

Lời nói đầu

Ngày nay với sự phát triển mạnh của ngành Công Nghệ Thông Tin, Internet và Web trở nên không thể thiếu trong cuộc sống. Vì thế, xây dựng các ứng dụng Web là cần thiết. Giáo trình **Thiết kế và lập trình Web** hướng dẫn cách xây dựng một ứng dụng Web từ cơ bản đến nâng cao bằng công nghệ **HTML** và **ASP**. Được biên soạn với phương châm đảm bảo tính khoa học, thiết thực, dễ hiểu nhằm giúp sinh viên ngành công nghệ thông tin có được tài liệu hữu ích cho việc học tập.

Tuy có nhiều cố gắng trong công tác biên soạn, nhưng do lần đầu xuất bản nên chắc chắn sẽ có nhiều khiếm khuyết. Chúng tôi rất mong nhận được các ý kiến đóng góp để hoàn thiện hơn trong các lần tái bản kế tiếp.

Nhóm biên soạn

Chương 1: Các khái niệm cơ bản

1. Mạng máy tính là gì?

Mạng máy tính là một tập các máy tính được nối với nhau bởi **đường truyền** theo một **cấu trúc** nào đó và qua đó các máy tính có thể trao đổi tin tức thông qua các giao thức truyền thông.

Đường truyền là một hệ thống các thiết bị truyền dẫn có dây hay không có dây như cáp xoắn, cáp đồng trục, cáp quang, dây điện thoại, sóng vô tuyến...

Tập các đường truyền tạo nên một cấu trúc mạng.

Mạng máy tính được phân thành 2 loại: mạng diện rộng và mạng cục bộ, việc phân loại mạng máy tính dựa vào khoảng cách địa lý.

- Mạng cục bộ (Local Area Networks) hay thường gọi là mạng LAN: là mạng được thiết kế để trao đổi thông tin giữa các máy tính trong một toà nhà, một khu nhà, một phân xưởng nhỏ.
- Mạng diện rộng (Wide Area Networks) hay còn gọi là mạng WAN: Là mạng được thiết lập để kết nối các máy tính ở những khu vực lại với nhau, ví dụ như giữa các thành phố, giữa các khu vực...

2. Internet là gì?

Internet ra đời vào giữa năm 1960. Người ta đã xây dựng Internet như một giao thức để trao đổi và chia sẻ thông tin giữa các viện nghiên cứu với nhau.

Ngày nay Internet cho phép hàng trăm triệu người trên khắp thế giới liên lạc và trao đổi thông tin với nhau thông qua tập các tập các giao thức gọi chung là TCP/IP (Transmission Control Protocol/Internet Protocol).

3. Địa chỉ IP là gì ?

IP là một địa chỉ dùng để xác định đối tượng gửi và nhận thông tin trên Internet, địa chỉ này có kích thước 32 bits (version 4), 64 bits (version 5 trở lên). Khi gửi một nội dung, thì địa chỉ **IP** của bạn sẽ được gửi cùng các gói tin nội dung đến người nhận. Khi người nhận nhận được yêu cầu từ bạn thì họ căn cứ vào địa chỉ **IP** để phản hồi thông tin lại cho bạn.

Địa chỉ **IP** gồm 2 phần: Phần địa chỉ mạng và phần địa chỉ máy. 32bits địa chỉ **IP** được chia thành 4 vùng, mỗi vùng có kích thước 1 byte (8 bits) được biểu diễn dưới dạng thập phân, thập lục phân hay nhị phân. Thông thường người ta dùng cách viết thập có dấu chấm để tách các vùng. Mục đích của địa chỉ **IP** là để định danh duy nhất cho một máy tính bất kỳ trên liên mạng.

Các máy tính trao đổi thông tin với nhau thông qua mô hình Client/Server. Mô hình Client/Server là mô hình trao đổi thông tin giữa các máy tính trong đó Server thường là máy cung cấp thông tin trong khi Client là một công cụ hay chương trình trên máy tính khác dùng để lấy thông tin từ máy Server. Tuy nhiên, máy Client cũng có thể đóng vai trò cung cấp dữ liệu cho máy Server.

Để trao đổi thông tin giữa các máy tính với nhau người ta đặt ra một số giao thức (protocol) truyền thông trên mạng, các quy định về việc trao đổi thông tin để các máy tính có thể nói chuyện với nhau thông qua mạng.

Với mức độ phổ biến của Internet ngày càng cao, số lượng người tham gia ngày càng lớn thì các giao thức truyền thông trở nên phổ biến và đa dạng. Sau đây là một số

giao thức thường gặp, cho phép người sử dụng Internet gửi/nhận thư điện tử (E-mail), tập tin (File), đọc tin và đưa tin.

4. Giao thức SMTP, POP3 (Simple Mail Transfer protocol)

Là giao thức dùng để gửi/nhận thư điện tử (E-mail) từ người dùng (user) này đến người dùng (user) khác được gửi từ user này đến user khác thông qua protocol này. Thông thường người ta dùng một e-mail client để gửi thông điệp (message), còn mail server trên internet quản lý, trả lời e-mail phúc đáp.

E-mail (electronic mail) là dịch vụ trao đổi thư điện tử trên mạng viễn thông. Nội dung thư điện tử thường được mã hóa dưới dạng mã ASCII khi gửi, tuy nhiên thư điện tử còn hỗ trợ việc trao đổi thông tin hình ảnh và âm thanh.

Để trao đổi thông tin bằng E-mail bạn cần tạo một hộp mail cho chính bạn. Một hộp mail có 3 thành phần chính sau:

- Địa chỉ hộp mail: Là định danh của hộp mail giúp xác định người gửi và người nhận. Chúng ta gửi E-mail thông qua địa chỉ này, địa chỉ mail thường có dạng tên@tênmiền, ví dụ: xuântt@yahoo.com, xuântt đóng vai trò là tên, yahoo.com là tên miền.
- Địa chỉ mail được quản lý bởi mail server thông qua các dịch vụ cung cấp mail như FPT, SaigonNet, vnEpress, Yahoo, Hotmail, vnn....tuy nhiên hiện nay có rất nhiều dịch vụ cung cấp mail miễn phí. Nhưng thông dụng nhất vẫn là 2 dịch vụ cung cấp mail Yahoo và Hotmail.
- Một hộp mail có một tên đăng nhập và một mật khẩu để truy cập hộp mail của mình. Tên đăng nhập và mật khẩu được tạo khi chúng ta đăng ký hộp mail. Điều này bảo đảm tính bảo mật của hộp mail của bạn và chỉ có bạn mới biết mật khẩu cùng tên đăng nhập của bạn để vào hộp mail mà thôi.

