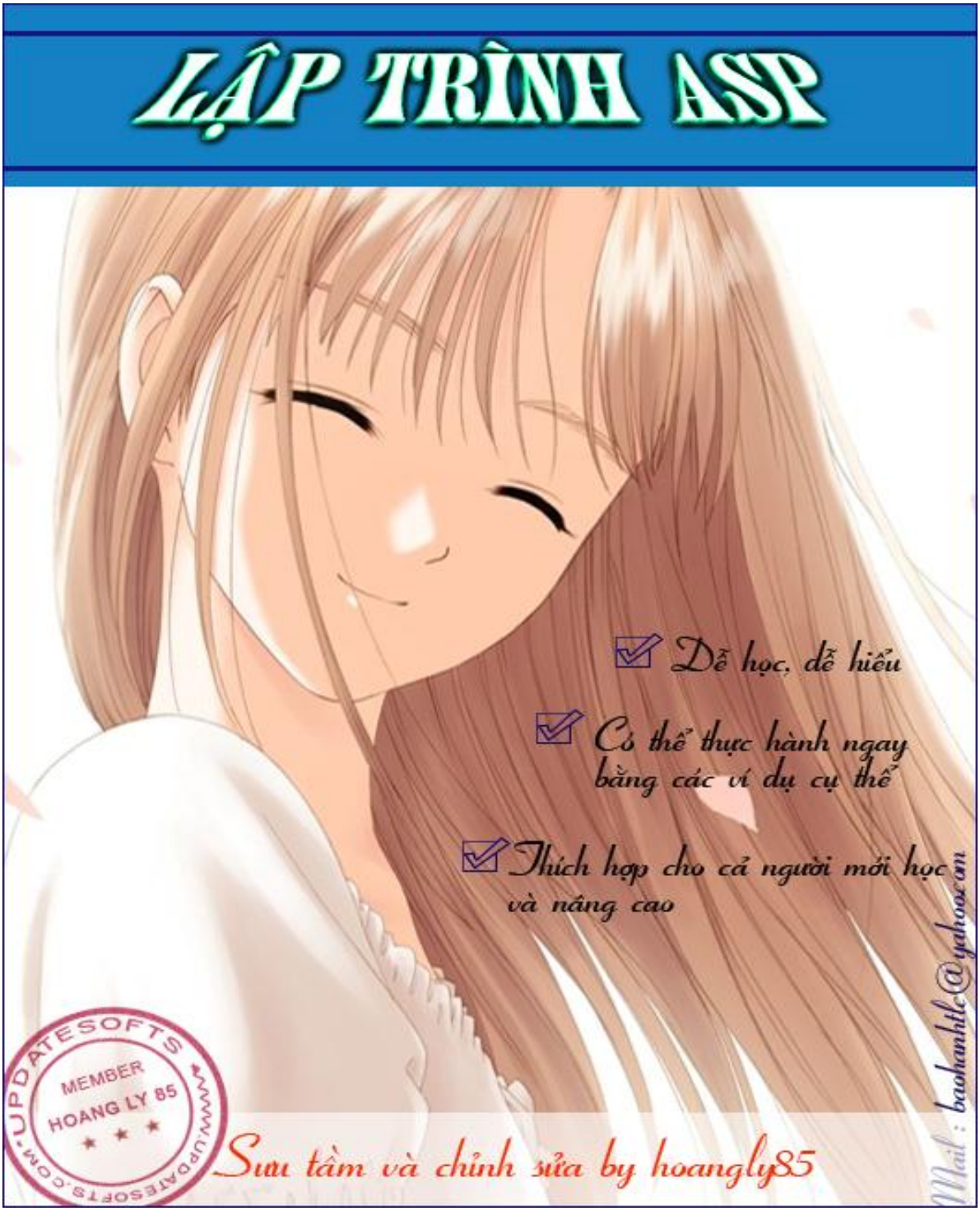

LẬP TRÌNH ASP
Dễ học, dễ hiểu
Có thể thực hành ngay bằng các ví dụ cụ thể
Thích hợp cho cả người mới học và nâng cao
UPDATESOFTS
MEMBER
HOANG LY 85
WWW.UPDATESOFTS.COM
Sưu tầm và chỉnh sửa by hoangly85
Mail : baohanhtle@yahoo.com

Lời nói đầu

Internet đã đưa nền công nghệ thông tin của thế giới bước vào một trang sử mới. Các trang Web cũng đã thực sự làm rạng rỡ kỷ nguyên thông tin bằng sức mạnh của nó. Với HTML là ngôn ngữ căn bản của mình, Web đã hiển thị và truyền tải được các văn bản, âm thanh và hình ảnh qua Internet, tuy nhiên HTML chỉ đơn giản mô tả cách thức văn bản, đồ hoạ và những dữ liệu khác hiện thị trên màn hình Web mà không mô tả bản thân dữ liệu và cũng không tiện dụng trong việc định vị, sửa đổi tài liệu. Bản thân HTML là tĩnh vì thế khi một trang Web đã được đưa lên Internet nó phải được sửa đổi, bổ sung ngoại tuyến và nạp lại thì mới có hiệu lực. Đó chính là những mặt hạn chế của HTML bởi vì sức mạnh tối thượng của Web chính khả năng chuyển thông tin mới cho khách hàng gần như theo thời gian thực và có khả năng tuỳ biến thông tin đó cho phù hợp với từng người, trong thế giới của Web, khả năng đó còn được gọi là khả năng cá biệt hoá.

Với các trang Web động các nhà quản trị và thiết kế Website chỉ cần tạo trang Web một lần , đó là một trang mẫu chung, sau đó server sẽ đổ dữ liệu chuyên biệt vào trang mẫu chung và tạo các trang chuyên biệt theo từng yêu cầu. Microsoft đã đưa ra một ngôn ngữ mới giúp cho việc lập trình máy chủ và tạo ra các Web động một cách hiệu quả - đó là ASP (Active Server Pages). ASP sẽ giúp cho các nhà thiết kế và nhà quản trị có thể tạo ra các Website có các trang Web động đa năng và mạnh mẽ.

Giáo trình này sẽ giúp bạn có những hiểu biết khá cơ bản về ASP, bắt đầu từng bước từ đơn giản đến phức tạp. Chúng tôi rất hy vọng nó thực sự hữu ích đối với bạn.

Trong quá trình xây dựng giáo trình này, chúng tôi không tránh khỏi nhiều thiếu xót, mong nhận được nhiều ý kiến và góp ý từ các bạn. Xin cảm ơn.

huudq@einfovn.com

Chương 2 - Lập trình CSDL với ASP

I. TỔNG QUAN VỀ ASP

Trong phần này chúng ta sẽ tìm hiểu tổng quan về ASP. Các bạn sẽ hiểu được

- *ASP là gì?*
- *Ta có thể làm được những gì với ASP?*
- *ASP hoạt động thế nào?*
- *Làm thế nào để thiết kế được một Website bằng ASP?*
- *Sau khi thiết kế được Website bằng ASP, làm thế nào để đưa (xuất bản - publish) nó lên mạng?*
- *Tìm hiểu ASP thông qua một ví dụ đơn giản*

1.1. ASP là gì?.

ASP (Active Server Pages) là một môi trường lập trình phía máy chủ, cho phép ta xây dựng các trang Web có nội dung động (interactive Web pages) và những ứng dụng mãnh mẽ trên nền Web.

Khác với những ngôn ngữ lập trình Web khác như Perl, PHP, Cold Fusion,… đều sử dụng các ngôn ngữ riêng của mình, ASP sử dụng các ngôn ngữ đã có sẵn và phổ biến như JavaScript hay VBScript. Đây là những ngôn ngữ rất đơn giản và dễ sử dụng.

Lưu ý rằng các script là những ngôn ngữ giới hạn, dùng để viết các chương trình đơn giản. Để thực hiện những công việc phức tạp như kết nối CSDL, lập trình đồ hoạ, giao tiếp với thiết bị phần cứng,... thì ta phải sử dụng các component (thành phần) đã được tạo trước bằng các ngôn ngữ lập trình mạnh mẽ hơn như C++, VB,... Các script của ASP chỉ có lệnh gọi đến các component này mà thôi.

Một trang ASP là một tệp (file) có **phần mở rộng là .ASP** và được đặt trên một thư mục Web trên máy chủ**. N**ội dung của nó là sự kết hợp của các thẻ HTML, ngôn ngữ kịch bản (scripting) như JavaScript, VBscript và các thành phần (components - như các DLL và ActiveX control..) được viết trong các ngôn ngữ khác nhằm tạo ra một ứng dụng Web mạnh mẽ và hoàn chỉnh và có khả năng bảo mật cao.

Nội dung của một trang ASP được mô tả trong hình sau:

TRANG.ASP

1. Nội dung văn bản
2. <Các thẻ HTML>
3. <script> Các Client-side script </script>
4. **<%Các Server-side script%>**

Như vậy, một trang ASP là sự kết hợp của 4 thành phần:

1. Các văn bản (những gì mà người sử dụng có thể nhìn được qua trình duyệt Web).
2. Các thẻ HTML (nằm trong căp thẻ <>) dùng để điều khiển việc hiển thị các văn bản.
3. Các client-side script (được viết bằng Javascript hoặc VBscript) dùng để tăng cường khả năng tương tác giữa trang Web của ta với người sử dụng và xử lý nội dung động của trang Web ngay tại trình duyệt trên máy trạm mà không cần liên lạc với server. Các client-side script thường được viết bằng ngôn ngữ JavaScript và phải được đặt giữa cặp thẻ <Script> và </Script>.

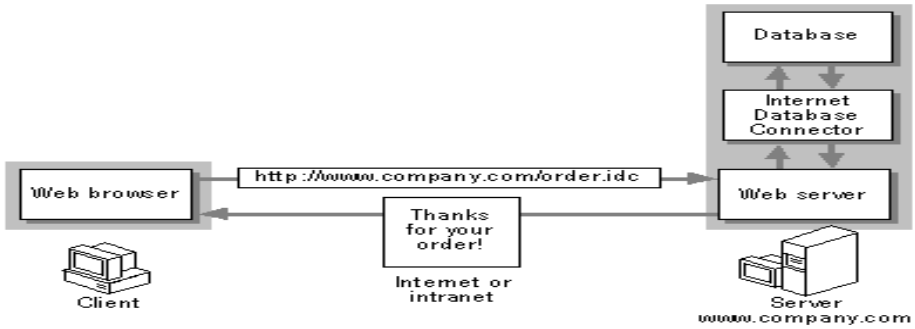
Với ba thành phần này, mã nguồn của nó phải được tải vào trong trình duyệt của máy trạm. Chính bản thân trình duyệt phải hiểu được chúng, xử lý chúng và hiển thị nội dung trang Web cho người dùng. Người dùng có thể xem được toàn bộ mã nguồn của ba thành phần này.

4. Thành phần thứ tư, mà ta sẽ tìm hiểu trong chương này, là các Server-side Script. Chúng là những đoạn mã script được viết bằng ngôn ngữ VBScript (hoặc cũng có thể bằng JavaScript), dùng để lập trình phía máy chủ (Server), nghĩa là chúng được biên dịch và xử lý ngay tại máy chủ bằng trình biên dịch ASP (được cài đặt kèm theo dịch vu IIS – trình bày bên dưới). Các Server-side script được đặt trong cặp thẻ "**<%**" và "**%>**". Ta có thể viết các mã lệnh kết nối CSDL, xử lý dữ liệu, đọc/ghi tệp, và tất cả những thao tác khác với hệ thống tại máy chủ nếu cần. Sau khi được biên dịch và thực thi, các mã server-side script sẽ không còn nữa. Kết quả của nó là mã của ba thành phần ***1., 2., 3.*** ở trên và IIS sẽ gửi về cho trình duyệt trên máy trạm.

Lưu ý là một trang ASP không bắt buộc phải có đầy đủ cả 4 thành phần trên . Nghĩa là, nó có thể chỉ có <%các mã Server Script%> hoặc chỉ có các mã HTML mà không có <%các mã Server-side Script%>. Trong trường hợp trong một trang ASP mà không có <%các mã Server-side Script%> thì toàn bộ mã nguồn của trang ASP đó sẽ được Web Server gửi trực tiếp cho trình duyệt của máy tràm mà không cần phải xử lý gì cả.

Khi thay đổi, sửa các file ASP trên server ta chỉ cần ghi lại (save) file trên server mà thôi. Vào những lần sau khi trang ASP này được gọi, các script trong file ASP sẽ được tự động biên dịch lại.

1.2. ASP làm việc như thế nào?



Khi kết hợp ASP vào trong một Website, ASP sẽ làm việc theo các tiến trình sau :

User ngồi ở máy trạm, mở trình duyệt (browser) ví dụ như Internet Explorer hoặc Netscape Navigator rồi nhập địa chỉ tới nơi đặt tệp ASP trên máy chủ.

Trình duyệt của user bắt đầu yêu cầu các ASP file trên server.

Web Server (IIS) nhận được yêu cầu và nó sẽ chuyển các tham số này đến trình biên dịch ASP và yêu cầu trình biên dịch ASP biên dịch file .ASP mà máy trạm yêu cầu.

Các scripts của trang ASP trên server bắt đầu được xử lý.

ASP xử lý các file được yêu cầu từ phía user một cách liên tục từ trên xuống dưới (top-down), thực thi các scripts chứa trong file ASP này rồi kết xuất thành một trang Web HTML.

Trang HTML vừa được sản xuất ra sẽ được gửi tới trình duyệt của user tại máy trạm.

Bởi vì các scripts chạy trên server cho nên Web server sẽ xử lý toàn bộ các tiến trình và trang HTML có thể được sản xuất và được gửi tới trình duyệt của user. Điều đó có nghĩa rằng trang Web của bạn bị hạn chế trong phạm vi những gì mà Web server có thể cung cấp.

Một trong các ưu điểm của việc đặt các script của bạn trên server là user sẽ không thể nhìn thấy source code nguyên mẫu của bạn được. Thay vào đó user chỉ có thể nhìn được nội dung của file HTML đã được sản xuất.

1.3. Giới thiệu về Internet Information Server (IIS):

1.3.1. IIS là gì?

Microsoft **I**nternet **I**nformation **S**ervices (các dịch vụ cung cấp thông tin Internet) là các dịch vụ dành cho máy chủ chạy trên nền Hệ điều hành Window nhằm cung cấp và phân tán các thông tin lên mạng, nó bao gồm nhiều dịch vụ khác nhau như Web Server, FTP Server,... Nó có thể được sử dụng để xuất bản nội dung của các trang Web lên Internet/Intranet bằng việc sử dụng "Phương thức chuyển giao siêu văn bản" - Hypertext Transport Protocol (HTTP).

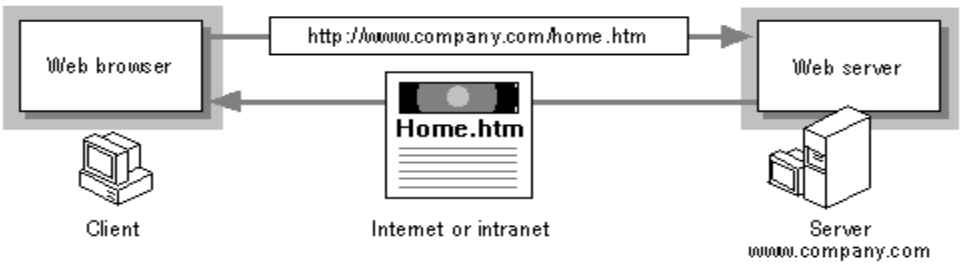
Như vậy, sau khi bạn thiết kế xong các trang Web của mình, nếu bạn muốn đưa chúng lên mạng để mọi người có thể truy cập và xem chúng thì bạn phải nhờ đến một Web Server, ở đây là IIS. Nếu không thì trang Web của bạn chỉ có thể được xem trên chính máy của bạn hoặc thông qua việc chia sẻ tệp (file sharing) như các tệp bất kỳ trong mạng nội bộ mà thôi.

1.3.2. IIS có thể làm được gì?

Nhiệm vụ của IIS là tiếp nhận yêu cầu của máy trạm và đáp ứng lại yêu cầu đó bằng cách gửi về máy trạm những thông tin mà máy trạm yêu cầu. Bạn có thể sử dụng IIS để:

- Xuất bản một Website của bạn trên Internet
- Tạo các giao dịch thương mại điện tử trên Internet (hiện các catalog và nhận được các đơn đặt hàng từ người tiêu dùng)
- Chia sẻ file dữ liệu thông qua giao thức FTP.
- Cho phép người ở xa có thể truy xuất database của bạn (gọi là Database remote access). Và rất nhiều khả năng khác ...

1.3.3. IIS hoạt động như thế nào?



IIS sử dụng các giao thức mạng phổ biến là HTTP (Hyper Text Transfer Protocol) và FPT (File Transfer Protocol) và một số giao thức khác như SMTP, POP3,... để tiếp nhận yêu cầu và truyền tải thông tin trên mạng với các định dạng khác nhau.

Một trong những dịch vụ phổ biến nhất của IIS mà chúng ta quan tâm trong giáo trình này là dịch vụ WWW (World Wide Web), nói tắt là dịch vụ Web. Dịch vụ Web sử dụng giao thức HTTP để tiếp nhận yêu cầu (Requests) của trình duyệt Web (Web browser) dưới dạng một địa chỉ URL (Uniform Resource Locator) của một trang Web và IIS phản hồi lại các yêu cầu bằng cách gửi về cho Web browser nội dung của trang Web tương ứng.

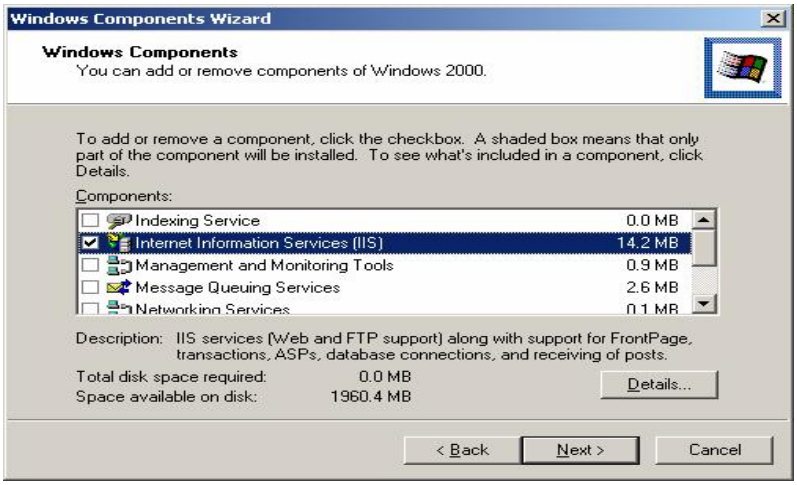
1.3.4. Cài đặt IIS

Hiện tại đã có các phiên bản 3.0, 4.0 và 5.1. Nói chung cách cài đặt không có gì khó và khác nhau lắm giữa các version.

Lưu ý : Tốt nhất là có bản cài ngoài (từ đĩa CD hoặc download từ Internet) hoặc tham khảo bảng sau :

Version	Cài đặt
Windows 98	Cài Personal Web Server trong Add/Remove Programs trong Control Panel
Window NT server 4.0	Cài Internet Information Server trong Add/Remove Programs trong Control Panel hoặc trong Windows NT Option Pack
Windows 2000	Cài Internet Information Server trong Add/Remove Programs trong Control Panel

Để cài IIS, sau khi đưa đĩa Win2K/WinXP vào ta chọn menu Star-> Settings->Control Panel-> Add/Remove Programs -> Add/Remove Windows components -> chọn Internet Information Services -> Next.

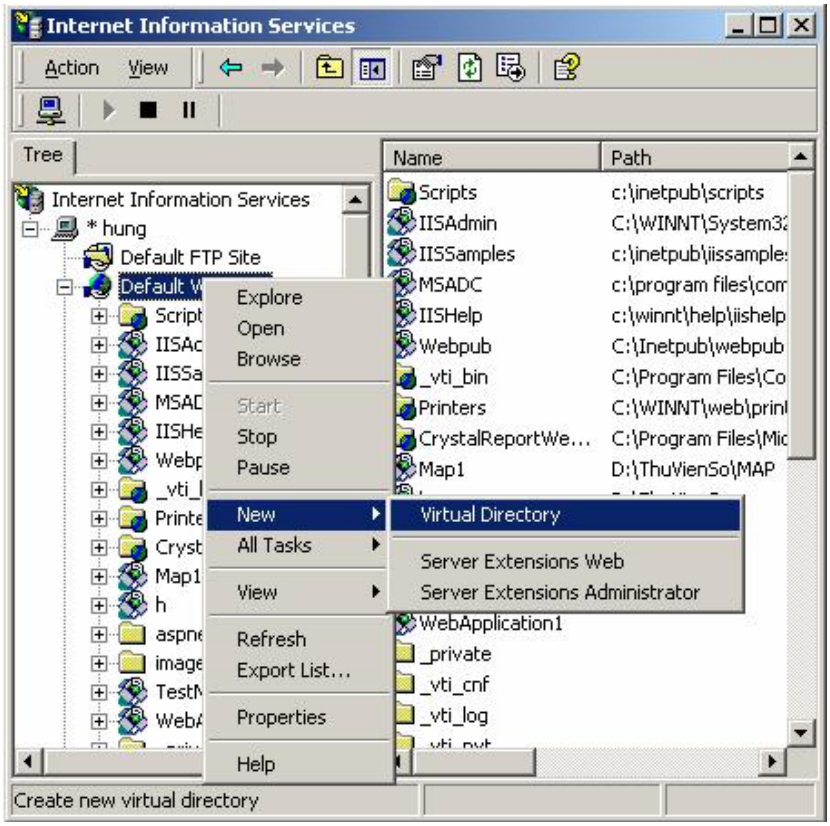


Giao diện màn hình cài IIS

1.3.5. Nạp một ứng dụng Web lên IIS

Sau khi cài trình Web chủ IIS, để xem một trang ASP trước tiên ta phải nạp ứng dụng chứa trang ASP lên trình chủ Web **IIS**, các bước thực hiện như sau:

1. **Bước 1**: Mở trình chủ Web IIS bằng cách vào menu Star -> Sittings -> Control panel -> Administrative Tools -> Internet Services Manager.
2. **Bước 2**: Tạo thư mục ảo (Virtual Directory) cho ứng dụng. Thông thường mỗi ứng dụng Web được đặt trong một thư mục và được tham chiếu đến thông qua địa chỉ URL.
    - **Cách tạo thư mục ảo**: Trên màn hình Internet Information Services ta vào **Default Web Site** -> **New** -> **Virtual Directory**.



Tạo thư mục ảo cho ứng dụng

- Trong ô nhập liệu Alias của hộp thoại **Virtual Directory Creation Wizard** ta nhập tên bí danh cho thư mục ảo, bấm next.
- Chọn đường dẫn thư mục vật lý chứa ứng dụng ta quan tâm. Thông thường thư mục chứa ứng dụng được đặt trong **C:\Inetpub\wwwroot**, chọn thư mục vật lý chứa ứng dụng xong ta bấm Next để đến màn hình cấu hình bảo vệ và đặt quyền cho thư mục ảo.

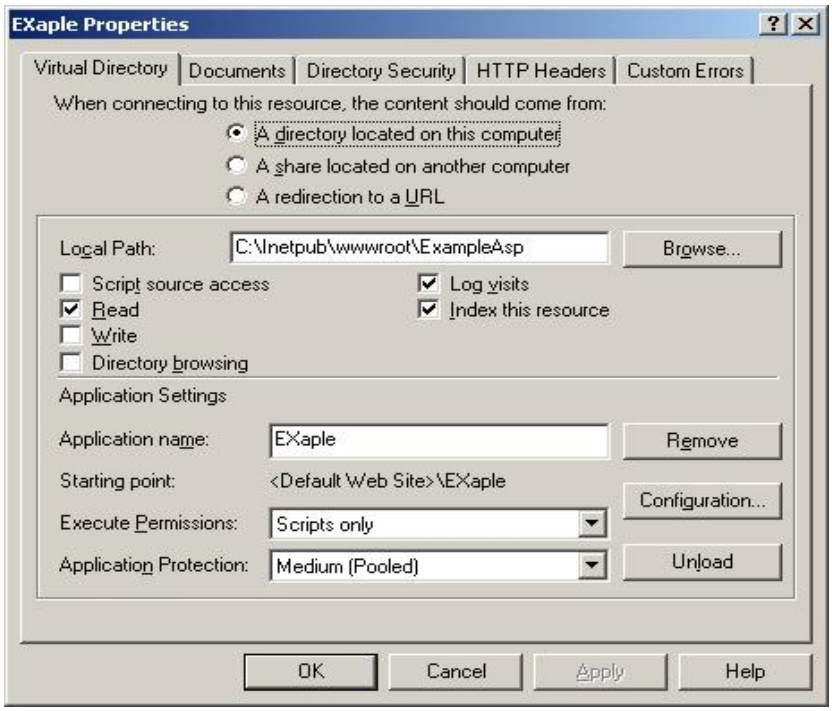
Chọn đường dẫn vật lý cho thư mục ảo

- Đặt quyền cho thư mục ảo như trong hình dưới. Có tất cả 5 quyền gồm **Read** (cho phép đọc nội dung trang), **Runscript** (cho phép thực thi trang kịch bản), **Execute** (thực thi các ứng dụng CGI), **Write** (cho phép ghi vào thư mục ảo), **Browse** (cho phép xem toàn bộ nội dung thư mục thay cho

trang Web mặc định). Hai quyền **Read** và **Run script** là cần thiết để trang ASP có thể truy xuất được.

Màn hình cấp quyền truy cập thư mục ảo

- Chúng ta đặt lại các quyền bảo vệ thư mục ảo và chế độ bảo mật bằng cách nhấn chuột phải lên thư mục ảo mới tạo, rồi vào Properties.

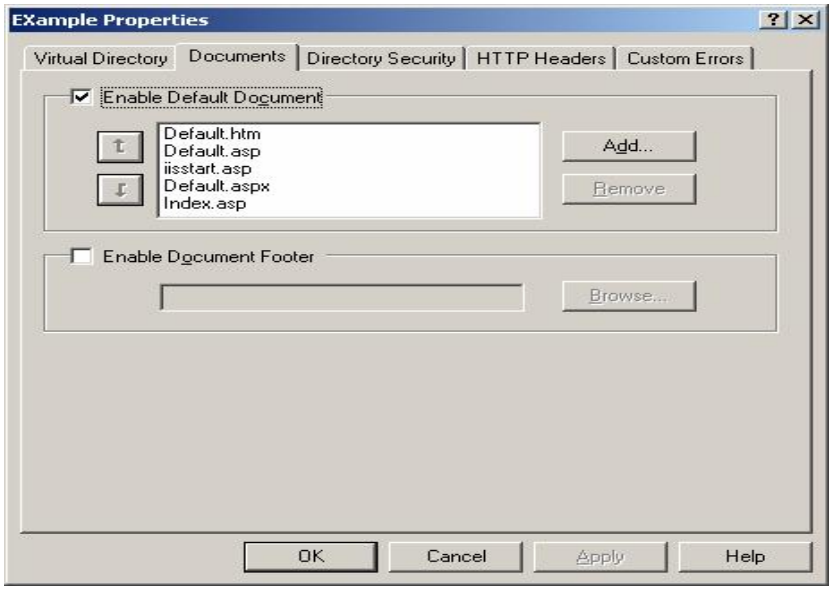


Màn hình Properties của thư mục ảo

3. **Bước 3:** Thiết lập trang mặc định cho thư mục ảo.

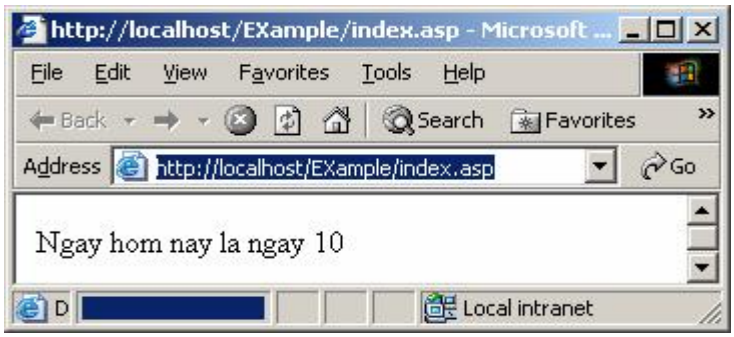
Khi máy client gõ một địa chỉ Web URL tham chiếu đến một ứng dụng mà không đưa ra tên trang cụ thể, lúc này trình chủ sẽ sử dụng trang mặc định. Ta có thể thiết lập một danh sách các trang mặc định, khi ấy **IIS** sẽ tìm theo thứ tự ưu tiên từ trên xuống dưới.

Để lập trang Web mặc định cho thư mục ảo. Từ màn hình trên, ta vào mục **Document**. Bạn có thể xóa hoặt thêm một trang mặc định vào danh sách bằng cách chọn **Add** hay **Remove**.



Màn hình thiết lập trang Web mặc định cho ứng dụng

4. **Bước 4**: Để xem trang Web ta mở trình duyệt Web **Internet Explorer** -> Gõ địa chỉ URL của trang Web vào mục **Address**.



Nếu chúng ta đã tạo trang mặc định cho thư mục ảo là "index.asp" thì ta chỉ cần gõ địa chỉ http://localhost/Example.

