AspTutorial.info

Active Server Pages tutorial for beginners

Contact

Active Server Pages: What are Active Server Pages?

What Are Active Server Pages?

Active Server Pages (ASPs) are Web pages that contain *server-side scripts* in addition to the usual mixture of text and HTML (Hypertext Markup Language) tags. Server-side scripts are special commands you put in Web pages that are processed before the pages are sent from your Personal Web Server to the Web browser of someone who's visiting your Web site. . When you type a URL in the Address box or click a link on a Web page, you're asking a Web server on a computer somewhere to send a file to the Web browser (sometimes called a "client") on your computer. If that file is a normal HTML file, it looks exactly the same when your Web browser receives it as it did before the Web server sent it. After receiving the file, your Web browser displays its contents as a combination of text, images, and sounds.

In the case of an Active Server Page, the process is similar, except there's an extra processing step that takes place just before the Web server sends the file. Before the Web server sends the Active Server Page to the Web browser, it runs all server-side scripts contained in the page. Some of these scripts display the current date, time, and other information. Others process information the user has just typed into a form, such as a page in the Web site's guestbook.

To distinguish them from normal HTML pages, Active Server Pages are given the ".asp" extension.

What Can You Do with Active Server Pages?

There are many things you can do with Active Server Pages.

- You can display date, time, and other information in different ways.
- You can make a survey form and ask people who visit your site to fill it out, send emails, save the information to a file, etc

What Do Active Server Pages Look Like?

The appearance of an Active Server Page depends on who or what is viewing it. To the Web browser that receives it, an Active Server Page looks just like a normal HTML page. If a visitor to your Web site views the source code of an Active Server Page, that's what they see: a normal HTML page. However, the file located in the server looks very different. In addition to text and HTML tags, you also see server-side scripts. This is what the Active Server Page looks like to the Web server before it is processed and sent in response to a request.

What Do Server-Side Scripts Look Like?

Server-side scripts look a lot like HTML tags. However, instead of starting and ending with lesser-than (<) and greater-than (>) brackets, they typically start with <% and end with %>. The <% is called an *opening tag*, and the %> is called a *closing tag*. In between these tags are

the server-side scripts. You can insert server-side scripts anywhere in your Web page--even inside HTML tags.

Do You Have to Be a Programmer to Understand Server-Side Scripting?

There's a lot you can do with server-side scripts without learning how to program. For this reason, much of the online Help for Active Server Pages is written for people who are familiar with HTML but aren't computer programmers.

Active Server Pages: Displaying Date, Time and Text

Displaying the Current Date and Time

The date and time described in this section are those that are on the server.

Date

To display the current date by itself in a Web page, type:

```
<% =date %>
```

at the point where you want it to appear. When you view the page in your browser, you should see something like this:

Thu, Jan 23, 1997

Note: Even though "=date" is a short script, it's actually made up of two parts. The "date" part tells the server, "Get me the date." The equal sign (=) tells the server to display the date in the Web page. If you typed just:

```
<% date %>
```

the server would get the current date from your system, but that's all. It wouldn't display it. There are times when it makes sense to use an ASP function without the equal sign.

Time

To display the current time by itself, type:

```
<% =time %>
```

where you want it to appear. When you view the page, you should see something like this:

4:19:46 PM

Now (Date and Time)

To display the current date and time, type:

```
<% =now %>
```

where you want them to appear. When you view the page, you should see something like this:

1/23/97 4:19:46 PM

Changing the Way Date and Time are Displayed

You can also use Active Server Pages (ASP) functions to customize the way the current date and time are displayed on your Web page. To do this, use the **now** function together with the following formatting functions.

Month and Monthname

To display the number of the current month in a Web page, type:

```
<% =month(now) %>
```

where you want it to appear. When you view the page in your browser, you'll see a 1 if the current month is January, 2 if it's February, and so on.

To display the name of the current month, type:

```
<% =monthname(month(now)) %>
```

where you want it to appear.

Day

To display the day of the current month, type:

```
<% =day(now) %>
```

where you want it to appear. When you view the page, you'll see a number between 1 and 31.

Year

To display the current year, type:

```
<% =year(now) %>
```

where you want it to appear.

Example

Suppose you wanted to display today's date as *day/month/year* instead of *month/day/year*. To do so, you would use the day, month, and year ASP functions together, by typing:

```
<% =day(now) %>/<% =month(now) %>/<% =year(now) %>
```

When you viewed the page, you would see something like this:

23/1/1997

Later we'll see how you can change this so only the last two digits of the year are displayed, like this:

23/1/97

Weekday and Weekdayname

To display the day of the week as a number from 1 to 7 in a Web page, type:

```
<% =weekday(now) %>
```

where you want it to appear. When you view the page in Internet Explorer, you'll see a 1 if today is Sunday, 2 if it's Monday, and so on.

To display the day of the week by name, type:

```
<% =weekdayname(weekday(now)) %>
```

where you want it to appear.

Hour, Minute, and Second

To display just the hour part of the current time, type:

```
<% =hour(now) %>
```

where you want it to appear. The hour function is based on a 24-hour clock. When you view the page, you'll see a number between 0 and 23.

To display just the minutes part of the current time, type:

```
<% =minute(now) %>
```

where you want it to appear. When you view the page, you'll see a number between 0 and 59.

To display just the seconds part of the current time, type:

```
<% =second(now) %>
```

where you want it to appear. When you view the page, you'll see a number between 0 and 59.

Example

Try typing this into a Web page:

```
The time is <% =time %>. That means it's <% =minute(now) %>
minutes past <% =hour(now) %> o'clock.
```

When you view the page in Internet Explorer, you should see something like this:

The time is 1:36:05 PM. That means it's 36 minutes past 13 o'clock.

Remember, the hour function is based on a 24-hour clock. Later we'll see how to convert from the 24-hour clock to a 12-hour clock.

Timevalue

You probably won't ever use the **timevalue** function. It takes the different ways you can write the time, such as "2:24PM" and "14:24," and returns them in this format: "2:24:00 PM." This

can be useful if you're using a function that needs to be given the time in that exact format.

Example

Earlier in this section we saw how you can use the hour, minute, and second functions to break up the time into hours, minutes, and seconds. With the **timevalue** function, you can put them back together. Type this into a Web page:

```
When it's 23 minutes and 5 seconds past 4 o'clock in the afternoon,
that means it's <% =timevalue("16:23:05") %>.
This is the same as <% =timevalue("4:23:05PM") %>
or <% =timevalue("16:23:05PM") %>.
```

Make sure you type "16:23:05PM" and not "16:23:05 PM." The "05" and the "PM." should be run together, not separated by a space. When you view the page in Internet Explorer, you should see:

When it's 23 minutes and 5 seconds past 4 o'clock in the afternoon, that means it's 4:23:05 PM. This is the same as 4:23:05 PM or 4:23:05 PM.