5. Giao thức FPT (File Transfer Protocol)

Đây là một giao thức để trao đổi các tập tin trên Internet với nhau. Nguyên tắc hoạt động của nó khá đơn giản. FTP dùng để tải các tập tin (file) từ máy tính này sang máy tính khác, các tập tin (file) này có thể là các tập tin chứa văn bản, âm thanh, hình ảnh. Các máy gửi yêu cầu tập tin qua lại thông qua nhiều chương trình khác nhau. Tuy nhiên, chúng ta có thể dùng một tập hợp lệnh như “open” (để kết nối với máy server) và “get” (để tải các tập tin này về máy client) hoặc có thể chọn tập tin mong muốn từ một giao diện của chương trình có sẵn để trao đổi các tập tin giữa các máy với nhau. FTP cũng có thể dùng để tải các chương trình, tập tin giữa các máy server với nhau.

6. Giao thức HTTP (HyperText Transfer Protocol)

Là giao thức dùng để hiển thị trang web dưới dạng văn bản, hình ảnh, âm thanh, video, và các liên kết (links) đến các trang web khác trên World Wide Web. Khi chúng ta chọn các liên kết thì HTTP sẽ mở một nội dung mới thông qua trình duyệt Web cho chúng ta. Đây là giao thức nền tảng trong tập các giao thức ICP/IP.

7. Giao thức NNTP (Network News Transfer Protocol)

Là giao thức phân phối thông điệp một cách rộng rãi với nhiều chủ đề khác nhau. Thông qua một chương trình tin tức Client như Collabra của Netscape hay chương trình Internet News của Microsoft bạn có thể đọc hay đưa các bài báo cáo trong những nhóm mới.

8. Giao thức Chat

Là giao thức cho phép người dùng trao đổi thông tin trực tiếp dưới dạng văn bản, hình ảnh và âm thanh.

Các chương trình Chat thông dụng nhất hiện nay là YahooMessenger, ICQ, MIRC...

9. URL

URL (Uniform Resource Locator) là địa chỉ của một trang Web hay bất kỳ một tập tin (file) nào khác trên Internet. Mỗi URL trên Web là duy nhất.

URL có một cú pháp đặc biệt. Tất cả các URL phải chính xác, thậm chí có một ký tự sai hay thiếu một dấu chấm cũng không được chấp nhận. Một ký tự sai trong URL có thể mang đến cho bạn một trang Web hoàn toàn khác biệt hoặc kết quả được trả về là một thông báo lỗi.

Một URL đơn giản dùng cho Web bao gồm tên của Web protocol, theo sau bởi dấu hai chấm (:) và hai dấu (/), tiếp đến là tên của domain và kết thúc bởi dấu (/).

Ví dụ: <http://www.hotmail.com/>

Cấu trúc của một URL có thể bao gồm các phần sau:

- Phần thứ nhất của URL là tên giao thức. Cho đến nay tất cả các URL đều bắt đầu là http:// (Hypertext Transfer Protocol) bởi vì đó là trang Web và HTML. Nhưng URL có thể chỉ đến các phần khác của Internet chứ không phải chỉ là Web.

Sau đây là một số giao thức được dùng trong URL:

ftp:// tên giao thức (File Transfer Protocol)

http:// tên giao thức (Hypertext Transfer Protocol)

mailto: địa chỉ thư điện tử (Electronic mail address) không yêu cầu hai ký tự ‘/’.

news: Giao thức cho phép mọi người tham gia vào nhóm tin nếu trang chủ của bạn có liên quan đặc biệt đến nhóm. Hoặc cho phép nhận các câu hỏi cần trả lời từ một nhóm tin nào đó. Như vậy bạn có thể hạn chế được số lượng e-mail nhận được (không yêu cầu hai ký tự ‘/’).

telnet: Giao thức cho phép trao đổi thông tin trực tiếp giữa các máy tính với nhau bằng cách yêu cầu người sử dụng nhập tên truy cập và mật khẩu (không yêu cầu hai ký tự ‘/’).

file:// Tài liệu mà bạn đang truy xuất được đặt trên máy tính của bạn thay vì trên Web.

- Phần thứ hai của URL là tên của domain. Domain đại diện cho tên của server mà bạn đang kết nối.
- Phần thứ ba của URL là đường dẫn đến tập tin cần truy cập, thường bắt đầu bằng tên tài khoản (account name). Nếu bạn muốn truy xuất đến trang Web được tạo bởi một người dùng (user) xác định thì phải thêm tên của tài khoản vào URL. Tên của tài khoản luôn bắt đầu bằng ký tự (~).

Ví dụ: <http://www.sonic.net/~japan>

- Phần thứ tư của URL là tên đường dẫn và tên tập tin (pathname và filename). URL thường chỉ ra tên đường dẫn và tên tập tin cụ thể.

Đây là một miền xác định trên máy chủ đang chứa tập tin cần truy xuất.

- Phần thứ năm của URL là tên của anchor (anchor name). Đây là một con trỏ chỉ đến một phần xác định của một file HTML. Nó luôn bắt đầu bằng ký tự (#). Anchor đặc biệt rất hữu dụng đối với các tài liệu lớn.

Tuy nhiên tên account, tên đường dẫn, tên file và tên anchor không phải là thành phần bắt buộc đối với một URL.

Ví dụ một URL đầy đủ:

<http://www.zdnet.com/~zdi/software/wind95/utls.html#WINZIP>

Trong đó:

http://	: Tên giao thức (Protocol)
www.zdnet.com	: Tên miền (domain)
~zdi	: Tên tài khoản (account)
software/win95	: Đường dẫn thư mục
utls.html	: Tên tập tin (file)
#WINZIP	: Tên neo (anchor)

10. Hyperlink là gì ?

Hyperlinks (hay còn gọi là links) rất cần thiết đối với World Wide Web. Dùng hyperlinks để liên kết từ tài liệu này đến tài liệu khác là một hoạt động rất phổ biến trên web.

Bạn có thể tạo links đối với bất kỳ đối tượng nào trên web. Links có thể kết nối trực tiếp với văn bản, hình ảnh, âm thanh hay một file HTML khác. Hyperlink thường được hiển thị với một màu sắc mặc định. Tuy nhiên chúng ta có thể thay đổi màu sắc nếu cần. Chỉ cần nhấn chuột vào hyperlinks, tài liệu được link sẽ được hiển thị. Thông thường, hình dạng con trỏ sẽ thay đổi thành hình bàn tay khi rê chuột qua vùng hyperlinks vì thế chúng ta sẽ biết được hyperlinks đang ở đâu để có thể nhấp chuột.