1.4. Bắt đầu với ASP qua một ví dụ nhỏ

1.4.1. Mục đích của chương trình

Ta sẽ xây dựng một ứng dụng ASP có chức năng như sau :

Khi user duyệt file này trên server, script sẽ kiểm tra giờ hiện tại nếu :

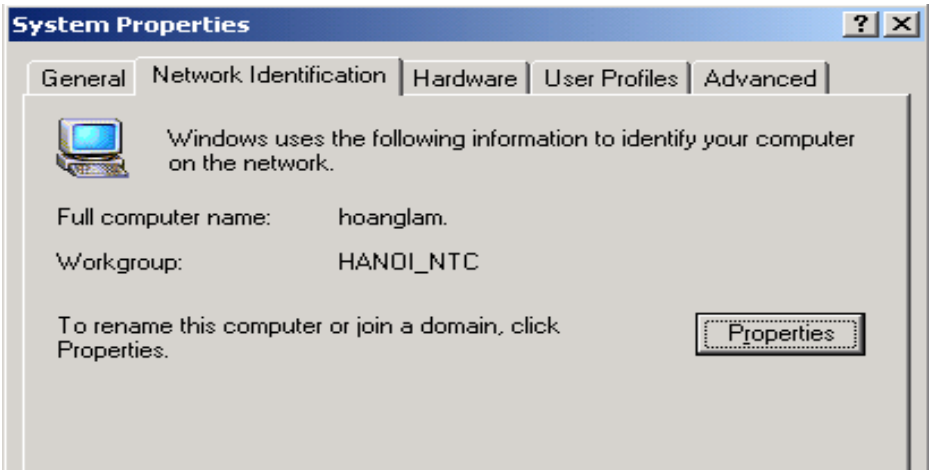
- Trước 12 giờ trưa sẽ hiện " Chào buổi sáng"
- Từ 12 giờ trưa tới 6 giờ chiều thì hiện " Chào buổi chiều "
- Sau 6 giờ chiều hiện " Chào buổi tối"

1.4.2. Các bước thực hiện

1. *Xác định tên máy chủ của bạn và cài đặt IIS*
2. *Tạo một thư mục để lưu trữ các trang .asp, .htm của bạn*
3. *Tạo ra một tệp văn bản thông thường (plan text) có phần mở rộng của tệp là **.asp** và viết các script của ASP, các client-side script (Java Script), các thẻ HTML cần thiết vào đó. Có thể dùng trình soạn thảo văn bản đơn giản như notepad để soạn nội dung của trang .asp này.*
4. *Tạo một thư mục ảo (Virtual Directory) bằng IIS Console để đưa (xuất bản) các trang .asp, .htm của bạn lên mạng.*
5. *Chạy thử bằng trình duyệt Web.*

Trước hết, ta phải cấu hình cho server và phải cài đặt Internet Information Services . Cấu hình cho server ví dụ như sau :

Đặt tên cho server – thực chất là đặt tên cho máy tính (ví dụ là Trungtam). Để đặt tên cho server ta kích nút chuột phải vào biểu tượng Network Neighborhood trên màn hình nền rồi chọn mục Network Identification.



Kích Properties rồi nhập tên server vào

Các thao tác trên chỉ cần làm duy nhất một lần đầu tiên mà thôi

Tạo một thư mục trên đĩa của server, tại đây ta sẽ lưu các tệp ASP, giả sử là C:\ViduASP

Mở một trình soạn thảo văn bản bất kỳ (Notepad của Windows chẳng hạn) để soạn thảo tệp ASP này. Nhập nội dung sau và ghi vào thư mục C:\ViduASP với tên là vd1.asp.

Chú ý là những dòng được **bôi đậm** trong ví dụ dưới đây là các Server-side Script của ASP

```
<html>
<title>Ví dụ về ASP</title>
<body bgcolor="blue">
<font face=".vntime" size=14 color="yellow">
<% TG=time() %>
<% If TG >= #00:00:00# And TG <= #12:00:00# Then%>
<p> Chao buoi sang </p>
<% else %>
<%   if  TG > #12:00:00# And TG <= #18:00:00# Then%>
<p> Chao buoi chieu </p>
<%    else %>
<p> Chao buoi toi </p>
<% end if%>
<% end if%>
</font>
</body>
</html>
```

Giải thích các dòng lệnh

STT dòng	Mã nguồn	Giải thích ý nghĩa của dòng lệnh tương ứng
1	<html>	Bắt đầu 1 tệp HTML
2	<title> Ví dụ về ASP </title>	Hiện tiêu đề của trang
3	<body bgcolor=”blue”>	Đặt nền trang màu xanh
4	< font face=”.Vntime” size = 14 color=”yellow”>	Đặt chữ kiểu .Vntime, kích cỡ 14, màu chữ vàng
5	<% TG=time() %>	Bắt đầu mã lệnh ASP , phải nằm trong cặp thẻ <% và %>. Gán cho biến TG giá trị là thời gian hiện tại. Lưu ý rằng đây là thời gian hiện tại trên server chứ không phải thời gian tại máy trạm (client)
6	<% If TG >= #00:00:00# And TG <= #12:00:00# Then%>	Sử dụng cú pháp if..then kiểm tra xem có phải là buổi sáng không , kiểu ngày phải cho trong #..#
7	Chao buoi sang	Nếu thoả mãn điều kiện trên thì dùng thẻ

		in ra màn hình
8	<% else %>	Sử dụng cú pháp else
9	<% if TG > #12:00:00 # And TG <= #18:00:00# Then%>	Kiểm tra xem có phải buổi chiều không.
10	Chao buoi chieu	Nếu đúng , in ra câu chào
11	<% else %>	Nếu không phải thì còn là trường hợp buổi tối
12	Chao buoi toi	
13	<% end if%>	Đóng lệnh if
14	<% end if%>	Đóng lệnh if
15	 </body> </html>	Các thẻ kết thúc của HTML

Sử dụng IIS để xuất bản (publish) thư mục C:\ViduASP lên mạng

Để cho các clients có thể duyệt đến thư mục C:\ViduASP ở trên server, ta phải tạo ra một Virtual directory (thư mục ảo) chỉ đến thư mục C:\ViduASP. Mỗi một thư mục ảo này có một bí danh (alias) tuỳ ý do ta đặt. Cách làm như sau

Chạy IIS

Kích phải chuột vào mục Default Web Directory và chọn Virtual directory. Sau đó ta duyệt đến thư mục C:\ViduASP. Kích Next, sau đó nhập bí danh (alias) cho thư mục ảo này ví dụ là VIDU và kích Finish

Nếu máy tính tên là Trungtam

Alias của thư mục C:\ViduASP là VIDU thì trên browser của client, user sẽ nhập địa chỉ sau

(có thể bỏ http:// đi cũng được, chỉ cần gõ Trungtam/Vidu/vd1.asp)

Chạy ứng dụng vd1.asp

Trên một máy trạm bất kỳ (tất nhiên phải cùng mạng và kết nối được tới máy chủ) hoặc chính máy chủ cũng được , ta mở trình duyệt IE (hoặc Netscape Navigator) và gõ địa chỉ sau vào ô địa chỉ :

Kết quả chương trình chạy (thời gian trên server lúc chạy là 20:00)

3. Kết luận :

Trên đây ta đã thấy rằng việc viết và thực thi một ứng dụng ASP khá đơn giản, nếu đã làm quen với môi trường Visual Basic rồi thì sẽ không gặp nhiều khó khăn khi xây dựng 1 ứng dụng ASP. Tuy nhiên ta hãy lưu ý một số điểm sau :

Nếu có nhiều câu lệnh ASP liên tục thì chỉ cần 1 cặp <%...%> là đủ

Ví dụ đoạn lệnh

```
<% a=5 %>
<% b=6 %>
<% c=a+b %>
```

thì tương đương với đoạn lệnh

```
<%
        a=5
        b=6
        c=a+b
%>
```

Hàm Time() ở trên sẽ trả về giờ hiện tại trên server chứ không phải giờ hiện tại trên client

Để in kết quả lên browser của client ta có thể dùng lệnh ASP sau :

Response. Write <xâu cần in >

Vậy thì ứng dụng vd1.asp có thể viết ngắn gọn hơn thành

```
<html>
<title>Ví dụ về ASP</title>
<body bgcolor="blue">
<font face=".vntime" size=14 color="yellow">
<%
TG=time()
If TG >= #00:00:00# And TG <= #12:00:00# Then
    Response. Write “<p> Chao buoi sang </p>”
else
    if  TG > #12:00:00# And TG <= #18:00:00# Then
        Response.Write “<p> Chao buoi chieu </p>”
    else
       Response.Write “<p> Chao buoi toi </p>”
   end if
end if
%>
</font>
</body>
</html>
```

Trong phần tiếp theo, chúng ta sẽ tìm hiểu cụ thể hơn về script hay được sử dụng trong ASP là VBScritp và các đối tượng của ASP hỗ trợ lập trình Web.

II. NGÔN NGỮ VBScript VÀ LẬP TRÌNH VỚI ASP

Nếu như ở phần trên bạn đã nắm được khái niệm căn bản về ASP cũng như cách viết một trang ASP đơn giản và xuất bản Website của mình lên mạng thì trong phần này chúng ta sẽ tìm hiểu cụ thể về ngôn ngữ lập trình VBScript, một ngôn ngữ được sử dụng để lập trình trong môi trường ASP, và các đối tượng có sẵn của ASP trợ giúp cho ta xây dựng trang Web bằng ASP. Trong phần này, bạn sẽ nắm được nhưng nội dung sau và có thể sử dụng như một tài liệu tham khảo khi cần thiết:

- Cách nhập/xuất dữ liệu trong ASP.
- Các kiểu dữ liệu của VBScript.
- Cách khai báo, sử dụng biến và hằng trong VBScript.
- Các cấu trúc điều khiển (rẽ nhánh, lặp,..) của VBScript.
- Tạo và sử dụng các chương trình con thủ tục (Sub) và hàm (function).
- Tham khảo các hàm hữu dụng, xử lý xâu ký tự, xử lý ngày tháng, toán học...
- Tham khảo các đối tượng có sẵn trong ASP như Request, Response, Server, Session, Application,...

2.1. Nhập/xuất dữ liệu

Để xuất dữ liệu của đoạn chương trình chạy phía Client (được đặt trong cặp tag <SCRIPT></SCRIPT>) dùng phương thức **document.write**. Tương tự, để xuất dữ liệu trong đoạn chương trình ASP (đoạn lệnh được đặt trong tag <% và %>) dùng phương thức **Response.write**.

Ngoài ra, ta có thể xuất dữ liệu bằng toán tử <%=***giá trị***%>. Giá trị ở đây có thể là một hằng hoặc một biến.

Ví dụ :

- Xuất chuỗi: Response.write “Learn ASP” hoặc <%=”Learn ASP”%>
- Xuất hằng kiểu số : Response.write 5 hoặc <%=5%>
- Xuất giá trị của biến **x**: Response.write **x** hoặc <%=**x**%>

Lệnh Response.write thường được dùng trong một khối lệnh của ASP mà ta chưa muốn kết thúc các script, sử dụng nó thì chương trình sẽ sáng sủa và dễ đọc hơn. Lệnh <%=giá trị%> thường được dùng như những script đơn lẻ mà có thể đóng ngay script, trong trường hợp ta muốn trộn các thẻ HTML với các giá trị (của các biến/hằng) có trong ASP.

Để nhập dữ liệu ta dùng phương thức **Request**.

Cụ thể hai phương thức yêu cầu (Request), trả lời (Response) sẽ được đề cập cụ thể ở mục sau.

2.2. Các kiểu dữ liệu của VBScript.

2.2.1. Khai báo biến :

Tuy rằng ASP không bắt buộc phải khai báo các biến trước khi sử dụng nhưng nếu lập trình viên cứ tuỳ tiện sử dụng biến thì sẽ dẫn đến tình trạng không kiểm soát nổi và khó bắt lỗi khi xây dựng 1 ứng dụng lớn. Vì vậy nên sử dụng <% **Option Explicit %>** ở ngay đầu mỗi tệp ASP, câu lệnh này có nghĩa là mọi biến phải được khai báo trước khi sử dụng, nếu không khi duyệt sẽ sinh ra lỗi.

Để khai báo biến ta dùng lệnh : Dim biến1, biến 2 ...

Việc khai báo biến chỉ đơn thuần như vậy (lưu ý không dùng cú pháp như trong Visual Basic là Dim <tên biến > as <tên kiểu> ví dụ Dim Hoten as String sẽ gây lỗi

Để khai báo kiểu biến mảng ta cũng dùng cú pháp Dim ở trên ví dụ Dim a(10) sẽ tạo ra một mảng a có chỉ số từ 0 .. 10 vậy có nghĩa là a có 11 phần tử

Trong lập trình chuyên nghiệp, khuyến cáo rằng các lập trình viên nên sử dụng các tiền tố (prefix) trước tên các biến ví dụ như biến chứa họ tên kiểu string thì nên đặt là strHoten (tiền tố thì viết chữ thường). Tham khảo cách viết chuẩn trong bảng sau :

Dạng dữ liệu	Tiền tố (prefix)
Boolean	bln
Byte	byt
Collection object	col
Currency	cur
Date-time	dtm
Double	dbl
Error	err
Integer	int
Long	lng
Object	obj
Single	sng
String	str
User-defined type	udt
Variant	vnt
ADO command	cmd
ADO connection	cnn
ADO field	fld
ADO parameter	prm
ADO recordset	rst

2.2.2.Khai báo hằng số

Sử dụng lệnh CONST tên hằng = giá trị

Ví dụ Const Max=100

2.3. Cấu trúc của chương trình ASP.

Như đã đề cập ở phần trước, các đoạn mã Script của ASP được đặt trong cặp dấu **"<%"** và **"%>"** và chúng được xử lý một cách tuần tự từ trên xuống, bắt đầu từ những dòng đầu tiên của tệp .asp. Để xử lý các thao tác phức tạp, ta phải dùng đến các cấu trúc rẽ nhánh, các vòng lặp cũng như là các chương trình con.

2.3.1. Các cấu trúc điều kiện :

a. Cấu trúc điều kiện IF...THEN...ELSE...END IF

Cú pháp:

```
<%
        IF <điều kiện> THEN
                <các câu lệnh 1>
        ELSE
                <các câu lệnh 2>
        END IF
%>
```

Nếu điều kiện thoả mãn <các câu lệnh 1> sẽ được thực thi nếu không <các câu lệnh 2> sẽ được thực thi.

Ta có thể dùng cấu trúc IF lồng nhau

```
<%
        IF <điều kiện 1> THEN
                <các câu lệnh 1>
        ELSE
                IF <điều kiện 2> THEN
                <các câu lệnh 2>
                ELSE
                <các câu lệnh 3>
END IF
END IF
%>
```

Hoặc có thể dùng cấu trúc IF...THEN ....ELSEIF.... END IF

```
<%
```

```
IF <điều kiện 1> THEN
        <các câu lệnh 1>
ELSEIF <điều kiện 2> THEN
<các câu lệnh 2>
        ELSE
<các câu lệnh 3>
END IF
%>
```

Xét ví dụ sau: ví dụ này sẽ kiểm tra xem số n có chia hết cho 123 hoặc 124 hay không.

```
<% Option Explicit %>
<html>
<title>Ví dụ về lệnh IF</title>
<body bgcolor="blue">
<font face=".vntime" size=14 color="yellow">
<%
Const n=123456
if n mod 123 = 0 then
    Response.Write "<p> " & n & "  chia hết cho 123 </p>"
elseif n mod 124 = 0 then
    Response.Write "<p> " & n & "  chia hết cho 124 </p>"
else
    Response.Write "<p> " & n & "  không chia hết cho 123 hay 124 </p>"
end if
%>
</font>
</body>
</html>
```

b. Cấu trúc chọn Select Case ...End Select

Trong trường hợp có nhiều điều kiện chọn ta có thể dùng cú pháp này

```
<%
Select case <tên biến >
        Case <giá trị 1 >
```

```
            <nhóm lệnh 1>
        Case <giá trị 2 >
            <nhóm lệnh 2>
        Case <giá trị 3 >
            <nhóm lệnh 3>
        .....
        Case <giá trị n >
            <nhóm lệnh n>
        Case Else
            <nhóm lệnh n+1>
End Select
%>
```

Xét ví dụ sau: Ta sẽ lấy ngày hiện thời trên máy chủ và xem nó là thứ mấy

```
<% Option Explicit %>
<html>
<title>Ví dụ về lệnh Select Case</title>
<body bgcolor="blue">
<font face=".vntime" size=14 color="yellow">
<%
dim dtmNgay
dtmNgay=Weekday(Date())
‘ Lấy thứ tự ngày trong tuần
‘ 1 là chủ nhật , 2 là thứ hai ...
Select Case dtmNgay
Case 1
    Response.Write “<p> Chủ nhật</p>”
Case 2
    Response.Write “<p> Thứ hai</p>”
Case 3
    Response.Write “<p> Thứ ba</p>”
Case 4
    Response.Write “<p> Thứ tư</p>”
Case 5
```

```
        Response.Write "<p> Thứ năm</p>"
    Case 6
        Response.Write "<p> Thứ sáu</p>"
    Case 7
        Response.Write "<p> Thứ bảy</p>"
    End Select
    %>
    </font>
    </body>
    </html>
```

2.3.2 Các cấu trúc lặp :

a. Cấu trúc lặp tuần tự FOR .... NEXT

Dùng để lặp với số lần đã biết, tuy nhiên có thể ngắt lệnh bằng EXIT FOR

Cú pháp như sau:

```
<%
FOR  <tênbiến>= <giá trị đầu> TO <giá trị biến> STEP  <bước nhảy>
        Các lệnh chương trình
        ' Nếu muốn ngắt dùng lệnh EXIT FOR
NEXT
%>
```

<bước nhảy> là tuỳ chọn, ngầm định là 1

Xét ví dụ: tính tổng từ 1+2+3+4+5+...+100

```
<% Option Explicit %>
<html>
<title>Tinh tổng 1+2+3+...+100</title>
<%
dim i,tong
tong=0
for i=1 to 100
   tong=tong+i
next
Response.Write "<font color=red>Kết quả là " & tong & "</font>"
%>
```

```
</html>
```

Xét ví dụ: tính tổng các số lẻ từ 1+3+5+...+101

```
<% Option Explicit %>
<html>
<title>Tinh tổng các số lẻ 1+3+...+101</title>
<%
dim i,tong
tong=0
for i=1 to 101 step 2
   tong=tong+i
next
Response.Write "<font color=red>Kết quả là " & tong & "</font>"
%>
</html>
```

Xét ví dụ in các số chia hết cho 3 bắt đầu từ số 1000 giảm dần đến 1

```
<% Option Explicit %>
<html>
<title>In </title>
<%
dim i
for i=1000 to 1 step -1
   if i mod 3=0 then
      Response.Write "<font color=blue>" & i & "</font><br>"
      ' thẻ <br> dùng để xuống dòng
  end if
next
%>
</html>
```

b. DO WHILE ...LOOP

Cú pháp

```
<%
      DO WHILE <điều kiện>
            Các câu lệnh
```

```
        LOOP
%>
```

Chừng nào điều kiện còn đúng thì vòng lặp sẽ được thực hiện. Muốn ngắt tại thời điểm bất kỳ dùng lệnh EXIT DO

Xét ví dụ: in các số từ 1 đến 10 ra browser

```
<% Option Explicit %>
<html>
<title>Lenh lap </title>
<%
dim  i
i=1
do while i<=10
  response.write i & "<br>"   ' thẻ <br> dùng để xuống dòng
  i=i+1
loop
%>
</html>
```

c. WHILE ....WEND

Cú pháp

```
<%
        WHILE <điều kiện>
                Các câu lệnh
        WEND
%>
```

Chừng nào điều kiện còn đúng thì vòng lặp sẽ được thực hiện.

Xét ví dụ in các số từ 1 đến 10 ra browser

```
<% Option Explicit %>
<html>
<title>Lenh lap </title>
<%
dim  i
i=1
```

```
while i<=10
  response.write i & "<br>"   ' thẻ <br> dùng để xuống dòng
  i=i+1
wend
%>
</html>
```

d. DO...LOOP UNTIL

Cú pháp

```
<%
        DO
                Các câu lệnh
        LOOP UNTIL <điều kiện>
%>
```

Chừng nào điều kiện còn sai thì vòng lặp sẽ được thực hiện.

Xét ví dụ in các số từ 1 đến 10 ra browser

```
<% Option Explicit %>
<html>
<title>Lenh lap </title>
<%
dim  i
i=1
Do
  response.write i & "<br>"   ' thẻ <br> dùng để xuống dòng
  i=i+1
loop until i>10
%>
</html>
```

2.3.3. Khai báo thủ tục (sub) và hàm (function)

a. Khai báo thủ tục (Sub)

```
<%
Sub <tên thủ tục>
```

```
        các câu lệnh
    End Sub
    %>
    Xét ví dụ sau :
    <%
    Sub  example
        Response.write " Hello World"
    End Sub
    ' Gọi sub bằng lệnh call
    Call example
    %>
    b. Khai báo hàm (Function)
    <%
    Function <tên hàm>
        - các câu lệnh
        - phép gán <tên hàm>= giá trị
    End function
    %>
    Xét ví dụ sau :
    <%
    Function Tinh
    a=100
    b=100
    Tinh=sqr(a)+b
    End function
    Response.write "Ket qua  = " & Tinh
%>
```

2.3.4. Một số hàm hữu dụng trong ASP

* Các hàm xử lý văn bản

a. Hàm TRIM(xâu as string) :

Hàm sẽ bỏ hết các ký tự space ở hai phía của xâu, cả bên trái và bên phải

Ví dụ :

```
<%
```

```
        x="    Hello  "
        y=Trim(x)
        Response.Write y
        ' y sẽ nhận giá trị ="Hello"
%>
```

b. Hàm LEFT(xâu as string,n as integer) :

Hàm sẽ lấy bên trái xâu trên n ký tự

Ví dụ :

```
<%
        x="Hello World"
        y=Left(x,5)
        Response.Write y
        ' y sẽ nhận giá trị ="Hello"
%>
```

c. Hàm RIGHT(xâu as string,n as integer) :

Hàm sẽ lấy bên phải xâu trên n ký tự

Ví dụ :

```
<%
        x="Hello World"
        y=Right(x,5)
        Response.Write y
        ' y sẽ nhận giá trị ="World"
%>
```

d. Hàm LCASE(xâu as string) :

Hàm sẽ chuyển xâu trên về chữ thường

Ví dụ :

```
<%
        x="Hello World"
        y=lcase(x)
        Response.Write y
        ' y sẽ nhận giá trị ="hello world"
%>
```

e. Hàm UCASE(xâu as string) :

Hàm sẽ chuyển xâu trên về chữ hoa

Ví dụ :

```
<%
        x=”Hello World“
        y=ucase(x)
        Response.Write y
        ‘ y sẽ nhận giá trị =”HELLO WORLD”
%>
```

f. Hàm MID(xâu as string, n1,n2) :

Hàm sẽ lấy n2 ký tự trong xâu trên bắt đầu tại vị trí n1

Ví dụ :

```
<%
        x=”Hello World“
        y=mid(x,3,2)
        Response.Write y
        ‘ y sẽ nhận giá trị =”ll”
%>
```

g. Hàm CSTR(biến) :

Hàm sẽ chuyển đổi (convert) giá trị của biến ở trên về kiểu string cho dù trước đó là kiểu gì đi nữa.

Ví dụ :

```
<%
        x=100
        y=cstr(x)
        Response.Write y
        ‘ y sẽ nhận giá trị =”100”
%>
```

h. Hàm SPLIT(xâu as string, ký tự ngăn cách) :

Hàm sẽ cắt xâu trên thành nhiều đoạn bằng cách xác định ký tự ngăn cách ở trên và cho các đoạn đó lần lượt vào một mảng

Ví dụ :

```
<%
        x=”Hà nội ; Hải phòng ; HCM“
        y=split(x,”;”)
```

```
        ‘ Lưu ý rằng lúc này y là một mảng có 3 phần tử đánh số là 0,1,2
        Response.Write y(0)
        ‘ y(0) sẽ nhận giá trị =”Hà nội”
%>
```

i. Hàm JOIN(mảng, ký tự ngăn cách) :

Ngược lại với hàm Split , hàm này sẽ nối toàn bộ các phần tử trong mảng và chèn thêm vào ký tự ngăn cách xen giữa các phần tử

Ví dụ :

```
<%
        dim x(2)
        x(0)=”Hà nội”
x(1)=”Hải phòng”
x(2)=”HCM“
        y=join(x,”-”)
        Response.Write y
        ‘ y sẽ nhận giá trị =”Hà nội – Hải phòng – HCM”
%>
```

2.3.5. Các hàm xử lý số

a.Hàm SQR(n)

Hàm khai căn bậc hai

Ví dụ :

```
<%
y=sqr(9)
response.write y
‘ y sẽ = 3
%>
```

b.Hàm INT(n)

Hàm lấy phần nguyên

Ví dụ :

```
<%
y=int(7.3434)
response.write y
‘ y sẽ = 7
%>
```

c. Toán tử MOD

Toán tử chia lấy phần dư cú pháp như sau : a mod b (lấy phần dư khi chia a cho b)

Ví dụ :

```
<%
y=7 mod 2
response.write y
‘ y sẽ = 1
%>
```

d. Toán tử \

Toán tử chia lấy phần nguyên cú pháp như sau : a \ b (lấy phần nguyên khi chia a cho b)

Ví dụ :

```
<%
y=7 \ 2
response.write y
‘ y sẽ = 3
%>
```

e.Hàm ROUND (số, n)

Hàm sẽ làm tròn số trên với n chữ số thập phân

Ví dụ :

```
<%
y=round(7.657,2)
response.write y
‘ y sẽ = 7.66
%>
```

f.Hàm RND()

Hàm trả về một số ngẫu nhiên bất kỳ trong [0,1]

Ví dụ :

```
<%
y=rnd()
response.write y
‘ y sẽ = 0.76565 chẳng hạn
%>
```

2.3.6. Các hàm xử lý ngày tháng và thời gian

a.Hàm NOW()

Hàm trả về ngày tháng giờ hiện thời

Ví dụ :

```
<%
y=now()
response.write y
‘ y sẽ = 01/05/03 AM 6:13:111
%>
```

b.Hàm TIME()

Hàm trả về giờ hiện thời

Ví dụ :

```
<%
y=Time()
response.write y
‘ y sẽ = “AM 6:13:111”
%>
```

c.Hàm DATE()

Hàm trả về ngày tháng hiện thời

Ví dụ :

```
<%
y=date()
response.write y
‘ y sẽ = 01/05/03
%>
```

d.Hàm WEEKDAY(ngày)

Hàm trả về thứ trong tuần ứng với ngày đã cho : 1 = chủ nhật, 2 = thứ hai ...

Ví dụ :

```
<%
y=weekday(date())
response.write y
‘ y sẽ = 3 (nghĩa là thứ ba)
%>
```

2.3.7. Các hàm và toán tử logic

a.Toán tử AND

Thực hiện phép và giữa 2 giá trị kiểu logic

Ví dụ :

```
<%
y= (7>2) and (5>4)
response.write y
‘ y = true
%>
```

b.Toán tử OR

Thực hiện phép hoặc giữa 2 giá trị kiểu logic

Ví dụ :

```
<%
y= (6>8) or (5>8)
response.write y
‘ y = false
%>
```

2.3.8. Các hàm về mảng

a. Hàm LBOUND (mảng)

Hàm trả về chỉ số nhỏ nhất của mảng

Ví dụ :

```
<%
dim y(10)
response.write lbound(y)
‘ sẽ có kết quả  = 0
%>
```

b.Hàm UBOUND (mảng)

Hàm trả về chỉ số lớn nhất của mảng

Ví dụ : <%

```
dim y(10)
response.write ubound(y)
‘ sẽ có kết quả  = 10
%>
```

2.4. Các đối tượng xây dựng sẵn (built–in objects) trong ASP

Đối tượng là khái niệm trừu tượng nói về một ”vật thể” (hay một structure) có khả năng lưu trữ dữ liệu và thao tác trên các dữ liệu để phục vụ cho một công việc nào đó. Trong đối tượng người ta gọi các dữ liệu là các thuộc tính còn các thao tác là các phương thức. Các đối tượng trong ASP cho phép người lập trình giao tiếp, tương tác với cả server lẫn client. Trong ASP có hai loại đối tượng đó là:

Các đối tượng cơ bản: Application, Session, Server, Request, Response, ObjectContext.

Các thành phần (component) xây dựng sẵn: Dictionary, FileSystemObject, AdRotator, Browser Capabilities,…

2.4.1 Đối tượng Request

Khi người dùng yêu cầu một trang hay đệ trình (submit) một biểu mẫu (form), đối tượng **Request** sẽ lưu trữ và cung cấp tất cả các thông tin từ browser (trình duyệt Web) gửi đến server, đối tượng này được xem như là đối tượng nhận dữ liệu. Các tập hợp (collection**),** thuộc tính (properties) và phương thức (method) của đối tượng này được mô tả như sau:

2.4.1.1. Các tập hợp (Collection) của đối tượng Request

Đối tượng **Request** cung cấp 5 **collection** cho phép chúng ta truy xuất tất cả các loại thông tin về yêu cầu của browser đối với server. Các **collection** của đối tượng **Request** bao gồm:

❖ **Client Cirtificate**

Một tập các giá trị của tất cả các trường (field) hay các mục (entry) trong **Client certificate** mà browser chuyển đi để trình cho server khi truy xuất một trang hay tài nguyên. Các thành phần của tập đều là giá trị chỉ đọc (read-only).

❖ **Cookies**

Cookies là một file văn bản có kích thước nhỏ được lưu trữ trên máy client. Mỗi khi người dùng thăm một Website, ta có thể bí mật gắn một tập tin chứa các thông tin mà mình muốn lên đĩa cứng của họ, chẳng hạn như thông tin về user, thông tin về số lần truy cập website,... Tuy nhiên các **Cookies** không phải được truy cập ngẫu nhiên bởi các Website mà chúng được truy cập bởi các domain tạo ra chúng.

Các **cookies** trong đối tượng **Request** đều là thuộc tính chỉ đọc (read-only) do đó ta chỉ có thể xem các giá trị **cookies** mà không thể sửa đổi giá trị của chúng. Để lấy giá trị của **cookies** ta sử dụng cú pháp sau:

```
Request.Cookies(name)[(key)|.attribute]
```

Trong đó:

- *name*: tên của cookie (kiểu chuỗi)

- *key*: khóa của cookie cần lấy giá trị (kiểu chuỗi)
- *attribute*: thông tin của cookie, là một trong các thông số sau:

+ **Domain**: (chỉ đọc – read only) cookie chỉ được gởi cho đối tượng Request của domain này.

+ **Expires**: (chỉ ghi - write only) chỉ định ngày mà **Cookies** hết hiệu lực (expires), nếu không chỉ định ngày thì cookie sẽ expires khi kết thúc phiên làm việc.

+ **HasKeys**: (chỉ đọc – read only) xác định khóa của cookie có tồn tại không.

+ **Path**: (chỉ ghi- write only) nếu thuộc tính này được xác lập thì chỉ cookie chỉ được gởi cho những Request của đường dẫn này, nếu không thì cookie chỉ được gởi cho những Request thuộc đường dẫn của ứng dụng.

+ **Secure** (chỉ ghi-write only) xác định cookie có bảo mật hay không.

Một **cookie** có thể chứa đựng một tập hợp các giá trị. Ta nói cookie đó có nhiều khóa.

Ví dụ:

```
<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>
```

❖ **Form**

Các **Form** cho phép người dùng nhập vào dữ liệu thông qua các control HTML như edit, radio button, check box, ... Khi người dùng submit một biểu mẫu thì tất cả các giá trị của các control trong phân đoạn <**FORM**> sẽ được gởi lên Web Server khi đặt giá trị của thuộc tính METHOD trong tag <**FORM** > là **POST**. Các thành phần của đối tượng này đều là giá trị chỉ đọc (read only).

Để truy xuất các giá trị của các control HTML mà người dùng **submit** bằng phương thức **POST** ta sử dụng cú pháp sau:

```
Request.Form(controlname)
```

Trong đó *controlname* là tên của control mà ta cần lấy giá trị.