Displaying Text

len

The **len** function tells you how many characters are in a word or sequence of words. (The name "len" is an abbreviation of "length.") All characters are counted, including the space character. For example, to find the length of the sentence "The cat is on the mat," type this into a Web page:

```
There are <% =len("The cat is on the mat.") %> characters in
"The cat is on the mat."
```

When you view the page in Internet Explorer, you should see this:

There are 22 characters in "The cat is on the mat."

left

You can use the **left** function to look at the first few characters of a word or sequence of words. For example, to find the first character of "Frankenstein," type this into a Web page:

```
"Frankenstein" begins with the letter <% =left("Frankenstein", 1) %>.
```

When you view the page, you should see this:

"Frankenstein" begins with the letter F.

right

To look at the last few characters of a word or sequence of words, use the **right** function. For example, to find the last three letters of "Wednesday," type this into a Web page:

```
The last three letters of "Wednesday" are: <% =right("Wednesday", 3) %>.
```

When you view this page, you should see this:

The last three letters of "Wednesday" are: day.

Example

What if you wanted to take a few letters from the *middle* of something? How would you specify exactly *where* in the middle you wanted to be? For example, how would you take out just the "apple" part of the word "pineapples"?

You could start with the fifth character from the left and then stop at the second character from the right. Or you could do it the following way.

Try typing this into a Web page:

```
<% =right("pineapples", 6) %> <% =left(right("pineapples", 6), 5) %>
```

This line takes the last six letters of the word "pineapples," which make up the word "apples." Then it takes the first five letters of the word "apples," which make up the word "apple."

When you view this page in Internet Explorer, you should see this:

apples apple

Then try typing this into a Web page:

```
<% =left("pineapples", 9) %> <% =right(left("pineapples", 9), 5) %>
```

This line takes the first nine letters of the word "pineapples," which make up the word "pineapple." Then it takes the last five letters of the word "pineapple," which make up the word "apple."

When you view this page, you should see this:

pineapple apple

Cool Things You Can Do with Date, Time, and Text

Here are some examples of interesting things you can do with date, time, and text functions.

Link of the Day

What if you wanted to have a link that pointed to a different page every day of the week? Here's how you can do that. First, choose the pages (HTML files) on your Web site that you want your link to point to. Name them "Sunday.htm," "Monday.htm," and so on. (If you don't have seven different HTML files, you can copy some of the files or make aliases on your Macintosh to them. The important thing is that there has to be one file or alias for every day of the week.)

To make the link, type

```
<a href= <% =weekdayname(weekday(now)) %>.htm>Link of the Day</a>
```

where you want it to appear. When you click this link in Internet Explorer, it will take you to today's page.

Another Way to Display Today's Date

Earlier we saw how to change the date display from month/day/year to day/month/year like this:

23/1/1997

We can also change the date display so only the last two digits of the year are included. To do this, type

```
<% =day(now) %>/<% =month(now) %>/<% =(year(now) - 1900)) mod 100 %>
```

Now when you view the page, you should see something like this:

23/1/97

Another Way to Display the Time

In an earlier example, we wrote a server-side script to display the current time in words, such as: "The time is 36 minutes and 5 seconds past 13 o'clock." This script used the ASP **hour** function, which returns just the hour part of the current time, based on a 24-hour clock.

In this example, we'll see how to change 24-hour clock times such as "13 o'clock" to 12-hour clock times ("1 o'clock PM"). To do this, we'll need to make the server-side script that uses the **hour** function a little more complicated. Instead of

```
<% =hour(now) %> o'clock
```

we'll need to write a script that looks at the hour and does one of the following:

- If the hour is 0 (zero), the script displays "midnight."
- If the hour is 12, the script displays "noon."
- If the hour is between 1 and 11, the script doesn't change it, but it displays "AM" after "o'clock."
- If the hour is between 13 and 23, the script subtracts 12 (to make it a number between 1 and 11) and displays "PM" after "o'clock."

The script is shown below. It isn't written quite the way a programmer would write it, but it works, and it's fairly easy to understand, since it follows the items in the bulleted list above exactly.

```
The hour is
<% if hour(now) = 0 then %>
   midnight.
<% end if
   if hour(now) = 12 then %>
  noon.
<% end if
   if (hour(now) >= 1) and (hour(now) <= 11) then
     =hour(now) %> o'clock AM.
<% end if
   if (hour(now) >= 13) and (hour(now) <= 23) then
     =hour(now) - 12 %> o'clock PM.
<% end if %>
```

If you type (or better yet, cut-and-paste) this script in a Web page, when you view the page, you should see something like this:

The hour is 4 o'clock PM.

Active Server Pages: Using Variables and Forms

Using Variables, and Forms in Active Server Pages

Forms are a convenient way to communicate with visitors to your Web site. Using forms, you can create a survey form and ask visitors to fill it out. When they fill out the form, you can process the results automatically.

With forms, there are two steps: first you create the form, and then you process it. To create a form for an Active Server Page, just create a standard HTML form.

To try out this example, create an HTML file ("form_response.html") and cut-and-paste the following text into it.

form_response.html

```
<html>
<head><title>Asking for information</title></head>
<body>
<form method="post" action="form_response.asp">
Your name: <input type="text" name="name" size="20"><BR>
Your email: <input type="password" name="email" size="15"><BR>
<input type="Submit" value="Submit">
</form>
</body>
</html>
```

Active Server Pages provide a mechanism for processing forms that, unlike CGI scripting, doesn't involve serious programming: the **Request.Form**.

Considering the form above, we may create the file bellow and get a response.

form_response.asp

```
<html>
<head><title>Responding to a form</title></head>
<body>
Your name is <% =Request.Form("name") %> <BR>
Your email is <% =Request.Form("email") %>
</body>
</html>
```

To display the contents of each field in the form, type:

<% =Request.Form(*fieldname*) %>

where *fieldname* is the name of the field.

Creating a Variable

You'll probably want to do more with your forms than display their contents in a Web page. For example, based on the contents of the form, you may want to create a variable and insert that variable in different places of your response page. You may need to create a variable. To do that, just make up a name and set it equal to the contents of the field.