Hyperlinks giúp chúng ta dễ dàng tìm kiếm các thông tin khác nhau về một chủ đề. Nếu bạn đang tìm kiếm một chủ đề nào đó, chỉ cần vào trang web có hyperlinks kết nối với vấn đề này, và chúng ta sẽ nhận được một danh sách liệt kê hàng trăm chủ thể có liên quan.

11. Web Browser là gì?

Web browser là một công cụ hay chương trình cho phép bạn truy xuất và xem thông tin trên web. Có nhiều web browser để truy xuất web. Mỗi web browser có những đặc điểm khác nhau và chúng sẽ hiển thị những trang web không hoàn toàn giống nhau. Hai web browser đang phổ biến hiện nay là *Netscape's Navigator* của công ty Netscape và *Microsoft's Internet Explorer* của công ty Microsoft. Chúng hiển thị được cả hình ảnh, âm thanh và rất dễ sử dụng. Ngoài việc truy xuất web, Netscape và IE (*Internet Explorer*) còn cho phép chúng ta thực hiện một số công việc khác nhau như gửi và nhận thư điện tử (e-mail), tải các tập tin (download file) từ máy chủ (server),.... Với hai web browser này web đã trở thành một hệ thống thông tin đa phương tiện có mối liên hệ với nhau.

12. Web Server là gì?

Web server đơn giản là một máy tính nối vào Internet và chạy các phần mềm được thiết kế để truyền tải nội dung dưới dạng trang HTML (Trang HTML có thể chứa

âm thanh, hình ảnh, video, văn bản) hay dưới dạng tập tin (tập tin có thể là tập tin hình ảnh, âm thanh, văn bản, video...). Máy chủ (server) phải đủ mạnh để đáp ứng nhiều kết nối Internet đồng thời.

Thông qua trình duyệt Web máy chủ sẽ cung cấp các dịch vụ được yêu cầu đến máy client.

Thông thường web server chạy trên các hệ điều hành khá mạnh như Unix, Linux, Microsoft Windows NT Server, Windows 2000 server.

13. Website là gì?

Là tập các trang web liên quan đến một công ty, một tập đoàn, một trung tâm hay một cá nhân nào đó. **Ví dụ:** Website của Trung Tâm Phát Triển Công Nghệ Thông Tin Thành Phố Hồ Chí Minh gồm tập các trang web được bắt đầu từ trang chủ có địa chỉ (URL) là www.citd.edu.vn.

14. World Wide Web (www) là gì?

World Wide Web là dịch vụ thông dụng ra đời vào năm 1990. Dịch vụ World Wide Web sử dụng giao thức HTTP (HyperText Transfer Protocol). Để sử dụng dịch vụ này chúng ta cần một trình duyệt Web (gọi là Web Browser).

Để truy cập một trang Web bạn cần phải biết địa chỉ (URL-Uniform Resource Locator) của trang web đó. Ví dụ nếu bạn muốn truy cập vào trang chủ của Yahoo thì bạn đánh địa chỉ <http://www.yahoo.com> cho trình duyệt web. Khi đó trình duyệt web sẽ mở trang chủ Yahoo cho bạn. Từ trang chủ này bạn có thể đi đến các trang khác bằng cách nhấn chuột vào đối tượng HyperLink trong trang web. Hình dạng con chuột thay đổi (thông thường là hình bàn tay) khi đi qua các đối tượng Hyperlink này.

Trang Web truyền tải nhiều thông tin khác nhau với nhiều hình thức khác nhau như văn bản, âm thanh, hình ảnh... Sự ra đời của trang web giúp con người trao đổi nhanh hơn, tiện lợi hơn, phong phú hơn và đa dạng về cả hình thức và nội dung.

World Wide Web (cũng được gọi là 'W3', 'WWW' hay gọi tắt là Web) là một hệ thống rộng lớn bao gồm nhiều HTTP server (máy chủ sử dụng giao thức HTTP - Hypertext Transfer Protocol), chúng đang thực hiện việc trao đổi file thông qua Internet.

15. Sự khác biệt giữa Internet và World Wide Web?

Nhiều người nghĩ rằng Web và Internet là một. Nhưng Web chỉ là một phần của Internet và nó đang phát triển với tốc độ nhanh hơn các phần khác.

Trên Internet chúng ta có thể gửi và nhận nhiều loại thông tin khác nhau như thư điện tử (e-mail), các bài báo, tán gẫu và các trang Web. Như vậy Web chỉ là một trong những dịch vụ của Internet.

16. Web page là gì?

Web page là trang web, là một loại tập tin đặc biệt được viết bằng ngôn ngữ siêu văn bản HTML. Web page có thể hiển thị các thông tin văn bản, âm thanh, hình ảnh... Trang Web này được đặt trên máy server sao cho máy client có thể truy cập được nó. Chúng ta cũng có thể đặt tập tin này trên ổ cứng máy tính của mình nhưng người khác sẽ không đọc được nó.

Chương 2 : lập trình web với ngôn ngữ siêu văn bản html

1. Khái niệm ngôn ngữ HTML

HTML (Hypertext Markup Language) là ngôn ngữ định dạng văn bản siêu liên kết, là ngôn ngữ lập trình Web căn bản, cho phép định dạng văn bản, bổ sung hình ảnh, âm thanh và video, cũng như lưu tất cả vào một tập tin dưới dạng văn bản hay dưới dạng mã ASCII, binary mà bất cứ máy tính nào cũng có thể đọc được thông qua trình duyệt Web (Web browser). HTML có hai đặc tính cơ bản sau:

- **Siêu văn bản:** Tạo các liên kết trong trang Web, cho phép truy cập thông tin từ nhiều nguồn khác nhau thông qua các Hyperlinks.
- **Tính tổng quát:** Hầu như máy tính nào cũng có thể thực thi các lệnh ngôn ngữ HTML. Điều này là do dữ liệu trong tập tin HTML chỉ là dữ liệu văn bản hay dữ liệu mã ASCII.

2. lập trình web với ngôn ngữ HTML

2.1 Các thành phần cơ bản của HTML

□ Tag là gì?

Tag là một tập các ký hiệu đặc trưng trong HTML có ý nghĩa cụ thể. Tag bắt đầu bằng ký hiệu nháy nhúng (<) theo sau bởi một từ khoá, và kết thúc bằng ký hiệu nháy nhúng (>). Nó quy định văn bản sẽ hiển thị trên màn hình như thế nào.