Ví dụ:

```
<HTML>
<BODY>
  Chao ban:
<%
   Response.Write (Request.Form("Ho_Lot"))
   Response.Write ("  " & Request.Form("Ten"))
%>
</BODY>
</HTML>
```

❖ QueryString

Khi người dùng yêu cầu 1 trang hay đệ trình (submit) một biểu mẫu với phương thức **GET** thì tất cả các control HTML trong phân đoạn <**FORM**> của biểu mẫu sẽ được Browser gắn vào URL theo từng cặp tên/giá trị.

QueryString được dùng để lấy về các giá trị trong một biểu mẫu với phương thức là **GET**. Tất cả các thông tin được gởi từ biểu mẫu với phương thức **GET** sẽ được gắn vào URL trên thanh address của browser và do đó mọi người có thể thấy được các thông tin này, tuy nhiên lượng thông tin được gởi này có giới hạn. Các thành phần của tập đều là giá trị chỉ đọc (read-only).

Để truy xuất các giá trị của các control HTML mà người dùng submit bằng phương thức **GET** ta sử dụng cú pháp sau:

```
Request.QueryString (controlname)
```

Ví dụ:

```
<HTML>
  Chào bạn:
<BODY>
   Response.Write
(Request.QueryString("Ho_Lot"))
   Response.Write ("  " &
Request.QueryString("Ten"))
</BODY>
</HTML>
```

❖ ServerVariables

Khi cần lấy giá trị các biến môi trường của Server ta dùng tập **ServerVariables**.

Cú pháp:

```
Request.ServerVariables (variable)
```

với *variable* chỉ định giá trị gì ta cần lấy. Sau đây là một số giá trị tiêu biểu của *variable*

Biến	**Mô tả**
ALL_HTTP	Trả về tất cả các header mà client đã gởi, luôn luôn theo sau HTTP_ và viết hoa
AL_RAW	Trả về tất cả các header ở dạng thô
APPL_MD_PATH	Trả về đường dẫn cho ứng dụng dùng cho DLL ISAPI
APPL_PHYSICAL_PATH	Trả về đường dẫn vật lý tương ứng của đường dẫn
AUTH_PASSWORD	Trả về giá trị đã nhập vào trên hộp thoại xác nhận của client
AUTH_TYPE	Cách thức mà server dùng để kiểm tra xác nhận người dùng
AUTH_USER	Trả về tên của người dùng (username)
CERT_COOKIE	Trả về ID duy nhất của client
CONTENT_LENGTH	Trả về kích thước của dữ liệu mà client gởi
CONTENT_TYPE	Trả về kiểu dữ liệu
GATEWAY_INTERFACE	
HTTP_ <*headername*>	Trả về giá trị chứa trong header *headername*
HTTP_USER_AGENT	Trả về một chuỗi mô tả browser gởi yêu cầu
LOCAL_ADDR	Trả về địa chỉ của server mà browser gởi yêu cầu tới

Ví dụ: Bạn có thể dùng vòng lặp để xem tất cả các biến của server như sau:

```
<%
    For each x in  Request.ServerVariables
       Response.Write (x & ”<BR>”)
    Next
%>
```

2.4.1.2. Thuộc tính (Property) của đối tượng Request

Đối tượng **Request** chỉ có một thuộc tính duy nhất đó là **TotalBytes**. Thuộc tính **TotalBytes** là thuộc tính chỉ đọc (read-only), nó trả về số byte dữ liệu mà người dùng chuyển lên server..

2.4.1.3.Phương thức (Method) của đối tượng Request

Đối tượng **Request** cũng chỉ có một phương thức đó là BinaryRead. Phương thức **BinaryRead** được dùng để lấy dữ liệu đã được client **POST** lên Server. Phương thức này trả về một mảng các giá trị.

Cú pháp:

```
Request.BinaryRead (count)
```

trong đó *count* là một con số nguyên chỉ rõ số byte cần đọc.

Phương thức này sẽ không nhận được dữ liệu nếu trước đó ta đã truy xuất đến tập **Request.Form**. Ngược lại nếu ta đã gọi phương thức này thì ta sẽ không nhận được dữ liệu của các control HTML khi truy xuất tập **Request.Form**.

Ví dụ: Dùng phương thức **BinaryRead** để đọc dữ liệu mà client **POST** lên và đưa vào một mảng.

```
<%
    Dim a,b
    a = Request.TotalBytes
    b = Request.BinaryRead (a)
%>
```

2.4.2. Đối tượng Response

Khi client có yêu cầu một trang từ server thì server có nhiệm vụ thực thi các đoạn VBScript trong trang ASP để tạo ra tập tin HTML rồi sau đó gởi cho client. Đối tượng **Response** sẽ đảm nhiệm việc chuyển kết quả từ server về cho client.

2.4.2.1. Các tập hợp (Collection) của đối tượng Response

Tập hợp của đối tượng **Response** chỉ có **cookies**. Đối tượng **Response** có thể xác lập giá trị của bất kỳ **cookies** nào mà ta muốn đặt trên hệ thống của client. Nếu **cookies** không tồn tại trên client thì nó sẽ được tạo ra.

2.4.2.2. Thuộc tính (Property) của đối tượng Response

- **Buffer**: Dùng để xác định xem kết quả được tạo ra bởi trang ASP có được giữ lại trong vùng đệm hay không. Thuộc tính Buffer nhận một trong 2 giá trị là true hoặc false. Nếu nhận giá True thì kết quả được tạo ra bởi trang ASP sẽ được server giữ trong vùng đệm cho đến khi tất cả các script của trang được xử lý xong, hay đến khi phương thức **Flush** hoặc phương thúc End được gọi. Giá trị này cần được xác lập trước tag <HTML> trong tập tin .asp. Còn nếu thuộc tính Buffer nhận giá trị False thì kết quả sẽ được gởi đi ngay khi nó được xử lý.

Cú pháp:

```
Response.Buffer [= true | false]
```

Trong **IIS** phiên bản từ 4.0 trở về trước false là giá trị mặc định, còn từ phiên bản 5.0 trở về sau thì true là giá trị mặc định.

Ví dụ 1: Kết quả sẽ không được gởi tới browser cho đến khi kết thúc vòng lặp.

```
<% Response.Buffer = true%>
<HTML>
<BODY>
 <%
    For i = 1 to 100 do
       Response.Write (i & ”<br>”)
   Next
 %>
</BODY>
</HTML>
```

Ví dụ 2: Kết quả sẽ được gởi tới browser mỗi lần lặp.

```
<% Response.Buffer = false%>
<HTML>
<BODY>
 <%
    For i = 1 to 100 do
       Response.Write (i & ”<br>”)
    Next
 %>
</BODY>
</HTML>
```

❖ CacheControl

Thuộc tính này dùng để xác định xem proxy server có thể cất giữ kết quả được tạo ra bởi ASP hay không. Mặc định thì proxy sẽ không cất giữ. **CacheControl** chỉ có thể nhận một trong hai giá trị đó là ”public hoặc ”private”. Nếu đặt thuộc tính này là “private” thì chỉ những vùng **cache** riêng mới có thể giữ còn proxy server sẽ không lưu trữ những trang này. Còn nếu đặt thuộc tính này là “public” thì proxy sẽ cất giữ những trang này.

Ví dụ:

```
<% Response.CacheControl = ”Public”%>
hoặc
<% Response.CacheControl = ”Private”%>
```

❖ Charset

Đây là thuộc tính kiểu chuỗi, thuộc tính này ghép tên của tập ký tự vào vùng **context-type** của đối tượng **Response**. Thuộc tính này chấp nhận bất cứ chuỗi ký tự nào bất chấp chuỗi đó đúng hay sai. Giá trị mặc định là **ISO-LATIN-1**

Cú pháp:

Response.Charset (*charsetname*)

Ví dụ:

<% Response.Charset = ””ISO-8859-1”%>

❖ ContentType

Đây là thuộc tính kiểu chuỗi, thuộc tính này đặt kiểu hiển thị của nội dung HTTP cho đối tượng **Response**. Nếu một trang ASP không chỉ định thuộc tính **ContentType** thì **content-type** mặc định sẽ là: **content-type**: text/html

Cú pháp:

Response.ContentType [= *contenttype*]

Sau đây là một vài giá trị *contenttype* thông dụng:

<% Response.ContentType = ”text/HTML”%>

<% Response.ContentType = ”image/GIF”%>

<% Response.ContentType = ”image/JPEG”%>

<% Response.ContentType = ”text/plain”%>

Ví dụ: Đoạn chương trình sau đây sẽ mở một **spreadsheet** trên browser (nếu bạn đã cài đặt Excel vào máy)

```
<%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>
```

❖ **Expires**

Thuộc tính **Expires** đặt thời gian bao lâu (tính theo phút) một trang sẽ được cất giữ ở browser trước khi nó hết hạn (expire). Nếu người dùng quay lại trang đó trước khi nó hết hạn thì trang đã cất giữ trước đó sẽ được hiển thị lên. Nếu ta muốn một trang không bao giờ hết hạn thì ta đặt thuộc tính **Expire** là -1.

Cú pháp:

```
Response.Expires [= number]
```

Ví dụ: Nếu ta muốn cho một trang sẽ hết hạn sau 24 giờ (= 1400 phút) ta đặt như sau:

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

❖ **ExpiresAbsolute**:

Tương tự như thuộc tính **Expires**, thuộc tính **ExpiresAbsolute** đặt một ngày và giờ xác định mà một trang được cất giữ trên browser sẽ hết hạn.

Nếu ta chỉ định thời gian mà không chỉ định ngày cụ thể thì trang sẽ hết hạn tại giờ chỉ định vào ngày mà script được thực thi. Còn nếu ta chỉ định ngày mà không chỉ định thời gian thì trang được browser cất giữ sẽ bị hết hạn vào lúc nửa đêm của ngày chỉ định.

Cú pháp:

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

Ví dụ: Đoạn mã sau đây chỉ định rằng trang sẽ hết hạn vào lúc 4h00 chiều ngày 11 tháng 10 năm 2003:

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

❖ **IsClientConnected**: Thuộc tính này xác định xem client có còn nối kết (connect) với server hay không. Thuộc tính này mang 1 trong 2 giá trị đó là true hoặc false. Mang giá trị true nếu client còn kết nối tới server và mang giá trị false trong trường hợp ngược lại.

Cú pháp:

```
Response.IsClientConnected
```

Ví dụ: Đoạn code sau đây kiểm tra người dùng còn kết nối hay không?

```
<%
 If  Response.IsClientConnected = true then
     Response.Write (”Nguoi dung con connect!”)
 Else
     Response.Write (”Nguoi dung khong con
connect!”)
 End If
%>
```

❖ **Pics**

Thuộc tính này thêm một giá trị vào nhãn **PICS** ở phần **header** của đối tượng **Response**.

Ví dụ:

```
<%
Response.PICS ("(PICS-1.1 <http://www.abc.com/file.html>
        by " & chr(34) & "xyz@yahoo.com" & chr(34) &
        " for " & chr(34) & "http://www.XXX.com" & chr(34) &
        " on " & chr(34) & "2002.10.05T02:15-0800" & chr(34) &
" r (n 2 s 0 v 1 l 2))")
%>
```

❖ **Status**

Thuộc tính này chỉ định giá trị của dòng trạng thái mà server trả về cho client và ta có thể dùng thuộc tính này để chỉnh sửa dòng trạng thái đó. Giá trị của dòng trạng thái bao gồm: ba con số dầu tiên là mã trạng thái và mô tả chi tiết của mã trạng thái đó (chẳng hạn như: 404 **Not Found**).

Cú pháp:

```
Response.Status = statusdescription
```

với *statusdescription* là dòng mô tả trạng thái.

Ví dụ: Đoạn code sau đây sẽ kiểm tra quyền của user dựa vào địa chỉ của họ

```
<%
  Dim IP
   IP = Request.ServerVariables(”REMOTE_ADDR”)
   If  IP <> 172.16.20.99”” Then
       Response.Status = ”401 Unauthorized”
       Response.Write (Response.Status)
       Response.End
   End If
%>
```

2.4.2.3. Phương thức (Method) của đối tượng Response

❖ **AddHeader**

Phương thức **AddHeader** thêm một header HTTP mới và một giá trị cho HTTP **response**. Một khi một header được thêm vào thì ta không thể gở bỏ nó ra.

Trong **IIS** 4.0, bạn phải gọi phương thức này trước bất kỳ kết quả nào gởi tới browser. Trong **IIS** 5.0 bạn có thể gọi phương thức **AddHeader** tại bất cứ nơi nào trong script nhưng phải đứng trước bất cứ lời gọi hàm **Response.Flush** nào trong trang.

Cú pháp:

```
Response.AddHeader name, value
```

Trong đó *name* là tên của header còn *value* là giá trị của header

Ví dụ:

```
<%
 Response.AddHeader ”Cảnh báo”, ”Máy của bạn có
Virus”
%>
```

Chú ý: Tên của header không được chứa dấu gạch dưới.

❖ **AppendToLog**

Phương thức này thêm một một chuỗi vào cuối mục **log** của trình chủ. Bạn có thể gọi phương thức này nhiều lần trong một script, mỗi lần gọi sẽ gắn thêm một chuỗi vào mục **log** của trình chủ.

Cú pháp:

```
Response.AppendToLog
(string)
```

Ví dụ:

```
<%
 Response.AppendToLog ”Client co virus!”
%>
```

Chú ý: Chuỗi cần ghi vào mục **log** không được chứa bất kỳ dấu phẩy (,) nào.

❖ **BinaryWrite**

Phương thức này ghi dữ liệu trực tiếp xuống browser mà không phải chuyển đổi bất kỳ ký tự nào. Phương thức này thường được dùng để ghi dữ liệu ảnh (BLOB) từ cơ sở dữ liệu xuống browser.

Cú pháp:

```
Response.BinaryWrite (data)
```

❖ **Clear**

Phương thức này xóa tất cả các kết xuất HTML được trình chủ đưa vào vùng đệm. Nhưng phương thức này không xóa phần **header** của đối tượng **Response** mà chỉ xóa phần nội dung của đối tượng **Response**. Nếu thuộc tính **Buffer** của đối tượng **Response** được đặt là false thì phương thức này sẽ gây ra lỗi lúc thi hành (vì không có vùng buffer thì lấy gì mà xóa!!!).

Cú pháp:

```
Response.Clear
```

Ví dụ:

```
<%
    Response.Buffer = true
%>
<HTML>
<HEAD>
<TITLE> Kiểm tra phương thức Clear </TITLE>
</HEAD>
<BODY>
<P> Đây là phần nội dung của trang Web. Nội dung này
sẽ được gởi tới người dùng</P>
<P> Bắt đầu xóa Buffer </P>
<%
   Response.Clear
%>
</BODY>
</HTML>
```

Kết quả khi duyệt trang web này là người dùng không thấy gì cả (vì trang HTML mà Server đưa vào trong vùng đệm chưa kịp gởi đã bị xóa bởi việc gọi phương thức clear.)

❖ **End**

Phương thức này dùng để dừng việc xử lý một script và trả về kết quả hiện tại. Nếu thuộc tính Buffer được đặt là true thì khi gọi phương thức này Server sẽ gởi các kết xuất HTML được lưu trong vùng đệm xuống browser. Nếu ta không muốn đưa kết quả xuống cho browser thì ta gọi phương thức clear trước khi gọi phương thức này.

Cú pháp:

```
Response.End
```

Ví dụ:

```
<HTML>
<BODY>
<P> Đoạn văn bản này sẽ được gởi tới browser và người dùng có thể đọc được</P>
<%
  Response.End
%>
<P> Đoạn văn bản này sẽ không được gởi và đã gọi phương thức End rồi</P>
</BODY>
</HTML>
```

❖ **Flush**

Gọi phương thức này để chuyển các kết xuất HTML mà Server lưu giữ lại trong vùng đệm xuống browser ngay lập tức. Nếu thuộc tính Buffer được đặt là false thì thuộc tính này sẽ gây ra lỗi lúc thi hành.

Cú pháp:

```
Response.End
```

Ví dụ:

```
<%
  Response.Buffer = true
%>
<HTML>
<BODY>
<P> Đoạn văn bản này sẽ được gởi tới người dùng ngay khi gọi phương thức Flush. <P>
<P> Một số đoạn văn bản khác sẽ được gởi sau một lúc nữa!!! </P>
<%
  Response.Flush
  Dim i
  For i = 1 to 1000
    Response.Write ” ”
   Response.Write ”Đây là đoạn văn bản tiếp theo!”
   Response.Flush
%>
</BODY>
</HTML>
```

❖ **Redirect**

Phương thức này dùng để chuyển người dùng đến một trang khác được chỉ định trong đường dẫn URL.

Cú pháp:

```
Response.Redirect (URL)
```

Ví dụ sau đây minh họa việc đăng nhập của người dùng.

Tạo tập tin **login.asp** với nội dung sau:

```
<HTML>
<HEAD>
<TITLE> Login to …</TITLE>
</HEAD>
<BODY>
<B>Login</B><BR>
<form method = “post” action = “validate.asp”>
Username: <input type = “text” size = “15%” name = “UserName”> <BR>
Password: <input type = “password” size = “15%” name = “Password”>
<P>
<input type = “submit”  value = “Login” name = “login”>
</P>
</FORM>
</BODY>
</HTML>
```

Tạo tập tin **validate.asp** với nội dung sau:

```
<HTML>
<BODY>
<%
   Dim User, Pass
    User = Request.Form(“UserName”)
    Pass = Request.Form(“Password”)
    If (User = “sv”) and (Pass = “1234”) Then
        Response.Redirect “success.asp”
    Else
        Response.Redirect “login.asp”
```

```
    End If
%>
</BODY>
</HTML>
```

Tạo tập tin **success.asp** với nội dung sau:

```
<HTML>
<BODY>
<P> Bạn đã đăng nhập thành công! </P>
</BODY>
</HTML>
```

❖ **Write**

Phương thức này dùng để ghi dữ liệu ra tập tin kết xuất dạng HTML để gởi cho browser. Dữ liệu này có thể là số, chuỗi, ngày, ...

Cú pháp:

```
Response.Write (text)
```

Ví dụ:

```
<HTML>
<BODY>
<%
 Response. Write ( “Chào bạn đến với ASP!” & “<BR>”)
 Dim x
 x = 100
 Response.Write x
%>
</BODY>
</HTML>
```

2.4.3. Đối tượng Session

Khi bạn mở, đóng ứng dụng hoặc đang làm việc với một ứng dụng nào đó, máy tính sẽ biết bạn là ai. Nhưng khi làm việc trên internet thì đó là một vấn khác: Web Server không biết bạn là ai và bạn đang làm gì bởi vì dòng địa chỉ http:// cung cấp trạng thái của bạn.

ASP giải quyết vấn đề này bằng cách tạo ra một **cookies** duy nhất cho mỗi người dùng, **cookies** này được gởi cho client và nó chứa đựng thông tin để nhận diện ra bạn. Giao tiếp này được gọi là đối tượng **Session.**

Đối tượng **Session** được dùng để lưu trữ thông tin về những thay đổi đối với một người dùng. Các biến được chứa trong đối tượng **Session** chứa thông tin về một người dùng và được dùng chung cho tất cả các trang trong một ứng dụng. Khi có một người

dùng mới, server tạo ra một đối tượng **Session** mới và sẽ hủy **session** đó khi người dùng không nối kết nữa hoặc khi session hết hạn.

2.4.3.1. Tập hợp của đối tượng Session

❖ **Contents**

Tập hợp **Contents** chứa tất cả các phần tử đã được gắn thêm vào đối tượng **Session** trong quá trình thực thi script.

Cú pháp:

```
Session.Contents (key)
```

Trong đó *key* là tên của phần tử cần lấy.

Ví dụ sau đây liệt kê tất cả các **session** đã được dùng trong ứng dụng.

```
<HTML>
<BODY>
<CENTER> Các session trong tập Contents </CENTER>
<%
   Dim x
   For each x in Session.Contents
      Response.Write (x & "=" &  Session.Contents (x) & "<BR>")
   Next
%>
</BODY>
</HTML>
```

❖ **StaticObjects**

Tập **StaticObjects** chứa tất cả các đối tượng gắn vào **session** với tag HTML <**object**>

Cú pháp:

```
Session.StaticObjects(key)
```

Ví dụ: Đoạn chương trình sau đây hiển thị tất cả các đối tượng trong tập **StaticObjects**

```
<HTML>
<BODY>
<center> Các đối tượng trong tập StaticObjects</center>
<%
   Dim x
   For each x in Session.Contents
      Response.Write (x &  "<br>")
   Next
%>
</BODY>
</HTML>
```

2.4.3.2. Các thuộc tính của đối tượng Session

❖ **CodePage**

Thuộc tính **CodePage** cho biết tập ký tự sẽ được dùng để hiển thị nội dung của trang. Sau đây là một vài giá trị **CodePage** và mô tả của chúng

1251 – American English and most European languages

932 – Japanese Kanji

Cú pháp:

```
Session.CodePage( = codepage)
```

Ví dụ: Đoạn chương trình sau đây hiển thị codepage của một trang.

```
<HTML>
<BODY>
<CENTER> CodePage của trang này là:
<%
      Response.Write (Session.CodePage)
%>
</CENTER>
</BODY>
</HTML>
```

❖ **LCID**

Ta dùng thuộc tính **LCID** để thiết lập hay nhận về một con số nguyên mà nó xác định một vùng nào đó. Dữ liệu ngày, giờ và tiền tệ sẽ được hiển thị dựa theo vùng đó.

Cú pháp:

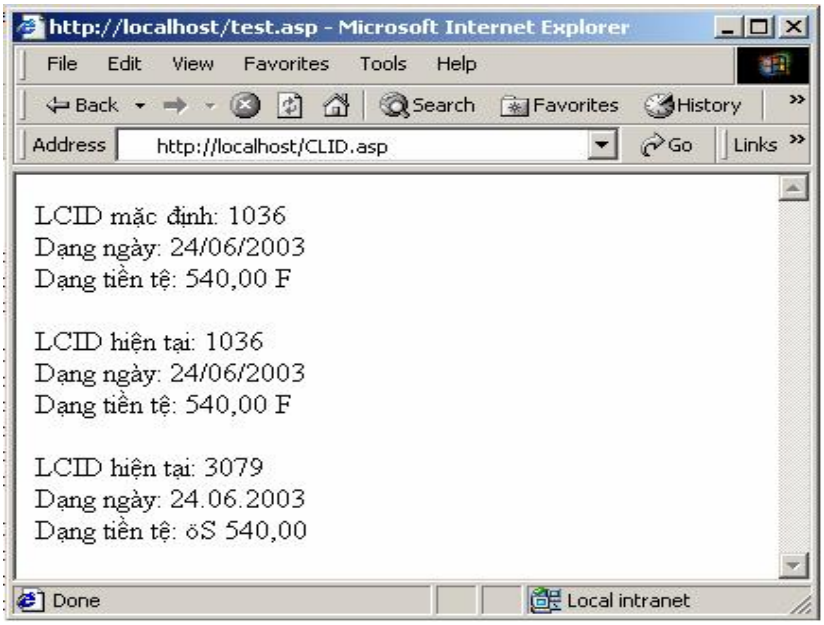
```
Session.LCID( = LCID)
```

Ví dụ:

```
<HTML>
<BODY>

<%
   Response.Write ("LCID mặc định: " & Session.LCID & "<br>")
   Response.Write ("Dạng ngày: " & date() & "<br>")
   Response.Write ("Dạng tiền tệ: " & FormatCurrency(540)& "<BR>")
   Session.LCID = 1036
```

```
    Response.Write (“<p>”)
    Response.Write (“LCID hiện tại: ” & Session.LCID & “<BR>”)
    Response.Write (“Dạng ngày: ” & date() & “<br>”)
    Response.Write (“Dạng tiền tệ: ” & FormatCurrency(540)& “<BR>”)
    Response.Write (“</P>”)

    Session.LCID = 3079
    Response.Write (“<P>”)
    Response.Write (“LCID hiện tại: ” & Session.LCID & “<BR>”)
    Response.Write (“Dạng ngày: ” & date() & “<br>”)
    Response.Write (“Dạng tiền tệ: ” & FormatCurrency(540)& “<BR>”)
    Response.Write (“</P>”)
%>
</BODY>
</HTML>
```

Khi đó kết quả của trình duyệt sẽ là:



❖ **SessionID**

Thuộc tính **SessionID** trả về một con số id duy nhất dùng để nhận diện cho mỗi người dùng. Con số này được server tạo ra và bạn không thể thay đổi giá trị này được.

Cú pháp:

```
Session.SessionID
```

Ví dụ: Đoạn chương trình sau đây hiển thị ra màn hình con số ID.

```
<HTML>
<BODY>
<CENTER> Số ID của bạn là:
<%
```

```
        Response.Write (Session.SessionID)
%>
</CENTER>
</BODY>
</HTML>
```

❖ **TimeOut**

Thuộc tính này dùng để thiết lập hay nhận về khoảng thời gian hiệu lực dành cho đối tượng **Session** trong ứng dụng (tính theo phút). Nếu người dùng không refresh hoặc yêu cầu một trang trong khoảng thời gian hiệu lực đó thì **session** sẽ kết thúc. Mặc định thời gian còn hiệu lực cho một trang là 20 phút.

Cú pháp:

```
Session.TimeOut [=number]
```

Ví dụ:

```
<HTML>
<BODY>
<P>
Thời gian hiệu lực mặc định là:
<% Response.Write (Session.TimeOut) %>
</P>
<% Session.TimeOut = 30%>
<P>
Thời gian hiệu lực bây giờ là:
<% Response.Write (Session.TimeOut) %>
</P>
</BODY>
</HTML>
```

2.4.3.3. Các phương thức của đối tượng Session

❖ **Abandon**

Phương thức **Abandon** dùng để kết thúc **session** của người dùng. Khi phương thức này được gọi, đối tượng **Session** hiện hành chưa bị xóa ngay mà sẽ tồn tại cho tới khi tất cả các Script của trang hiện hành được xử lý xong. Điều này có nghĩa là bạn có thể truy cập các biến **session** trong cùng trang mặc dù bạn đã gọi phương thức **Abandon** trước đó, nhưng truy cập các biến **session** từ những trang khác thì không được.

Cú pháp:

```
Session.Abandon
```

Ví dụ: Tạo 2 tập tin **file1.asp** và tập tin **file2.asp** trong cùng một ứng dụng với nội dung sau:

file1.asp

```
<HTML>
<BODY>
<%
    Session (“Ten”) = “Bill Gate”
    Session.Abandon
    Response.Write (Session (“Ten”))
 %>
</BODY>
</HTML>
```

file2.asp

```
<HTML>
<BODY>
<%
    Response.Write (Session (“Ten”))
 %>
</BODY>
</HTML>
```

Khi người dùng yêu cầu trang **file1.asp** thì kết quả in ra màn hình là “Bill Gate” nhưng khi người dùng yêu cầu tiếp trang **file2.asp** thì kết quả không hiển thị Bill Gate như mong muốn bởi vì **Session**(“Ten”) đã bị kết thúc ở **file1.asp** do gọi phương thức **Abandon**.

❖ **Contents.Remove**

Phương thức này dùng để xóa một phần tử ra khỏi tập Contents của đối tượng **Session**.

Cú pháp:

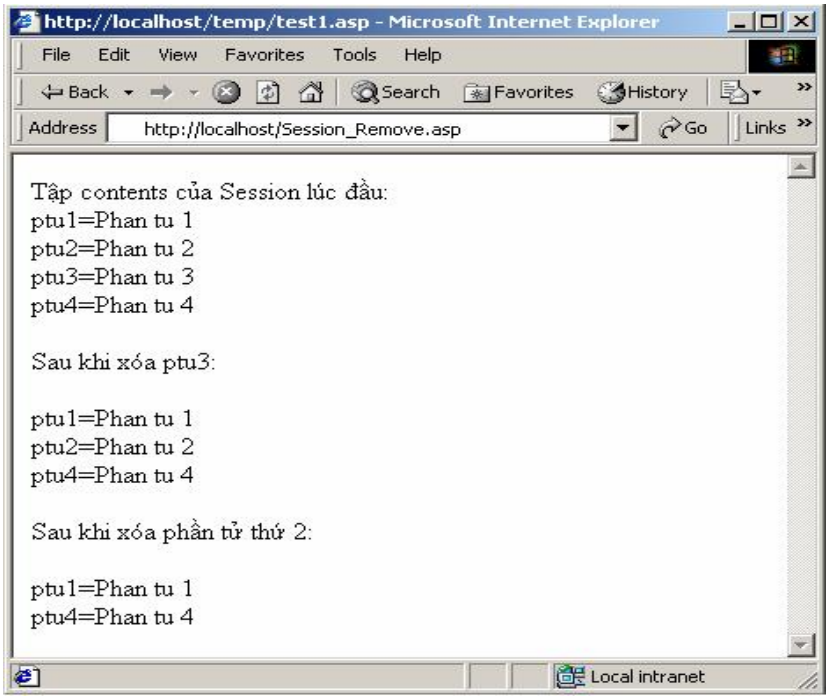
```
Session.Contents.Remove (name | index)
```

Khi gọi phương thức này ta có thể truyền vào tên của phần tử cần xóa hoặc vị trí của phần tử trong tập **Contents**.

Ví dụ:

```
<HTML>
<BODY>
<%
  Session("ptu1") = ("Phan tu 1")
  Session("ptu2") = ("Phan tu 2")
  Session("ptu3") = ("Phan tu 3")
  Session("ptu4") = ("Phan tu 4")
  Response.Write ("Tập contents của Session lúc đầu: <br>")
  Dim x
  For each x in Session.Contents
    Response.Write (x & "=" & Session.Contents(x) & "<BR>")

  Session.Contents.Remove("ptu3")
  Response.Write ("<p> Sau khi xóa ptu3: </p>")
  For each x in Session.Contents
    Response.Write (x & "=" & Session.Contents(x) & "<br>")

  Session.Contents.Remove(2)
  Response.Write ("<p> Sau khi xóa phần tử thứ 2: </p>")
  For each x in Session.Contents
    Response.Write (x & "=" & Session.Contents(x) & "<BR>")
%>
</BODY>
</HTML>
```

Kết quả khi thực hiện trang này như sau:



❖ **Contents.RemoveAll()**

Thay vì chỉ xóa một phần tử ta dùng phương thức **Remove** thì phương thức này xóa tất cả các phần tử ra khỏi tập **Contents**

Cú pháp:

```
Session.Contents.RemoveAll()
```

2.4.3.4. Các sự kiện của đối tượng Session

❖ **Session_OnStart**

Sự kiện này xuất hiện khi trình chủ tạo một **session** mới. Cài đặt của sự kiện này được đặt trong tập tin ***global.asa***.

❖ **Session_OnEnd**

Sự kiện này xuất hiện khi **session** kết thúc. Cài đặt của sự kiện này cũng được đặt trong tập tin **global.asa**.