For example, if you have a field called "CatName" in your form, you can save it into a variable called "TheName" by typing:

```
<% TheName = Request.Form("CatName") %>
```

If you want to display "VisitorName" several times within a text you only need to include the variable in the text. For example:

My cat´s name is <% =TheName %>. Do you want to see <% =TheName %>?.

Example

The form in this example asks users to introduce their names and their favorite color: red, blue, or green. When the form is received, the server responds displaying these data.

nameandcolor.html

```
<html>
<head><title>Name and Color</title></head>
<body>
<FORM ACTION="nameandcolor.asp" METHOD=POST>
Let me know your Name and Favorite Color:
<P>YOUR NAME:
<INPUT TYPE="TEXT" NAME="YOURNAME" SIZE=20>
<P>COLOR:
<INPUT TYPE="RADIO" NAME="COLOR" VALUE="1" CHECKED>Red
<INPUT TYPE="RADIO" NAME="COLOR" VALUE="2">Green
<INPUT TYPE="RADIO" NAME="COLOR" VALUE="3">Blue
<P>
<INPUT TYPE="SUBMIT" VALUE="OK">
</FORM>
</body>
</html>
```

Now, create an ASP file ("nameandcolor.asp") and cut-and-paste the following text into it.

nameandcolor.asp

```
<html>
<head><title>Name and Color</title></head>
<body>
<% TheName = Request.Form("YOURNAME) %>
<% colornumber = Request.Form("COLOR") %>
Hi, <% =Thename %>.<BR>
I know your favorite color is
<% if colornumber = "1" then %>
red
<% end if %>
<% if colornumber = "2" then %>
green
<% end if %>
<% if colornumber = "3" then %>
blue
<% end if %>.
</body>
</html>
```

Active Server Pages: If...Then and For...Next instructions

If....Then...Else

The If....Then...Else instructions sequence is very similar to the one we may find in different kind of scripting languages. Let's check an example.

```
<%
AA="water"

If AA="water" Then
 response.write ("I want to drink water")
Else
 response.write ("I want to drink milk")

End If
%>
```

We may use it this way:

```
<%  AA="water"
If AA="water" Then %>

      I want to drink water

<% Else %>

      I want to drink milk

<% End If %>
```

In both cases we have checked a condition (AA="water"), and we have get a positive instruction (to write the sentence "I want to drink water"). We are allowed to execute any kind of instructions (including If....then....Else) and as many instructions as we want .

For....Next

This instructions is also similar in different programming languages. Let's see a typical example.

```
I want to say "Hello" 10 times<BR>
<% For mynumber = 1 to 10 %>
<% =mynumber %> Hello<BR>
<% Next %>
END
```

In this case we have defined a variable ("mynumber") and using the For...Next instruction we have repeated 10 times line 4. Similarly to If....Then....Else instruction, we are allowed to execute any kind of instructions and as many of them as we want .

The For...Next instruction allows to define the value of the increment.

```
<% For mynumber = 1 to 20 STEP 2
response.write("Hello<BR>")
Next %>
```

```
<% For mynumber = 20 to 1 STEP -2
response.write("Hello<BR>")
Next %>
```

In both cases we will get the same response ("Hello" 10 times). The increment may be positive or negative as shown in the example.

Active Server Pages: Do...Loop / Select...Case instructions

Do While...Loop

Again, we will define a condition and one or more instructions:

```
<%
mynumber=0
Do While mynumber<10

   response.write("Hello<HR>")
   mynumber=mynumber+1

Loop
%>
```

In this example the condition is "mynumber<10" and the instructions defines a response text and an increment of the variable "mynumber". In the example, mynumber will be increased until it gets a value of 10. Then the loop will be abandon. Several instruction may be used within the loop.

Do Until....Loop

Quite similar to the previous one, it also includes a condition and one or more instructions:

```
<%
mynumber=0
Do Until mynumber=10

   response.write("Hello<HR>")
   mynumber=mynumber+1

Loop
%>
```

In this example the condition is "mynumber=10", so mynumber will increased until it is equal to 10, and then the loop will be abandon.

Let's see an example using this Do Until...Loop:

```
<%
myfirstnumber=0
mysecondnumber=0

Do Until myfirstnumber=15

    Do Until mysecondnumber=15
            response.write("X")
            mysecondnumber=mysecondnumber+1
    Loop
    Response.write ("<BR>")
    myfirstnumber=myfirstnumber+1
    mysecondnumber=myfirstnumber

Loop

Response.write ("END")
%>
```

The result of the script is this one:

```
XXXXXXXXXXXXXXX
XXXXXXXXXXXXXX
XXXXXXXXXXXXX
XXXXXXXXXXXX
XXXXXXXXXXX
XXXXXXXXXX
XXXXXXXXX
XXXXXXXX
XXXXXXX
XXXXXX
XXXXX
XXXX
XXX
XX
X
END
```

Select Case....End Select

This is a very useful instruction in case we want to check different values for variable. Lets check an example:

```
<%
mynumber=3
Select Case mynumber
  Case 1
     Response.write ("Number 1")
  Case 2
     Response.write ("Number 2")
  Case 3
     Response.write ("Number 3")
  Case 4
     Response.write ("Number 4")
  Case 5
     Response.write ("Number 5")
  Case Else
     Response write ("Mynumber is higher than 5")
End Select
%>
```

In this example above, we have defined mynumber as 3, so they are executed the instructions following line 8 (in this case only one instruction is executed, but they may be several instructions). Case Else is not necessary.

Let's try a second example:

```
<%
username=request.form("username")
Select Case username
  Case "Peter"
     Response.write ("Hello, Peter")
  Case "John"
     Response.write ("Hello, John")
  Case "Joe"
     Response.write ("Hi, Joe")
  Case Else
     Response write ("I do not know you")
End Select
```

```
13 %>
```

Let's see a different example:

backgroundform.html

```
<html>
<head><title>Chose background color</title></head>
<form action="backgroundresponse.asp" method="post">
Which color do you prefer to use as your background?
<BR>
<input type="radio" name="kindofcolor" value="defined" checked>
Defined color
<select name="definedcolor">
<option value="#FFFFFF">White</option>
<option value="#FF0000">Red</option>
<option value="#00FF00">Green</option>
<option value="#0000FF">Blue</option>
</select>
<BR>
<input type="radio" name="kindofcolor" value="custom">
Custom color
<input type="text" size="8" name="mycolor"></input>
<BR><input type="Submit" value="Submit"></input>
</form>
</body>
</html>
```

backgroundresponse.asp

```
<%
kindofcolor=Request.form("kindofcolor")

Select Case kindofcolor
   case "defined"
    colorofbackground=Request.form("definedcolor")
       Select Case colorofbackground
                case "#FFFFFF"
                    texttoshow="White"
                case "#FF0000"
                    texttoshow="Red"
                case "#00FF00"
                    texttoshow="Green"
```

```
            case "#0000FF"
                texttoshow="Blue"
        End select
    case "custom"
        colorofbackground=Request.form("mycolor")
        texttoshow="Custon color"
End select
%>

<html>
<head><title>Chose background color</title></head>
<body bgcolor="<% =colorofbackground %>">
<center>
<H1><% =texttoshow %></H1>
</center>
</form>
</body>
</html>
```

You may try this example (works only online):

Which color do you prefer to use as your background?

