disk is </qtext>
    <choices>
      <choice>Hard to break</choice>
      <choice>Primary Memory Storage</choice>
      <choice>Temporary Memory Storage</choice>
      <choice>Secondary Memory Storage</choice>
    </choices>
    <answer>4</answer>
  </question>
  <question id="6">
    <qtext>Computer Monitor is used </qtext>
    <choices>
      <choice>To monitor activities</choice>
      <choice>To control Computer</choice>
      <choice>As display unit</choice>
      <choice>None</choice>
    </choices>
    <answer>3</answer>
  </question>
  <question id="7">
    <qtext>XML stands for</qtext>
    <choices>
      <choice>Extended Markup Language</choice>
      <choice>Extended Model Language</choice>
      <choice>Extensible Markup Language</choice>
      <choice>Extensible Model Language</choice>
    </choices>
    <answer>3</answer>
  </question>
  <question id="8">
    <qtext>ASP stands for</qtext>
    <choices>
      <choice>Active Server Page</choice>
      <choice>Application Service Provision</choice>
      <choice>As Soon as Possible</choice>
      <choice>All</choice>
    </choices>
    <answer>1</answer>
  </question>
</data>
```

ASP and Flash? How?

Macromedia's Flash and Microsoft's Active Server Pages technologies are two products with an extremely heavy impact on the world of web site development. Flash allows you to create zippy, vector-based animation and interactivity in a small ActiveX control, and ASP allows you to create dynamic HTML content on the fly. Hopefully, by the time you're done skimming this article, you'll be confident enough to use ASP in the creation of zippy, vector-based Flash content on the fly.

To many developers, a site done in Flash is worth many done in HTML. The ease of use and graphical appeal of Flash is unmatched. Where many of us get stuck, however, is at the question "How am I going to get data into Flash from an external source?" The answer to this question is also the key to a frighteningly successful web site.

The Concept

The first thing you need to know about Flash is the method in which it handles variables from remote files. Flash will take an URL-encoded query string and transform it into a list of variables within its own memory. An example of a string that Flash recognizes as variable definitions looks like this:

```
size=Medium&color=Navy+Blue&style=Mandarin+Collar
```

When you import this string into Flash, you will have three new variables: size, color, and style. These variables will already be initialized; loaded with the data you passed in the string.

size	Medium
color	Navy Blue
style	Mandarin Collar

Be aware that Flash is incredibly easy to please as long as you employ the Server.URLEncode method. Flash does all the decoding itself. In the same respect, when Flash sends variables to a script, they will already be URL-encoded.

Techniques

Returning a valid string from ASP

The code below will return a valid string to the Flash movie.

```
<%@Language="VBScript"%>
<%
   Option Explicit
   Dim var(3), i, count

   i = 0
   count = 3

   var(0) = "Winken"
   var(1) = "Blinken"
   var(2) = "Nod"

   Do While i < count
      Response.Write "var" & i & "=" & var(i) & "&"
      i = i + 1
   Loop

   Response.Write "i=" & i

%>
```

When you execute this program, ASP returns this string:

```
var0=Winken&var1=Blinken&var2=Nod&i=3
```

Passing values from ASP to Flash

Remember that query string I mentioned above? You'll be using this format with ASP next. To pass variables from ASP to Flash, **you must write a string** back to the response in the above format.

To load variables from an ASP file, you should use the following action script:

```
Load Variables ("myscript.asp", 0)
```

(The Load Variables command can be found by choosing Load/Unload Movie from the dropdown list and selecting the Load Variables Into Location radio button.)

Generally, you should not use Get URL unless you are a) rewriting the object code to embed a new movie or b) generate HTML in a new page.

The zero in the above script represents the level at which you will load the movie. If you have movies stacked upon one another, you should reference them by their z-order. _level0 is the first movie, _level1 is the movie stacked above the first, and so on. For our current purpose, we'll assume there is only one movie in the picture and use 0 for the level specification.

Passing values from Flash to ASP

Surely you will want to send out form data from Flash to ASP. To send variables from Flash to an ASP program, use the following ActionScript code:

```
Load Variables ("myscript.asp", 0, vars=POST)
```

When you send variables using POST, you can access them in your ASP script using one of two methods:

1. Request.Form("flash_var_1")
2. Request("flash_var_1")

That's right... when you POST variables, Flash sends them in the form collection.

When you send Flash variables using GET, however, they are written into the query string, and subsequently you should use Request.QueryString("flash_var_1").

When you specify "Don't send" as a destination for the variables, Flash assumes you're loading data from the ASP file. You can use a query string parameter for the URL value of the Load Variables call and still retain the "don't send" variable transmission specification to avoid overwriting variable values.

Exercise: Transferring Values From Flash to ASP to a Database to ASP to Flash

Download support materials for Exercise 1 from here.

View the code here.

The files are named as such:

employee.mdb	Access database
employee.asp	ASP script
employee.fla	Flash file
employee.swf	Flash movie
employee.html	Movie mounting page

We will be employing ADO database routines in this example to make all this useful to you in the future. I have created a table called Employees in a file called employee.mdb which resides in the same directory as the script. The table looks like the one below:

ID	NameLast	NameFirst	Position
1	SMITH	JOHN	CEO
2	BROWN	SUSAN	MARKETING DIRECTOR
3	STANFORD	ROBERT	SALES REPRESENTATIVE

That's it; three simple records. Notice how all the data is in capital letters: In a real-world application, this would really only be relevant to the fields upon which you would search. Putting the data in capital (or lowercase) letters guards against case mismatch in searches, because we can UCase the data from Flash before sending it to ADO.

Save your table as Employees and save your database as employee.mdb in a directory by itself.

Next, we'll make the input screen in Flash.

Open up Flash or use the support materials available.

The Data Entry Frame

In the first frame, create a screen with a text label that reads "Please enter employee's last name." Next to it, create a text field (push the |ab button down to make it an editable field) and right-click on it. Go to Properties, and where it says Variable, enter NameLast. You should go to the first frame's Properties and add an action script that simply says "Stop." This prevents the movie from advancing until you have entered a value and clicked Submit.

The Submit Button

Create a button that says Submit. Then drop it onto the stage and go to its Instance Properties by double clicking it. In the Actions tab, enter this action script:

```
Go To and Play (2)
```

This sends the user to the second frame.

Loading The Variables Behind The Scenes

Create, in the second frame, a blank keyframe with one frame action:

```
Load Variables ("employee.asp", 0, vars=GET)
```

The "Loading Data" Loop

In the third frame, you will want to create a short animation that tells the user that ASP is loading data. In this case, I used frames 3-8 for this animation. At frame 3, I entered the following frame action:

```
If (Position ne "")
   Go To and Stop (9)
End If
```

This checks to make sure that ADO has pulled our last variable, Position, from the database and returned it to the response. Otherwise it continues to play the Loading sequence until frame 5, at which we have this action:

```
Go To and Play (3)
```

This causes the animation to play over and over again and continue to check the value of Position. When ASP is finished, the animation will automatically go to frame 6.

The Result Frame

At frame 6, we should have a small text label that reads "ASP has returned the following results." Below this text label, we should have three more text labels in a row which read "Last Name", "First Name", and "Position". These are captions for our editable text fields below. Insert editable text fields (click the |ab button on the toolbar when you drop it, just like in the first frame). Set one's Variable equal to "NameLast", set another one's equal to "NameFirst", and the last one's equal to "Position" (all without the quotation marks).

Publishing

Save your document file as employee.fla in the same directory as your database.

Then, go to File menu -> Publish, and you will find employee.swf and employee.html residing in the same directory you saved employee.fla.

Okay, that's it, you're done with the Flash side of it. Now it's time to move on to ASP.

Open up your favorite ASP Editor (Mine is Textpad!), such as InterDev or even Notepad. You will be using an ADODB recordset and connection with a SQL connection string. You will not return anything to the response except the URL-encoded variable definition string. Are you ready? Well, get that way, because here's the code.

```
<%@Language="VBScript"%>
<%
   Option Explicit    ' Don't ever let me catch you without this line!

   Dim oRS, oConn     ' Recordset and connection objects

   Set oConn = Server.CreateObject("ADODB.Connection")
   oConn.ConnectionString = "Driver={Microsoft Access Driver (*.mdb)};" & _
      "DBQ=" & Server.MapPath("employees.mdb")
   oConn.Open

   Set oRS = Server.CreateObject("ADODB.Recordset")
   oRS.Open "SELECT * FROM Employees", oConn, 2, 3

   ' The next line looks for the specified name and UCases the
   ' last name we searched for to avoid case-sensitive issues.
   oRS.Find "NameLast = '" & UCase(Request.QueryString("NameLast")) & "'"

   ' If the last name does not exist, then return Not Found to the response.
   ' Otherwise return appropriate variables.
   If oRS.EOF Then
      Response.Write "NameLast=Not+Found&NameFirst=Not+Found" & _
         "&Position=Not+Found"
   Else
      Response.Write "NameFirst=" & Server.URLEncode(oRS("NameFirst")) & _
         "&NameLast=" & Server.URLEncode(oRS("NameLast")) & _
         "&Position=" & Server.URLEncode(oRS("Position"))
   End If