Chú ý: Trong cài đặt của sự kiện **Session_OnEnd** ta không sử dụng được phương thức **MapPath** bởi vì ở đây phương thức này không còn hiệu lực.

2.4.4. Đối tượng Application

Một ứng dụng bao gồm một tập hợp các file kết hợp với nhau để xử lý hoặc phục vụ cho một mục đích nào đó. ASP cung cấp một đối tượng dùng để kết hợp các file đó lại với nhau, đó là đối tượng **Application**.

Đối tượng **Application** được dùng để lưu trữ các biến, qua đó các trang có thể truy cập đến các biến này. Không giống như đối tượng **Session** chỉ dùng cho một nối kết cho mỗi người dùng, đối tượng **Application** được dùng chung cho tất cả các người dùng. Do đó đối tượng **Application** nên chứa các thông tin mà có thể được truy cập

bởi nhiều trang trong ứng dụng (như thông tin nối kết cơ sở dữ liệu, thông tin về số người dùng truy cập, ...) nghĩa là bạn có thể truy cập các thông tin này từ bất cứ trang nào trong ứng dụng, nhưng chú ý là khi thay đổi các thông tin này sẽ ảnh hưởng đến tất cả các trang khác trong ứng dụng.

2.4.4.1. Tập hợp của đối tượng Application

❖ **Contents**

Tập hợp **Contents** chứa tất cả các phần tử đã được gắn thêm vào đối tượng **Application** trong quá trình thực thi script.

Cú pháp:

```
Application.Contents (Key)
```

Trong đó *key* là tên của phần tử cần lấy.

Ví dụ sau đây liệt kê tất cả các Application đã được dùng trong ứng dụng.

```
<HTML>
<BODY>
<CENTER> Các biến Application trong tập Contents
</CENTER>
<%
   Dim x
   For each x in Application.Contents
     Response.Write (x & "=" &
          Application .Contents (x) & "<br>")
   Next
%>
</BODY>
</HTML>
```

❖ **StaticObjects**

Tập hợp **StaticObjects** chứa tất cả các đối tượng được gắn vào ứng dụng với tag HTML <**object**>

Cú pháp:

```
Application.StaticObjects(Key)
```

Ví dụ: Đoạn code sau đây liệt kê tất cả các **object**.

```
<%
  Dim x
  For each x in Application.StaticObjects
   Response.Write( x & "<br>")
%>
```

2.4.4.2. Các phương thức của đối tượng Application

❖ **Contents.Remove**

Phương thức này dùng để xóa một phần tử ra khỏi tập Contents của đối tượng **Application**.

Cú pháp:

```
Application.Contents.Remove (name | index)
```

Khi gọi phương thức này ta có thể truyền vào tên của phần tử cần xóa hoặc vị trí của phần tử trong tập Contents.

Ví dụ:

```
<%
 Application("ptu1") = ("Phan tu 1")
 Application("ptu2") = ("Phan tu 2")
 Application("ptu3") = ("Phan tu 3")
 Application.Contents.Remove("ptu3")
 %>
```

❖ **Contents.RemoveAll**

Thay vì chỉ xóa một phần tử ta dùng phương thức **Remove** thì phương thức này xóa tất cả các phần tử ra khỏi tập **Contents**

Cú pháp:

```
Application.Contents.RemoveAll()
```

❖ **Lock và Unlock**

Bởi vì tất cả các người dùng đều có thể truy cập đến các biến **Application** nên có thể cùng lúc 2 hay nhiều người dùng cùng thay đổi giá trị của biến và điều này dẫn đến sai lệch giá trị của biến. Để khắc phục điều này đối tượng **Application** cung cấp hai phương thức **Lock** và **Unlock**. Phương thức **Lock** ngăn cản người dùng khác thay đổi biến trong đối tượng **Application** (dùng để đảm bảo rằng tại một thời điểm chỉ có một người dùng thay đổi các biến trong đối tượng Application). Phương thức **Unlock** cho phép người dùng thay đổi giá trị các biến trong đối tượng **Application**.

Cú pháp:

```
Application.Lock
Application.Unlock
```

Lưu ý: Khi gọi phương thức **Lock** thì ta phải nhớ gọi phương thức **Unlock** ngay khi thực hiện xong.

Ví dụ:

```
<%
  Application.Lock
  Application("visits") = Application("visits") +1
  Application.Unlock
%>
Trang này được truy cập:
<% = Application("visits") %> lần!
```

2.4.4.3. Các sự kiện (Events) của đối tượng Application

❖ **Application_OnStart**

Sự kiện này xuất hiện trước khi một phiên nối kết mới đầu tiên được hình thành. Sự kiện này được đặt trong file **global.asa**

❖ **Application_OnEnd**

Sự kiện này xuất hiện khi ứng dụng kết thúc (khi web server dừng). Sự kiện này được đặt trong file **global.asa**

2.4.5. Đối tượng Server

Đối tượng **Server** cung cấp nhiều thuộc tính và phương thức dùng để truy cập server. Đây là đối tượng dùng để quản lý những đặc trưng của trình chủ **IIS** và các hành động liên quan tới dịch vụ HTTP. Ngoài ra đối tượng **Server** còn cung cấp khả năng tạo kế thừa các thành phần COM trên Server.

2.4.5.1. Các thuộc tính của đối tượng Server

Đối tượng **Server** chỉ có duy nhất một thuộc tính đó là **ScriptTimeOut.** Thuộc tính này quy định thời gian lớn nhất mà các lệnh kịch bản còn được thực hiện. Giá trị mặc định là 90 giây.

Lưu ý là giá trị *timeout* sẽ không hiệu lực khi server thực hiện các lệnh kịch bản.

Cú pháp:

```
Server.ScriptTimeOut = [number]
```

2.4.5.2. Các phương thức của đối tượng Server

❖ **CreateObject**

Phương thức **CreateObject** dùng để tạo một thực thể của một đối tượng. Các đối tượng do phương thức này tạo ra chỉ có hiệu lực trong phạm vi một trang, do đó chúng sẽ bị hủy khi server xử lý xong trang ASP hiện hành.

Để tạo một đối tượng mà phạm vi của nó như **Session** hay **Application**, bạn có thể dùng tag <**object**> trong file **Global.asa** hoặc lưu trữ đối tượng trong biến **Session** hay **Application**.

Cú pháp:

```
Server.CreateObject (progID)
```

Trong đó *progID* là kiểu của đối tượng cần tạo.

Ví dụ:

```
<%
Dim adrot
Set adrot = Server.CreateObject("MSWC.AdRotator")
 ……
 ……
Set adrot = nothing
%>
```

❖ **Execute**

Thuộc tính Execute thực thi một trang ASP bên trong một trang khác. Sau khi thực thi xong file ASP được gọi thì quyền điều khiển được trả về cho file ASP ban đầu (file gọi).

Cú pháp:

```
Server.Execute (path)
```

Với *path* là đường dẫn tới tập tin ASP cần thực thi.

Ví dụ: Tạo 2 tập tin **file1.asp** và **file2.asp** và đặt trong cùng thư mục với nội dung sau:

File1.asp

```
<HTML>
<BODY>
<%
  Response.Write "Đang ở file 1"
  Server.Execute ("File2.asp")
  Response.Write " Trở về file 1"
%>
</BODY></HTML>
```

File2.asp

```
<HTML>
<BODY>
<%
  Response.Write "Đang ở file 2"
%>
</BODY>
      </HTML>
```

❖ **GetLastError**

Phương thức này trả về một đối tượng **ASPError** mô tả lỗi xuất hiện. Mặc định trang web dùng tập tin **\iishelp\common\500-100.asp** để xử lý các lỗi trong ASP. Nếu cần thì bạn có thể tạo hoặc thay đổi tập tin để đưa ra những câu thông báo thân thiện hơn,..

Chú ý: Phương thức này được dùng trước khi tập tin ASP gởi bất cứ nội dung gì xuống bowser.

Cú pháp:

```
Server.GetLastError()
```

Ví dụ: Trong ví dụ sau đây sẽ xuất hiện một lỗi chia cho 0

```
<HTML>
<BODY>
<%
 Dim i, tong, j
i = 0
j = 0
tong = 0
for i = 1 to 10 do
   tong = tong + i
next
tong = tong/j
%>
</BODY>
</HTML>
```

❖ **HTMLEncode**

Phương thức này dùng để mã hóa dạng HTML một chuỗi

Cú pháp:

```
Server.HTMLEncode(string)
```

Ví dụ: Đoạn chương trình sau đây cho phép người dùng nhập vào **username** và **password**, sau đó nhấn nút login. Nếu người dùng **login** sai thì sẽ bắt người dùng nhập lại **password**.

```
<HTML>
<BODY>
<% Dim uname,upass
  uname=Request.Form("uname")
  upass=Request.Form("upass")
  if (uname="test") and (upass="test") then
     Response.Redirect("main.asp")
   elseif (uname<>"") or (upass<>"") then
       Response.Write("Account nay khong hop le <BR>")
   end  if
%>
<form name="login" method="POST" action="login.asp">
<TABLE border = "2">
  <tr>
    <TD> Dang nhap </TD></TR>
 <TR>
  <TD>
    <TABLE border = "1">
      <TR>
        <TD> Username: </td>
        <TD><input type="text" name= "uname"
value="<%=server.HTMLEncode(uname)%>">
        </TD></TR>
      <TR>
       <TD>Password:</TD>
      <TD><input type="password" name= "upass"></td></tr>
   </TABLE>
  </TD>
 </TR>
 <TR>
    <TD><input type="submit" name= "submit" value="Login "></TD>
  </TR>
</TABLE>
```

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

❖ **MapPath**

Phương thức này ánh xạ một đường dẫn nào đó sang một đường dẫn vật lý. Phương thức này không được dùng trong sự kiện **Session_OnEnd** và **Application_OnEnd**.

Cú pháp:

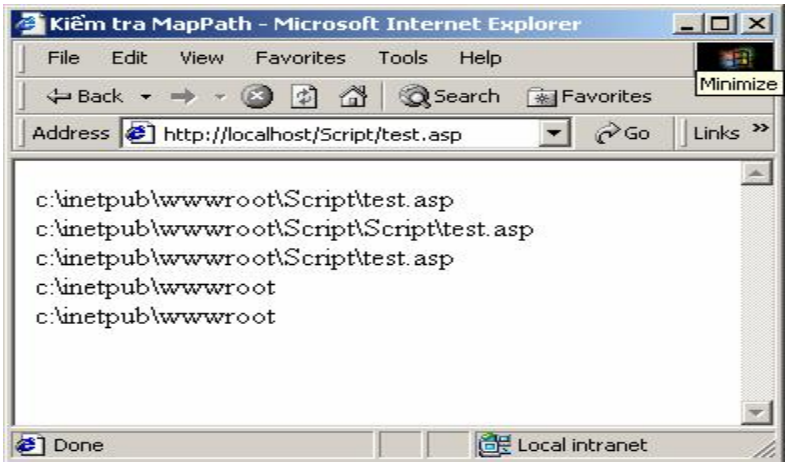
```
Server.MapPath (path)
```

Chú ý: Nếu *path* bắt đầu bằng ký tự / hoặc \ thì các ký tự này đại diện cho đường dẫn vật lý của thư mục ảo của tập tin ASP hiện tại.

Ví dụ: Giả sử bạn có tập tin test.asp đặt trong thư mục **C:\Inetpub\wwwroot\Script** với nột dung sau:

```
<HTML>
<HEAD>
<TITLE> Kiểm tra MapPath</TITLE>
</HEAD>
<BODY>
<%
Response.Write(Server.MapPath("test.asp") & "<br>")
Response.Write(Server.MapPath("Script/ test.asp") & "<BR>")
Response.Write(Server.MapPath("/Script/ test.asp") & "<br>")
Response.Write(Server.MapPath("/") & "<br>")
Response.Write(Server.MapPath("\") & "<br>")
%>
</BODY>
</HTML>
```

Khi duyệt trang test.asp này ta được kết quả như sau:



❖ **Transfer**

Phương thức này gởi (chuyển) tất cả các thông tin về trạng thái (các biến **Session**, các biến **Application**, các dữ liệu trong tập **Request**...) của tập tin ASP hiện tại cho một tập tin ASP thứ hai. Khi trang thứ hai thực hiện xong thì quyền điều khiển không trả về cho trang trước đó (xem thêm phương thức **Execute**).

Phương thức **Transfer** là một dạng khác của phương thức **Response.Redirect** nhưng lại hiệu quả hơn bởi vì phương thức **Response.Redirect** buộc Server phải giữ lại một **Request** giả trong khi phương thức **Server.Transfer** thì chuyển quyền điều khiển cho một trang ASP khác trên server. (xem thêm phương thức **Response.Redirect**)

Cú pháp:

```
Server.Transfer (path)
```

Ví dụ: Tạo 2 tập tin **file1.asp** và **file2.asp** và đặt trong cùng thư mục với nội dung 2 file như sau:

File1.asp

```
<HTML>
<BODY>
<%
  Response.Write "Dòng 1 trên file1.asp"
  Server.Transfer("File2.asp")
  Response.Write " Dòng 2 trên file1.asp"
%>
</BODY>
</HTML>
```

File2.asp

```
<HTML>
<BODY>
<%
  Response.Write "Dòng 1 trên file2.asp"
 Response.Write "Dòng 2 trên file2.asp"
```

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

Mở trình duyệt lên và thực thi **file1.asp**. So sánh kết quả này với kết quả ở ví dụ của phương thức Server.Execute.

❖ **URLEncode**

Phương thức này dùng để mã hóa một chuỗi URL.

Cú pháp:

```
Server.URLEncode (stringURL)
```

2.4.6. Đối tượng ASP Error

Đối tượng **ASPError** được dùng để hiển thị thông tin chi tiết của bất cứ lỗi nào xuất hiện trong các kịch bản của trang ASP. Đối tượng **ASPError** được tạo ra khi phương thức **Server.GetLastError** được gọi, vì thế thông tin về các lỗi chỉ có thể được truy cập bằng việc gọi phương thức **Server.GetLastError**.

Đối tượng **ASPError** được bổ sung vào ASP từ phiên bản 3.0 trở đi và chỉ có sẵn trong **IIS** 5.

Đối tượng **ASPError** không có phương thức nào mà chỉ có các thuộc tính để cung cấp các thông tin về lỗi xuất hiện. Dưới đây là các thuộc tính của đối tượng **ASPError**:

❖ **ASPCode**

Thuộc tính này cho biết mã lỗi được tạo ra bởi **IIS**

Cú pháp:

```
ASPError.ASPCode
```

❖ **ASPDescription**

Thuộc tính này trả về một chuỗi mô tả chi tiết lỗi xuất hiện.

Cú pháp:

```
ASPError.ASPDescription
```

❖ **Category**

Thuộc tính này cho biết nơi nào đưa ra lỗi (do IIS hay do ngôn ngữ kịch bản hay do một thành phần phụ thêm nào đó).

Cú pháp:

```
ASPError.Category
```

❖ **Column**

Thuộc tính này cho biết vị trí cột thứ mấy trong tập tin ASP đã gây ra lỗi.

Cú pháp:

```
ASPError.Column
```

❖ **Description**

Thuộc tính này mô tả ngắn gọn lỗi.

Cú pháp:

```
ASPError.Description
```

❖ **File**

Thuộc tính này trả về tên tập tin ASP đã gây ra lỗi.

```
ASPError.File
```

❖ **Line**

Thuộc tính này cho biết dòng thứ mấy trong tập tin ASP đã gây ra lỗi.

Cú pháp:

```
ASPError.Line
```

❖ **Number**

Thuộc tính này trả về mã lỗi COM chuẩn của lỗi tạo ra.

Cú pháp:

```
ASPError.Number
```

❖ **Source**

Thuộc tính này trả về đoạn mã của dòng gây ra lỗi.

Cú pháp:

```
ASPError.Source
```

Ví dụ:

```
<HTML>
<BODY>
<%
Dim objErr
Set objErr = Server.GetLastError()
Response.Write (“ASPCode = “ & objErr.ASPCode)
Response.Write (“<br>”)
Response.Write(“ASPDescription= “&objErr. ASPDescription)
Response.Write (“<BR>”)
Response.Write (“Category = “ & objErr.Category)
Response.Write (“<BR>”)
Response.Write (“Column = “ & objErr.Column)
Response.Write (“<BR>”)
```

```
Response.Write (“Description = “ & objErr. Description)
Response.Write (“<BR>”)
Response.Write (“File = “ & objErr.File)
Response.Write (“<BR>”)
Response.Write (“Line = “ & objErr.Line)
Response.Write (“<BR>”)
Response.Write (“Number = “ & objErr.Number)
Response.Write (“<BR>”)
Response.Write (“Source = “ & objErr.Source)
Response.Write (“<BR>”)
%>
</BODY>
</HTML>
```

2.4.7. Đối tượng DICTIONARY

Đối tượng **Dictionary** được dùng để lưu trữ thông tin theo cặp tên/giá trị. Đối tượng **dictionary** có thể xem tương tự như mảng, tuy nhiên đối tượng **Dictionary** được tạo ra để thao tác với dữ liệu một cách hiệu quả hơn.

❖ So sánh đối tượng **Dictionary** với các mảng ta thấy:

+ Đối tượng **Dictionary** dùng từ khoá (key) để nhận diện các phần tử (item) còn mảng thì sử dụng chỉ số.

+ Bạn không thể dùng **ReDim** để thay đổi kích thước của đối tượng **Dictionary** còn mảng thì được.

+ Khi xoá một phần tử khỏi đối tượng **Dictionary** thì các phần tử còn lại sẽ tự động thay thế, còn các mảng thì không.

+ Mảng có thể có nhiều chiều còn đối tượng **Dictionary** thì không.

+ Đối tượng **Dictionary** được xây dựng với nhiều chức năng hơn.

+ Đối tượng **Dictionary** truy cập thường xuyên các phần tử một cách ngẫu nhiên hiệu quả hơn mảng.

+ Đối tượng **Dictionary** định vị các phần tử dựa trên nội dung hiệu quả hơn.

2.4.7.1. Tạo đối tượng Dictionary

Đối tượng **Dictionary** được tạo ra bởi đối tượng Server bằng việc gọi phương thức **CreateObject** như sau:

```
<%
   Dim Dic
   Set Dic = Server.CreateObject("Scripting.Dictionary")
 %>
```

Bởi vì hàm **CreateObject** của đối tượng Server trả về một đối tượng nên để gán đối tượng cho biến **Dic** ta dùng lệnh **Set**.

Khi sử dụng xong thực thể của đối tượng **Dictionary** ta phải hủy bỏ thực thể đó bằng cách:

```
Set Dic = nothing
```

2.4.7.2. Các thuộc tính của đối tượng Dictionary

❖ **CompareMode**

Ta dùng thuộc tính **CompareMode** để thiết lập hoặc nhận về chế độ so sánh để so sánh các khoá trong đối tượng **Dictionary**.

Cú pháp:

```
Dictionary.CompareMode [=mode]
```

Trong đó *mode* có thể nhận một trong các giá trị sau:

0 = vbBinaryCompare – So sánh nhị phân

1 = vbTextCompare – So sánh dạng văn bản

2 = vbDatabaseCompare – So sánh cơ sở dữ liệu

Ví dụ:

```
<HTML>
<BODY>
<%
   Dim Dic
   Set Dic = Server.CreateObject("Scripting.Dictionary")
       Dic.CompareMode = 1
       Dic.Add "HN", "Hà Nội"
       Dic.Add "HCM", "Hồ Chí Minh"
       ' Phương thức Add sau đây sẽ sai bởi vì khoá "hn"
       ' đã có rồi!
       Dic.Add "hn", "Hà Nam"
 %>
</BODY>
</HTML>
```

❖ **Count**

Thuộc tính này trả về số cặp tên/giá trị (số phần tử) trong đối tượng **Dictionary**.

Cú pháp:

```
Dictionary.Count
```

Ví dụ:

```
<HTML>
<BODY>
<%
 Dim Dic
 Set Dic = Server.CreateObject("Scripting.Dictionary")
  Dic.CompareMode = 1
  Dic.Add "HN", "Hà Nội"
  Dic.Add "HCM", "Hồ Chí Minh"
Dic.Add "HP", "Hải Phòng"
Response.Write ("Số cặp tên/giá trị là: " & Dic.Count)
Set Dic = nothing
 %>
</BODY>
</HTML>
```

❖ **Item**

Dùng thuộc tính này để gán hoặc lấy về giá trị của một phần tử trong đối tượng **Dictionary**.

Cú pháp:

```
Dictionary.Item (key)[ = newitem]
```

Ví dụ:

```
<HTML>
<BODY>
<%
 Dim Dic
 Set Dic = Server.CreateObject("Scripting.Dictionary")
  Dic.Add "HN", "Hà Nội"
  Dic.Add "HCM", "Hồ Chí Minh"
Dic.Add "HP", "Hải Phòng"
Response.Write ("Giá trị của khoá HN là: " & Dic.Item("HN"))
Set Dic = nothing
```

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

❖ **Key**

Để thay đổi tên của một khoá đã có trong đối tượng **Dictionary** ta dùng thuộc tính key theo cú pháp sau

```
Dictionary.Key (key)[ = newkey]
```

Ví dụ:

```
<HTML>
<BODY>
<%
 Dim Dic
 Set Dic = Server.CreateObject("Scripting.Dictionary")
  Dic.Add "HN", "Hà Nội"
  Dic.Add "HCM", "Hồ Chí Minh"
Dic.Add "HP", "Hải Phòng"
Dic.Key("HN") = "HNOI"
Response.Write ("Giá trị của khoá HNOI là: " & Dic.Item("HNOI"))
Set Dic = nothing
 %>
</BODY></HTML>
```

7.3. Các phương thức của đối tượng Dictionary

❖ **Add**

Phương thức **Add** dùng để thêm một cặp khoá/giá trị mới vào đối tượng **Dictionary**. Nếu khoá này đã có trong **Dictionary** thì phương thức này sẽ bị sai.

Cú pháp:

```
Dictionary.Add (key, value)
```

Ví dụ:

```
<HTML>
<BODY>
<%
 Dim Dic
 Set Dic = Server.CreateObject(“Scripting.Dictionary”)
  Dic.Add “Đ”, “Đỏ”
  Dic.Add “X”, “Xanh”
Dic.Add “V”, “Vàng”
Dic.Key(“T”) = “Tím”
Response.Write (“Giá trị của khoá T là: “ & Dic.Item(“T”))
Set Dic = nothing
 %>
</BODY>
</HTML>
```

❖ **Exists**

Để kiểm tra một khoá đã có trong đối tượng **Dictionary** hay chưa ta dùng phương thức **Exists**. Phương thức này trả về true nếu khoá đã có trong **Dictionary** và trả về **false** nếu khoá này tồn tại.

Cú pháp:

```
Dictionary.Exists (key, value)
```

Ví dụ:

```
<HTML>
<BODY>
<%
 Dim Dic
 Set Dic = Server.CreateObject(“Scripting.Dictionary”)
  Dic.Add “Đ”, “Đỏ”
  Dic.Add “X”, “Xanh”
Dic.Add “V”, “Vàng”
Dic.Key(“T”) = “Tím”
If  Dic.Exists (“V”) = true Then
   Response.Write “Khoá V tồn tại!”
Else
   Response.Write “Khoá V không tồn tại!”
End If
Set Dic = nothing
 %>
</BODY></HTML>
```

❖ **Items**

Không phải lúc nào ta cũng thao tác trên các khoá của đối tượng **Dictionary** mà đôi lúc ta cũng phải thao tác trên dữ liệu của các khoá như: tìm kiếm một giá trị nào đó, sửa đổi giá trị,... Nếu ta duyệt lần lượt trên các khoá và lấy giá trị của chúng để so sánh thì ắt hẳn sẽ mất nhiều thời gian. Đối tượng **Dictionary** cung cấp cho ta phương thức **Items** để lấy một mảng các giá trị của các khoá, và nhờ vào mảng này ta sẽ thao tác trên dữ liệu dễ dàng hơn. Cú pháp của phương thức **Items** như sau:

```
Dictionary.Items
```

Ví dụ:

```
<HTML>
<BODY>
<%
 Dim Dic
 Set Dic = Server.CreateObject("Scripting.Dictionary")
  Dic.Add "Đ", "Đỏ"
  Dic.Add "X", "Xanh"
Dic.Add "V", "Vàng"
Dic.Key("T") = "Tím"
Response.Write("<p>" & "Các giá trị: " & "</p>")
Dim Arr, i
Arr = Dic.Items
For i = 0  to Dic.Count-1
   Response.Write (Arr(i) & "<br>")
Next
Set Dic = nothing
 %>
</BODY>
</HTML>
```

❖ **Keys**

Thay vì trả về một mảng các giá trị như phương thức **Items** thì phương thức **Keys** lại trả về một mảng các khoá trong đối tượng **Dictionary**.

Cú pháp:

```
Dictionary.Keys
```

Ví dụ:

```
<HTML>
<BODY>
<%
 Dim Dic
 Set Dic = Server.CreateObject(“Scripting.Dictionary”)
  Dic.Add “Đ”, “Đỏ”
  Dic.Add “X”, “Xanh”
Dic.Add “V”, “Vàng”
Dic.Key(“T”) = “Tím”
Response.Write(“<p>” & “Các khoá:  ” & “</p>”)
Dim Arr, i
Arr = Dic.Keys
For i = 0  to Dic.Count-1
   Response.Write (Arr(i) & “<br>”)
Next
Set Dic = nothing
 %>
</BODY>
</HTML>
```

❖ **Remove**

Phương thức này xoá một phần tử (một cặp khoá/giá trị) ra khỏi đối tượng **Dictionary**. Cú pháp của phương thức này như sau:

```
Dictionary.Remove (key)
```

Ví dụ:

```
<HTML>
<BODY>
<%
 Dim Dic
 Set Dic = Server.CreateObject(“Scripting.Dictionary”)
  Dic.Add “Đ”, “Đỏ”
  Dic.Add “X”, “Xanh”
Dic.Add “V”, “Vàng”
Dic.Key(“T”) = “Tím”
Dic.Remove(“Đ”)
Set Dic = nothing
 %>
</BODY></HTML>
```

❖ **RemoveAll**

Phương thức này dùng để xoá tất cả các phần tử của đối tượng **Dictionary**.

Cú pháp:

```
Dictionary.RemoveAll
```

2.4.8. Đối tượng FILESYSTEMOBJECT

Đối tượng **FileSystemObject** cung cấp thông tin về hệ thống tập tin, thư mục trên trình chủ và ta có thể sử dung đối tượng này để thao tác với các tập tin, thư mục,...

2.4.8.1. Tạo đối tượng filesystemobject

Bởi vì đối tượng **FileSystemObject** thao tác trên hệ thống tập tin của trình chủ (server) nên đối tượng này đối tượng này được tạo ra bởi Server theo cú pháp sau:

```
<%
Dim fso
Set fso = Server.CreateObject(“Scripting.FileSystemObject”)
%>
```

Lưu ý: Khi dùng xong thực thể của đối tượng **FileSystemObject** ta phải hủy thực thể đó đi bằng cách:

Set fso = nothing

2.4.8.2. Các thuộc tính của đối tượng filesystemobject

Đối tượng **FileSystemObject** chỉ có một thuộc tính duy nhất đó là:

Drives: Thuộc tính này cho biết một tập tất cả các ổ dĩa trên máy tính.

Cú pháp:

```
[drivecoll = ] FileSystemObject.Drives
```

2.4.8.3. Các phương thức của đối tượng FileSystemObject

❖ **BuildPath**

Phương thức này gắn một chuỗi vào một đường dẫn đã có để tạo ra một đường dẫn mới.

Cú pháp:

```
[newpath = ]FileSystemObject.BuildPath(path, name)
```

Trong đó:

+ *path*: là đường dẫn đã tồn tại

+ n*ame*: là tên cần gắn thêm vào Path

+ n*ewpath*: là đường dẫn mới sau khi đã gắn tên vào

Ví dụ:

```
<html>
<body>
<%
Dim fso, NewPath
Set fso = Server.CreateObject("Scripting.FileSystemObject")
NewPath = fso.BuildPath("C:\My Documents", "BT")
Response.Write (NewPath)
Set fso = nothing
 %>
</body>
</html>
```

Trong ví dụ trên, sau khi gọi phương thức **BuildPath** thì biến **NewPath** sẽ có giá trị là "**C:\My Documents\BT**"

❖ **CopyFile**

Phương thức này sao chép một hoặc một số tập tin từ thư mục này tới thư mục khác.

Cú pháp:

```
FileSystemObject.Copy src, des [,ovr]
```

Trong đó:

src: là đường dẫn tới tập tin cần sao chép, tên tập tin cần chép có thể chứa các ký tự thay thế như (*, ?).

des: Là đường dẫn của thư mục cần chép tới, đường dẫn này không được chứa ký tự thay thế (*, ?).

ovr: nhận giá trị true hoặc false. Nếu *ovr* là true có nghĩa là cho phép chép đè lên các tập tin đã có trong *des*. Nếu false thì không cho chép đè. Giá trị mặc định của *ovr* là true.

Ví dụ: Đoạn chương trình sau đây sao chép tất cả các tập tin có đuôi .asp trong thư mục **C:\Web** sang thư mục **D:\ASP**

```
<HTML>
<BODY>
<%
Dim fso
Set fso = Server.CreateObject("Scripting.FileSystemObject")
fso.Copy "C:\Web\*.asp", "D:\ASP"
Set fso = nothing
 %>
```

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

❖ **CopyFolder**

Phương thức này sao chép một hoặc nhiều thư mục.

Cú pháp:

```
FileSystemObject.CopyFolder src, des [,ovr]
```

Ví dụ: Sao chép tất cả các thư mục con của thư mục **C:\Web** vào thư mục **D:\ASP**

```
<HTML>
<BODY>
<%
Dim fso
Set fso = Server.CreateObject("Scripting.FileSystemObject")
fso.CopyFolder "C:\Web\*", "D:\ASP"
Set fso = nothing
 %>
</BODY>
</HTML>
```

❖ **CreateFolder**

Phương thức này tạo một thư mục mới.

Cú pháp:

```
FileSystemObject.CreateFolder (Foldername)
```

Ví dụ: Tạo thư mục **C:\ASP**

```
<HTML>
<BODY>
<%
Dim fso
Set fso = Server.CreateObject("Scripting.FileSystemObject")
Fso.CreateFolder "C:\ASP"
Set fso = nothing
 %>
</BODY>
</HTML>
```

❖ **CreateTextFile**

Phương thức này tạo một tập tin văn bản trong thư mục hiện hành và trả về một đối tượng **TextStream** dùng để đọc hoặc ghi dữ liệu lên file.

Cú pháp:

FileSystemObject.CreateTextFile(*filename*[,*Ovr*[,*Uni*]])

❖ **DeleteFile**

Phương thức này xoá một hoặc nhiều tập tin. Nếu tập tin không tồn tại thì sẽ xuất hiện lỗi.