(•) Defined color White

( ) Custom color

Submit

Active Server Pages: Subroutines and Include/virtual

Subroutines

Subroutines have the same utility in ASP as it has in other languages.

In the next two examples, we have asked our visitors his name, and depending on that answer a different response is sent to the client. The response will be the same in both cases, but in the second one subroutines are used. The use of subroutines may be very useful when there

are a lot of instructions to be perform within a subroutine. This way it will allow us to simplify the structure of our script.

Example 1

```
<%
TheName=request.form("name)

if TheName="John" then
response.write ("Hi, John. How are you?")
response.write ("<br>Did you know I got married last
month?")
else
response.write ("Hi. How are you?")
end if
%>
```

Example 2

```
<%
TheName=request.form("name)

if TheName="John" then
ResponseToJohn()
else
ResponseToUnknown()
end if

Sub ResponseToJohn()
response.write ("Hi, John. How are you?")
response.write ("<br>Did you know I got married last
month?")
End Sub

Sub ResponseToUnknown()
response.write ("Hi. How are you?")
End Sub

%>
```

In order to call a subroutine, we will use this kind of code:

Whatever()

Where Whatever is the name of the subroutine (it is recommended to use a very descriptive name of the task we want to perform within the subroutine to make it easier to understand the script). We may also provide information to the subroutine in order to perform the specified task. The data will be provided this way:

Whatever(data1, data2 ... dataN)

In the following example we will provide different data to a unique subroutine depending on the Name of the person provided throw a form:

Example 3

```
<%                                                          
TheName=request.form("name)                                 1
                                                            2
if TheName="John" then                                      3
ResponseToVisitor(35,Sue,New York)                          4
else                                                        5
   if TheName="Peter" then                                  6
   ResponseToVisitor(33,Sally,Los Angeles)                  7
   else                                                     8
   response.write("Who are you?")                           9
   end if                                                   10
end if                                                      11
                                                            12
Sub ResponseToVisitor(AA,BB,CC)                             13
response.write ("I know your are" & AA & "years             14
old, ")                                                     15
response.write ("you are married to" & BB & ",              16
and")                                                       17
response.write ("you are living in " & CC)                  18
End Sub                                                     19
%>
```

In line 14 it is specified AA is the first variable to be get, BB the second one, and CC the third one. The values for the three variables are provided in the same order in line 5 or line 8.

The example above also shows subroutines are very useful to avoid repeating a specific number of tasks several times within the script, so that the script looks more organized and it is smaller.

Include/virtual

Server Site includes or SSI is a very simple programing language but it also has a very limited number of instructions. We will consider only one option SSI allows us to use within our asp scripts: include/virtual.

In the next example we will use the include option of SSI in a our asp script (response.asp). This command allows as to add a set of instructions from different files (file1.txt and file2.txt bellow) and execute them.

Example 4

response.asp

```
<%
TheName=request.form("name)

if TheName="John" then
   %>
   <!--#include virtual="/file1.html" -->
   <% else %>
   <!--#include virtual="/file2.asp" -->
   <%
end if %>
```

File1.html

```
Hi, John.<br>
I know your are 31 years old, you are married to Sue,
and you are living in New York.
```

File2.asp

```
<%
for i=1 to 3
response.write(Thename & "...<BR>")
next
response.write("Who are you?")
%>
```

In this case, if the name of the person who is visiting our page is John, then we will respond with file1.html. If not, then we will execute some asp instructions from file2.asp.

The include file must be a text file (.txt, .html, .htm, .shtml, .asp...). Although we have used file1.html and file2.asp, the script will work exactly in the same way with file1.txt and file2.txt (changing the name of the files would have no effect).

By using SSI and asp we may also get a secret page:

secret_page.asp

```
<%
UserName=request.form ("username")
Password=request.form("password")
if UserName="myusername" and
Password="mypassword" then
%>
<!--#include virtual="/cgi-bin/secret_info.txt" -->
<% else %>
<Form Action=secretpage.asp method=post>
Username:  <input type=text name=username
size=15><BR>
Password:  <input type=text name=password
size=15><BR>
<input type=Submit Value=Send>
</form>
<% end if %>
```

secret_info.txt

```
This is my secret information:<BR>
My name is John.<BR>
My surname is Smith.<BR>
<BR>End of secret information.
```

In this case it is convenient to save secret_info.txt file in the cgi-bin directory (the .txt file is not accessible by visitors from this directory, but it will be accessible from our top directory).

Active Server Pages: Session and Application

In this section we will learn how to keep information from the user in our server (**Session** method) and how to share information between users (**Application** method). This is only a basic tutorial for beginners, so only basic features will be described.

The **Session** method

The first time a user accesses to a our pages some connections and disconnections took place. During this process the server and the client will interchange information to identify each other. Due to this exchange of information our server will be able to identify a specific user and this information may be use to assign specific information to each specific client. This relationship between computers is call a session. During the time a session is active, it is possible to assign information to a specific client by using **Session** method. We will use an example to explain this method:

Let's suppose we want to allow specific user to access the information on our site or directory and we want to show a username in all pages visited by the user. In this case we may use the Session method.