   ' Clean up and say goodbye.
   oRS.Close
   Set oRS = Nothing
   oConn.Close
   Set oConn = Nothing
%>
```

Save this code as employee.asp

Now, access employee.html through the local intranet. You should, if you haven't already, stick all these files somewhere in your web root and access it via http://localhost/employee/employee.html (or wherever you have stored the files). Since ASP is processed server-side, you cannot access employee.html via the hard drive and expect values from the ASP application. This means you will have an endless "Loading" loop

Once you have pulled up the page on the Intranet, enter a name which you know is in the database, such as Smith. ASP will return the values to Flash, and you will see the results when ADO is done processing.

Good Luck with this! If you have troubles, please contact me at dan@catapultic.com.

-Dan

A Simple Content Management System Tutorial

by Jesus Torres

Simple CMS

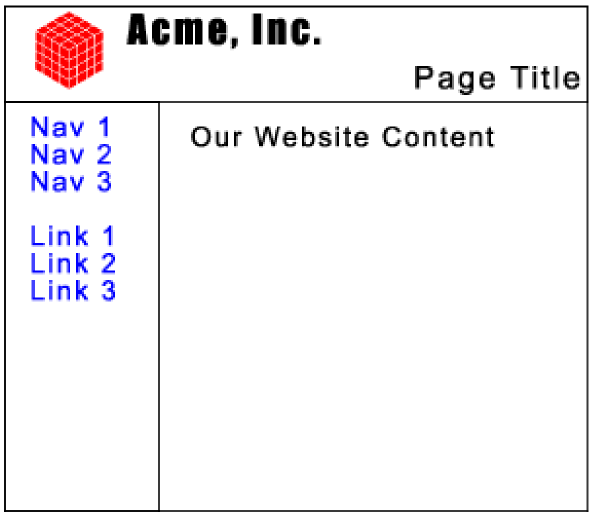
The definition of a content management system (CMS) is as diverse as all the people creating them, but at the very core of a CMS is the concept of separating Website layout and design from content. By doing this you provide a means for the Web Developer to do his job and not get bogged down with endless requests for content changes/additions, more importantly though, this empowers the non-Developer, (content creators) to update/create their own web pages.

I will show you in this tutorial how to create a simple CMS. This tutorial will do the following:

1. Create a Website template
2. Create an MS Access Database for the Content
3. Show you how to use the FileSystemObject to read a text file