Cú pháp:

FileSystemObject.DeleteFile(*filename*[,*bReadOnly*])

Trong đó *bReadOnly* nhận một trong hai giá trị. Nếu nhận giá trị true thì các tập tin mang thuộc tính chỉ đọc (read-only) cũng sẽ bị xoá. Nếu nhận giá trị false thì các tập tin mang thuộc tính read-only sẽ không bị xoá.

❖ **DeleteFolder**

Phương thức này xoá một hoặc nhiều thư mục. Nếu thư mục không tồn tại thì phương thức này sẽ gây ra lỗi.

Cú pháp:

FileSystemObject.DeleteFolder(f*oldername*[,*bReadOnly*])

❖ **DriveExists**

Phương thức **DriveExists** kiểm tra trên hệ thống tập tin của server có tồn tại một ổ dĩa nào đó hay không? Nếu có thì phương thức này trả về **true**, còn nếu không thì sẽ trả về **false**.

Cú pháp:

FileSystemObject.DriveExists(*drive*)

Trong đó *drive* là tên của ổ dĩa cần kiểm tra

Ví dụ:

```
<HTML>
<BODY>
<%
Dim fso
Set fso = Server.CreateObject("Scripting.FileSystemObject")
if fso.DriveExists("C:") = true then
  Response.Write ("Ổ dĩa C tồn tại!")
else
  Response.Write ("Ổ dĩa C không tồn tại!")
end if
Set fso = nothing
 %>
</BODY>
</HTML>
```

❖ **GetAbsolutePathName**

Phương thức này trả về đường dẫn dạng đầy đủ của một đường dẫn tương đối.

Cú pháp:

```
FileSystemObject.GetAbsolutePathName(path)
```

Ví dụ: Giả sử đường dẫn hiện hành là **C:\ASP**. Đoạn chương trình sau đây sẽ in ra màn hình browser dòng **C:\ASP\Data\list.txt**

```
<HTML>
<BODY>
<%
Dim fso, path
Set fso = Server.CreateObject("Scripting.FileSystemObject")
path = fso.GetAbsolutePathName("Data\list.txt")
Response.Write (path)
Set fso = nothing
 %>
</BODY>
</HTML>
```

❖ **GetBaseName**

Phương thức này trả về phần tên của một tập tin hoặc tên của thư mục ở cuối một đường dẫn.

Cú pháp:

```
FileSystemObject.GetBaseName(path)
```

Ví dụ: Nếu *path* = "C:\ASP\Data\list.txt" thì hàm này sẽ trả về "list"

❖ **GetDrive**

Phương thức này trả về một đối tượng **Drive** mô tả một ổ đĩa. Có được đối tượng **Drive** ta có thể thao tác trên ổ đĩa mà **Drive** mô tả bằng cách sử dụng các phương thức của đối tương **Drive**.

Cú pháp:

```
FileSystemObject.GetDrive(Drive)
```

Ví dụ: Đoạn chương trình sau đây trả về đối tượng **Drive** mô tả ổ đĩa C.

```
<%
Dim fso, drvC
Set fso = Server.CreateObject("Scripting.FileSystemObject")
set drvC = fso.GetDrive("C:\")
Set fso = nothing
 %>
```

❖ **GetDriveName**

Phương thức này trả về một chuỗi là tên của ổ đĩa trong một đường dẫn.

Cú pháp:

```
FileSystemObject.GetDriveName(path)
```

Ví dụ: Nếu *path* = "C:\ASP\Data\list.txt" khi gọi hàm GetDriveName(*path*) ta sẽ nhận được chuỗi "C:"

❖ **GetExtensionName**

Phương thức này trả về phần mở rộng của một tập tin (không bao gồm dấu chấm phân cách giữa phần tên và phần mở rộng).

Cú pháp:

```
FileSystemObject.GetExtensionName(path)
```

Ví dụ: Nếu *path* = "C:\ASP\Data\list.txt" thì khi gọi hàm GetExtensionName(*path*) ta sẽ nhận được chuỗi "txt"

❖ **GetFile**

Phương thức **GetFile** trả về đối tượng File mô tả một tập tin đã được chỉ định trong đường dẫn truyền vào.

Cú pháp:

```
FileSystemObject.GetFile(path)
```

❖ **GetFileName**

Phương thức này chỉ trả về phần tên của một tập tin hay một thư mục.

Cú pháp:

FileSystemObject.GetFileName(*path*)

Ví dụ: Nếu *path* = "C:\ASP\Data" thì khi gọi hàm GetFileName(*path*) ta sẽ nhận được chuỗi "Data". Nếu *path* = "C:\ASP\Data\list.txt" thì khi gọi hàm GetFileName(*path*) ta sẽ nhận được chuỗi "list".

❖ **GetFolder**

Phương thức **GetFolder** trả về đối tượng **Folder** của một thư mục.

Cú pháp:

FileSystemObject.GetFolder(*path*)

❖ **GetParentFolderName**

Phương thức này trả về thư mục cha của một thư mục.

Cú pháp:

FileSystemObject.GetParentFolderName(*path*)

Ví dụ: Nếu *path* = "**C:\ASP\Data**" thì khi gọi hàm **GetParentFolderName**(*path*) ta sẽ nhận được chuỗi ASP". Đây là thư mục cha của thư mục Data

❖ **GetSpecialFolder**

Phương thức này trả về đường dẫn tới một số thư mục đặc biệt của hệ điều hành.

Cú pháp:

FileSystemObject.GetSpecialFolder(*foldername*)

Trong đó *foldername* nhận một trong các giá trị sau:

+ **WindowsFolde**r hay 0: Nếu muốn nhận về thư mục của hệ điều hành (mặc định Windows 98 đó là thư mục Windows, đối với Windows 2000 thì đó là thư mục Winnt).

+ **SystemFolder** hay 1: Nếu muốn nhận về đường dẫn tới thư mục System của hệ điều hành.

+ **TemporaryFolder** hay 2: Nếu muốn nhận về đường dẫn tới thư mục tạm thời (TEMP) của hệ điều hành.

Ví dụ: Đoạn chương trình sau đây lấy thư mục hệ thống của hệ điều hành. Nếu dùng Windows 2000 thì trên màn hình browser sẽ xuất hiện dòng "**C:\WINNT\System32**"

```
<HTML>
<BODY>
<%
Dim fso, path
Set fso = Server.CreateObject("Scripting.FileSystemObject")
path = fso.GetSpecialFolder(1)
Response.Write (path)
Set fso = nothing
 %>
</BODY>
</HTML>
```

❖ **GetTempName**

Phương thức này trả về một tên tập tin hoặc thư mục tạm thời được phát sinh ngẫu nhiên.

Cú pháp:

```
FileSystemObject.GetTempName
```

Ví dụ:

```
<HTML>
<BODY>
<%
Dim fso, tfolder, tname, tfile
Set fso = Server.CreateObject("Scripting.FileSystemObject")
tfolder = fso.GetSpecialFolder(2)
tname = fso.GetTempName
Set tfile = tfolder.CreateTextFile(tname)
Response.Write (tfile)
Set fso = nothing
 %>
</BODY>
</HTML>
```

❖ **MoveFile**

Phương thức này di chuyển một hoặc nhiều tập tin từ nơi này sang nơi khác.

Cú pháp:

FileSystemObject.MoveFile (*src, des*)

Trong đó *src* là nơi chứa các tập tin cần di chuyển di, *des* là nơi mà các tập tin cần chép đến.

❖ **MoveFolder**

Phương thức này di chuyển một hoặc nhiều thư mục từ nơi này sang nơi khác.

Cú pháp:

FileSystemObject.MoveFile (*src, des*)

Trong đó *src* là nơi chứa các tập tin cần di chuyển di, *des* là nơi mà các tập tin cần chép đến.

❖ **OpenTextFile**

Phương thức này mở một tập tin và trả về một đối tượng TextStream được dùng để truy cập đối tượng này.

Cú pháp:

FileSystemObject.OpenTextFile(*fname, mode, creat, format*)

Trong đó:

+ *fname*: là tên của tập tin cần mở

+ *mode*: dùng để chỉ cách thức mở.

+ *create*: dùng để chỉ định rằng nếu tập tin không tồn tại thì có tạo tập tin mới hay không.

+ *format*: dùng để chỉ ra rằng mở tập tin dùng chuẩn ASCII hay Unicode.

2.4.8.4. Ví dụ minh họa

Sau đây là một ví dụ hiển thị một cửa sổ đăng nhập (login) cho phép người dùng gõ vào tên truy cập (username) và mật khẩu (password). Chương trình sẽ kiểm tra cặp username và password này có tồn tại trong tập tin **password.txt** trong thư mục hiện tại của ứng dụng không? Nếu có thì chương trình sẽ đưa người dùng đến trang main.asp. Nếu không thì chương trình sẽ thông báo và bắt người dùng đăng nhập lại.

Tập tin **common.asp** chứa các hàm mà người dùng định nghĩa trong đó có hàm CheckAccount dùng để kiểm tra xem username và password truyền vào có tồn tại trong tập tin password.txt không? Nếu có thì hàm này trả về **true**. Nếu không thì hàm này sẽ trả về **false**. Nội dung của **common.asp** như sau:

```
<%
Function CheckAccount(uname,upass)
Dim fso        ' Bien chua doi tuong FileSystem
Dim ftxt       ' Bien chua doi tuong File
Dim stLine
Dim path
```

```
Dim uname_pass

  uname=CStr(uname)
  upass=CStr(upass)
  uname_pass = uname & ":" & upass
  CheckAccount=false        ' Mac dinh ban dau
  path = Server.MapPath(".") & "\Password.txt"

Set fso =
Server.CreateObject("Scripting.FileSystemObject")
 Set ftxt = fso.OpenTextFile(path)

While (ftxt.AtEndOfStream <> true) and
 (CheckAccount=false)
                stLine = ftxt.Readline
                if (uname_pass = stLine) then
                        CheckAccount= true
          end if
          Wend
          ftxt.Close
          Set ftxt=nothing
          Set fso=nothing
End Function
%>
```

Tập tin **login.asp** là tập tin mô tả giao diện với người dùng. Nội dung của **login.asp** như sau:

```
<!--#include file = "common.asp" -->
<html>
<head>
<title> Login to …</title>
</head>
<body>
<b><font size="6">Login</font></b></p>
<%
Dim uname, upass
uname = Request.Form("Uname")
upass = Request.Form("Upass")
if (uname <> "") and (upass <> "") then
```

```
    if CheckAccount(uname, upass) = true then
        response.redirect ("main.asp")
    else
        response.write ("<font color = red> Please check username and password! </font>")
    end if
elseif uname <> "" then
    response.Write ("<font color = red> Please enter password! </font>")
 elseif upass <> "" then
        response.Write ("<font color = red> Please enter username! </font>")
end if
%>
<form method="POST" action = "login.asp">
<table border="1" cellpadding="0" cellspacing="0" width="27%">
  <tr>
    <td width="10%">Username </td>
    <td><input type = text size="20" name="Uname"
     value = '<% = Server.HTMLEncode(Request.Form("Uname"))%>'>
    </td>
  </tr>
  <tr>
    <td width="10%">Password </td>
    <td><input type="password" size="20" name="Upass"> </td>
  </tr>
  <tr>
    <td width="100%" colspan="2">
    <p align="center">
        <input type="submit" value="Login" name="B3"></td>
  </tr>
</table>
</form>
</body>
</html>
```

Tạo tập tin main.asp. Khi bạn viết một ứng dụng web thực sự thì trang main.asp chính là trang chính của ứng dụng. Giả sử tập tin main.asp với nội dung sau:

```
<html>
<head>
<title> Trang Web chính… </title>
</head>
<body>
<%
 Response.Write (“Đăng nhập thành công. Chào mừng bạn đến trang Web của chúng tôi!”)
%>
</body>
</html>
```

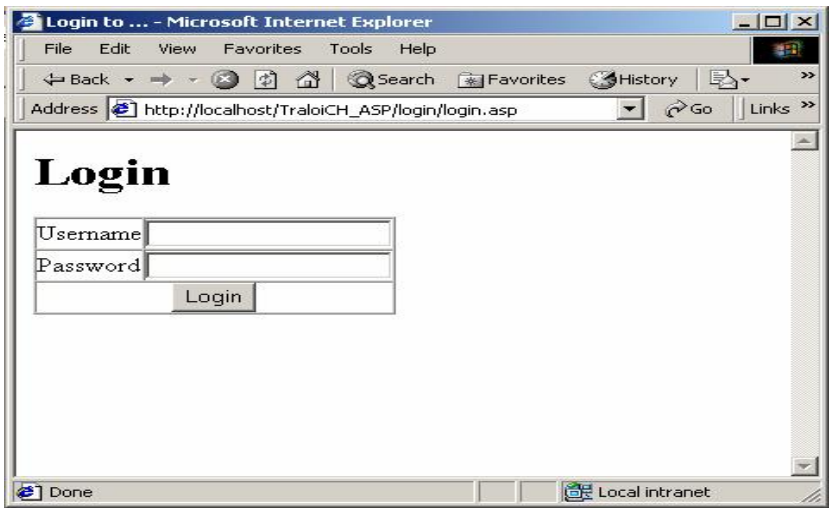
Tạo tập tin Password.txt và đặt cùng thư mục với ba tập tin trên với nội dung sau:

```
abc:abc
cobe:becon
nvlong:long1280
hung1254:meocon
```

Đây chính là danh sách các username và password mà ứng dụng cho phép đăng nhập vào. Nếu muốn thêm người dùng, bạn thêm vào tập tin này các dòng tương ứng.

Đặt 4 tập tin vừa tạo vào trong cùng một thư mục và tạo một thư mục ảo với tên MyWeb chỉ đến thư mục chứa 4 tập tin này.

Mở trình duyệt và gõ vào http://localhost/MyWeb/login.asp. Kết quả sẽ hiển thị lên màn hình như sau:



Khi người dùng gõ vào đúng username và password trong tập tin password.txt thì khi nhấn nút **login** người dùng sẽ được chuyển sang trang main.asp như sau:

Nhưng khi gõ sai username hay password thì một câu thông báo sẽ được hiện lên và bắt người dùng đăng nhập lại như sau:

2.4.9. Đối tượng ADROTATOR

Đối tượng **AdRotator** được dùng để hiển thị các ảnh khác nhau mỗi khi người dùng yêu cầu hoặc **refresh** một trang. Các thông tin về các ảnh hiển thị được đặt trong tập tin văn bản.

2.4.9.1. Cách tạo đối tượng AdRotator

Để tạo đối tượng **AdRotator** ta dùng cú pháp sau:

```
Set ad = Server.CreateObject("MSWC.AdRotator")
ad.GetAdvertisement("textfile.txt")
```

2.4.9.2 Định dạng tập tin văn bản

```
REDIRECT URL
WIDTH 480
HEIGHT 100
BORDER 0
*
CITD.gif
http://www.ioit.vast.ac.vn/
Đến với IOIT
80
Microsoft.gif
http://www.microsoft.com/
Đến với Microsoft
20
```

Các dòng ở phía dưới dấu * là các dòng chỉ hình ảnh, địa chỉ trang Web, dòng văn bản để hiển thị nếu không hiển thị được ảnh, và tỉ lệ phần trăm số người dùng truy cập để hiển thị các ảnh.

2.4.9.3. Các thuộc tính của đối tượng AdRotator

- **Border**: chỉ định kích thước của đường viền bao quanh phần quảng cáo.
- **Clickable**: Chỉ định phần quảng cáo có **hyperlink** không.
- **TargetFrame**: tên của frame để hiển thị ảnh quảng cáo.

Ví dụ:

```
<%
Dim adrot
Set adrot = Server.CreateObject("MSWC.Adrotator")
adrot.Border = "2"
adrot.Clickable = false
adrot.TargetFrame = "targer='_blank'"
Response.Write(adrot.Advertisement("ads.txt"))
Set fso = nothing
 %>
```

2.4.9.4. Các phương thức của đối tượng AdRotator

GetAdvertisement: Phương thức này trả về đoạn HTML mà hiển thị mục quảng cáo trên trang Web.

Cú pháp:

```
AdRotator.GetAdvertisement(path)
```

Trong đó *path* là đường dẫn tới tập tin văn bản mô tả các mục quảng cáo.

2.5. CHỈ THỊ #include

Khi muốn chèn nội dung của một tập tin ASP vào tập tin ASP khác trước khi server thực thi chúng ta dùng chỉ thị **#include**. Thông thường các nội dung đó chứa các hàm toàn cục, các biến toàn cục, các **header**, các **footer** hoặc những gì dùng chung cho nhiều trang.

Cú pháp:

```
<!--#include file = filename -->
hoặc
<!--#include virtual = filename -->
```

Trong đó *filename* là tên của tập tin mà nội dung của tập tin đó cần include vào.

Từ khoá **file** để chỉ rằng đường dẫn đến tên tập tin cần include là đường dẫn tương đối, đường dẫn này bắt đầu bằng thư mục chứa tập tin. Còn từ khoá **virtual** để chỉ ra rằng đường dẫn tới tập tin bắt đầu bằng thư mục ảo.

Ví dụ: Giả sử ta có tập tin time.inc có chứa hàm dùng để ghi ra màn hình của browser giờ hiện hành. Còn tập tin **distime.asp** là tập tin **include** tập tin **time.inc**. Hai tập tin này được đặt trong cùng thư mục và với nội dung sau:

time.inc

```
<%
Sub DisplayTime
   Response.Write (Time)
End Sub
 %>
```

distime.asp

```
<!-- #include file = "time.inc" -->
<HTML>
<BODY>
<%
   Response.Write ("Bây giờ là:")
   DisplayTime ' Gọi hàm trong tập tintime.inc
%>
</BODY>
</HTML>
```

Lưu ý: Dòng chỉ thị #**include** không được đặt trong đoạn chứa các lệnh kichk bản.

2.6. TẬP TIN GLOBAL.ASA

ASP cung cấp cho bạn file cấu hình **global.asa**, trong file này bạn có thể đặt các script xử lý các sự kiện hay các hàm, thủ tục, biến mang tính toàn cục. File **global.asa** phải được đặt trong thư mục gốc của ứng dụng và mỗi ứng dụng chỉ được phép có duy nhất một file **global.asa**. Khi trang asp của ứng dụng được triệu gọi lần đầu tiên, trình chủ **IIS** sẽ tìm xem trong thư mục hiện tại của ứng dụng có file **global.asa** không. Nếu có thì trình chủ sẽ nạp và xử lý các sự kiện được cài đặt trong file này, sau đó chuyển giao quyền xử lý lại cho trang ASP. Trong file **global.asa**, bạn chỉ được phép cài đặt và xử lý các sự kiện sau:

❖ **Application_OnStart**: Sự kiện này được phát sinh khi người dùng đầu tiên triệu gọi bất kỳ trang nào trong ứng dụng. Khi trình chủ **IIS** khởi động lại hoặc khi nội dung file **global.asa** bị hiệu chỉnh thì sự kiện này được phát sinh trở lại. Sau khi xử lý xong sự kiện này, trình chủ bắt đầu xử lý sự kiện **Session_OnStart** để chuẩn bị cho phiên nối kết. Các biến **Application** thường được khởi tạo bên trong sự kiện này.

❖ **Session_OnStart:** Sự kiện này được gọi mỗi khi có một người dùng mới yêu cầu trang asp của ứng dụng Web trong lần đầu tiên. Các biến **session** của người dùng cũng thường được khởi tạo bên trong sự kiện này.

❖ **Session_OnEnd:** Sự kiện này được gọi khi phiên làm việc của người dùng chấm dứt. Phiên làm việc được xem là chấm dứt khi nó hết hạn (timeout hay expired), mặc định cho thời gian làm việc của **session** là 20 phút, bạn có thể tăng hay giảm thời gian này bằng cách thay đổi giá trị của thuộc tính **Timeout** của đối tượng **session.**

❖ **Application_OnEnd**: Sự kiện này được gọi khi không còn người dùng nào tương tác với ứng dụng web của bạn nữa. Thông thường thì sự kiện này được gọi khi trình chủ **IIS** ngừng hoạt động. Thông qua sự kiện này bạn có thể giải phóng vùng nhớ đã cấp phát trước đó hoặc lưu lại các thông tin, trạng thái cần thiết xuống đĩa cứng để phục vụ cho quá trình khởi động trở lại sau đó.

Bạn cài đặt thủ tục xử lý sự kiện trong file **global.asa** theo mẫu sau:

```
<script language = “vbscript” runat = “server”>

Sub Application_OnStart
  …….
End Sub

Sub Session_OnStart
  ……..
End Sub

Sub Session_OnEnd
  ……..
```

```
End Sub

Sub Application_OnEnd
  ……….
End Sub
</script>
```

Ví dụ: Dưới đây là ví dụ minh họa cách cài đặt và xử lý sự kiện trong file **global.asa**.

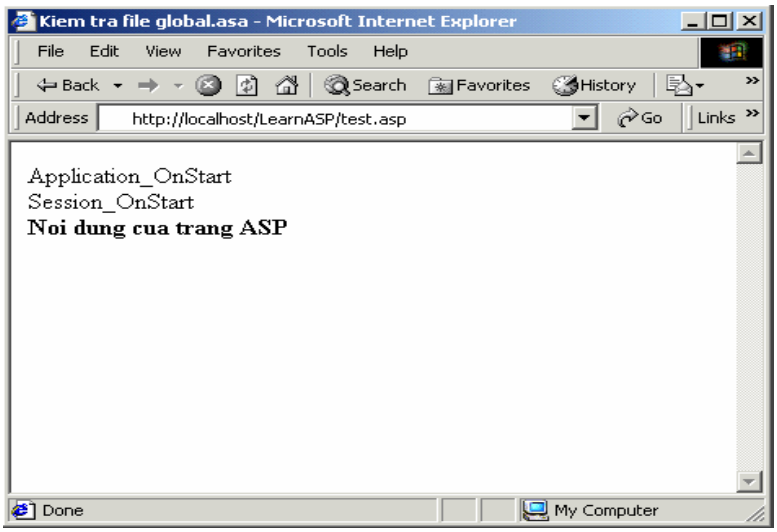
```
<script language = "vbscript" runat = "server">

Sub Application_OnStart
   Application("Status") = "Application_OnStart"
End Sub

Sub Session_OnStart
  Response.Write (Application("Staus") + "<br>")
  Response.Write ("Session_OnStart" + "<br>")
End Sub

Sub Session_OnEnd
End Sub

Sub Application_OnEnd
End Sub
</SCRIPT>
```

Bạn lưu file **global.asa** vào thư mục của ứng dụng (giả sử là LearnASP). Kế tiếp là bạn tạo một tập tin để kiểm tra file **global.asa** với tên **test.asp** và đặt cùng thư mục với tập tin **global.asa** với nội dung sau:

```
<HTML>
<HEAD>
<TITLE> Kiem tra file global.asa </TITLE>
</HEAD>
<BODY>
<B>
<%
   Response.Write "Noi dung cua trang ASP"
```

```
%>
</B>
</BODY>
</HTML>
```

Mở trình duyệt lên và bạn triệu gọi file **test.asp**. Kết quả sẽ được thể hiện như sau:



Ngoài ra bạn có thể đặt các hàm hay thủ tục xử lý trong file **globsl.asa** để có thể các trang trong ứng dụng có thể triệu gọi các hàm này.

III- KẾT NỐI VÀ TRUY XUẤT DATABASE

Phần lớn các ứng dụng Web động đều cần phải lưu trữ được các thông tin cần thiết để cung cấp cho các trang Web. Có nhiều phương pháp khác nhau để lưu trữ thông tin, ví dụ ta có thể sử dụng các tệp Text, nhưng đơn giản nhất là ta sử dụng một Hệ quản trị CSDL nào đó như SQL Server, MS Access, Oracle, MySQL,... vì lý do các Hệ quản trị CSDL này đã có sẵn các chức năng tổ chức dữ liệu, khai thác dữ liệu rất hiệu quả. Vấn đề ở đây là ta phải biết sử dụng các Hệ quản trị CSDL này để tạo ra CSDL và khai thác chúng bằng ASP.

Có rất nhiều cách để có thể kết nối tới database như dùng ADO (ActiveX Data Object), ODBC (Open Database Connectivity).. ở đây ta sẽ nghiên cứu qua ADO.

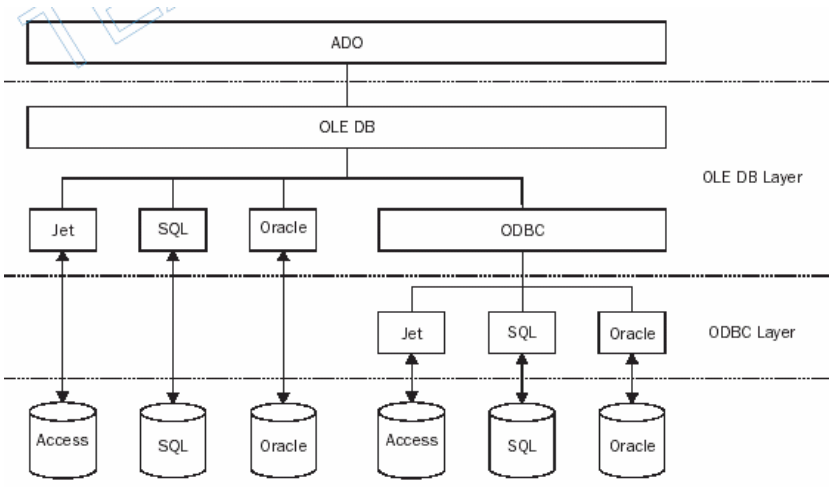
Trong phần này không đề cập lại các vấn đề của các Hệ quản trị CSDL, coi như bạn đã có kiến thức về một số Hệ quản trị CSDL thông dụng như MS Access hay SQL Server. Sau khi bạn đã có một CSDL, phần này sẽ giúp bạn biết cách dùng các đối tượng của ADO trong môi trường ASP để:

- Kết nối với CSDL.
- Thực hiện các thao tác với dữ liệu trong CSDL như: Truy vấn, thêm, sửa, xoá dữ liệu.

3.1. Giới thiệu về ADO.

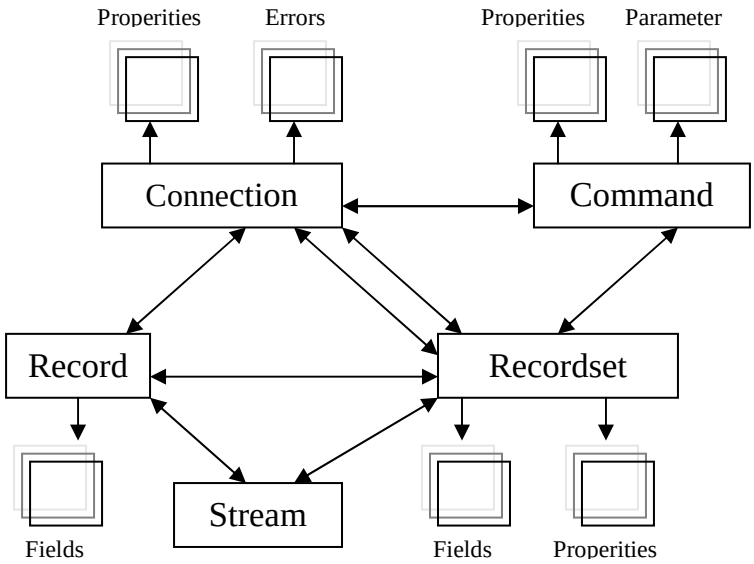
ADO (ActiveX Data Object) là một công nghệ mới của Microsoft, như tên gọi ADO là một thành phần ActiveX tạo ra một giao diện lập trình giúp cho việc truy xuất dữ liệu trong database. Và nó được cài đặt tự động khi cài Microsoft IIS.

ADO nối kết với cơ sở dữ liệu có thể xem như thông qua tầng OLEDB hay tầng ODBC; từ tầng OLEDB có thể nối kết trực tiếp dựa trên Provider được cung cấp cho từng hệ cơ sở dữ liệu riêng biệt hoặc thông qua ODBC cung cấp Driver cho từng hệ cơ sở dữ liệu như mô hình và ta có các bảng liệt kê bên dưới:



3.2. Các đối tượng của ADO

ADO cung cấp một tập các đối tượng cho lập trình viên sử dụng để kết nối đến CSDL và thực hiện các thao tác trên CSDL, đó là các đối tượng Connection, Command, Recordset, Record, Stream và các tập hợp Errors, Fields, Properitites, Parameters. Sự liên hệ giữa các đối tượng của ADO được biểu diễn bởi hình sau:



a. Đối tượng Connection

Đối tượng cho phép bạn nối kết với cơ sở dữ liệu thông qua phương thức Open(). Qua đó, nó cung cấp sự kết nối giữa ADO và một cơ sở dữ liệu nào đó. Nó chứa ba thông tin:

Cơ sở dữ liệu

Giao thức (driver/ provider) để trao đổi thông tin với

Username và Password

Đối tượng Command

Bạn có thể sử dụng phương thức Execute() của đối tượng này để thực hiện các câu lệnh ở dạng các câu lệnh SQL như SELECT, INSERT, UPDATE, DELETE, hay các câu lệnh SQL thay đổi cấu trúc dữ liệu như ALTER TABLE hay DROP INDEX, hoặc có thể chứa các store procedure. Đối tượng này thường được dùng để chạy câu lệnh SQL không trả về bất kỳ mẫu tin nào.

b. Đối tượng Recordset

Đối tượng chứa tập hợp các bản ghi là kết quả truy vấn được từ cơ sở dữ liệu. Nó cho phép thay đổi dữ liệu như thêm, xóa, sửa dữ liệu, hay di chuyển giữa các bản ghi trong tập bản ghi mà nó biểu diễn. Tại một thời điểm đối tượng này đứng ở một bản ghi mà thôi.

c. Đối tượng Record

Đối tượng Record lưu trữ một hàng (mẫu tin) trong Recordset, một thư mục hay tập tin trong File System

d. Đối tượng Stream

Đối tượng được thiết kế để quản lý dữ liệu dạng binary, nó được dùng để quản lý dữ liệu BLOB (Binary Large Object) như hình ảnh hay mảng dữ liệu lớn

3.3. Truy xuất database bằng ADO trong ASP :

Cách thức cơ bản để truy xuất dữ liệu trong database trong ASP như sau :

- Xây dựng một kết nối ADO tới database.
- Thực thi kết nối đó để kết nối vơi database.
- Tạo một đối tượng ADO recordset (tập các record).
- Mở recordset này (sử dụng SQL để thu thập các record).
- Lấy dữ liệu kết quả từ recordset này.
- Đóng recordset lại.
- Đóng kết nối lại.