In this example, we will ask the username of the person in our index.asp page

respondtoforms.asp

```
<% IF Request.form="" THEN %>

<html>
<title>Our private pages</title>
<body>
In order to access this pages fill the form below:<BR>
<form method="post" action="index.asp">
Username: <input type="text" name="username" size="20"><BR>
Password: <input type="password" name="password" size="15"><BR>
<input type="Submit" value="Submit">
</form>
</body>
</html>

<% ELSE %>

<%
```

```
IF Request.form("username")="Joe" AND Request.form("password")="please"
THEN
%>
<%
Session("permission")="YES"
Session("username")="Joe"
%>

<html>
<title>Our private pages</title>
<body>

Hi <% =Session("username") %>, you are allow to see these pages: <BR>
<A HREF="page1.asp">Page 1</A><BR>
<A HREF="page2.asp">Page 2</A>

</body>
</html>

<% ELSE %>

Error in username or password

<% END IF %>

<% END IF %>
```

Let's explain how this page works:

In line 1 it is checked whether information is submitted throw a form. If the answer is negative (Request.form=""), a form is displayed asking for username and password.

After filling the form and submitting it, as Request.form is not "" and the script will jump to line 15. In line 17 they are checked the username and password. If user name is "Joe" and Password is "please", then two variables are set for the client (lines 21-22):

```
Session("permission")="YES"
Session("username")="Joe"
```

These variables will be kept in the server during the time the session is active (normally it will expire after 20 minutes without contact).

Finally, if username and password are correct, a response page with links is send to the client with the name of the user in the top. In this example, if the username or password are incorrect the response page will include the text in line 38.

Now, let's suppose the user clicks in the link "Page 1" (page1.asp). The code of page1.asp will be the following one:

page1.asp

```
<% IF Session("permission")="YES" THEN %>

<html>
<title>Page 1</title>
<body>

Hi <% =Session("username") %>, welcome to Page 1 <BR>
This page is empty at the moment, but it will be very interesting in the next future

</body>
</html>

<% ELSE %>

You are not allowed to access this page

<% end IF %>
```

In line 1 it is check whether the value for Session("permission") is "YES". If the answer is positive a page with information is send to the client. If the answer is negative, the text in line 15 is send.

NOTES:

- Session method is suitable for sites with a limited number of visitors. For sites with a bigger number of visitors it is preferable to keep the information in the clients computer (by using cookies).
- To create more variables associated to a specific client we must substitute the text between brackets in Session("text").
- The corresponding security features in the client's browser must be enable.

The **Application** method

With **Session** method we have defined a value for Session("whatever")="Joe", but this information can not be share between visitors (Session("whatever") has a unique value for each visitor). To allow sharing information **Application** method is used.

For a better understanding of this method we will create a counter which will be shown in the same page. In order to make it work, copy the code below to your server:

counter.asp

```
<%
Aplication.Lock
Application("pagevisits")=Application("pagevisits")+1
Application.Unlock
%>

<html>
<title>Page under construction</title>
<body>

Under construction<BR><BR>
Page views: <% =Application("pagevisits") %>

</body>
</html>
```

In the first part of this code, as Application method is shared between different clients, it is necessary to prevent other clients from modifying the information in Application("pagevisits"). **Application.Lock** will avoid that by stopping the information to be shared, and **Application. Unlock** will allow the information to be shared again. Line 3 increases the value for the counter.

Finally a html code is send to the client, including the value of the counter.

NOTES:

- The information save as Application("whatever") as shown in this tutorial is lost each time the server is restarted.

Session and **Application** method has been used to create a simple chat script (copy and paste the code to your site and it will work immediately.

Active Server Pages: Dictionary

The Dictionary object

In order to learn how Dictionary object works we will create a small script which will translate number 1 to 10 from English to Spanish.

translate.asp

```
<%
SET MyDictionary=CreateObject("Scripting.Dictionary")

        MyDictionary.Add "one","uno"
        MyDictionary.Add "two","dos"
        MyDictionary.Add "three","tres"
        MyDictionary.Add "four","cuatro"
        MyDictionary.Add "five","cinco"
        MyDictionary.Add "six","seis"
        MyDictionary.Add "seven","siete"
        MyDictionary.Add "eight","ocho"
        MyDictionary.Add "nine","nueve"
        MyDictionary.Add "ten","diez"

EnglishNumber="four"
SpanishNumber=MyDictionary.Item (EnglishNumber)
Response.Write(SpanishNumber)
%>
```

How the script works

- Fist we have define a Dictionary named "Mydictionary" (line 2)
- We have add to the dictionary the data corresponding to the different number in English and Spanish (lines 4 to 13).

  When adding pairs of English and Spanish numbers to the Dictionary object, the number

writen in English is a **Key**, and the number writen in Spanish a **Item**.

- In line 15 we have defined a variable named EnglishNumber and we have provided a value for this variable (in red).
- In line 16 we have defined a new variable (SpanishNumber) and we have get its value from the dictionary by indicating we want to get the **Item** corresponding to a specific **Key** (EnglishNumber).
- In line 17 the translated number is send to our visitor. The response will be "cuatro".

We may change the values in our dictionary by using this kind of code:

- MyDictionary.Key ("one")="1" example

  In our original script the key "one" will be substitute by a new key value ("1"). The item "uno" will not be changed.
- MyDictionary.Item ("two")="2" example

  In our original script the item corresponding to key "two" will be substitute by a new item value ("2"). The key "two" will not be changed.

We may display the number of element pairs in the dictyonary by using this code:

- MyDictionary.Count example

If we want to check whether a key exists in our dictionary before responding to our visitor we will use this kind of comparisoncode

```
if MyDictionary.Exists ("ten")=True  then
        Response.Write("this key is included in the dictionary")
lse
        Response.Write("Error: no such a key in the dictionary")
end if
```

Example: Translation of a number from English to Spanish

This example uses most of the elements explained above. Try it online

- If there is no information posted to the script (line 6), the script will send to the visitor the form in the Sendform() subrouting (lines 34-40).
- When a request to translate a number is get the script will check whether the number corresponds to a key in the dictionary (line 25). If the response is afirmative the corresponding item is send to the visitor. In case the key does not exists, a "No

translation available" response is send to the visitor (line 29).

translation.asp

```
<html>
<title>Page under construction</title>
<body>

<%
if request.form="" then
     Sendform()
else
     SET MyDictionary=CreateObject("Scripting.Dictionary")

          MyDictionary.Add "one","uno"
          MyDictionary.Add "two","dos"
          MyDictionary.Add "three","tres"
          MyDictionary.Add "four","cuatro"
          MyDictionary.Add "five","cinco"
          MyDictionary.Add "six","seis"
          MyDictionary.Add "seven","siete"
          MyDictionary.Add "eight","ocho"
          MyDictionary.Add "nine","nueve"
          MyDictionary.Add "ten","diez"

     EnglishNumber=request.form("EnglishNumber")
     Response.Write("English number: " & EnglishNumber)

     if MyDictionary.Exists (EnglishNumber)=True then
          SpanishNumber=MyDictionary.Item (EnglishNumber)
          Response.Write("<BR>Spanish number: " & SpanishNumber)
    else
          Response.Write("<BR>Spanish number: " & "No translation available")
    end if
end if
%>

<% Sub Sendform() %>
<form action=translation.asp method=post>
Write a number in English<BR>
<input type=text size=30 name=EnglishNumber><BR>
<input type=submit Value="Enter to my Secret Page">
</form>
```

```
<% End Sub %>

</body>
</html>
```