Ví dụ:

- + Tag định phong cách cho văn bản là **chậm**.
- + Tag<I> định phong cách nghiêng.
- + Tag<TITLE> tiêu đề cho trang web.
- + Tag<HTML> bắt đầu một trang web.
- + Tag<P> bắt đầu một đoạn văn bản.

Mọi tag trong HTML có một ý nghĩa riêng, nó khá rõ ràng và dễ hiểu. Tag không phân biệt chữ hoa chữ thường vì thế các tag <Title>, <title>, <TITLE> và <titLe> đều có nghĩa như nhau.

Có hai loại tag: Tag bắt đầu và tag kết thúc. Dữ liệu bắt đầu sẽ nằm giữa tag bắt đầu và tag kết thúc. Tag kết thúc giống tag bắt đầu nhưng có thêm dấu xuy t phi (/) phía trước.

Ví dụ: <TITLE> Lập trình web với HTML</TITLE>

Trong đó:

<TITLE> : Tag bắt đầu.

Lập trình web với HTML: chuỗi dữ liệu bắt đầu.

</TITLE> : Tag kết thúc.

• Thuộc tính

Có nhiều tag còn có thuộc tính kèm theo. Thuộc tính sẽ nhập vào ngay trước dấu ngoặc đóng (>) của tag. Có thể sử dụng nhiều thuộc tính trong một tag. Thuộc tính này khác biệt với thuộc tính khác, phân cách nhau bởi khoảng trống.

Ví dụ: <TABLE BORDER>

Trong đó: < TABLE> : Tên tag

<BORDER> : Thuộc tính

• Giá trị

Ngoài các thuộc tính không có giá trị còn có các thuộc tính có tag có giá trị. Ví dụ: thuộc tính **CLEAR** của tag **
** có ba giá trị chỉ định là: left, right, all.

Ví dụ: <BR CLEAR = "left">

Trong đó: **
**: tên tag
CLEAR: thuộc tính
left: giá trị của thuộc tính CLEAR.

Giá trị nên đặt giữa hai dấu nháy "". Tuy nhiên có thể bỏ dấu nháy trong các trường hợp giá trị chỉ định là chữ (A -> Z, a -> z), số (0 -> 9), dấu gạch ngang (-), dấu chấm (.).

- **Tag lồng nhau**

Thường lồng nhau dùng để nhấn mạnh cách trình bày nội dung trong một trang web. Ví dụ chúng ta muốn trình bày chương nghiêng trong một vài chương quan trọng trong tiêu

Không phải tag nào cũng chấp nhận lồng nhau.

Trình soạn thảo phải chấp nhận tag lồng nhau. Thứ tự ưu tiên sẽ là theo thứ tự sau cùng.

Ví dụ: Dòng ưu tiên đúng, dòng thứ hai sai

Đúng: **<H1> Phần <I> Nội Dung </I></H1>**

Sai: **<H1> Phần <I> Nội Dung </H1></I>**

- **Khoảng trắng**

Trình duyệt bỏ qua các khoảng trống giữa các tag trong tệp tin HTML.

- **Tên tệp tin**

Nên sử dụng chữ thường khi đặt tên tệp tin, điều này giúp ích cho bộ nhớ trình duyệt. Ví dụ, sẽ giúp ích khi tìm kiếm các trang Web. Việc này dùng, sẽ dễ dàng khi truy cập trang web.

Tên tệp tin phải kết thúc bằng ".htm" hay ".html". Giúp trình duyệt Web nhận ra loại tài liệu khi duyệt.

2.2 Tạo trang Web

Có thể tạo trang Web trên bất cứ chương trình xử lý văn bản nào như chương trình soạn thảo văn bản Notepad, WordPad, Word... Đây chúng ta sẽ thấy khi tạo trang Web trên trình soạn thảo văn bản Notepad, chương trình soạn thảo có sẵn trên Windows, bằng cách vào menu **Start -> Programs -> Accessories -> Notepad**.

2.3. Cấu trúc của một tệp tin HTML

Một trang Web luôn bắt đầu bằng tag **<HTML>** và kết thúc bởi tag **</HTML>**.

Huht các trang Web được chia thành 2 phần. Phần đầu và phần thân. Phần đầu là phần chứa tag **<HEAD>** và tag **</HEAD>**. Phần thân là phần chứa tag **<BODY>** và tag **</BODY>**.

Phần đầu là nơi ghi rõ tiêu đề, nội dung của tiêu đề chứa tag **<TITLE>** và **</TITLE>**.

Cú pháp: <HEAD> <TITLE> tên tiêu đề của trang Web

<TITLE> </HEAD>

Huht các trình duyệt Web, tiêu đề xuất hiện trên thân tiêu đề của trình duyệt.

Phần thân chứa nội dung của trang Web là phần chính của trang Web, phần cung cấp thông tin cần thiết cho trình duyệt Web.

Cú pháp: <BODY> Nội dung trang Web </BODY>

Ví dụ: Tạo một trang web đầu tiên, mở trình soạn thảo văn bản NodePad và gõ nội dung HTML như trong hình sau:

2.4. Xem trang HTML bằng trình duyệt Web

Sau khi thiết kế trang Web xong bạn dùng một trình duyệt Web để xem kết quả thiết kế. Vì không biết nên dùng trình duyệt Web nào. Nên tốt nhất nên xem nó trong nhiều trình duyệt Web khác nhau theo các bước sau.

Khi mở trình duyệt Web **Internet Explorer** -> Nhấp **File** -> **Open** -> Trong hộp thoại **Open** bạn gõ địa chỉ tệp tin rồi nhấp **OK** như hình sau:

Hoặc nhấp vào nút **Browser..** để chọn tệp tin trong hộp thoại vừa mở ra, nhấp tiếp nút **Open** trong hộp thoại thì trình duyệt Web sẽ mở trình duyệt. Khi này nội dung của trang Web trong ví dụ trên sẽ hiển thị như sau:

2.5. Các tag cơ bản trong HTML

2.5.1. Tag chú giải

Tag này sẽ thêm vào tài liệu HTML những nhận xét để giúp ích cho các dòng lệnh. Trình duyệt không tính nội dung những tag ghi chú này.

Cú pháp: <!-- nội dung chú giải -->

2.5.2 Các tag định dạng văn bản

- **Định phong chữ:** tag **FONT** để định dạng văn bản hiển thị.