4. Show you how to use the VBScript function Replace()

CMS Template - Part I

First we need to create a Website template (how to do this is a tutorial to itself). You can use any HTML editor out there (I personally prefer Dreamweaver MX). We want to create a modular layout with the Name and Logo for our Website, a place for our Website navigation and the content area:

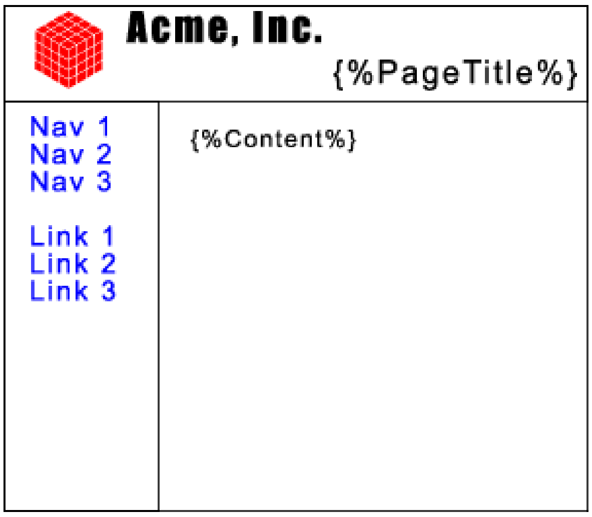


CMS Template - Part II

In order to make this work, we will create a template and use tokens as place holders for the content:

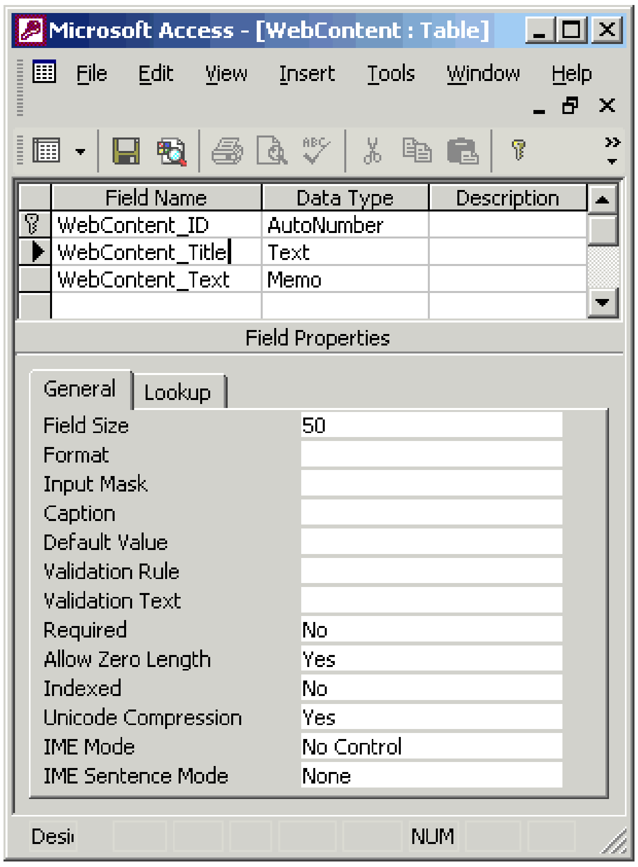
```
{%PageTitle%}
{%Content%}
```

Our Template should look like this:



CMS Content Database

We need to create an MS Access database to store the Web Content. The one I created is called "SimpleCMS" and the Table is called "WebContent":

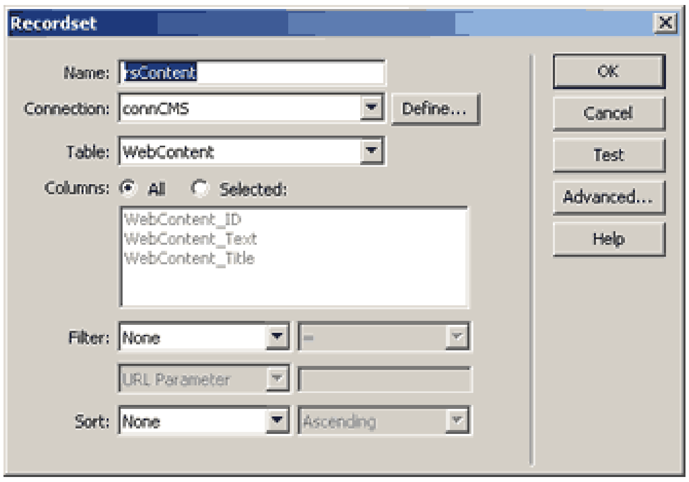


Field Name	Data Type	Description
WebContent_ID	AutoNumber	
WebContent_Title	Text	
WebContent_Text	Memo	

Field Size	50
Format	
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	Yes
Indexed	No
Unicode Compression	Yes
IME Mode	No Control
IME Sentence Mode	None

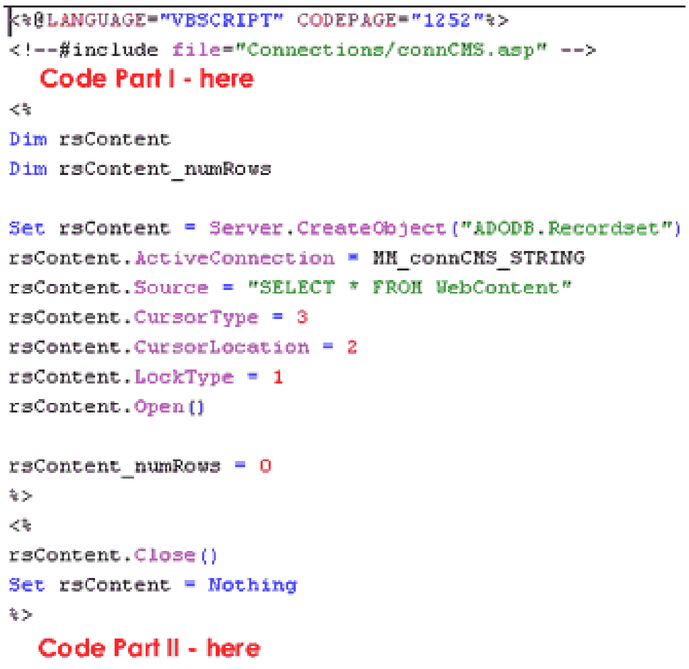
CMS ASP Script

Now we will put it all together with the ASP script. The script will be called "Default.asp". The first thing we do is create new page in Dreamweaver and delete all the HTML, then create a recordset:



Name: rsContent
Connection: connCMS
Table: WebContent
Columns: ● All ○ Selected:
WebContent_ID
WebContent_Text
WebContent_Title
Filter: None =
URL Parameter
Sort: None Ascending

Next we have to do some hand coding so hang on to your hats:



```
<%@LANGUAGE="VBSCRIPT" CODEPAGE="1252"%>
<!--#include file="Connections/connCMS.asp" -->
  Code Part I - here
<%
Dim rsContent
Dim rsContent_numRows

Set rsContent = Server.CreateObject("ADODB.Recordset")
rsContent.ActiveConnection = MM_connCMS_STRING
rsContent.Source = "SELECT * FROM WebContent"
rsContent.CursorType = 3
rsContent.CursorLocation = 2
rsContent.LockType = 1
rsContent.Open()

rsContent_numRows = 0
%>
<%
rsContent.Close()
Set rsContent = Nothing
%>
  Code Part II - here
```

Code Part I

This is the code to read the contents of our Template file (Template.htm):

```
<%
Page_Template = ""
Set objTemplate = Server.CreateObject("Scripting.FileSystemObject")
Set objText =
objTemplate.OpenTextFile(Server.MapPath("Template.htm"),1,false)
Page_Template = objText.ReadAll
objText.Close
Set objText = Nothing
Set objTemplate = Nothing
%>
```

Code Part II

This is the CMS Code Part II where we read the Database table "WebContent" based on the URL parameter "Page", if no URL parameter is specified we read from the first record:

```
 1. <%
 2. MyWebPage = "That Web Page is not Found."
 3. If Request.QueryString("Page") <> "" Then
 4.    rsContent.filter = "WebContent_ID = " &
CStr(Request.QueryString("Page"))
 5. End If
 6. If Not (rsContent.eof or rsContent.bof) Then
 7.    MyPageTitle = (rsContent.Fields.Item("WebContent_Title").Value)
 8.    Page_Content = (rsContent.Fields.Item("WebContent_Text").Value)
 9.    MyWebPage = Replace(Page_Template,"{%Content%}",Page_Content)
10.    MyWebPage = Replace(MyWebPage,"{%PageTitle%}",MyPageTitle)
11. End If
12. Response.Write(MyWebPage)
13. %>
```

First we determine if a URL parameter has been passed. The URL parameter would be used to specify a particular page which would be stored in the database, i.e.:

http://www.website.com/default.asp?Page=7

6. If we have not traveled to the end of the table without finding a record,
7. we proceed to read the Page Title from the field "WebContent_Title" and
8. the Page Content from the field "WebContent_Text".
9. Here we take the contents of Template (stored in the variable "Page_Template") and replace the token "{%Content%} with the contents of the field "WebContent_Text".
10. Here we take the contents of our Web Page (stored in the variable "MyWebPage" and replace the token "{%PageTitle%}" with the contents of the field "WebContent_Title".

A Guestbook Application (Text File Based)

by Darren Death

Instructions

The guest book has been designed to be easily deployed just drop the files into a directory, customize the file paths, and set the appropriate permissions. The guest book does however allow for advanced configuration options which allows for bad word filtering, IP filtering and advanced email options.

Features

- Easy integration into your present web site with the use of customizable templates, cascading style sheet (CSS) and optional user defined input and exit links.
- Configurable logging option allows the guest book to log descriptive error and user access message to the web server log.
- Configurable bad words filter which allows you to block bad words which are passed to the guest book.
- Configurable IP Address blocking filter which will block unwanted visitors from accessing the guest book.
- Configurable age blocking filter which prevents under age visitors from signing the guest book.
- Configurable email auto-responder which allows you to selectively turn email on or off, send an administrative email, and send mail in either HTML or ASCII format.
- Configurable page titles and form titles.
- Guest Book can be called via HTTP or HTTPS. There are no hard coded links in the guest book which would prevent this.
- Security feature which prevents an unauthorized user from sending a form post to the guest book’s primary processing page. Without this feature an unauthorized user could potentially use the guest book as a source to send unauthorized email to unknown users.
- The guest book generates a new "view" once it has reached a predefined file size. A user that views the guest book entries will be presented by an abbreviated view with a drop down list of additional guest book views.
- Client Side form validation to ensure users input the appropriate data. Client side form validation takes the processing burden off of the server.

A Guestbook Application (Database Based)

by Darren Death

Overview

The guest book has been designed to be easily deployed. Just drop the files into a directory, customize the database path, and set the appropriate permissions. The guest book does however allow for advanced configuration options which allows for bad word filtering, IP filtering, advanced email options, and more.

Features

- Easy integration into your present web site with the use of custom header/footer files, cascading style sheet (CSS) and optional user defined entry and exit links.
- Configurable logging option allows the guest book to log descriptive errors and user access message to the applications database.
- Configurable bad words filter which allows you to block bad words which are passed to the guest book.
- Configurable IP Address blocking filter which will block unwanted visitors from accessing the guest book.
- Configurable age blocking filter which prevents under age visitors from signing the guest book.
- Configurable email auto-responder which allows you to selectively turn email on or off, send an administrative email, and send mail in either HTML or ASCII format.
- Configurable page titles and form titles.
- Guest Book can be called via HTTP or HTTPS. There are no hard coded links in the guest book which would prevent this.
- Security feature which prevents an unauthorized user from sending a form post to the guest book’s primary processing page. Without this feature an unauthorized user could potentially use the guest book as a source to send unauthorized email to unknown users.
- The guest book will page database entries (5 entries per page).
- Server Side form validation to ensure users have input the appropriate data.
- All features can be enabled or disabled per your requirements.

ASP using Dreamweaver UltraDev and PWS

Andrew Starling

May 17, 2001

Macromedia Dreamweaver UltraDev is a great tool for creating ASP pages. Combine it with Microsoft Personal Web Server (PWS) and you can check that the VB and SQL elements of your ASP work fine while the pages are still in progress. Here are some practical tips to help you get the combination up and running.

Macromedia Dreamweaver UltraDev is a great tool for HTML coders who want to spread their wings and begin creating ASP pages. It allows you to create simple ASP pages without any knowledge of VB or SQL and only a basic grasp of databases. Once you've created a few pages you can look at the code that UltraDev has created and begin to adapt it for more sophisticated uses, giving you a welcome boost along the learning curve as you move from very basic ASP to something more valuable in the commercial world.

In this article we'll look at the basics of setting up UltraDev on a PC, along with Microsoft Personal Web Server (PWS) for testing the ASP pages you create. Rather than repeat all the information in the fine UltraDev tutorials that come with the software, and in the help systems of both UltraDev and PWS, we'll home in on some of the missing bits - tips that will help you get to grips with the combination in the minimum amount of time.

The first item to look at is PWS. It's essential to have some kind of server software on your machine for live testing. If you're running Windows 2000 then you'll find Microsoft IIS (Internet Information Server) on your OS disk and this will almost certainly be your server software of choice. But for this example we're going to look at PWS, because it's free and also because it's a little quirky. It's highly likely you'll use PWS if you're running Win95 or Win98.

You can download PWS direct from Microsoft, where it's included as part of the fairly large NT Option Pack of December 1997. You may also come across this as a CD, which will save you a lot of downloading. It doesn't matter that this is an NT pack - the program can still be transferred to a Win95 or Win98 machine. But the best place to get hold of the program is from the Windows 98 OS disk, where it can be found in the directory *Add-Ons/PWS*. The version on the Win98 disk gives you a better chance of a successful install than the version in the NT pack.