Example: Password protected information

In this example keys and items are used as usernames and passwords. It is very similar to the one above. Try it online

secretpage.asp

```
<%
if request.form="" then
    Sendform()
else
    SET MyDictionary=CreateObject("Scripting.Dictionary")
    MyDictionary.Add "John","123"
    MyDictionary.Add "Peter","456"
    MyDictionary.Add "Anna","789"
    Username=request.form("Username")
    Password=request.form("password")

    if MyDictionary.Exists (Username)=True AND Password=MyDictionary.Item
(Username) then
        SecretInfo()
    else
        Response.Write("Error: incorrect userame or password")
    end if
end if
%>

<% Sub Sendform() %>
<form action=secretpage.asp method=post>
Username: <input type=text size=30 name=Username><BR>
Password: <input type=password size=30 name=Password><BR>
<input type=submit Value="Translate to Spanish">
</form>
```

```
<% End Sub %>
<% Sub SecretInfo() %>
<html>
<head>
<title>My Secret Page</title>
</head>
<body bgcolor=FFFFFF>
<center>
<h1>This is my secret info</h1>
Hello !<BR>
Do you want to be my friend?
</center>
</body>
</html>
<% End Sub %>
```

Active Server Pages: Dictionary examples

Example: Modify a key

In this example we have change the value "one" to "1".
The resulting page will show the value "uno".

translate.asp

```
<%
SET MyDictionary=CreateObject("Scripting.Dictionary")

        MyDictionary.Add "one","uno"
        MyDictionary.Add "two","dos"
        MyDictionary.Add "three","tres"
        MyDictionary.Add "four","cuatro"
        MyDictionary.Add "five","cinco"
        MyDictionary.Add "six","seis"
        MyDictionary.Add "seven","siete"
        MyDictionary.Add "eight","ocho"
        MyDictionary.Add "nine","nueve"
        MyDictionary.Add "ten","diez"

        MyDictionary.Key ("one")="1"

EnglishNumber="1"
SpanishNumber=MyDictionary.Item (EnglishNumber)
Response.Write(SpanishNumber)
%>
```

Example: Modify an item

In this example we have change the value "dos" to "2".
The resulting page will show the value "2".

translate.asp

```
<%
SET MyDictionary=CreateObject("Scripting.Dictionary")

        MyDictionary.Add "one","uno"
        MyDictionary.Add "two","dos"
        MyDictionary.Add "three","tres"
        MyDictionary.Add "four","cuatro"
        MyDictionary.Add "five","cinco"
        MyDictionary.Add "six","seis"
        MyDictionary.Add "seven","siete"
        MyDictionary.Add "eight","ocho"
        MyDictionary.Add "nine","nueve"
```

```
        MyDictionary.Add "ten","diez"

        MyDictionary.Item ("two")="2"

EnglishNumber="two"
SpanishNumber=MyDictionary.Item (EnglishNumber)
Response.Write(SpanishNumber)
%>
```

Example: Count elements in the dictionary

The value of the response page will be "10"

translate.asp

```
<%
SET MyDictionary=CreateObject("Scripting.Dictionary")

        MyDictionary.Add "one","uno"
        MyDictionary.Add "two","dos"
        MyDictionary.Add "three","tres"
        MyDictionary.Add "four","cuatro"
        MyDictionary.Add "five","cinco"
        MyDictionary.Add "six","seis"
        MyDictionary.Add "seven","siete"
        MyDictionary.Add "eight","ocho"
        MyDictionary.Add "nine","nueve"
        MyDictionary.Add "ten","diez"

Response.Write(MyDictionary.Count)
%>
```

Active Server Pages: Cookies

Cookies method is very similar to Session method: the basic difference is that with Cookies method the information is save in the clients computer and not in the server, so it is more suitable for sites with a lot of visitors. This method implies sending information to the client and

requesting it whenever the information is needed. Additionally, we will learn how to delete the information save in the clients computer when it is not necessary anymore.

When the visitor gets to our asp file we may save information related with him in his computer. The order will be like this one:

<% response.Cookies ("whatever")="information" %>

When this line is executed, the visitor will have the information in his computer, and whenever we need that information, we may request it using this code:

<% =request.Cookies ("whatever") %>
or
<% variable1=request.Cookies ("whatever") %>

Let's try an example using Cookies method: let's consider we have visitors checking our site several times and we want to let them know how many times they have accessed to our computer.

```
<% If Request.Cookies ("NumberVisits")="" Then %>
       <% Response.Cookies ("NumberVisits")=1 %>
       This is your first visit to this page. Welcome.
<% Else %>
      <%  VarNumberVisits=Request.Cookies ("NumberVisits")
      VarNumberVisits=NumberVisits+1
      Response.Cookies("NumberVisits")=VarNumberVisits %>

      Welcome back to this page. You have visited this page <% =VarNumberVisits %>
times.
     <BR>Check my great links
     <BR>.....
     <BR>.....

<% End If %>
```

Cookies method may be used to show visitors specific information we have requested throw a form, as for example a list of links related to a specific theme, information to allow access to the content of a page or to personalize the page (background color, frames or not frames...), information to fill a form automatically, etc.

Active Server Pages: Open, Read and Create files

Open and Read content from a text file

Example 1: This one will be the basic code we need to open a text file:

```
1  <%
2  Set fs = CreateObject("Scripting.
   FileSystemObject")
3
4  Set wfile = fs.OpenTextFile("c:\Mydir\myfile.txt")
5  filecontent = wfile.ReadAll
6
7  wfile.close
8  Set wfile=nothing
9  Set fs=nothing
10
11 response.write(filecontent)
12 %>
```

Line 2 will create the appropriate environment which allows to perform the operations involving files in the server. We have defined a variable named "fs" to do it (we may change the name of this variable).

In line 4 we have create a new variable named "wfile" and we have apply the method OpenTextFile to variable "fs". We have also define which is the exact location of the file we want to open in this line (the complete path is necessary).