3.3.1. Kết nối với CSDL bằng đối tượng ADODB.Connection

ADO cung cấp đối tượng Connection để hỗ trợ cho việc tạo và quản lý kết nối cơ sở dữ liệu cần xử lý. Các thuộc tính và phương thức của đối tượng này cho phép bạn mở, đóng kết nối, đồng thời cho phép thực hiện các câu truy vấn dữ liệu.

Để có thiết lập kết nối cơ sở dữ liệu, ta cần phải thực hiện các bước sau:

- Tạo một thể hiện (instance) của đối tượng Connection từ phía server bằng lệnh: **Server.CreateObject("ADODB.Connection")**
- Sử dụng phương thức **Open** để mở kết nối cơ sở dữ liệu. Tham số của phương thức này là chuỗi **connection string**, chuỗi này sẽ tương ứng với cơ sở dữ liệu.

Mấu chốt quan trọng nhất để có thể kết nối được đến một CSDL qua ADODB.Connection là bạn phải thiết lập được connection string. Connection string là một xâu ký tự chứa tất cả những thông tin cần thiết để có thể kết nối được tới một CSDL như:

- Giao thức kết nối (driver/provider)
- Tên máy chủ CSDL (nếu sử dụng SQL Server, Oracle,...)
- Tên CSDL
- User name và Password (nếu cần)

Có nhiều cách kết nối CSDL khác nhau, mỗi cách tương ứng với một cách xây dựng connection string, như:

a. Kết nối thông qua ODBC không dùng DSN (Data Source Name).

b. Kết nối thông qua OLE DB.

c. Kết nối thông qua ODBC sử dụng DSN.

a. Kết nối qua ODBC không dùng DSN (data source name – tên nguồn dữ liệu)

Đây là phương pháp kết nối khá đơn giản và linh động, dễ cài đặt và triển khai vì nó không yêu cầu ta phải thực hiện thao tác nào với hệ thống (như phương pháp c. dùng DSN) ngoài những mã lệnh ta thấy dưới đây.

Ví dụ1: Ta có một tệp cơ sở dữ liệu (tạo bằng ACCESS chẳng hạn) là Dulieu.mdb được đặt trong thư mục C:\Data\Dulieu.mdb thì ta có thể kết nối đến database này bằng đoạn lệnh sau :

```
<%
 'Tạo thể hiện của đối tượng ADODB.Connection
 Set Conn=server.CreateObject("ADODB.Connection")
 'Tạo connection string
 strDSN="Driver={Microsoft Access Driver(*.mdb)}; DBQ=C:\Data\Dulieu.mdb"
 ' Mở nối kết qua bằng connection string
 Conn.Open strConn
%>
```

Trong ví dụ trên, trong Connection string ta phải chỉ rõ **Driver={Microsoft Access Driver(*.mdb)}** (gọi là Provider) để ADO hiểu là ta muốn kết nối đến Hệ quản trị CSDL là MS Access và chỉ rõ đường dẫn vật lý tới database qua tham số **DBQ.**

Ví dụ 2: Giả sử ta có CSDL trong SQL Server có tên là pubs và tên máy chủ CSDL là Popmap, Username là sa, Password là 123. Để truy cập được CSDL này qua ODBC không dùng DSN, ta phải xây dựng Connection string như sau:

```
<%
 'Tạo thể hiện của đối tượng ADODB.Connection
 Set Conn=server.CreateObject("ADODB.Connection")
 'Tạo connection string
 strDSN= "Driver = {SQL Server}; Server = Popmap; uid = sa; pwd=123 ;
 database= pubs"
 ' Mở nối kết qua bằng connection string
 Conn.Open strConn
%>
```

Trong ví dụ trên, trong Connection string ta phải chỉ rõ **Driver = {SQL Server};** để ADO hiểu là ta muốn kết nối đến Hệ quản trị CSDL là MS SQL Server và tên máy chủ được chỉ ra qua tham số **Server,** Username được chỉ ra trong tham số **uid**, Passworrd được chỉ ra trong tham số **pwd,** tên CSDL chỉ ra qua tham số **database.**

b. Kết nối thông qua OLE DB

Đây cũng là phương pháp hiệu quả và dễ cài đặt.

Trở lại ví dụ 1 ở trên, ta có mã lệnh như sau:

```
<%
 'Tạo thể hiện của đối tượng ADODB.Connection
 Set Conn=server.CreateObject("ADODB.Connection")
 'Tạo connection string
 strDSN="Provider=Microsoft.Jet.OLEDB.4.0;Data Source= C:\Data\Dulieu.mdb"
 ' Mở nối kết qua bằng connection string
 Conn.Open strConn
%>
```

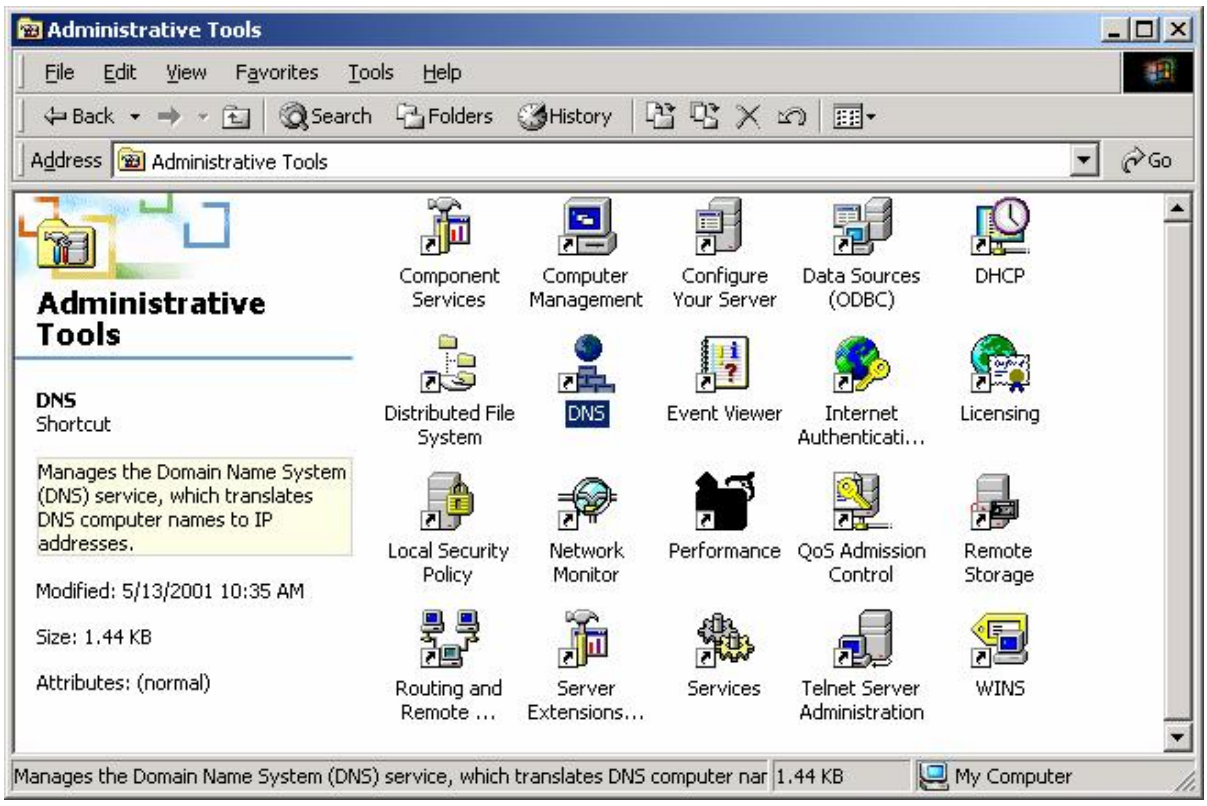
Qua những ví dụ trên, bạn có thể tự so sánh và thấy rằng ADO sẽ dựa vào các từ khoá trong Connection string để phân biệt giữa hai phương pháp kết nối trên. Bạn cũng nên nhớ rằng đây chỉ là một vài ví dụ kết nối CSDL thông dụng nhất, ta cũng có thể dùng Connection string để kết nối đến các hệ quản trị CSDL khác mà ADO hỗ trợ. Về vấn đề này thì bạn có thể tham khảo trên mạng hoặc các tài liệu khác.

c. Tạo một kết nối dữ liệu bằng ODBC (Open Database Connectivity) .

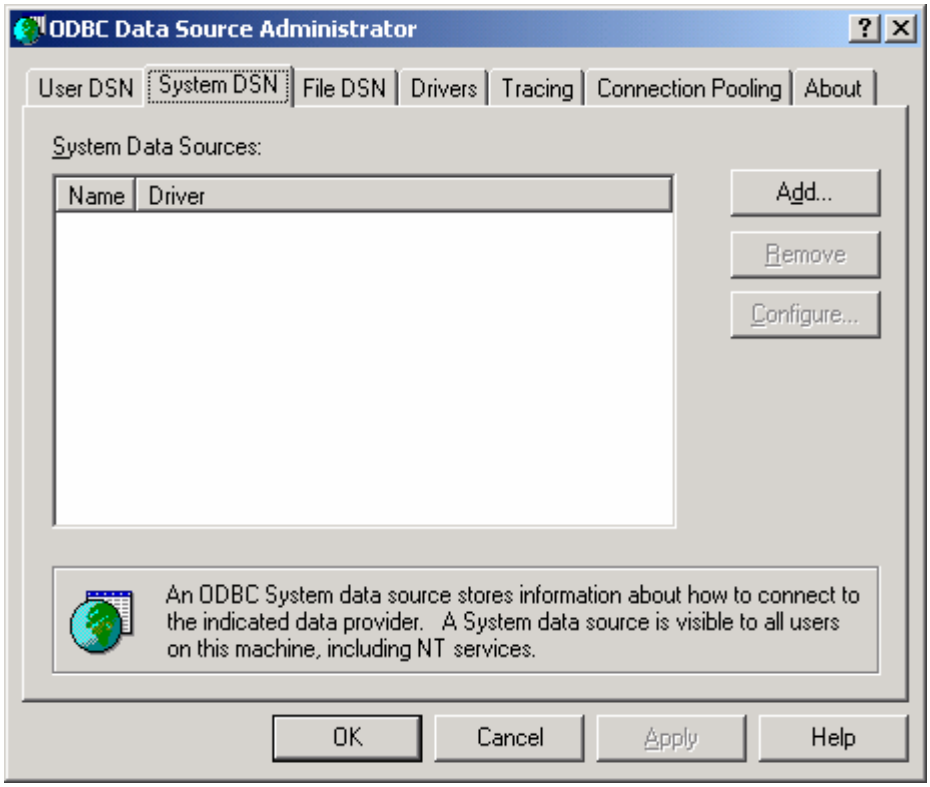
Theo cách này ta không cần chỉ rõ đường dẫn cũng như provider cho kết nối. Tất cả đã được khai báo trong ODBC.

Ví dụ ta tạo một kết nối ODBC tới dữ liệu của MS Access ta làm như sau :

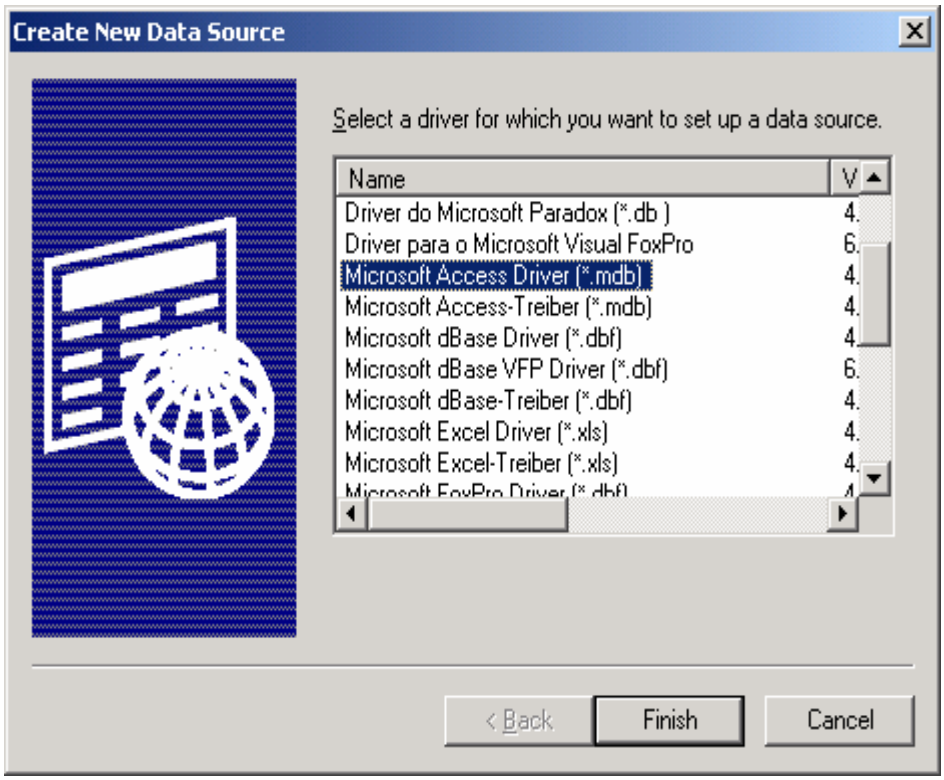
Chọn Start / Settings / Control Panel/ ODBC



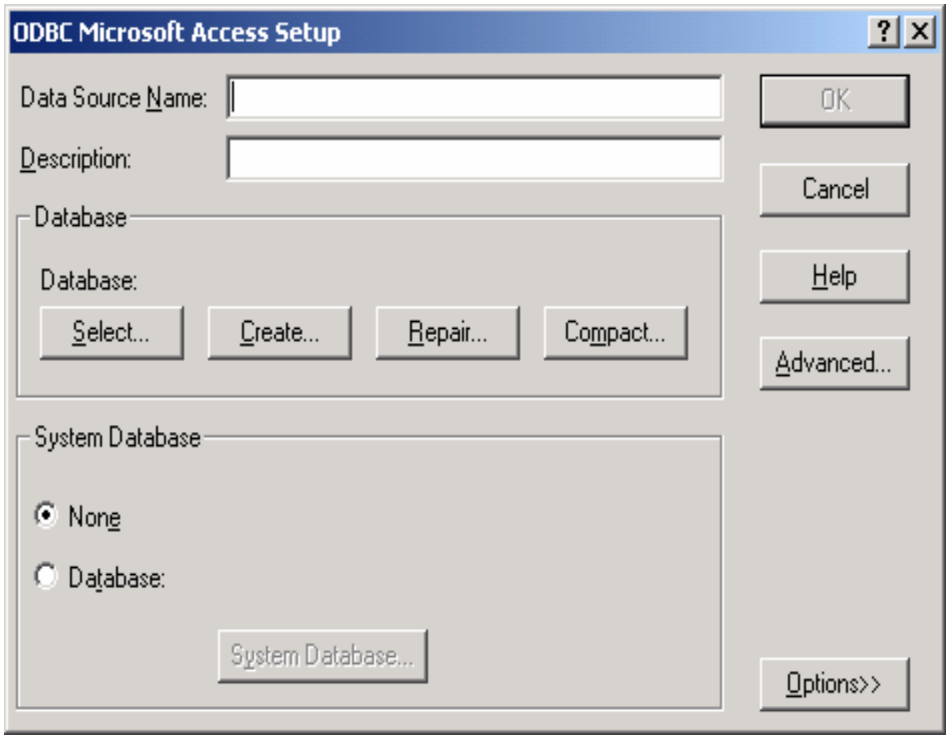
Chọn System DNS



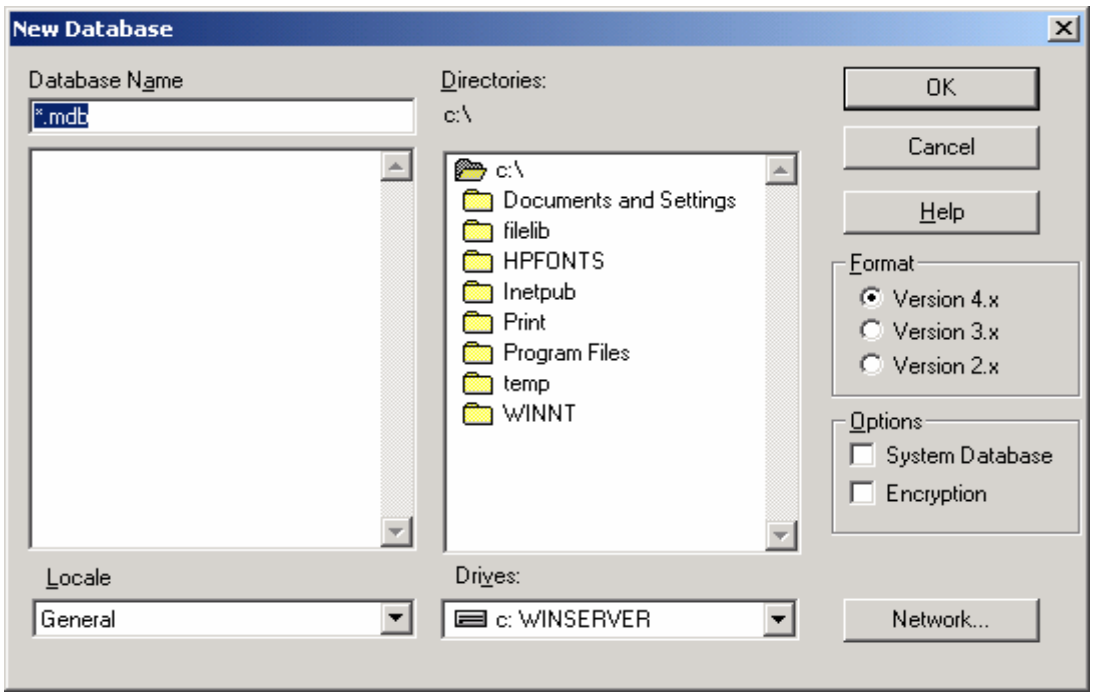
Chọn nút Add và chọn Microsoft Access Driver



Kích Finish và màn hình sau hiện ra



Trong Data Source Name ta nhập tên nguồn dữ liệu vào chẳng hạn là BangLuong và ta chọn nút CREATE màn hình sau hiện ra



Ta chọn duyệt đến nơi đặt database và chọn OK

Sau đó trong ASP muốn kết nối tới database đó ta làm như sau :

```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Open "bangluong"
%>
```

Chú ý :

- Nếu tệp Dulieu.mdb được đặt trong thư mục chứa tệp ASP thì có thể dùng cách server.mappath("dulieu.mdb") để trả về đường dẫn đầy đủ một cách linh động vì nó không phụ thuộc vào đường dẫn chính xác, cố định.

```
<%
 Set Conn=server.CreateObject(“ADODB.Connection”)
 strDSN=”Provider=Microsoft.Jet.OLEDB.4.0;
                    Data Source=” + server.mappath("dulieu.mdb")
 Conn.Open strConn
%>
```

- Ta cũng có thể gán xâu kết nối cho đối tượng ADODB.Connection bằng cách khác thông qua thuộc tính ConnectionString của nó, và khi gọi phương thức Open, ta không phải chỉ ra tham số cụ thể.

```
<%
 Set Conn=server.CreateObject(“ADODB.Connection”)
 Conn.ConnectionString=”Provider=Microsoft.Jet.OLEDB.4.0;
                    Data Source=”+server.mappath("dulieu.mdb")
 Conn.Open
%>
```

- Sau khi đã kết nối thành công đến một CSDL, bạn có thể thực hiện các thao tác với CSDL đó. Khi nào không dùng kết nối CSDL nữa bạn phải đóng kết nối và giải phóng tài nguyên bằng các lệnh sau:

```
<%
 Conn.Close
 Set Conn=Nothing
%>
```

3.3.2. Thao tác với CSDL sử dụng ADODB.Recordset

Sau khi đã kết nối thành công với CSDL thông qua đối tượng ADODB.Connection, ta đã có thể thao tác được với các đối tượng bên trong CSDL đó. Có nhiều cách để làm điều này, nhưng đơn giản nhất là thao tác với các đối tượng của CSDL thông quan ADODB.Recordset. Nó là đối tượng dùng để chứa tập các bản ghi dữ liệu mà ta lấy ra từ CSDL, nó cung cấp cho ta các thuộc tính và các phương thức để ta có thể thao tác với tập bản ghi mà nó chứa như: dịch chuyển con trỏ giữa các bản

thi, lấy thông tin từ các trường, sửa thông tin của các trường, thêm bản ghi, xoá bản ghi...

Ví dụ: Giả sử ta có tệp DuLieu.mdb có hai bảng sau

Bảng 1 : tên là LUONG

MaNV	HoTen	LuongCB	PhuCap	TongTien

Bảng 2 : tên là DIACHI

MaNV	Tel	SoNha	Pho	Quan

Tạo một ADO Table Recordset : Trong trường hợp này recordset sẽ chứa trong nó 1 table.

Giả sử rằng ta đã có một DSN là "BangLuong" kết nối tới tệp DuLieu.MDB, trong tệp DuLieu.mdb này có 2 bảng là Luong và DiaChi

```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Open "BangLuong"
set rs = Server.CreateObject("ADODB.recordset")
rs.Open "Luong", conn
%>
```

Vậy trong RS sẽ chứa bảng LUONG

Tạo một ADO SQL Recordset : Sẽ sử dụng ngôn ngữ SQL để trích vấn các record

```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Open "bangluong"
set rs = Server.CreateObject("ADODB.recordset")
rs.Open "Select * from DIACHI", conn
%>
```

Để lấy dữ liệu trong ADO recordset ta hãy xem qua các ví dụ sau :

Ví dụ 1: In tất cả mọi người trong bảng LUONG ra màn hình

```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Open "bangluong"
set rs = Server.CreateObject("ADODB.recordset")
```

```
rs.Open "Select * from LUONG", conn
for each x in rs.fields
  response.write(x.name) ‘ in tên cột
  response.write(" = ")
  response.write(x.value) ‘ in giá trị
  response.write “<br>”
next
set rs=nothing
set conn=nothing
%>
```

Bây giờ, ta sẽ tìm hiểu một số phương thức của Recordset

3.3.2.1.Các phương thức của đối tượng Recordset

Phương thức	Diễn giải
AddNew	Tạo mới record
Cancel	Hủy các thao tác đang thực thi
Close	Đóng đối tượng recordset và các đối tượng liên quan
Delete	Xóa record hay một tập record hiện hành
Find	Tìm một record thoả điều kiện
GetRows	Lấy nhiều record đưa vào một mảng
GetString	Trả về recordset dưới dạng một chuỗi
MoveFirst	Đưa vị trí của record hiện hành về record đầu tiên trong recordset
MoveLast	Đưa vị trí của record hiện hành về record cuối cùng trong recordset
MoveNext	Đưa vị trí của record hiện hành về record kế
NextRecordset	Xóa đối tượng Recordset hiện hành và trả về đối tượng recordset kế tiếp
Open	Mở một recordset
Requery	Cập nhật lại dữ liệu bằng cách thực hiện lại câu lệnh truy vấn ban đầu

Resync	Refesh lại dữ liệu trong đối tượng Recordset hiện hành
Save	Lưu Recordset xuống file
Seek	Tìm chỉ mục của recordset
Update	Lưu các thay đổI

Đối tượng Recordset có nhiều phương thức để xử lý thao tác dữ liệu như bảng liệt kê ở trên, trong đó các phương thức thường sử dụng như để tác động đến sự thay đổi mẫu tin như **AddNew, Update, Delete**; di chuyển vị trí các mẫu tin như **MoveFirst, MovePrevious, MoveNext, MoveLast**; đóng mở recordset như **Open, Close.** Ta sẽ lần lượt đi vào chi tiết cách thức sử dụng các phương này một các cụ thể.

1. Phương thức Open:

Phương thức Open có thể coi như điểm bắt đầu của Recordset, nó cho phép ta lấy về một tập bản ghi thông qua tên của bảng (TABLE) một cách trực tiếp, hoặc thông qua một câu truy vấn kết nối một hay nhiều bảng với nhau, hay thực hiện một thủ tục lưu trữ (Stored Procedure) của SQL Server mà trả về tập bản ghi.

Khi sử dụng phương thức này, bản phải đặc biệt chú ý các tham số CursorType và LockType. Đó là những tham số được sử dụng để giới hạn sự tương tác với tập bản ghi như: có cho phép dịch chuyển con trỏ hay không?, có cho phép cập nhật dữ liệu hay chỉ được phép đọc dữ liệu?,...

Cú pháp:

objRs.Open Source, Connection, CursorType,LockType, Options

trong đó:

Source	Xâu ký tự biểu diễn tên bảng hay câu lệnh SQL, hoặc Stored Procedure
ActiveConnection	chứa instance đối tượng Connection đã được khai báo hay chuỗi kết nối (Connection String)
CursorType	Kiểu con trỏ mà cơ sở dữ liệu sử dụng khi mở Recordset.
LockType	Kiểu khóa sẽ được sử dụng trong Recordset. Bao gồm 4 kiểu khóa:
Options	Kiểu của truy vấn hay bảng được miêu tả bởi Source

Tham số **CursorType –** dùng để khai báo kiểu con trỏ dữ liệu:

Hằng số	Giá trị	Chức năng
adOpenForwardOnly	0	Truy xuất tuần tự trong Recordset. Đây là cursor mặc định
AdOpenKeyset	1	Không được truy xuất đến record đang được user khác truy xuất
adOpenDynamic	2	Cho phép sửa đổi, thêm hay xóa ngay cả recordset đang được mở bởi user khác
adOpenStatic	3	Không được phép thay đổi record khi nó đang được mở bởi user khác

Tham số **LockType**: có 4 kiểu khóa

Hằng số	Giá trị	Chức năng
adLockReadOnly	1	Khóa mặc định, các trường trong recordset chỉ có thể đọc không thể cập.
adLockPressimistic	2	Sử thay đổi dữ liệu sẽ có tác động ngay lập tức trên recordset
adLockOptimistic	3	Khóa mẫu tin hiện hành khi gọi phương thức Update.
adLockBatchOptimistic	4	Thực hiện việc cập nhật theo lô.

Tham số **Options** – khai báo kiểu của tham số **Source** là gì:

Hằng	Giá trị	Loại của CommandText
AdCmdText	1	Tham số Source là câu lệnh SQL
AdCmdTable	2	Tham số Source là Tên bảng
AdCmdStoredProc	4	Tham số Source là stored procedure hay câu truy

		vấn
AdCmdUnknown	0	Tham số Source không xác định
AdCmdFile	256	Tham số Source là file
AdCmdTableDirect	512	Tham số Source là tên bảng

2. Phương thức AddNew: Phương thức này cho phép tạo mới mẫu tin, gán dữ liệu mới vào các field của mẫu tin, và nó chỉ được cập nhật vào cơ sở dữ liệu khi ta gọi phương thức Update hay UpdateBatch

3. Phương thức Update: Phương này được dùng để cập nhật lại mẫu tin hiện thời trong cơ sở dữ liệu

Ví dụ: Sinh viên có mã số CV-012003, muốn thay đổi số điện thoại '9817442'

Trước hết ta tìm Sinh viên có mã số CV-012003, nếu tồn tại ta sẽ cập nhật số điện thoại

objRs.Find "MASV='CV-012003'"

objRs.("Phone")='9817442'

objRs.Update

4. Phương thức Delete: phương thức này cho phép xóa mẫu tin trong Recordset

Cú pháp: objRs.Delete

hay objRs.Delete < tham số>

Tham số	Mô tả
AdAffectCurrent	Xóa mẫu tin hiện hành
AdAffectGroup	Xóa mẫu tin thỏa điều kiện lọc

Ví dụ: Xoá Sinh viên mang họ tên 'Nguyen Thang'

objRs.Find "HOTEN='Nguyen Thang"

If objRs.EOF Then

Response.Write "Không có sinh viên Nguyen Thang"

Else

objRs.Delete

End If

5. Phương thức Close: Để ngắt kết nối với cơ sở dữ liệu, ta dùng phương thức Close có trong đối tượng Recordset cũng như có trong đối tượng Connection. Sau đó để giải phóng tài nguyên hệ thống đã dùng trong các đối tượng này, ta dùng lệnh gán giá trị Nothing cho các biến đối tượng này

```
<%
objRs.Close
Set objRs = Nothing
Conn.Close
Set Conn = Nothing
%>
```

3.3.2.2. Các thuộc tính (properties) của Recordset

Properties	Mô tả
BOF	Trả về TRUE nếu record hiện thời là ở trước record thứ nhất, ngược lại trả FALSE.
EOF	Trả về TRUE nếu record hiện thời là ở sau record cuối, ngược lại trả FALSE.
Fields	Bao gồm toàn bộ các trường.
RecordCount	Trả về số record trong recordset.
Sort	Chỉ ra danh sách tên các cột được sắp xếp.

3.3.2.3. Lấy dữ liệu từ Recordset

Khi muốn lấy được dữ liệu của một trường (field) trong một mẫu tin hiện hành, ta lấy chuỗi tên của trường đó như là đối số cho đối tượng Recordset hay đối số của thuộc tính Fields của đối tượng Recordset. Ví dụ để lấy dữ liệu của trường HOTEN trong bảng STUDENTS ta có thể dùng **objRS("HOTEN")** hay **objRS.Fields("HOTEN")**

Khi muốn dịch chuyển qua lại đến các mẫu tin được lưu trong đối tượng Recordset, ta sử dụng phương thức **MoveNext, MovePrevious, MoveFirst, MoveLast** và phải đi kèm với việc kiểm tra mẫu tin hiện hành có đang ở vị trí đầu hay ở cuối mẫu tin.

Khi muốn kiểm tra vị trí con trỏ mẫu tin hiện hành là trước mẫu tin đầu hay mẫu tin cuối trong Recordset, ta dùng thuộc tính **BOF** hoặc **EOF** để kiểm tra

3.3.2.4. Sử dụng câu truy vấn SQL

Bạn có thể xây dựng sẵn một câu lệnh truy vấn và thực thi nó bằng phương thức Open của đối tượng ADODB.Recordset để lấy ra tập bản ghi thoả mãn câu truy vấn đó. Sử dụng câu truy vấn SQL bạn có thể lọc ra những dữ liệu cần thiết theo một số điều kiện, đồng thời có thể sắp xếp được dữ liệu.