Cú pháp:

Trong đó *fontname1* là phong chữ mặc định, gõ tên của phong chữ (m (Bold), nghiêng (Italic), gạch dưới (Underline)). Ta có thể bổ sung thêm phong chữ *fontname2* phòng khi trình duyệt không cài loại phong chữ *fontname1*, các phong chữ sẽ viết cách nhau dấu phẩy.

Ví dụ: Ta thêm 3 dòng như sau vào ví dụ trên

Như vậy đoạn lệnh được viết lại như sau:

Mã HTML

```
<HTML>
<HEAD><TITLE>To chuc</TITLE></HEAD>
<BODY>
<FONT FACE = "VNI-Litthos, VNI-Times", bold>
<H1>Giám đốc</H1></FONT>
<FONT FACE = "VNI-Times", bold>
<H2>Phó giám đốc</H2></FONT>
<FONT FACE = "VNI-Centur, VNI-Times", bold>
<H3>Nhân viên</H3>
</FONT>
</BODY>
</HTML>
```

Kết quả hiển thị trên trình duyệt
Giám đốc Phó giám đốc Nhân viên

- **Định kích thước chữ**

Cú pháp: <BASEFONT SIZE = “n”>

Với n mang giá trị từ 1 đến 7, giá trị mặc định là 3. Tag **BASEFONT** dùng để định kích thước cho toàn bộ văn bản, nếu muốn thay đổi một đoạn hay một từ trong văn bản thì dùng tag **FONT**, tag **BIG**, tag **SMALL**.

- **Định màu cho văn bản**

Cú pháp:

Với *color* là màu dùng cho chữ, các giá trị màu có thể gõ bằng chữ hoặc gõ bằng chữ số ở hệ hexa (hệ 16).

Nếu gõ ở hệ hexa thì thành phần *color* là sự kết hợp thứ tự giữa 3 giá trị màu tương ứng là Red (đỏ), Green (xanh lục), Blue (xanh đậm), mỗi giá trị màu được biểu diễn bởi 2 chữ số thập phân.

Ví dụ: Ta muốn màu chữ là màu đỏ thì ta gõ “Red” hoặc “FF0000”, nếu màu vàng thì Yellow hoặc “FFFF00”, màu xanh đậm thì “Green” hoặc “0000FF”...

Như vậy tag **FONT** không những dùng để định dạng kích thước văn bản mà còn định dạng màu cho văn bản.

Ví dụ: Ta muốn định màu xanh đậm cho đoạn nội dung “Lop chuyen vien” nên ta bỏ chúng vào 2 tag <COLOR> và </COLOR>

Mã HTML
<pre> <HTML> <HEAD><TITLE>To chuc</TITLE></HEAD> <BODY> <H1>Dao tao tu xa qua mang</H1> <BASEFONT SIZE = “4”> Lop chuyen vien cong nghe thong tin </BODY> </HTML> </pre>
Kết quả hiển thị trên trình duyệt
Dao tao tu xa qua mang Lop chuyen vien cong nghe thong tin

- **Định dạng chữ**

Cú pháp:

nội dung hoặc nội dung

<I>nội dung</I> hoặc

Ta định dạng chữ đậm chữ nghiêng cho văn bản bằng cách sử dụng 2 tag:

Định dạng chữ đậm dùng tag hoặc tag

Định dạng chữ nghiêng dùng tag <I> hoặc tag

Ví dụ: Từ ví dụ trên muốn phần nội dung “lop chuyen vien” được in đậm còn phần nội dung “cong nghe thong tin” được in nghiêng thì 2 nội dung trên được bỏ vào 2 tag như sau:

Lop chuyen vien

<I>cong nghe thong tin</I>

Ghi chú:

- Ta có thể kết hợp các thuộc tính vào chung một tag

Ví dụ: Ta kết hợp cùng lúc 3 thuộc tính định dạng phông chữ là **FACE** (kiểu chữ hiển thị), **SIZE** (kích thước văn bản), **COLOR** (màu của văn bản) vào tag như đoạn tag sau:

nội dung văn bản

Ví dụ: Xét ví dụ sau

Mã HTML
<pre><HTML> <HEAD><TITLE>Thong tin</TITLE></HEAD> <BODY> <H1>Lop chuyen vien</H1> chào các bạn đến với chương trình đào tạo chuyên viên </BODY> </HTML></pre>
Kết quả hiển thị trên màn hình
<p>Lop chuyen vien chào các bạn đến với chương trình đào tạo chuyên viên</p>

- Sử dụng tag <BIG> và tag <SMALL> để thay đổi từng phần hoặc cả thể trong nội dung văn bản. tag **BIG** dùng để thay đổi phần nội dung thành chữ lớn, tag **SMALL** dùng để thay đổi phần nội dung thành chữ thường.

Cú pháp: <BIG>...</BIG> và <SMALL>...</SMALL>

- Định dạng chỉ số trên, chỉ số dưới**

Chữ (hoặc số) nằm hơi cao hoặc hơi thấp hơn so với phần văn bản chính được gọi là chỉ số trên và chỉ số dưới.

Để định dạng chỉ số trên hay chỉ số dưới ta dùng hai tag **SUP** (định dạng chỉ số trên), **SUB** (định dạng chỉ số dưới).

Cú pháp: *Chỉ số trên:*
Chỉ số dưới:

Ví dụ: Ta muốn hiển thị phương trình bậc hai

$AX^2 + BX + C = 0$, $C + O_2 = CO_2$ lên Web thì ta dùng đoạn lệnh sau:

Mã HTML
<pre><HTML> <HEAD><TITLE>đinh dang chi so</TITLE></HEAD> AX<SUP>2</SUP>=BX+C=0
</pre>

```
C+O<SUB>2</SUB>=CO<SUB>2</SUB>
</FONT>
</HTML>
```

Kết quả hiển thị trên trình duyệt

```
AX2+BX+C=O
C+O2=CO2
```

Trong đoạn lệnh trên ta sử dụng tag
, tag này dùng để sang dòng mới mà ta sẽ đề cập cụ thể trong các phần kế tiếp.

- **Phân đoạn**

Dùng tag <P> để phân đoạn văn bản. Muốn canh chỉnh đoạn trên trang Web ta sử dụng thêm thuộc tính **ALIGN**.