In line 5 we have read all the content of the file to a variable named "filecontent" using the instruction "ReadAll".

Lines 7 to 9 are use to let the server know we have finished all operations involving files.

In line 11 we have response to the client with the content in the variable "filecontent".

Example 2: Let's suppose we have a file with different kind of information in each line (a name in the first line, the last name in the second one, and the age in the third one), and we want to use them separately. This one will be the script we may use:

```
1  <%
2  Set fs = CreateObject("Scripting.
   FileSystemObject")
3
4  Set wfile = fs.OpenTextFile("c:\Mydir\myfile.txt")
5  firstname = wfile.ReadLine
6  lastname = wfile.ReadLine
7  theage = wfile.ReadLine
8
9  wfile.close
10 Set wfile=nothing
11 Set fs=nothing
12
13 %>
14
15 Your first name is <% =firstname %><BR>
16 Your last name is <% =firstname %><BR>
17 Your are <% =firstname %> years old<BR>
```

This example is very similar to the previous one, but in this case each line we have read from "myfile.txt" has been saved to a different variable (lines 5 to 7), and they have been used in lines 15 to 17 to respond to the client.

Example 3: This example will read all lines in the file, and the response page will include the content of each line with its line number.

```
<%
Set fs = CreateObject("Scripting.
FileSystemObject")

Set wfile = fs.OpenTextFile("c:\Mydir\myfile.txt")

counter=0
do while not wfile.AtEndOfStream
 counter=counter+1
 singleline=wfile.readline
 response.write (counter & singleline & "<br>")
loop

wfile.close
Set wfile=nothing
Set fs=nothing

%>
```

In line 6 we will define the variable "counter", and in line 7 to 11 we will repeated instructions within the Do_While _Loop until the file does not reach the end of the file (the condition is "not wfile.AtEndOfStream").

Example 4: Let's suppose we have a file with a number in line 1 and a second number in line 2.

```
<%
Set fs = CreateObject("Scripting.
FileSystemObject")

Set wfile = fs.OpenTextFile("c:\Mydir\myfile.txt")

number1 = Clng(wfile.ReadLine)
number2= Clng(wfile.ReadLine)

number1and2 = number1 + number2
response.write (number1and2)

wfile.close
Set wfile=nothing
```

```
14 Set fs=nothing
15
16 %>
```

In the previous examples we were able to save the value in a line to a variable, but that variable was a string class variable. In this example we have saved the content of line 1 and line 2 to variables "number1" and number2" by using the function "Clng". This function has allow us to add both numbers in line 9 and send the result to the client (line 10).

Create and Write a text file

Example 1: The basic code we need to create a file is very similar to that one we have used to open a file:

```
   <%
1  thetext="Write this text in the file"
2
3  Set fs = CreateObject("Scripting.
4  FileSystemObject")
5
6  Set wfile = fs.CreateTextFile("c:\Mydir\myfile.txt",
7  True)
8  wfile.Write (thetext)
9
10 wfile.close
11 Set wfile=nothing
12 Set fs=nothing
13
14 response.write("Text created")
   %>
```

The differences to instructions when opening a file are line 6 and line 7:

The method used in line 6 is "CreateTextFile"; it is necessary to indicate the complete path to the file we want to create; in line 6 we may use the instruction True (to allow over-writing an existing file) or False (if the file exits, it is not over-written).

Line 7 will write in the file the string in variable "thetext".

We may also use this instruction to add content to the file

wfile.WriteLine (thetext1)
wfile.WriteLine (thetext2)
...

In this case we will write the content in variable "thetext1" in line 1, content in "thetext2" in line 2 etc.

Example 2: Let suppose we want to record the IP address of all visitor to our page to a file named "mylog.txt".

```
1  <%
2  VisitorsIP=Request.ServerVariables ("REMOTE_ADDR")
3
4  Set fs = CreateObject("Scripting.FileSystemObject")
5
6  Set wfile = fs.OpenTextFile("c:\Mydir\mylog.txt",
7  forappending)
8  wfile.WriteLine (VisitorsIP)
9
10 wfile.close
11 Set wfile=nothing
12 Set fs=nothing
13
14 response.write("IP registered")
   %>
```

The IP address is requested in line 2 (check Functions and Procedures). In this case we have open the file "mylog.txt" in line 6 with the instruction "forappending". this instruction will allow us to open the file and add at the end of it the IP address of our last visitor.

Active Server Pages: Global.asa

Global.asa is a text file locate in your main directory (/global.asa). Bellow is shown the basic extructure of a global.asa file.

global.asa

```
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">

Sub Application_OnStart
.......
End Sub

Sub Application_OnEnd
.......
End Sub

Sub Session_OnStart
.......
End Sub

Sub Session_OnEnd
.......
End Sub

</SCRIPT>
```

This file will be activated in this cases:

- When the first visitor accesses our pages
- When a new session starts.

In both cases, we may determine a series of events to be execute in the file above.

1. Application_OnStart

   It is execute before the first session is started.
2. Application_OnEnd

   It is execute when the application is finished.
3. Session_OnStart

It is execute when the server creates a new session (when a new client acccesses our server).

4. Session_OnEnd

   It is execute when a session is abandon of after certain period of time without contact between the client and the server (normaly after 20 minutes or so from the last request from a specific client, the server will consider he is not going to come back, so it will delete all the information related to that session).

Lets try a very simple example:

Active Users Counter

Just copy the code in the table to a text file and save it in the main directory of your site ("/global.asa").

global.asa

```
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">

Sub Application_OnStart
application("activevisitors")=0
End Sub

Sub Application_OnEnd
End Sub

Sub Session_OnStart
application.lock
application("activevisitors")=application("activevisitors")+1
application.unlock
End Sub

Sub Session_OnEnd
application.lock
application("activevisitors")=application("activevisitors")-1
application.unlock
End Sub

</SCRIPT>
```

The first time a visitor gets to our pages, global.asa will be executed, and consequently, **application("activevisitors")** in line 4 will get a value equal to "0". Immediately (as a new session has started), in line 12, **application("activevisitors")** will be increased by one. Each time a new visitor gets to our pages **application("activevisitors")** will be increased by one, and identically, each time a session is finished, this parameter will be reduce by one (line 18).

In case we want to show the number of visitors in our page, we must use this kind of code :

```
index.asp
There are  <% =application("activevisitors") %> active visitors.
```

Active Server Pages: Active Server Pages Server-Side Scripting

Functions and Procedures

abs (*n*: number)

Function. Returns the absolute value of *n*.

chr (*asciicharcode*: number)

Function. Returns a string containing the ASCII character specified by *asciicharcode*.