Ví dụ 1: Hiện những người có Họ tên bắt đầu là chữ N

```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Open "bangluong"
sql="SELECT * FROM LUONG WHERE HOTEN LIKE 'N%' "
set rs=Server.CreateObject("ADODB.recordset")
rs.Open sql, conn
%>
<table border="1" width="100%">
<%do until rs.EOF%>
  <tr>
  <%for each x in rs.Fields%>
    <td><%Response.Write(x.value)%></td>
  <%next
  rs.MoveNext%>
  </tr>
<%loop
rs.close
conn.close
%>
</table>
```

Ví dụ 2 : Hiện toàn bộ nhưng Họ tên được sắp xếp

```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Open "bangluong"
set rs=Server.CreateObject("ADODB.recordset")
sql="SELECT * FROM LUONG ORDER BY  HOTEN"
```

```
rs.Open sql, conn
%>
<table border="1" width="100%">
<%do until rs.EOF%>
  <tr>
  <%for each x in rs.Fields%>
    <td><%Response.Write(x.value)%></td>
  <%next
  rs.MoveNext%>
  </tr>
<%loop
rs.close
conn.close
%>
</table>
```

Có một cách khác để thực hiện câu truy vấn là sử dụng phương thức Execute của đối tượng ADODB.Connection rồi gắn kết quả trả về của phương thức này cho đối tượng ADODB.Recordset. Trong trường hợp này, ta không cần phải khai báo trước đối tượng ADODB.Recordset.

Thí dụ sau tương đương với ví dụ 1 ở trên, nhưng ta sử dụng phương thức ADODB.Connection.Execute để thực hiện câu truy vấn.

```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Open "bangluong"       ‘Mở CSDL gắn với DSN là “bangluong”
sql="SELECT * FROM LUONG WHERE HOTEN LIKE 'N%’ ”
‘Bỏ lệnh này set rs=Server.CreateObject("ADODB.recordset")
‘Bỏ lệnh này rs.Open sql, conn
Set rs=conn.Excute(sql)  ‘Không cần khai báo trước đối tượng rs
%>
<table border="1" width="100%">
<%do until rs.EOF%>
  <tr>
  <%for each x in rs.Fields%>
    <td><%Response.Write(x.value)%></td>
```

```
    <%next
    rs.MoveNext%>
    </tr>
<%loop
rs.close
conn.close
%>
</table>
```

3.3.2.5. Một số ví dụ

Ví dụ 1: Ta hiện toàn bộ danh sách trong bảng lương nhưng có thêm tiêu đề các cột cho dễ đọc :

```
<html>
<body>
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Open “bangluong”
    set rs = Server.CreateObject("ADODB.recordset")
sql="SELECT * FROM LUONG"
rs.Open sql, conn
%>
<table border="1" width="100%">
<tr>
<% for each x in rs.Fields
   response.write("<th>" & x.name & "</th>")
next%>
</tr>
<%do until rs.EOF%>
   <tr>
   <% for each x in rs.Fields%>
     <td><% Response.Write(x.value)%></td>
   <%next
   rs.MoveNext%>
   </tr>
<%loop
rs.close
conn.close
%>
</table>
```

```
</body>
</html>
```

Ví dụ 2 : Thêm record vào database

```
<html>
<body>
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Open “bangluong”,2,3 ‘ Tham số để có thể truy xuất , cập nhật database
set rs = Server.CreateObject("ADODB.recordset")
sql="SELECT * FROM LUONG"
rs.Open sql, conn
rs.addnew
rs.fields(“MaNV”)=5
rs.fields(“Hoten”)=”Hoang Hai”
rs.fields(“LuongCB”)=100000
rs.fields(“PhuCap”)=40000
rs.fields(“TongTien”)=140000
rs.update
rs.close
conn.close
%>
</table>
</body>
</html>
```

Ví dụ 3: Trong các trang Web ta hãy thêm hit-counter vào. Đây là một kỹ thuật để đếm số lượt người dùng đên thăm trang Web. Ta xây dựng một CSDL chứa giá trị hiện thời số lượt người đến, mỗi một lượt duyệt Web nữa tăng giá trị trong cơ sở dữ liệu lên 1.

Ta tạo một tệp ASP chuyên dùng để đếm lượt duyệt Web có tên là DEM.ASP sau đó trên toàn bộ các trang Web khác ta sẽ gọi tệp ASP này ở trong mỗi tệp ASP của ta (include file).

Giả sử ta có tệp counter_db.mdb có bảng hit_count. Trong bảng này có hai trường là Page_name và Hit_count. Page_name chứa tên file ASP và Hit_count chứa số lần user duyệt trang đó

```
<%
Function RetrieveAndIncrementCount()
Const adOpenKeyset = 1
```

```
        Const adLockPessimistic = 2
        Const adCmdText = &H0001
        Dim strFilename
        Dim strSQL
        Dim rsCounter
        Dim iCount
        strFilename = Request.ServerVariables("SCRIPT_NAME")
        strSQL = "SELECT page_name, hit_count FROM hit_count WHERE
page_name='" & strFilename & "';"
        Set rsCounter = Server.CreateObject("ADODB.Recordset")
        rsCounter.Open strSQL, "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=" & Server.MapPath("counter_db.mdb") & ";", _
        adOpenKeyset, adLockPessimistic, adCmdText
        If rsCounter.EOF Then
                rsCounter.AddNew
                iCount = 0
                rsCounter.Fields("page_name").Value = strFilename
        Else
                rsCounter.MoveFirst
                iCount = rsCounter.Fields("hit_count").Value
        End If
                ‘ Tăng giá trị lên 1 và update vào DB
        rsCounter.Fields("hit_count").Value = iCount + 1
        rsCounter.Update
        rsCounter.Close
        Set rsCounter = Nothing

        ' Trả về giá trị hiện thời
        RetrieveAndIncrementCount = iCount
        End Function
        %>
```

Trong các tệp ASP của ta , phần đầu tệp ta dùng lệnh

<!--#include file="DEM.ASP”-->

chỗ nào ta muốn sử dụng giá trị đếm ta gọi hàm RetrieveAndIncrementCount

IV - CÁC KỸ THUẬT XÂY DỰNG ỨNG DỤNG ASP HOÀN CHỈNH

4.1. Quản lý môi trường trong ASP

4.1.1. Lệnh Request.ServerVariables :

Nhằm để biết được tất cả những gì client cho phép ví dụ như browser của client là gì, địa chỉ IP là gì, đang duyệt trang Web nào... tất cả những thứ đó ta đều có thể biết được. Ví dụ Request.ServerVariables("REMOTE_ADDR") cho ta biết IP qua Internet của client.

Hãy chạy ví dụ sau :

```
<%
Dim Item
%>
<!--Hiện thông báo chào đón user -->
Hello visitor from <%= Request.ServerVariables("REMOTE_ADDR")
%>!o Your browser identifies itself as <%=
Request.ServerVariables("HTTP_USER_AGENT") %>.
<BR>
<BR>
<!—Hiện toàn bộ các biến server-->
<TABLE BORDER=2>
<TR>
<TD><B>Server Variable</B></TD>
<TD><B>Value</B></TD>
</TR>
<% For Each Item In Request.ServerVariables %>
<TR>
<TD><FONT SIZE="-1"><%= Item %></FONT></TD>
<TD><FONT SIZE="-1"><%= Request.ServerVariables(Item)
%> </FONT></TD>
</TR>
<% Next %>
</TABLE>
```

Kết quả chạy script trên như sau :

Hello visitor from 202.167.117.130! Your browser identifies itself as Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0).

Server Variable	Value
ALL_HTTP	HTTP_ACCEPT:image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword, */* HTTP_ACCEPT_LANGUAGE:en-us HTTP_CONNECTION:Keep-Alive HTTP_HOST:www.asp101.com HTTP_REFERER:http://www.asp101.com/samples/index.asp HTTP_USER_AGENT:Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0) HTTP_COOKIE:NGUserID=a0201d5-1156-986206461-1; ASPSESSIONIDGQQQQMOU=IOIONDOBLBADFBFHLGPHHJOC HTTP_ACCEPT_ENCODING:gzip, deflate
ALL_RAW	Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword, */* Accept-Language: en-us Connection: Keep-Alive Host: www.asp101.com Referer: http://www.asp101.com/samples/index.asp User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0) Cookie: NGUserID=a0201d5-1156-986206461-1; ASPSESSIONIDGQQQQMOU=IOIONDOBLBADFBFHLGPHHJOC Accept-Encoding: gzip, deflate
APPL_MD_PATH	/LM/W3SVC/3/Root
APPL_PHYSICAL_PATH	D:\Webs\asp101\www\
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	0
CONTENT_TYPE	
GATEWAY_INTERFACE	CGI/1.1
HTTPS	off
HTTPS_KEYSIZE	
HTTPS_SECRETKEYSIZE	
HTTPS_SERVER_ISSUER	
HTTPS_SERVER_SUBJECT	
INSTANCE_ID	3
INSTANCE_META_PATH	/LM/W3SVC/3
LOCAL_ADDR	10.2.3.180
LOGON_USER	
PATH_INFO	/samples/servvars.asp
PATH_TRANSLATED	D:\Webs\asp101\www\samples\servvars.asp
QUERY_STRING	
REMOTE_ADDR	202.167.117.130
REMOTE_HOST	202.167.117.130
REMOTE_USER	
REQUEST_METHOD	GET
SCRIPT_NAME	/samples/servvars.asp
SERVER_NAME	www.asp101.com

SERVER_PORT	80
SERVER_PORT_SECURE	0
SERVER_PROTOCOL	HTTP/1.0
SERVER_SOFTWARE	Microsoft-IIS/5.0
URL	/samples/servvars.asp
HTTP_ACCEPT	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword, */*
HTTP_ACCEPT_LANGUAGE	en-us
HTTP_CONNECTION	Keep-Alive
HTTP_HOST	www.asp101.com
HTTP_REFERER	http://www.asp101.com/samples/index.asp
HTTP_USER_AGENT	Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
HTTP_COOKIE	NGUserID=a0201d5-1156-986206461-1; ASPSESSIONIDGQQQQMOU=IOIONDOBLBADFBFHLGPHHJOC
HTTP_ACCEPT_ENCODING	gzip, deflate

Hãy tham khảo các tên biến server ở trên để tạo cho mình một trang Web hoàn chỉnh và chuyên nghiệp hơn.

4.1.2. Điều khiển cache (bộ nhớ đệm) :

Các proxy server – kiến trúc server dùng để giảm nhẹ số lượng cộng việc thu thập thông tin qua Internet- nó được thực hiện bởi việc sử dụng bộ nhớ đệm. Nó lưu trữ các thông tin trong memoy và sẽ cung cấp cho user nếu user muốn sử dụng lại nó. Mặc định thì IIS gửi tất cả các trang Web với header rằng Proxy server không sử dụng cache. Nếu ta muốn sử dụng cache thì phải dùng lệnh sau ở đầu mỗi trang ASP <% Response.CacheControl="Public" %>. Lệnh này nói cho proxy server biết rằng có sử dụng kỹ thuật caching.

4.1.3. Sử dụng Cookie trong trang ASP:

Cookie là những file có kích thước nhỏ được đặt trong thư mục Temporary Internet Files của Window, đây là một cách khá hay để "cá nhân hoá- personalize" trang ASP . Đoạn ví dụ sau mô tả một trang ASP có sử dụng kỹ thuật cookie. Đầu tiên sẽ yêu cầu user nhập họ tên của họ cho lần duyệt đầu tiên, cho những lần duyệt sau, sẽ kiểm tra xem nếu user đã duyệt rồi thì hiện câu chào đón.

```
<%
IF NOT (Request.Form("Name") = "") THEN
Response.Cookies("Name") = Request.Form("Name")
Response.Cookies("Name").Expires = "Jan 1, 2000"
END IF
%>
<HTML>
<HEAD><TITLE>Welcome Page</TITLE></HEAD>
<BODY>
<%
IF Request.Cookies("Name") = "" THEN
Response.Write("<FORM Method=Post>Enter Your Name: ")
Response.Write("<INPUT Type=Text Name=Name> ")
Response.Write("<INPUT Type=Submit></FORM>")
ELSE
Response.Write("Welcome back, "&Request.Cookies("Name"))
END IF
%>
</BODY>
</HTML>
```

4.1.4. Đóng và xoá bỏ các objects :

Các đối tượng mà ta sẽ không dùng nữa trong ASP thì ta nên close lại, xét ví dụ sau:

```
<%
Dim objConn, objRS
Set objConn = Server.CreateObject("ADODB.Connection")
objConn.ConnectionString = ...
objConn.Open
Set objRS = objConn.Execute("...")

objRS.Close
Set objRS = Nothing

objConn.Close
Set objConn = Nothing
```

Các đối tượng được tạo ra bằng Server.CreateObject thì nên set thành nothing.

4.1.5. Sử dụng đối tượng Application:

Đối tượng này cho phép tạo các biến kiểu Application. Đó là các biến toàn cục (global) dùng chung cho tất cả các user của cùng một ứng dụng web, được duy trì kể cả khi stop web server. Tất cả các user mà yêu cầu các trang web từ một thư mục web có thể chia xẻ với nhau các biến định nghĩa trong các trang đó.

```
<%      Application.Lock
```

Application(“Company”) = “IOIT”

Application.Unlock %>

ở đây định nghĩa 1 biến là Company, có giá trị là IOIT. Vì rằng biến có thể bị tranh chấp nên phải đặt giữa Lock và Unlock.

<% Application.Lock

Application(“Time”) = Now

Application.Unlock %>

4.1.6. Sử dụng Session object

Một session là một phiên làm việc giữa browser với web server. Nếu đóng browser lại thì các biến session mất đi.

Đối tượng Session cho phép ta tạo các biến toàn cục (global), nhưng khác với đối tương Application, biến kiểu Session chỉ có phạm vi trong 1 session, tức là nó chỉ có thể được sử dụng cho một user duy nhất mà tạo ra nó. Nói cách khác biến kiểu Session là biến toàn cục nhưng ở mức độ user. Nó hoàn toàn hữu ích khi sử dụng đối tượng Session để lưu trữ thông tin vì nó không bị mất đi khi user nhảy từ trang Web này sang trang Web khác. Nó chỉ bị xoá khi user đóng browser, hoặc duyệt trang Web mới hay giới hạn thời gian tồn tại của Session bị hết (khoảng 20’) . Thực ra mỗi biến sẽ được server phát sinh và gán cho nó một giá trị định danh GUID (Globally Unique Identifier, 128 bit) và gửi tới Browser. Browser sẽ lưu trữ GUID đó và sử dụng để yêu cầu dữ liệu từ biến có GUID tương ứng trên Server.

Xét ví dụ sau :

Ta có tệp thứ nhất tên là VD1.ASP

```
<%
  session(“hoten”)=”Hoang Tuan”
  session(“luong”)=40000
%>
```

Ta có tệp thứ hai tên là VD2.ASP

```
<%
 strHoten= session(“hoten”)
  intLuong=session(“luong”)
%>
```

Vậy ta có 2 biến strHoten và intLuong nhận giá trị từ tệp thứ nhất.

Muốn xoá session ta dùng lệnh Session.Abandon

Ngoài ra nếu muốn truyền giá trị cho một tệp ASP nào đó có thể dùng QueryString. Ví dụ trong tệp VD1.ASP

```
<%
‘ Các giá trị cần truyền đặt theo quy tắc tên biến= giá trị và ngăn cách bởi dấu &
‘ Có thể  truyền nhiều giá trị
%>
<a href=”VD2.ASP?Hoten=Minh&Tuoi=16> Xem chi tiết </a>
```

Trong tệp VD2.ASP ta có thể lấy 2 giá trị trên

```
<%
 strHoten=Request.QueryString(“Hoten”)
intLuong=Request.QueryString(“luong”)
%>
```

4.1.7. File Global.asa:

Một ứng dụng web = tập các trang Web, asp trong một thư mục web và các thư mục con.

File Global.asa xác định khởi đầu và kết thúc của một ứng dụng web cũng như của các session của từng user đơn thể đối với ứng dụng.

File này cho phép bạn đưa vào đặt 4 thủ tục: Session_OnStart, Session_OnEnd, Application_OnStart, Application _OnEnd.

- ✓ Session_OnStart: thực hiện lần đầu tiên khi có một user yêu cầu trang web trong ứng dụng. (ứng với user đó)
- ✓ Session_OnEnd: thực hiện khi user thoát khỏi ứng dụng hoặc timeout.
- ✓ Application_OnStart: thực hiện 1 lần khi trang web đầu tiên của ứng dụng được chạy lần đầu tiên bởi bất kỳ user nào.
- ✓ Application _OnEnd: thực hiện một lần khi web server shutdown hoặc khi tất cả các session đã đóng.

Ta thường sử dụng tệp này để cài đặt các lệnh cần thiết khi một Ứng dụng Web, hay một phiên làm việc của một user bắt đầu khởi đông. Ví dụ, ta có thể đặt lệnh khởi tạo giá trị cho các biến kiểu Session trong thủ tục Session_OnStart, hay đặt lệnh đếm số lần truy cập cho toàn bộ Website của ta.

Ví dụ, ta có tệp CSDL Access tên là data.mdb, trong đó có bảng **tblCounter**, trong bảng này có cột **Counter_Session** dùng để ghi nhận số lần truy cập của Website. Ta sẽ tạo ra file global.asa với nội dung như sau:

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

```
Sub Session_OnStart
        Session("UserLoginState") = 0
        Session("UserName")= ""
        Session("LangID") = 2   '1 - Tieng Viet;  = 2 la English
        Session("UserLevel")  = 1
        ' CAP NHAP BIEN DEM VAO CSDL
        strConn = "provider=microsoft.jet.oledb.4.0;data source=" &
Server.MapPath("data.mdb")
        Set db = Server.CreateObject("ADODB.Connection")
        db.Open strConn
        db.Execute "UPDATE tblCounter SET Counter_Session=Counter_Session+1"
        db.Close
End Sub

Sub Session_OnEnd

End Sub
</SCRIPT>
```

4.1.8. Chèn file (Include)

Chèn file là một kỹ thuật rất quan trọng khi bạn thiết kế Website bằng ASP. Nó cho phép:

- Tổ chức thư viện các chương trình con mà được sử dụng lại nhiều lần vào một thuvien.asp và khi nào cần sử dụng chúng trong các file.asp khác, ta không phải viết lại mà chỉ cần chèn file thuvien.asp đã có bằng lệnh <!--#include file = "thuvien.asp"> vào đầu mỗi tệp.
- Xây dựng Website có 1 giao diện thống nhất mà không phải thiết kế lại giao diện cho mỗi trang ASP khác nhau.
  Nghĩa là, thông thường các Website đều có một **khung của giao diện bên ngoài** chung cho mọi chức năng. Nếu làm theo cách đơn giản thì với mỗi một chức năng của Website bạn tạo ra một file.asp độc lập, nó bao gồm khung giao diện chung và các chức năng riêng của trang ASP đó. Như vậy, ngoài việc viết các script của ASP, bạn phải ghép nó với

khung giao diện chung bằng cách copy lại giao diện bên ngoài của Website. Nếu như lúc nào đó cần phải thay đổi lại khung giao diện bên ngoài thì bạn phải thay đổi lại giao diện lần lượt cho từng trang ASP riêng biệt một cách rời rạc. Làm như vậy rất tốn thời gian và khó khăn nhưng cũng không đảm bảo được sự thống nhất của giao diện Website. Để giải quyết vấn đề này, thông thường các Website chuyên nghiệp bằng asp thường chỉ sử dụng một file Index.asp duy nhất là file chính. Nó sẽ chứa khung giao diện chung bên ngoài (bao gồm banner bên trên, các menu bên trái và phải) còn ở chính giữa là phần nội dung thì nó chỉ chứa các lệnh Include từng trang ASP con. Các trang ASP con này thì chỉ có các lệnh script để thực hiện chức năng chính của nó và **đổ dữ liệu vào nơi nó được Include** mà không cần quan tâm đến giao diện bên ngoài. Làm như vậy thì ta có thể xây dựng các chức năng của Website một cách độc lập và thoải mái mà không cần quan tâm đến giao diện chung, ta chỉ quan tâm đến giao diện chung một cách tương đối mà thôi. Nếu như giao diện chung bên ngoài cần có sự thay đổi thì chỉ một file Index.asp phải thay đổi mà thôi.

- Ví dụ: bạn có thể tham khảo qua ví dụ ở phần 4.4.2 dưới đây.

4.2. Xây dựng hệ thống bảo mật :

Có rất nhiều trường hợp trang Web cần phải được bảo vệ vì những lí do như chỉ cho phép một số người nhất định có khả xem trang Web (kiểm tra để login vào hòm thư cá nhân chẳng hạn) hoặc là với mỗi người có thẩm quyền khác nhau thì được xem một mức thông tin chi tiết ở một mức độ nào đấy v.v

4.2.1. Sử dụng request.servervariables :

Có rất nhiều biến trong ASP cung cấp các thông tin về client, ta có thể kiểm tra để biết xem user đó có hợp lệ hay không :

```
<%
If request.servervariables("REMOTE_ADDR") = "200.200.157.4" then
Response.Buffer = TRUE
Response.Status = ("401 Unauthorized")
Response.End ‘ dừng không load trang web nữa
End If
%>
```

4.2.2. Sử dụng CSDL để quản lý username và password

Lưu trữ các thông tin vào CSDL, mỗi lần có user duyệt trang web thì sẽ bắt user nhập username và password vào, sau đó sẽ kiểm tra trong CSDL xem có đúng không .

Xem ví dụ sau :

Trước hết ta xây dựng một CSDL chứa username và password. Tệp này có tên là SECURITY.MDB gồm có bảng LIST với các cột sau :

USERNAME	PASSWORD

Minh	Mn12345
Tuan	Hjgtsf
Hai	13-937

Vậy ta tạo một tệp có tên là LOGIN.ASP chứa các hộp textbox cho user nhập :

```
<html>
<head>
<title>Login</title>
</head>
<body>
<form method="POST" action="VALIDATE.ASP">
  <p><font face=".VnTimeH">Nhập username và password</font></p>
  <p><font face=".VnTime" size="4">UserName<input type="text"
name="username" size="20"></font></p>
  <p><font face=".VnTime" size="4">Password <input type="password"
name="password" size="20"></font></p>
  <p><font face=".VnTime" size="4"><input type="submit" value="Login"
name="Login"><input type="reset" value="Clear" name="Clear"></font></p>
</form>
</body>
</html>
```

Ta tạo tệp có tên VALIDATE.ASP để truy xuất vào database để kiểm tra :

```
<%
strUN=request(“username”)
strPW=request(“password”)
set conn=Server.CreateObject("ADODB.Connection")
conn.ConnectionString="Provider=Microsoft.Jet.OLEDB.4.0;Data Source="&_
server.mappath("security.mdb")
conn.open
set rs = Server.CreateObject("ADODB.recordset")
sql="SELECT * FROM LIST WHERE USERNAME='" & strUN & “’ and “ & _
“PASSWORD = ‘” & strPW & “’”
rs.Open sql, conn,2,3
if not (rs.eof or rs.bof) then
   Response.write “Bạn không có quyền được truy cập trang này”
```

```
            Response.end ‘ ngừng duyệt
         else
            Response.write “Chào mừng bạn ”
        end if
        rs.close
        set rs=nothing
        conn.close
        set conn=nothing
%>
```

Tuy nhiên vẫn phải lưu ý trường hợp sau : giả sử ta có hệ thống trang Web gồm 5 files mà file nào cũng cần phải kiểm tra quyền hạn truy nhập của user. Vậy ngoài màn hình login ra ta còn phải bảo vệ cho hệ thống các trang web vì đề phòng trường hợp user biết địa chỉ các trang Web này mà user vào thẳng luôn. Vậy ta làm như sau:

Trang login kiểm tra username và password như ở trên. Tuy nhiên sau khi kiểm tra xong ta phải lưu kết quả kiểm tra được vào session (session(“login”) = “ok” chẳng hạn)

```
        <%
        strUN=request(“username”)
        strPW=request(“password”)
        set conn=Server.CreateObject("ADODB.Connection")
conn.ConnectionString="Provider=Microsoft.Jet.OLEDB.4.0;Data Source="&_
server.mappath("security.mdb")
conn.open
        set rs = Server.CreateObject("ADODB.recordset")
sql="SELECT * FROM LIST WHERE USERNAME=’" & strUN & “’ and “ & _
        “PASSWORD = ‘” & strPW & “’”
rs.Open sql, conn,2,3
        if not (rs.eof or rs.bof) then
            Response.write “Bạn không có quyền được truy cập trang này”
            Response.end ‘ ngừng duyệt
         else
              Session(“Login”)=”ok” ‘ Ghi nhận việc thành công login
        end if
        rs.close
        set rs=nothing
        conn.close
```

```
        set conn=nothing
%>
```

Ta tạo một tệp có tên là CheckLogin.asp . Mục đích của tệp này sẽ kiểm tra biến Session(“login”) nếu nó bằng “ok” nghĩa là user hợp lệ không thì ngược lại :

```
        <%
        if  not (session(“login”)=”ok”) then
            response.write “ Bạn không được quyền truy nhật trang web này “
            response.redirect “login.asp”  ‘ tự động quay sang trang login
        end if
        %>
```

Và ở đầu mỗi tệp ASP mà ta cần bảo vệ việc truy cập trái phép ta chèn thêm dòng lệnh sau : <!--#CheckLogin.asp-->

4.3. Một số kỹ thuật với CSDL :

4.3.1. Phân trang RecordSet

Khi select => cho kết quả 1 bảng gồm nhiều hàng. Tuy nhiên mỗi lần chỉ muốn sử dụng một vài hàng, ví dụ: bảng 50 hàng, mỗi lần hiển thị 10 hàng => sử dụng kỹ thuật paging. ADO cung cấp các thuộc tính của Recordset như: PageSize, PageCount, AbsolutePage.

PageCount: số trang, PageSize: số hàng.

ở đây ta dùng biến session CurrentPage để ghi nhận trang hiện hành.

```
<% Select     Case Request.QueryString(“Direction”)
              Case “”          ‘Null
                    Session(“CurrentPage”) = 1
              Case “Next”
                    Session(“CurrentPage”) = Session(“CurrentPage”) + 1
              Case “Previous”
                    Session(“CurrentPage”) = Session(“CurrentPage”) - 1
End Select %>
```

✓ Chú ý: Request.QueryString(“Direction”) <=> QueryString(“Direction”)

Tiến hành kết nối truy vấn dữ liệu.

```
<%
const adOpenKeySet = 1
dim objConnection }  không cần khai báo cũng được
dim objRexordset
Set objConnection = Server.CreateObject(“ADODB.Connection”)
```

```
Set ObjConnection.Open “Biblio”, “”, “”
Dim strSQL
StrSQL = “Select * from Authors”
Set objRecordSet  = Server.CreateObject(“ADODB.RecordSet”)
ObjRecordSet.PageSize = 10
ObjRecordSet.Open strSQL, objConnection, adOpenKeyset
ObjRecordSet.AbsolutePage = CLng(Session(“CurrentPage”))
%>
<P> Page = <%=Session(“CurrentPage”) %> of
<%=ObjRecordSet.PageCount>
<%     Dim i
For i = 1 to ObjRecordSet.PageSize %>
<% = ObjRecordSet(“Authors”) %>
...
<% ObjRecordSet.MoveNext
Next %>
<% if CLng(Session(“CurrentPage”)) < ObjRecordSet.PageCount Then %>
<P> <A HREF = “paging.asp?Direction=Next”>Next Page</A></P>
<%Enf if%>
<% If CLng(Session(“CurrentPage”)) > 1 then %>
<P><A Href = “paging.asp?Direction=Previous”>Previous Page</A></P>
<%Enf if%>
<%     ObjRecordSet.close
ObjConnection.close
Set objRecordSet = nothing
Set objConnection = nothing %>
```

4.3.2. Gọi Stored Procedure trong ASP

```
<%     Set Cnn1 = Server.CreateObject(“ADODB.Connection”)
StrCnn = “driver = {SQL Server}; Server = smas; uid = sa; pwd= ; database=
pubs”
Cnn1.Open strCnn
Et sp = CreateObject(“ADODB.Command”)
Set sp.ActiveConnection = Cnn1
Sp.CommandType = adCmdStoredProc
```

```
Sp.CommandText = “sp_Help”
Set pm = sp.CrateParameter(“objname” = 200, 1, 92, “authors”)
Sp.parameters.Append pm
Set rs = sp.Excute()
Do Until rs is Nothing
        Reponse.Write “<Table border = 1 bgColor = White><TR>”
        Call PrintHeadings
        Call PrintAllRecordsInRS
        Loop Reponse.Write “</Table><BR><BR>”
        Set rs=rs.NextRecordSet

Call CleanUp %>
<% Sub PrintHeadings()
        For each hdFld in rs.Fields
                Reponse.Write “<TH>” & hdFld.Name & “</TH>”
        Next
        Reponse.Write “<TR>”
End sub
Sub printAllRecordsInRS()
        Do until rs.EOF
                For each fld in rs.Fields
                        Reponse.Write “<TD>” & fld
                Next
                Reponse.Write “<TR>”
                Rs.MoveNext
        Loop
End Sub
Sub CleanUp()
        Set rs = nothing
        Cnn1.Close
        Set Cnn1 = nothing
End sub %>
```

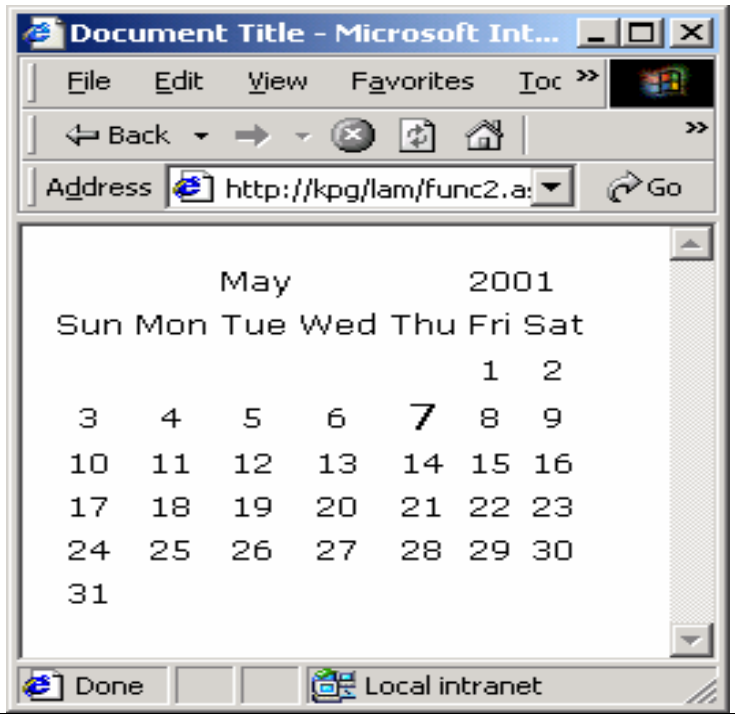
Tóm lại các bước thực hiện:

- Tạo một kế nối Connection, active nó.
- Tạo đối tượng kiểu Command, gán nó với một Connection đã kích hoạt (thuộc tính ActiveConnection), CommandText: tên thủ tục;
- Tạo ta đối tượng tham số, bằng phương thức CreateParameter để truyền tham số cho thủ tục.
- Thực thi và sử dụng kết quả.