One big tip is to make sure you install PWS *before* you install Dreamweaver UltraDev. I tried it the other way round and no matter what I did I couldn't get the combination to work. But by simply uninstalling both programs and reinstalling with PWS first, I was able to get everything functioning without any need for tweaks.

The second tip is to make sure your back-ups are in order and you've been nice to all your heavy-duty computing friends in the run-up to installation. The first time I installed PWS it crashed my machine so badly I couldn't even get Windows to work in safe mode, and had to revert to an old version of the registry using Regedit in DOS. The stress reduced my lifespan by a couple of days, but hey ho, it's free software.

Getting PWS Up and Running

Once you've successfully installed PWS you need to get it fully active as a server. Frankly, the documentation that comes with the program isn't wonderful and is more likely to send you off on a wild goose chase than take you where you want to go. Most of the menu icons are fairly useless too. Probably the only one you'll ever use is *Advanced*, down at the bottom. This is the one that allows you to identify directories you'd like to make available through your server.

You'll need to create a *virtual directory* for your ASP files, and this can be slightly confusing if you're not used to servers. It's the place where the server holds the files that it publishes, and it will refer to these locations by shortcut names that aren't the same as normal computer drive filepaths.

There are many alternative ways to set up your files for publishing. In my case I used Windows Explorer to create a new folder called Inetfolders. This has the filepath C:\Inetfolders and is where I place all published files. Then I created a subdirectory within this directory called Trial, and placed a very simple Web page in the subdirectory, called index.html.

To make PWS aware that I wanted to publish the contents of the Trial subdirectory, I went in through the Advanced menu and chose Add. I could then identify the subdirectory by entering its regular filepath, and give it a shortcut (*virtual*) name. I called it TrialOne.

Now I could access the file through my Web browser on the local Intranet that PWS creates on my machine. To access this Intranet I need to start my URL with http://OEMComputer, then add on the virtual name. So if I enter http://OEMComputer/TrialOne/index.html, I get my browser to show the simple Web page I put in the subdirectory. Best of all, this tells me that PWS is functioning properly.

Note that the expression *OEMComputer* varies according to your PC. It's the name the PC calls itself. You can find this under *Start* > *Settings* > *Control Panel* > *Network* > *Identification*. Other popular names are *MyMachine* and *localhost*.

Here are a couple more items on PWS. When you're in the Advanced area you should see a checked box for "Allow Directory Browsing". And the individual directories you want to publish should have "Read" and "Scripts" checked so ASP will work.

PWS and Internet Explorer can get themselves in a real mess with IE's offline mode. Normally IE will prompt you if it has managed to get itself into offline mode and you want to view something it has to go online to see. But if that page is served by PWS it may just tell you it can't be found. Watch out for this kind of false error. Generally it's best to avoid offline mode when using PWS. If you ever get the prompt *Work Offline* or *Try Again*, opt for *Try Again*, which will lead to a 404 error but stops you going into offline mode.

PWS can also significantly slow down your machine when it's running. You can switch it off using the Stop button on the main front page of the program, but you'll need to press Start and then reboot to get it going again.

Finally, don't forget that PWS isn't a pretend server, it's publishing your pages for real. If you connect to the Internet while PWS is running, it's theoretically possible for anybody with the right skills to find the files you've published and look at them through your Internet connection. In most cases this is probably no big deal. But if you're working on confidential material you'll need to bear it in mind.

Starting with UltraDev - Page 2

Andrew Starling
May 17, 2001

With PWS up and running, you can now turn your attention to Dreamweaver UltraDev. It's really two separate programs - Dreamweaver the HTML editor, and UltraDev the ASP editor - welded together. But unlike many twin-sets this one's not a double whammy. The joint is clearly visible but it's a fine piece of welding.

You'll be at a definite advantage if you're already familiar with Dreamweaver. It's also worth knowing a little about databases, at least the basics of tables, fields and records. You won't be able to progress very far with ASP unless you understand databases.

Macromedia is well-known for producing fine tutorials for its software and UltraDev is no exception. It's definitely worth spending a couple of hours following the UltraDev tutorial that's included with the package. You won't finish up as an ASP expert after that brief amount of time, but you will have produced a couple of pages, and more importantly you'll have been gently guided through the configurations that are necessary to get your ASP working and UltraDev showing you that it works even while you're still coding.

The first of these configurations is defining the local site (directory) where you're going to hold your working files. Once you've got these files in good shape, you'll be sending them (or to use the correct terminology, "putting" them) to the remote site, which is a server directory - one that you've told PWS it can publish. It's very similar to the relationship between a local and remote site in FTP. The difference is that in this case it's likely that both the directories will be on your hard drive.

It can seem a bit of a performance, shifting files around like this when they're probably in neighbouring directories to start with, but it's good practice and mirrors the way you deal with files that are published on real servers.

Another set-up item that quickly comes into play either with the Ultradev tutorial or with a genuine new project is configuring UltraDev's Live Data window. This is probably the single strongest feature of UltraDev and one you'll be using time after time.

If you're familiar with Dreamweaver, you'll know it allows you to check that the HTML you've just messed with works well in a browser - using the Preview in Browser feature. Live Data is similar but more advanced. It allows you to run your ASP code and see that it works, without leaving the development window.

That's where PWS comes in. UltraDev latches on to it as a real server, puts your page through it for real and then shows the output on screen. Since there's no browser involved, some elements of layout are not perfectly WYSIWYG, but the important thing is that the code is run and any problems with the ASP and database connections should show up. It's a true peach of a feature.

The instructions included in the UltraDev tutorial for configuring the Live Data window are thorough and there's not much point in repeating them here. Likewise the guidelines for connecting your pages to the database that drives them, which is Access97 in the tutorial but might be SQLServer or other options for real projects. You'll need to set up a DSN (Data Source Name) which can be done through ODBC accessed either via Control Panels or directly through UltraDev.

Once these one-off configurations are done, you can move on to defining the data you're intending to pull from the database into your pages - your recordsets.

Recordsets and Data - Page 3

Andrew Starling
May 17, 2001

Recordsets are where you define how much of the data in the database you want to pull into your ASP page. The simplest choice is a selection of columns from a single table or query. In UltraDev you define the recordset for a page in the *Data Bindings* window. Here you give the recordset a name, identify the database connection it should use, and which table and fields you wish to select for publication on your page.

Assuming you've asked Ultradev to write Visual Basic — the most common choice for ASP pages — the recordset code you finish up with on your page will look something like this:

```
<%
  set rsReleaseSummaries =
Server.CreateObject("ADODB.Recordset")
  rsReleaseSummaries.ActiveConnection =
"dsn=scaalcoffee;"
  rsReleaseSummaries.Source = "SELECT PRID, PRTitle,
FROM
  PressReleases"
  rsReleaseSummaries.CursorType = 0
  rsReleaseSummaries.CursorLocation = 2
  rsReleaseSummaries.LockType = 3
  rsReleaseSummaries.Open
  rsReleaseSummaries_numRows = 0
  %>
```

This sits very close to the beginning of the page, before any HTML. If you're new to ASP it's very interesting to look at the mix of VB and HTML produced on your first Ultradev pages. The two languages alternate through the page. The HTML elements are passed through the server untouched and sent to the viewer's browser. The VB elements are processed by the server and usually result in more lines of dynamic HTML code, which are also sent to the browser as they are created.

Defining a recordset is a preparatory element and doesn't produce HTML itself, it just paves the way for later VB code to do so. The first line of the code shown above names the recordset "rsReleaseSummaries." The second line shows the database connection that should be used, called "scaalcoffee." And the third line identifies the fields that will be included in the page, which are "PRID, PRTitle, PRShort, PRDate" from the table "PressReleases." The remaining lines define standard elements of the recordset and for most simple examples these stay the same.

To get the data showing on your page, you now need to connect the database fields to text elements on your page. Continuing with the example started above, we can create a regular HTML table with four cells containing the text - PRID, PRTitle, PRShort, and PRDate. In the UltraDev Data Bindings window we then highlight each field and drag and drop it on to the corresponding text in a cell. UltraDev changes the cell contents, for example the text PRTitle changes to:

```
<%=(rsReleaseSummaries.Fields.Item("PRTitle").Value)%>
```

This will create the cell content dynamically from the database.

There's one final step in this simple example, which is setting up the HTML table so it is repeated for each row of data in the database. In UltraDev that's done in the *Server Behaviors* window by highlighting the table and adding the behavior "Repeat Region." This adds the following VB code before and after the table.

```
<%
  While ((Repeat1__numRows <> 0) AND (NOT
rsReleaseSummaries.EOF))
  %>

<Table HTML goes here>

<%
  Repeat1__index=Repeat1__index+1
  Repeat1__numRows=Repeat1__numRows-1
  rsReleaseSummaries.MoveNext()
  Wend
  %>
```

That's the simple example completed. Selecting *View > Live Data* should now show the ASP in action and return a page full of data.

Debugging and Transfer to a Real Server

Although UltraDev is a great piece of software, when you're creating your first ASP pages there are many things you need to get right, and it shouldn't be too much of a surprise if your pages fail the first few times you try to run them. Here are a couple of practical debugging tips.

When you've created your recordset, there's a Test button available in the recordset window, under *Data Bindings*. It's a good idea to use this. This confirms (or otherwise) that your database connection is working as it should, that the fields and tables you've selected are valid and that any SQL statements you've made are also correct. If you've got something wrong, the error message will tell you whether it's your connection or your SQL. It will even try to tell you which bit of your SQL is wrong, though often without success. Any syntax error you get here will relate to your SQL, not your VB, which of course UltraDev is writing flawlessly.

Whenever a page fails it's a good idea to go back to this Test button and try it out. In one case I had a page that had previously worked and then failed. I wasted time looking at the server and at file versions, then pressed the Test button and found that a colleague had renamed all the fields in the database.

If Live Data won't work, especially at the beginning of a work session, it's a good idea to check your server and offline/online mode. One easy way to do this is to have a simple HTML file on the server and bookmark it in your browser. Then access the file and press shift refresh/reload to check that you're not getting a cached version. If you can see the file then you know the server's working. In practice I find that I use Live Data in the early stages of building a page, but towards the later stages when I'm changing cell sizes and appearance I'm more likely to "put" the page to the remote site after each change and view it in a browser.

Finally, the UltraDev plus PWS combination is aimed at the development and testing phase. Once your pages are working, you'll need to transfer your work to a real server connected to the Internet or an intranet. This means that some of the details on your pages may change. One that will almost certainly change is the Data Source Name (dsn) in the first line of any recordset. If you developed using an Access database and the real system uses SQLServer or other alternatives then you may also need to change the names of tables, queries and even individual fields.

That's where the Dreamweaver element of Dreamweaver UltraDev steps in, with its powerful global editing facility. In the *Site* window, select the files or folders you'll be transferring - but don't open them. Then choose *Edit > Replace* and make sure *HTML Source* is selected rather than *Text*. Now you can make changes across multiple files. Personally, I like to create a small text file with all the necessary changes listed. Then I copy and paste the individual lines into the fields of Dreamweaver's Replace feature. Eight or ten global changes take just a few minutes.

Like every other aspect of Dreamweaver Ultradev, it's fast and effective. This is one program that will repay your purchase investment through saved time in just a few days.

Recommended Links:

WDVL ASP articles
More on ASP, plus a host of links