For a list of ASCII characters and character codes, see http://www.microsoft.com/workshop/author/newhtml/htmlr018.htm.

date

Function. Returns the current system date as a string.

day (*thedate*: string)

Function. Extracts the day of the month from *thedate* and returns it as a number.

hour (*thetime*: string)

Function. Extracts the hour value from *thetime* and returns it as a number.

left (*thestring*: string, *n*: number)

Function. Returns a string containing the first *n* characters of *thestring*.

len (*thestring*: string)

Function. Returns the number of characters in *thestring*.

minute (*thetime*: string)

Function. Extracts the minutes value from *thetime* and returns it as a number.

month (*thedate*: string)

Function. Extracts the month value from *thedate* and returns it as a number.

monthname (*themonth*: number)

Function. Returns a string containing the name of the month whose number is specified by *themonth*.

now

Function. Returns the current system date and time.

Request.Form (*fieldname*: string)

Function. Returns the contents of the field whose name is *fieldname*.

ServerVariables

> **Request.ServerVariables ("HTTP_User-Agent")**
>
> Function. Returns the client browser type as a string.
>
> **Request.ServerVariables ("REMOTE_ADDR")**

Function. Returns the client's IP (Internet protocol) address as a string.

Request.ServerVariables ("REMOTE_HOST")

Function. Returns the client's domain name as a string.

Request.ServerVariables ("SERVER_NAME")

Function. Returns the domain name of the server as a string. If the server has no domain name, returns the server's IP address as a string.

Additional ServerVariables

Response.redirect (*anotherURL*: string)

Procedure. Redirects the current request to one specified by *anotherURL*.

right (*thestring*: string, *n*: number)

Function. Returns a string containing the last *n* characters of *thestring*.

second (*thetime*: string)

Function. Extracts the seconds value from *thetime* and returns it as a number.

Server.URLEncode (*thestring*: string)

Function. Converts *thestring* from a string to a URL by "escaping" certain characters. For example, each space character is replaced with %20. Returns the URL.

time

Function. Returns the current system time as a string.

timevalue (*thetime*: string)

Function. Extracts the time value from *thetime* and returns it as a string.

weekday (*thedate*: string)

Function. Extracts the weekday value from *thedate* and returns it as a number.

weekdayname (*weekday*: number)

Function. Returns a string containing the name of the weekday whose number is specified by *weekday*.

year (*thedate*: string)

Function. Extracts the year value from *thedate* and returns it as a number.

Operators

The following operators are supported in ASP server-side scripts:

- Addition (+)
- And
- Assignment (=)
- Concatenation (&)
- Division (/)
- Exponentiation (^)
- Is
- Equals (=)
- Greater Than (>)
- Greater Than or Equal to (>=)
- Less Than (>)
- Less Than or Equal to (>=)
- Mod
- Multiplication (*)
- Negation (-)
- Not
- Not Equal To (<>)
- Or
- Subtraction (-)

  Xor

Active Server Pages: Server Variables

Server Variables

All available server variables may be obtained with the script bellow. Just save the script to your server and it will show you the values for each of the ServerVariables (use an url like http://www.yoursite.com/yourdir/servervariables.asp?a=hello&b=end)

servervariables.asp

```
<html><head><TITLE>Server Variables</TITLE></head>
<body bgcolor=FFFFFF>
<%
For Each Key in Request.ServerVariables
response.write Key & ": " & request.servervariables(Key) & " <br><br>"
Next
%>
</body></html>
```

ALL_HTTP
ALL_RAW
APPL_MD_PATH
APPL_PHYSICAL_PATH
AUTH_PASSWORD
AUTH_TYPE
AUTH_USER
CERT_COOKIE
CERT_FLAGS
CERT_ISSUER
CERT_KEYSIZE
CERT_SECRETKEYSIZE
CERT_SERIALNUMBER
CERT_SERVER_ISSUER
CERT_SERVER_SUBJECT
CERT_SUBJECT
CONTENT_LENGTH
CONTENT_TYPE
GATEWAY_INTERFACE
HTTPS
HTTPS_KEYSIZE
HTTPS_SECRET_KEYSIZE
HTTPS_SERVER_ISSUER
HTTPS_SERVER_SUBJECT
INSTANCE_ID

INSTANCE_META_PATH
LOCAL_ADDR
LOGON_USER
PATH_INFO
PATH_TRANSLATED
QUERY_STRING
REMOTE_ADDR
REMOTE_HOST
REMOTE_USER
REQUEST_METHOD
SCRIPT_NAME
SERVER_NAME
SERVER_PORT
SERVER_PORT_SECURE
SERVER_PROTOCOL
SERVER_SOFTWARE
URL
HTTP_PRAGMA
HTTP_HOST
HTTP_ACCEPT_ENCODING
HTTP_ACCEPT_LANGUAGE
HTTP_ACCEPT_CHARSET
HTTP_COOKIE
HTTP_VIA
HTTP_X_FORWARDED_FOR
HTTP_CACHE_CONTROL
HTTP_ACCEPT
HTTP_USER_AGENT
HTTP_REFERER
ASP_VERSION
ASP_VERSION_MAJOR
ASP_VERSION_MINOR
ASP_OS

The scripts included in this tutorial are simple examples of what we may get by using Active Server Pages (ASP). Our server may allow us to use ASP only in the "cgi-bin" directory or in all the site. Depending on this fact we may need to change the path to the files we are manipulating. Here, we will consider a server which allows using ASP in the entire site.

ASP counter - A simple counter. It will save the result to a file.
Form to mail - Easy to use. Three examples for different porpouses. It may not work in your server.
Free For All Link Page - Start your own FFA with as many categories as you want.
Submision to search engines - A Free URL submission service you may add to your site.
"Search the web" utility - Search the web from your site.
Chat - A fully customizable copy and paste script. Chatrooms are now available in your site.
Search files - This script will allow you to find a term or phrase within a directory in your site.
Find links - A small script to help you find links to your site in the net. A good way to spy your competitors.
HTML code reduction - This script will remove all redundant spaces within your HTML pages (allowing a faster transmission). Just let the script know the directory you want HTML files to be reduced.
Active Users counter - This script allows to know how many clients are visiting your site and to show it in your pages.
Simple ad rotator - By using this script you may show to your visitors a different ad in each page avoiding repetition and showing them in a specific order independely of the page visited.