Cú pháp: <P ALIGN="direction">

Với direction mang 1 trong 3 giá trị sau:

- Left (nội dung trong đoạn được canh trái)
- Right (nội dung trong đoạn được canh phải)
- Center (nội dung trong đoạn được canh giữa)

Ví dụ: Muốn phân hai dòng $AX^2 + BX + C = 0$ và $C + O_2 = CO_2$ thành hai đoạn riêng biệt, nội dung đoạn thứ nhất nằm ở giữa màn hình, đoạn thứ hai nằm bên trái màn hình thì ta viết lại đoạn lệnh như sau:

```
<HTML>
<HEAD><TITLE>định dạng chỉ số</TITLE></HEAD>
<FONT SIZE = "3">
<P ALIGN = "center">
AX<SUP>2</SUP>=BX+C=0
<P ALIGN = "left">
C+O<SUB>2</SUB>=CO<SUB>2</SUB>
</FONT>
</HTML>
```

- **Tag phân cấp đề mục**

Đề mục trong trang Web giúp cho người duyệt hiểu rõ hơn cấu trúc của trang.

Cú pháp: <Hn></Hn> với n mang giá trị từ 1 -> 6

Trong HTML cho phép sử dụng 6 cấp đề mục trong trang Web, tuy nhiên trong thực tế ta chỉ sử dụng 3 đến 4 cấp đề mục là đủ.

Ví dụ: ta xem xét đoạn lệnh sau:

```
Mã HTML
<HTML>
<HEAD><TITLE>To chuc</TITLE></HEAD>
<BODY>
<!--các cấp trong một công ty-->
<H1>Giam doc</H1>
<H2>Pho giam doc</H2>
<H3>Nhan vien</H3>
</BODY>
```

</HTML>
Kết quả hiển thị trên trình duyệt
Giam doc Pho giam doc Nhan vien

2.5.3. Các tag định dạng hình ảnh

Có nhiều loại hình ảnh trên một trang Web: Logo (biểu tượng), ảnh chụp, ảnh vẽ từ các chương trình xử lý ảnh như: CorelDraw, Paint, Photoshop...

Hình ảnh được đưa vào trang Web dưới dạng tập tin hình ảnh.

Nhớ lưu hình ảnh dưới dạng tập tin có phần mở rộng “*gif*”, “*jpg*” hoặc “*png*”. Hạn chế lưu ảnh dưới dạng tập tin có phần mở rộng “*bmp*” như thế, chỉ có người dùng trình duyệt Web Internet Explorer mới xem được hình ảnh này, mặt khác kích thước của các tập tin hình lưu dưới dạng “*bmp*” lớn hơn nhiều lần so với tập tin lưu dưới dạng “*jpg*”, “*gif*”, “*png*”.

- **Chèn hình ảnh vào trang Web**

Để chèn hình ảnh vào trang Web ta dùng tag .

Cú pháp:

Tên tập tin chứa hình ảnh có cả đường dẫn thư mục, nếu tập tin hình ảnh được đặt cùng thư mục với thư mục chứa trang Web thì không cần đường dẫn thư mục.

Tạo thư mục có tên **images** trong thư mục chứa trang Web, sau đó lưu tập tin hình ảnh vào thư mục **images**. Khi đó SRC=”*images/tên tập tin*”

- **Thêm đường viền chung quanh vào hình ảnh**

Để thêm đường viền vào xung quanh hình ảnh ta dùng thêm thuộc tính **BORDER** vào tag

Cú pháp:

Với *n* là độ dày mảnh của đường biên ảnh, tính bằng pixel.

- **Canh chỉnh hình ảnh**

Để canh chỉnh hình ảnh ta dùng thuộc tính **ALIGN** trong tag với 3 giá trị: left, center và right.

Cú pháp: *nội dung*

Trong đó *direction* mang 1 trong 3 giá trị sau:

- **ALIGN = left** : Hình ảnh nằm bên trái màn hình.
- **ALIGN = center** : Hình ảnh nằm giữa màn hình.
- **ALIGN = right** : Hình ảnh nằm bên phải màn hình.

- **Thêm chữ xung quanh hình ảnh**

Muốn thêm chữ xung quanh hình ảnh ta thêm thuộc tính **ALIGN** vào tag .

Cú pháp: *nội dung* muốn chèn

Trong đó *direction* mang 1 trong 3 giá trị sau:

- **ALIGN = top** : Vị trí văn bản nằm ở phía trên màn hình.
- **ALIGN = middle** : Vị trí văn bản nằm ở giữa hình ảnh.
- **ALIGN = bottom** : Vị trí văn bản nằm ở phía bên dưới hình ảnh.

Ví dụ:

Mã HTML

```

<HTML>
<HEAD><TITLE>hien thi hinh anh</TITLE></HEAD>
<BODY>
<H1>hien thi hinh anh qua 3 cach</H1>
<H2>tren</H2>
<IMG SRC="..\002.jpg" BORDER=1 ALIGN="top">van ban o phia tren.
<H3>giua</H3>
<IMG SRC="..\002.jpg" BORDER=2 ALIGN="middle">van ban o giua.
<H4>giua</H4>
<IMG SRC="..\002.jpg" BORDER=3 ALIGN="bottom">van ban o phia duoi.
</BODY>
</HTML>

```

Kết quả hiển thị trên trình duyệt

2.5.4. Các tag định dạng trang

- **Đường kẻ ngang trang Web**

Dùng tag <HR> để kẻ đường ngang trên trang Web, giúp trang Web rõ ràng hơn.

Cú pháp : <HR SIZE ="*n*" WIDTH= "*w*"
ALIGN= "*direction*" NOSHADE>

Trong đó:

- **SIZE:** d y/m ng c a ng k tính b ng Pixel.
- **WIDTH:** Chi u r ng c a ng k tính b ng Pixel.
- **ALIGN:** Thu c tính này có ba giá tr *left, right, center* dùng ch nh v trí ng ngang.

- **Định màu nền cho trang Web**

nh màu n n cho trang Web ta thêm thu c tính **BGCOLOR** trong tag **BODY**.

Cú pháp: <BODY BGCOLOR= "*color*">
color là màu c n nh cho trang Web

Ví dụ: Ta mu n t o màu n n c a trang là màu vàng (yellow) cho Web thì ta dùng o n l nh sau:

```

<HTML>
<HEAD><TITLE>mau nen</TITLE></HEAD>
<BODY BGCOLOR="yellow">
</BODY>
</HTML>

```

- **Định ảnh nền cho trang Web**

Dùng hình nh làm n n cho toàn trang Web thay vì chúng ta s d ng màu n n. nh n n giúp trang Web c a chúng ta p h n, h p d n h n, trang Web có s c lôi cu n ng i dùng h n.

nh hình nh nên cho trang Web ta s d ng thu c tính **BACKGROUND** trong tag **BODY** thay vì dùng thu c tính **BGCOLOR** nh màu n n.