V- MỘT SỐ ỨNG DỤNG ASP MẪU

I.Xây dựng ứng dụng Calender:

Đây là một ứng dụng đơn giản, nó sẽ hiện lịch của tháng hiện tại.




```
<%
Option Explicit
Function Calender(datDate)
Dim Months(12)
Dim DaysMonth(12)
Dim Days(7)
Dim strTmp, I, intTmp, strTmp2

 Months(1) = "Januari"
 Months(2) = "Februari"
 Months(3) = "March"
 Months(4) = "April"
 Months(5) = "May"
 Months(6) = "June"
 Months(7) = "Juli"
 Months(8) = "August"
```

```
        Months(9) = "September"
        Months(10) = "October"
        Months(11) = "November"
        Months(12) = "December"

        DaysMonth(1) = "31"
        DaysMonth(2) = "29"
        strTmp = Cstr(Year(datDate) / 4) 'Xem thang 2 co 28 hay 29 ngay
        If Instr(1,strTmp,",",1) = False then
          DaysMonth(2) = "28"
        End If
        DaysMonth(3) = "31"
        DaysMonth(4) = "30"
        DaysMonth(5) = "31"
        DaysMonth(6) = "30"
        DaysMonth(7) = "31"
        DaysMonth(8) = "31"
        DaysMonth(9) = "30"
        DaysMonth(10) = "31"
        DaysMonth(11) = "30"
        DaysMonth(12) = "31"

        Days(1) = "Sun"
        Days(2) = "Mon"
        Days(3) = "Tue"
        Days(4) = "Wed"
        Days(5) = "Thu"
        Days(6) = "Fri"
        Days(7) = "Sat"
        Calender = Calender & "<table cellspacing=0 cellpadding=2 border=0>"
& Vbcrlf
        Calender = Calender & "<tr>" & VbCrlf
        Calender = Calender & "<td colspan=5 align=center><font size=2>" &
Months(Month(datDate)) & "</td>" & VbCrlf
```

```
        Calender = Calender & "<td colspan=2><font
size=2>"&Year(datDate)&"</td>" &VbCrlf
        Calender = Calender & "</tr>" & VbCrlf
        Calender = Calender & "<tr>" & VbCrlf
        ' In toan bo ngay ra
        I = 1
        Do Until I = Ubound(Days) + 1
          Calender = Calender & "<td align=right><font size=2>"&Days(I)&"</td>"
& VbCrlf
        I = I + 1
        Loop
        Calender = Calender & "</tr>" & VbCrlf
        ' In ngay dau tien cua thang vao vi tri hop le
        strTmp = "1/"&Month(datDate)&"/"&Year(datDate)
        I = 0
        Do until I = Cint(WeekDay(strTmp)) - 1
          strTmp2 = strTmp2 & "<td> </td>" & VbCrlf
        I = I + 1
        Loop
        Calender = Calender & "<tr>" & VbCrlf
        I = 1
        intTmp = Cint(Weekday(strTmp)) - 1
        Do Until I = Cint(DaysMonth(Month(datDate))) + 1
          If intTmp = 0 then Calender = Calender & "<tr>" & vbcrlf
          Calender = Calender & strTmp2
          strTmp2 = ""
          Calender = Calender & "<td align=center><font size=2>"
          If I = Cint(Day(datDate)) Then Calender = Calender & "<font size=3><b>"
          Calender = Calender & I
          If I = Cint(Day(datDate)) Then Calender = Calender & "</b>"
          Calender = Calender & "</td>" & VbCrlf
          If intTmp >= 6 then
            Calender = Calender & "</tr>" & VbCrlf
            intTmp = 0
```

```
    Else
      intTmp = intTmp + 1
    End If
  I = I + 1
  Loop
  Calender = Calender & "</table>" & Vbcrlf
  ' Xoa bo cac mang
  erase Months
  erase DaysMonth
  erase Days
End Function
%>
<HTML>
<HEAD>
<TITLE>Vi du ve Calendar</TITLE>
</HEAD>
<BODY>
<font size=2 face=verdana>
<%=Calender(Date())%>
</BODY>
</HTML>
```

II. Xây dựng ứng dụng GuestBook :

Thông thường các trang Web đều có sử dụng GuestBook, với guest book khi user duyệt có thể ghi lại các ý kiến cũng như các cảm nhận của mình .

Trong ví dụ này ta có sử dụng component là Scripting.FileSystemObject, đối tượng này cho phép ta có rất nhiều thao tác hữu ích đối với file, thư mục.

Để chạy ví dụ này hãy nhập đoạn code sau và bạn tạo sẵn 1 tệp có tên guestbook.txt trong cùng thư mục với tệp ASP này

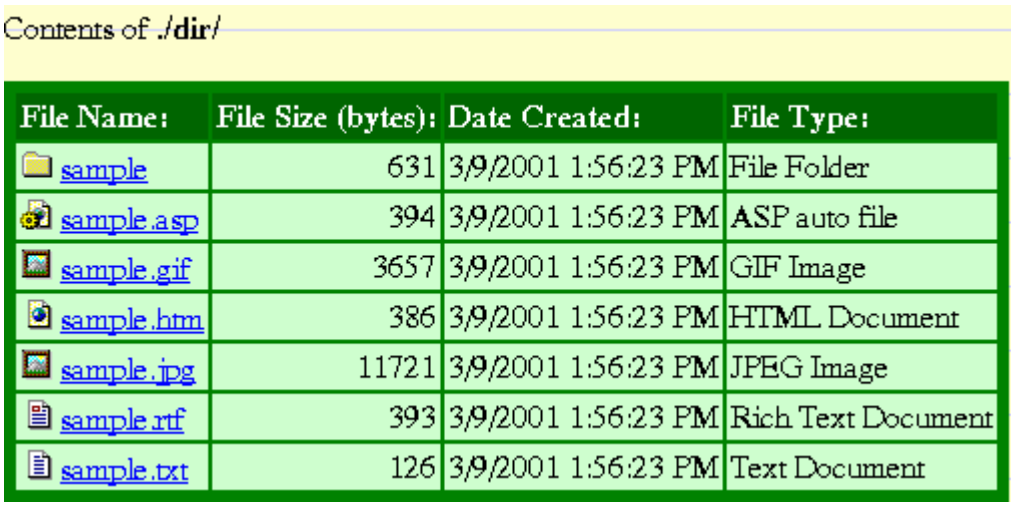
```
<%
Const bDeleteEntries = True
Dim bForce
bForce = Request.QueryString("force")
Dim strFile
strFile = Server.MapPath("guestbook.txt")
If Request.Form.Count = 0 Then
%>
<H3>Sign Our Guestbook:</H3>
<FORM ACTION="guestbook.asp" METHOD="post">
<TABLE>
<TR>
<TD ALIGN="right"><B>Name:</B></TD>
<TD><INPUT TYPE="text" NAME="name"
SIZE="15"></INPUT></TD>
</TR>
<TR>
<TD ALIGN="right"><B>Comment:</B></TD>
<TD><INPUT TYPE="text" NAME="comment"
SIZE="35"></INPUT></TD>
</TR>
</TABLE>
<INPUT TYPE="submit" VALUE="Sign Guestbook!"></INPUT>
</FORM>
<BR>
<H3>Today's Comments:</H3>
<!--#INCLUDE FILE="guestbook.txt"-->
```

```
<%
Else
Dim objFSO
Dim objFile
Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
Set objFile = objFSO.OpenTextFile(strFile, 8, True)
objFile.Write "<B>"
objFile.Write Server.HTMLEncode(Request.Form("name"))
objFile.Write ":</B> "
objFile.Write Server.HTMLEncode(Request.Form("comment"))
objFile.Write "<BR>"
objFile.WriteLine ""
objFile.Close
Set objFile = Nothing
Set objFSO = Nothing
%>
<H3>Your comments have been written to the file!</H3>
<A HREF="./guestbook.asp">Back to the guestbook</A>
<%
End If
If bDeleteEntries Then
Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
Set objFile = objFSO.GetFile(strFile)
If DateDiff("d", objFile.DateLastModified, Date()) <> 0 Or bForce <> ""
Then
        Set objFile = Nothing
        Set objFile = objFSO.CreateTextFile(strFile, True)
        objFile.Write "<B>Hello</B> "
        objFile.WriteLine "I hope you like this guestbook!<BR>"
        objFile.Close
End If
Set objFile = Nothing
Set objFSO = Nothing
```

```
End If
%>
```

II. Xây dựng ứng dụng Directory List

Trong ứng dụng này ta sẽ duyệt một thư mục và hiện lên browser dưới dạng một bảng biểu, ví dụ như sau :



Contents of **./dir/**

File Name:	File Size (bytes):	Date Created:	File Type:
sample	631	3/9/2001 1:56:23 PM	File Folder
sample.asp	394	3/9/2001 1:56:23 PM	ASP auto file
sample.gif	3657	3/9/2001 1:56:23 PM	GIF Image
sample.htm	386	3/9/2001 1:56:23 PM	HTML Document
sample.jpg	11721	3/9/2001 1:56:23 PM	JPEG Image
sample.rtf	393	3/9/2001 1:56:23 PM	Rich Text Document
sample.txt	126	3/9/2001 1:56:23 PM	Text Document

Để thực hiện làm ví dụ này hãy tạo trong thư mục chứa tệp ASP này một thư mục con có tên là images chứa các hình ảnh làm biểu tượng :

STT	Kiểu file	Tên tệp icon
1	asp	dir_asp
2	dir	dir_dir
3	gif, jpg	dir_img
4	htm, html	dir_htm
5	txt	dir_txt
6	các kiểu khác	dir_misc

Nhập đoạn code sau :

```
<%
Function ShowImageForType(strName)
Dim strTemp
strTemp = strName
If strTemp <> "dir" Then
        strTemp = LCase(Right(strTemp, Len(strTemp) - InStrRev(strTemp, ".", -1, 1)))
```

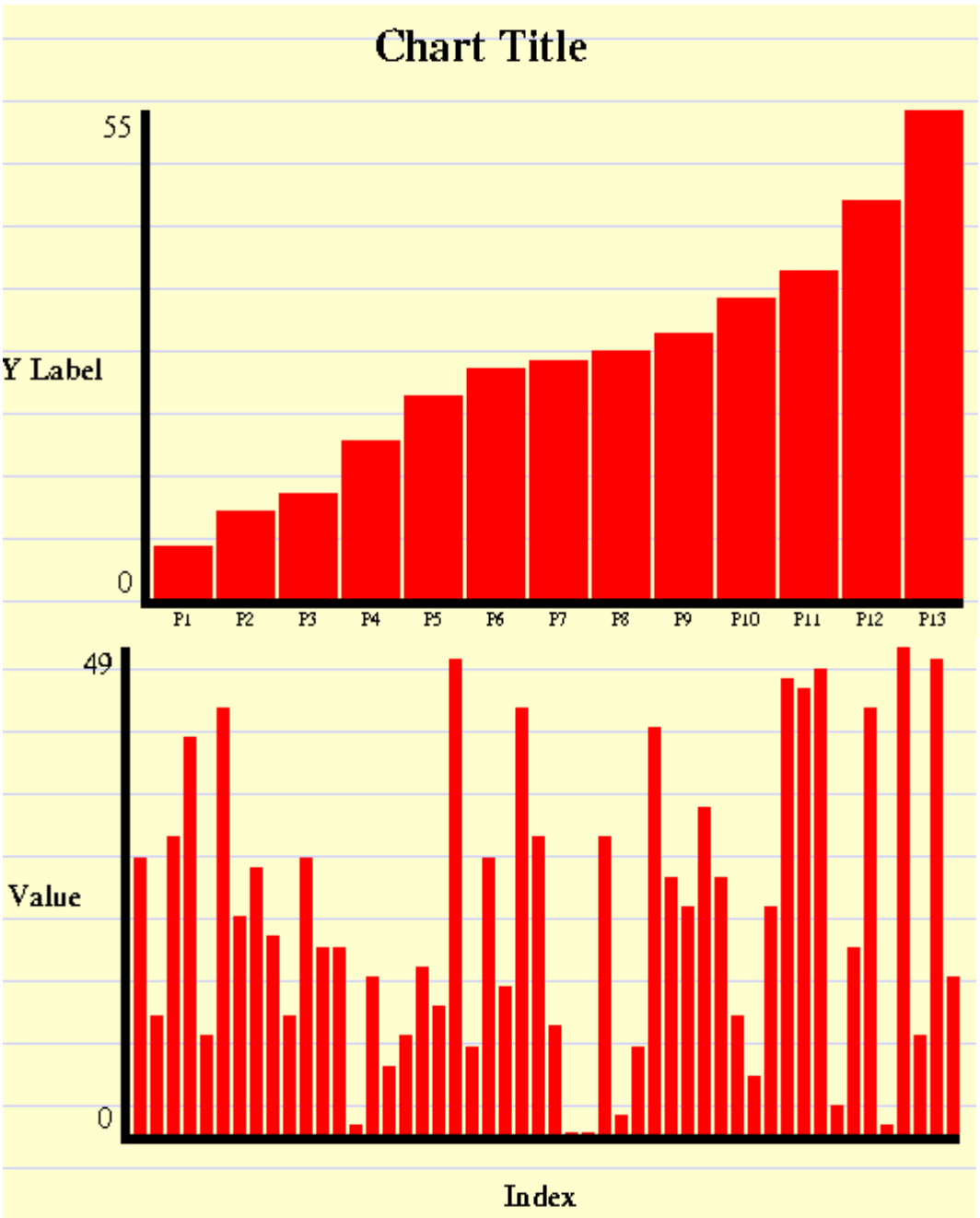
```
        End If
        Select Case strTemp
                Case "asp"
                        strTemp = "asp"
                Case "dir"
                        strTemp = "dir"
                Case "htm", "html"
                        strTemp = "htm"
                Case "gif", "jpg"
                        strTemp = "img"
                Case "txt"
                        strTemp = "txt"
                Case Else
                        strTemp = "misc"
        End Select
        strTemp = "<IMG SRC=""./images/dir_" & strTemp & ".gif"" WIDTH=16
HEIGHT=16 BORDER=0>"
        ShowImageForType = strTemp
        End Function
        %>
        <%
        Dim strPath
        Dim objFSO
        Dim objFolder
        Dim objItem

        strPath = "./dir/" ‘ đặt đường dẫn thư mục cần xem ở đây
        ‘ Sử dụng đối tượng FileSystemObject
        Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
        Set objFolder = objFSO.GetFolder(Server.MapPath(strPath))
        %>
        Contents of <B><%= strPath %></B><BR>
        <BR>
```

```
<TABLE BORDER="5" BORDERCOLOR="green" CELLSPACING="0" CELLPADDING="2">
<TR BGCOLOR="#006600">
<TD><FONT COLOR="#FFFFFF"><B>File Name:</B></FONT></TD>
<TD><FONT COLOR="#FFFFFF"><B>File Size (bytes):</B></FONT></TD>
<TD><FONT COLOR="#FFFFFF"><B>Date Created:</B></FONT></TD>
<TD><FONT COLOR="#FFFFFF"><B>File Type:</B></FONT></TD>
</TR>
<%
For Each objItem In objFolder.SubFolders
If InStr(1, objItem, "_vti", 1) = 0 Then
%>
<TR BGCOLOR="#CCFFCC">
<TD ALIGN="left" ><%= ShowImageForType("dir") %> <A HREF="<%= strPath & objItem.Name %>"><%= objItem.Name %></A></TD>
<TD ALIGN="right"><%= objItem.Size %></TD>
<TD ALIGN="left" ><%= objItem.DateCreated %></TD>
<TD ALIGN="left" ><%= objItem.Type %></TD>
</TR>
<%
End If
Next
For Each objItem In objFolder.Files
%>
<TR BGCOLOR="#CCFFCC">
<TD ALIGN="left" ><%= ShowImageForType(objItem.Name) %> <A HREF="<%= strPath & objItem.Name %>"><%= objItem.Name %></A></TD>
<TD ALIGN="right"><%= objItem.Size %></TD>
<TD ALIGN="left" ><%= objItem.DateCreated %></TD>
<TD ALIGN="left" ><%= objItem.Type %></TD>
```

```
</TR>
<%
Next
Set objItem = Nothing
Set objFolder = Nothing
Set objFSO = Nothing
%>
</TABLE>
```

IV.Xây dựng ứng dụng BAR CHART

Ta xây dựng ứng dụng này để mô tả cách xây dựng một bar chart (biểu đồ cột) trong ASP. Tất nhiên có rất nhiều component giúp cho việc tạo các chart. Sau khi duyệt biểu đồ sản sinh ra có dạng như sau



spacer.gif : dùng ngăn cách giữa các cột.

spacer_black.gif : dùng để cho việc vẽ hai trục .

spacer_red.gif : dùng vẽ cột (ví dụ trên có màu đỏ).

```
<%
```

```
Sub ShowChart(ByRef aValues, ByRef aLabels, ByRef strTitle, ByRef strXAxisLabel, ByRef strYAxisLabel)
Const GRAPH_WIDTH  = 450
Const GRAPH_HEIGHT = 250
Const GRAPH_BORDER = 5
Const GRAPH_SPACER = 2
Const TABLE_BORDER = 0
Dim I
Dim iMaxValue
Dim iBarWidth
Dim iBarHeight
iMaxValue = 0
For I = 0 To UBound(aValues)
    If iMaxValue < aValues(I) Then iMaxValue = aValues(I)
Next
iBarWidth = (GRAPH_WIDTH \ (UBound(aValues) + 1)) - GRAPH_SPACER
%>
<TABLE BORDER="<%= TABLE_BORDER %>" CELLSPACING="0" CELLPADDING="0">
<TR>
<TD COLSPAN="3" ALIGN="center"><H2><%= strTitle %></H2></TD>
</TR>
<TR>
<TD VALIGN="center"><B><%= strYAxisLabel %></B></TD>
<TD VALIGN="top">
<TABLE BORDER="<%= TABLE_BORDER %>" CELLSPACING="0" CELLPADDING="0">
<TR>
<TD ROWSPAN="2"><IMG SRC="./images/spacer.gif" BORDER="0" WIDTH="1" HEIGHT="<%= GRAPH_HEIGHT %>"></TD>
<TD VALIGN="top" ALIGN="right"><%= iMaxValue %> </TD>
</TR>
<TR>
```

```
<TD VALIGN="bottom" ALIGN="right">0 </TD>
</TR>
</TABLE>
</TD>
<TD>
<TABLE BORDER="<%= TABLE_BORDER %>" CELLSPACING="0"
CELLPADDING="0">
<TR>
<TD VALIGN="bottom"><IMG SRC="./images/spacer_black.gif"
BORDER="0" WIDTH="<%= GRAPH_BORDER %>" HEIGHT="<%=
GRAPH_HEIGHT %>"></TD>
<%
For I = 0 To UBound(aValues)
iBarHeight = Int((aValues(I) / iMaxValue) * GRAPH_HEIGHT)
If iBarHeight = 0 Then iBarHeight = 1
%>
<TD VALIGN="bottom"><IMG SRC="./images/spacer.gif"
BORDER="0" WIDTH="<%= GRAPH_SPACER %>" HEIGHT="1"></TD>
<TD VALIGN="bottom"><IMG SRC="./images/spacer_red.gif"
BORDER="0" WIDTH="<%= iBarWidth %>" HEIGHT="<%= iBarHeight %>"
ALT="<%= aValues(I) %>"></A></TD>
<%
Next
%>
</TR>
<TR>
<TD COLSPAN="<%= (2 * (UBound(aValues) + 1)) + 1 %>"><IMG
SRC="./images/spacer_black.gif" BORDER="0" WIDTH="<%=
GRAPH_BORDER + ((UBound(aValues) + 1) * (iBarWidth +
GRAPH_SPACER)) %>" HEIGHT="<%= GRAPH_BORDER %>"></TD>
</TR>
<% If IsArray(aLabels) Then %>
<TR>
<TD></TD>
<% For I = 0 To UBound(aValues)  %>
<TD></TD>
```

```
        <TD ALIGN="center"><FONT SIZE="1"><%= aLabels(I)
%></FONT></TD>
        <% Next %>
        </TR>
        <% End If %>
        </TABLE>
        </TD>
        </TR>
        <TR>
        <TD COLSPAN="2"></TD>
        <TD ALIGN="center"><BR><B><%= strXAxisLabel %></B></TD>
        </TR>
        </TABLE>
        <%
        End Sub
        %>
        <%
        ShowChart Array(6, 10, 12, 18, 23, 26, 27, 28, 30, 34, 37, 45, 55),
Array("P1", "P2", "P3", "P4", "P5", "P6", "P7", "P8", "P9", "P10", "P11", "P12",
"P13"), "Chart Title", "X Label", "Y Label"
        Response.Write "<BR>" & vbCrLf
        Response.Write "<BR>" & vbCrLf
        Response.Write "<BR>" & vbCrLf
        Dim I
        Dim aTemp(49)
        Randomize
        For I = 0 to 49
        aTemp(I) = Int((50 + 1) * Rnd)
        Next
        ShowChart aTemp, "Note that this isn't an Array!", "Chart of 50 Random
Numbers", "Index", "Value"
        %>
```

The process of uploading an image involves:

1. Verify if the file was uploaded (that is, verify that someone isn't trying to upload a file already on the server). This could be done using:

```
is_uploaded_file( $_FILES[ 'field' ]['tmp_name'] )
```

2. Verify the extension of the file for that of an image.
3. Check that the type of upload is that of an image. This could be done using:

```
private $uploadTypes = array( 'image/gif', 'image/jpg',
    'image/jpeg', 'image/pjpeg', 'image/png' );
if( in_array( $_FILES[ 'field' ]['type'], $this->uploadTypes ) )
```

4. Move the uploaded file, using:

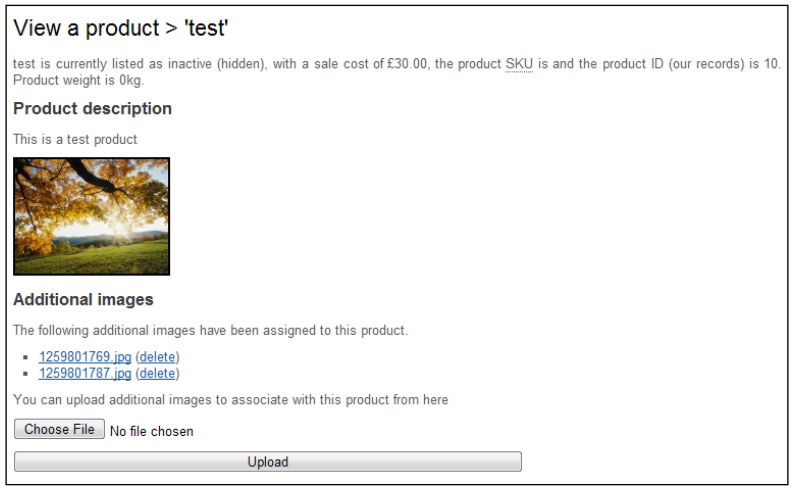
```
move_uploaded_file( $_FILES[ 'field']['tmp_name'] , $path );
```

5. Resize the image. The functions used for this vary depending on the type of the image—we do this by scaling the height based on a defined width.

```
$new = imagecreatetruecolor($x, $y);
imagecopyresampled($new, $theimage, 0, 0, 0, 0, $x, $y,
    $newwidth, $newheight);
imagejpeg( $new, $location, $quality);
```

Additional photographs

Although the product has a primary photograph associated with it, which we have just discussed, often products need to have a number of photographs to fully show off the product to the customer. The product details on the edit page could be shown along with the primary image, a list of additional images, and the additional image upload form as follows:

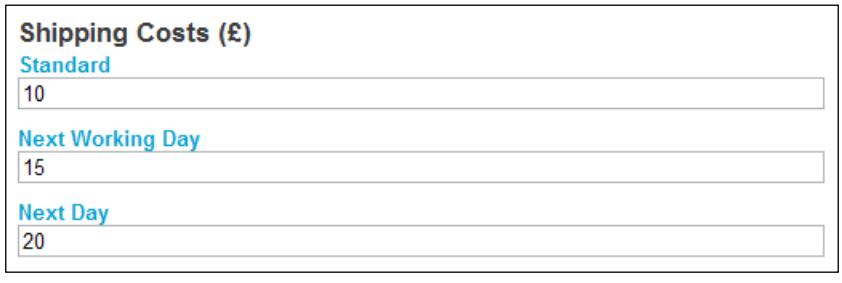


Shipping costs

When creating the product, we must save a shipping cost to be associated with each shipping method in the framework. This stage actually needs to wait until we have created the product record in the database, as we would:

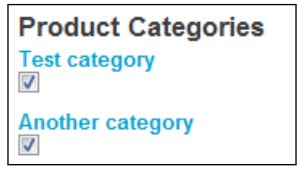
1. Take note of the product ID.
2. Query the shipping methods in the framework and store the results in an array.
3. Iterate through the array of shipping methods:
    - Lookup the value of a shipping cost field, which is suffixed with the ID of the shipping method.
    - Store the shipping cost in the shipping costs table, referencing the product ID, the shipping method ID, and the shipping cost.

As illustrated below, the shipping methods should be listed once for each product, with their corresponding default price:



Categories

This is something else which has to wait until the product has already been created; for each category checkbox that has been checked, we create an associated record in the database table, which relates products to categories.



Customizable products

Finally, as some of our products can be customized by the customer before they place their order, we need to take some options into account; these options include:

- Variations of the product, for example sizes and colors
- If the customer can upload a file
- If the customer can enter any free text, and if so, what the free text fields should be called, and how many of them there should be

Editing a product

Editing a product should be very similar to creating a product. We need to take all of the same aspects into account, and update the relevant database records accordingly, and where appropriate, upload new images.

When editing a product, we would want all of the product information to be pre-populated in the edit form. This includes pre-checked boxes indicating which categories the product belongs to, and textboxes for each shipping method.

Save existing or new variant

One very useful timesaver would be to allow the administrator to create a new product based on an existing product. To facilitate this, we could have an option when saving changes to a product to either save the changes to the existing product, or to create a new variant of the product—in which case, we would then need to actually create a new product from the submitted data.

Categories

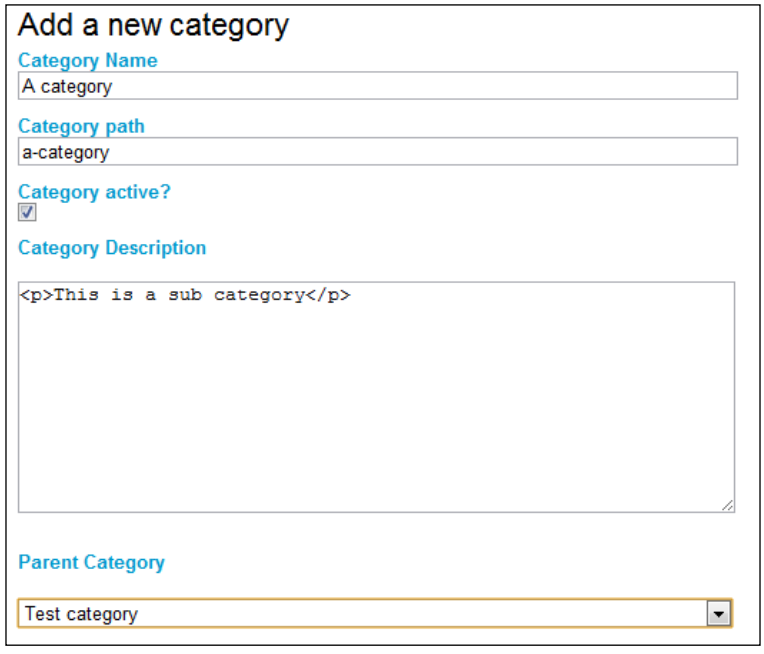
As we associate products with categories and our customers can browse these categories, we also need to be able to administer categories from the administration area.

Creating a category

Creating a category is just a case of:

- Storing the name of the category
- Storing the parent category
- Storing the order the category should display within a hierarchy of other categories
- Generating the search engine-friendly URL for the category

The following form captures this data for a new category:



Editing a category

Again, editing a category is very simple. It is just a case of updating the same details we stored when creating the category.

Deleting a category

Deleting a category requires two stages: first we need to delete the category from the database, and next we need to delete all product category associations with this category.

Orders and customers

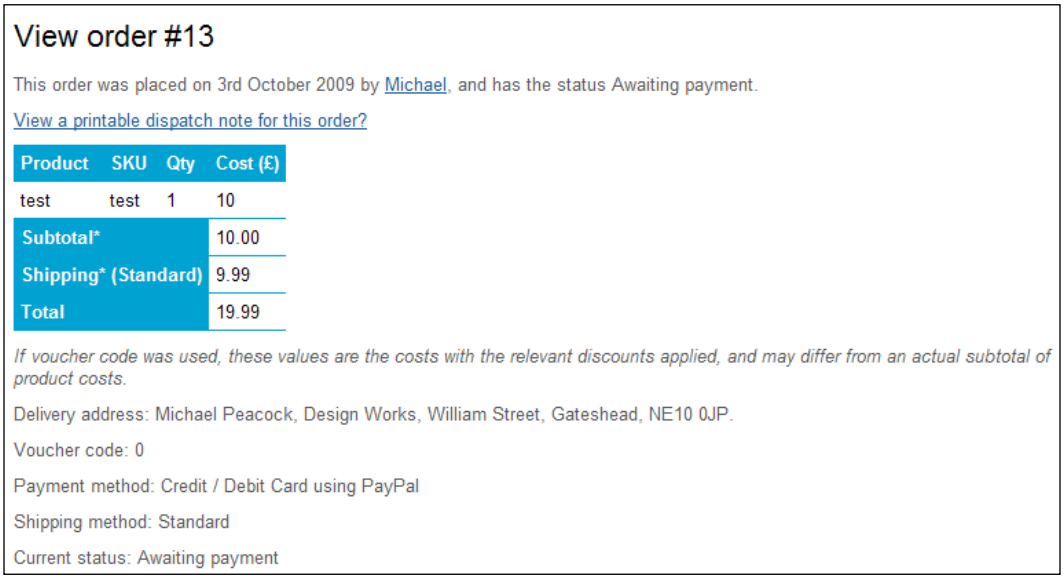
Now that our administration area can create and manage products, we need to be able to view orders and customers so that we can actually fulfill orders.

Orders

First, let's look at orders; we need to be able to:

- View an order
- Update (that is, process) an order, and inform the customer
- Print a dispatch note
- Process refunds where appropriate

The following screenshot illustrates viewing an order, displaying the status, the date it was placed, customer, products, delivery details, payment details, and shipping details:



Updating an order

When we update an order, we cannot rely on the customer checking their user account area to see that the status of the order has changed. Instead, we should e-mail the customer automatically to inform them of the change in order status.

When dispatching orders, we may often want to inform the customer of a tracking code, so they can contact the courier to see where their order is, and to get a delivery estimate. To accommodate this, the order update area should have a drop-down list of all of the possible order statuses and a free text area for the administrator to enter some text.