Cú pháp : **<BODY BACKGROUND="tên file">**

Tên file là tên t p tin hình nh c n làm n n cho trang Web.

Ví dụ: Mu n nh t p tin nh có tên “b.jpg” trong th m c images làm n n cho trang Web ta dùng o n l nh sau:

```
<HTML>
<HEAD><TITLE>mau nen</TITLE></HEAD>
<BODY BACKGROUND="images/b.jpg">
</BODY>
</HTML>
```

• Xuống dòng trong trang Web

Khi dùng tag **<P>** phân o n thì kho ng cách gì a các o n khá l n. tránh khuy t i m này ta dùng tag **
** xu ng dòng.

2.5.5. Các tag tạo danh sách (list)

• Tạo danh sách theo thứ tự

S d ng tag **** t o danh sách theo th t

Cú pháp: **<OL TYPE=X>**

**** n i dung m c 1

**** n i dung m c 2

...

**** n i dung m c n

Trong ó **X** nh ki u ký hi u s c s d ng trong danh sách.

- A là ch hoa
- a là danh sách c ánh theo th t ch th ng a,b,c...z
- I danh sách c ánh theo th t s La Mã I, II, III...
- i danh sách c ánh theo th t s La Mã i, ii, iii...
- 1 danh sách c ánh d u theo th t 1,2,3...

Ngoài ra còn có thu c tính **START=n**, n là giá tr b t u c a danh sách.

Ví dụ: Xét o n l nh sau:

Mã HTML

```
<HTML>
<HEAD><TITLE>mau nen</TITLE></HEAD>
<BODY>
<OL TYPE=1 START=2>
<LI>cong cha nhu nui thai son
<LI>nghia me nhu nuoc trong nguon chay ra
<LI>mot long tho me kinh cha
<LI>cho tron chu hieu moi la dao con
</BODY>
</HTML>
```

K t qu hi n th trên trình duy t

2. cong cha nhu nui thai son

3. nghĩa mẹ như nước trong nguồn chảy ra
4. một lòng thờ mẹ kính cha
5. cho tròn chu hiếu mới là đạo con

- **Tạo chấm tròn (Bullet) cho danh sách**

Dùng tag để tạo chấm tròn cho danh sách.

Cú pháp: <UL TYPE="kiểu chấm tròn">

nội dung mục thứ 1

nội dung mục thứ 2

...

nội dung mục thứ n

“kiểu chấm tròn” mang một trong 3 giá trị sau:

- disc: kiểu dấu chấm tròn thông thường (mặc định khi viết danh sách cấp thứ nhất).
- circle: kiểu dấu chấm tròn rỗng (mặc định khi viết danh sách cấp 2).
- square: kiểu dấu chấm vuông (mặc định khi viết danh sách cấp thứ 3 trở đi)

Ví dụ: Dưới đây là một dấu chấm tròn cho danh sách.

Mã HTML

```
<HTML>
<HEAD><TITLE>mau nen</TITLE></HEAD>
<H1>Dau cham tron cho danh sach</H1>
<HR>
<UL TYPE="disc">
<LI>muc mot
<LI>muc hai
<LI>muc ba
<LI>muc nam
<LI>muc sau
</BODY>
</HTML>
```

Kết quả hiển thị trên màn hình

Dấu chấm tròn cho danh sách

- muc mot
- muc hai
- muc ba
- muc nam
- muc sau

Về cách tạo Bullet này trông có vẻ đơn giản, nhưng nó lại rất thú vị. Bạn hãy thử vẽ những hình ảnh nhỏ, sau đó dùng những hình ảnh này để tạo Bullet bằng tag .

Ví dụ: Ta vẽ một Bullet và một tập tin hình ảnh này với phần mở rộng là “gif”. Khi viết danh sách bằng một mã HTML sau:

Mã HTML

<pre> <HTML> <HEAD><TITLE>Bullet tu tao</TITLE></HEAD> <BODY> <H2><CENTER>Cac huong trong nganh CNTT</CENTER></H2> Cong nghe phan mem
 Cong nghe tri thuc
 He thong thong tin
 Mang may tinh </BODY> </HTML> </pre>
K t qu h i n th trên trình duy t
<p>Cac huong trong nganh CNTT</p> <ul style="list-style-type: none"> • Cong nghe phan mem • Cong nghe tri thuc • He thong thong tin • Mang may tinh

• Tạo danh sách định nghĩa

Là lo i danh sách có d ng m t t hay m t c m t kèm theo n i dung dài, r t thích h p gi i thích ý ngh a c a n i dung.

làm c i u này ta dùng các tag <DL>, <DT>, <DD>.

- Tag <DL>: t o danh sách nh ngh a
- Tag <DT> : ánh d u thu t ng c nh ngh a trong danh sách.
- <DD>: Gi i thích thu t ng trên

Ví dụ: Ta xét o n l nh sau:

Mã HTML
<pre> <HTML> <HEAD><TITLE>Danh sach dinh nghia</TITLE></HEAD> <BODY> <H2>Cac huong trong nganh cong nghe thong tin</H2> <DL> <DT>Huong cong nghe phan mem <DD>Dao tao ra nhung lap trinh vien, dap ung nhu cau can thiet cua xa hoi, sau khi ra truong Sinh Vien duoc gioi thieu viec lam o cac khu cong nghiep phan mem </BODY> </HTML> </pre>
K t qu h i n th trên trình duy t
<p>Cac huong trong nganh cong nghe thong tin</p> <p>Huong cong nghe phan mem</p> <p>Dao tao ra nhung lap trinh vien, dap ung nhu cau can thiet cua xa hoi, sau khi ra truong Sinh Vien duoc gioi thieu viec lam o cac khu cong nghiep phan mem</p>

2.5.6. Tạo liên kết (link)

Liên kết là đặc trưng của Word Wide Web, chúng cho phép người dùng chuyển từ một trang sang một trang khác trong cùng trang Web hoặc chuyển từ trang Web này sang trang Web khác.

- **Liên kết trong cùng trang Web**

Cho phép liên kết các mục bên trong một trang Web. Làm việc này trước tiên phải tạo điểm neo (Bookmark) và sau đó tạo liên kết đến điểm neo này.

Cú pháp:

Tạo điểm neo: `Nội dung`

Tạo liên kết: ``

Dấu “#” báo cho trình duyệt biết liên kết là nằm bên trong trang Web.

Ví dụ: Xét nguồn như sau:

```
<HTML>
<HEAD><TITLE>Hyperlink</TITLE></HEAD>
<BODY>
<P><A HREF="#Ch1">Chương 1</A></P>
<P><A NAME="Ch1">Chương 1:</A></P>
<P>Bài 1: Một số khái niệm liên quan đến mạng máy tính</P>
<P>Bài 2; Lập trình với ngôn ngữ siêu liên kết HTML</P>
</BODY>
</HTML>
```

- **Tạo liên kết đến các trang Web khác**

Cú pháp: ``

Ví dụ:

- `Yahoo`: liên kết đến trang chủ Yahoo.
- `Home`: liên kết đến trang có tên home.htm trong cùng thư mục.
- `Click`: gọi Open của JavaScript.
- `Download`: mục tải tệp tin nén “code.zip”.
- ` my mail`: địa chỉ gửi mail.
- `News`: một trang Web có tên “news.htm” trong các cửa sổ mới. Thuộc tính **TARGET**, trong đó **TARGET** mang một trong những giá trị sau:
 - **name**: Nạp trang news.htm lên các cửa sổ có tên Name.
 - **_blank**: Nạp trang news.htm vào một cửa sổ trống mới, cửa sổ mới này không có tên.
 - **_parent**: Nạp trang news.htm vào cửa sổ cha gần nhất của trang Web hiện hành.
 - **_self**: Nạp trang news.htm vào cùng cửa sổ với trang Web hiện hành.

- **_top**: Nạp trang news.htm vào cửa sổ cao nhất.

2.5.7. Một số ký tự đặc biệt trong HTML

Vì HTML dùng các ký tự '<', '>' để đánh dấu các tag, do đó để thể hiện các ký tự đặc biệt này ta phải dùng các nhóm ký tự thay thế.

Ký tự cần hiển thị	Nhóm ký tự thay thế
<	<
>	>
&	&
“	"
Khoảng trắng	

2.5.8. Các tag dùng thiết kế bảng

Để thiết kế một bảng biểu dùng các tag cơ bản sau, với các thuộc tính đi kèm như sau:

- **<TABLE></TABLE>**: Bắt đầu một bảng mới với các thuộc tính đi kèm sau:
 - **BGCOLOR**: Định màu nền cho bảng
 - **BORDER**: Định độ dày/mảnh của đường viền.
 - **BORDERCOLOR**: Định màu cho đường viền.
 - **BORDERCOLORDARK**: Định màu sậm cho phần hắt bóng của đường viền.
 - **BORDERCOLORLIGHT**: Định màu nhạt cho phần sáng hơn của đường viền.
 - **CELLPADDING**: Định khoảng cách giữa nội dung và đường viền.
 - **CELLSPACING**: Khoảng cách giữa các ô.
 - **FRAME**: Hiện thị đường viền ngoài.
 - **HEIGHT**: Định chiều cao bảng.
 - **WIDTH**: Định chiều rộng bảng.
 - **RULES**: Hiện thị đường viền trong.
- **<TR> </TR>**: Bắt đầu một hàng mới trong bảng với các thuộc tính sau:
 - **ALIGN/VALIGN**: Canh chỉnh nội dung hàng theo phương ngang/dọc
 - **BGCOLOR**: thay đổi màu nền của bảng.
- **<TH></TH>, <TD></TD>**: Bắt đầu một đề mục mới cho bảng với các thuộc tính sau:
 - **ALIGN/VLIGN**: Canh chỉnh nội dung ô theo phương ngang/dọc.
 - **BGCOLOR**: thay đổi màu nền của ô.
 - **COLSPAN**: mở rộng ô qua nhiều cột.
 - **ROWSPAN**: kéo dài ô xuống nhiều hàng.
 - **NOWRAP**: Giữ nội dung ô nằm trên một dòng.
 - **WIDTH, HEIGHT**: định chiều rộng, cao cho ô.

• Tạo bảng

Ví dụ: Tạo một bảng đơn giản bằng đoạn HTML sau:

Mã HTML
<HTML>
<BODY>

```

<HEAD><TITLE>tao bang</TITLE></HEAD>
<TABLE BORDER="1">
<TR><TH>TEN</TH>
      <TH>DIEM TB</TH>
      <TH>LOAI</TH>
<TR><TD>Nguyen Thi An</TD>
      <TD>7</TD>
      <TD>KHA</TD>
<TR><TD>Nguyen Van Tuan</TD>
      <TD>8</TD>
      <TD>Gioi</TD>
</TABLE></BODY></HTML>

```

kết quả hiển thị trên màn hình

TEN	DIEM TB	LOAI
Nguyen Thi An	7	KHA
Nguyen Van Tuan	8	GIOI

- Dùng tag <ROWSPAN> và <COLSPAN> để mở rộng một ô qua nhiều dòng, nhiều cột.

Ví dụ: Xét đoạn HTML sau:

Mã HTML

```

<HTML>
<HEAD><TITLE>BANG DIEM</TITLE></HEAD>
<BODY>
<TABLE BORDER="1">
<TR><TD ALIGN="center" ROWSPAN="3">HOC KY I</TD>
      <TD ALIGN="center" COLSPAN="3">DIEM TB</TD>
</TR>
<TR><TD>NGUYEN THI AN</TD>
      <TD>7</TD>
      <TD>KHA</TD>
<TR><TD>NGUYEN VAN TUAN</TD>
      <TD>8</TD>
      <TD>GIOI</TD>
</TABLE></BODY></HTML>

```

Kết quả hiển thị trên trình duyệt

HOC KY I	DIEM TRUNG BINH		
	NGUYEN THI AN	7	KHA
	NGUYEN VAN TUAN	8	GIOI

- **Tạo khung viền cho bảng**

Cú pháp: <TABLE BORDER="n">

Với “n” là độ dày của đường viền (tính bằng Pixel), nếu n=0 thì bảng không có đường viền.

- **Thay đổi màu khung viền**

Cú pháp: <TABLE BORDERCOLOR="màu">

Ví dụ: Ta muốn độ dày viền=2 và màu khung viền là đỏ thì

<TABLE BORDER="2" BORDERCOLOR="red">

- **Tạo màu nền bảng**

Cú pháp: < TABLE BGCOLOR="màu">

- **Màu nền cho các ô**

Cú pháp: <TH BGCOLOR="màu">...</TH> hoặc

<TD BGCOLOR="màu">...</TD>

- **Định kích thước bảng**

Cú pháp: <TABLE WIDTH="x" HEIGHT="y">...

Với “x”, “y” thứ tự là chiều rộng và chiều cao của bảng.

- **Định kích thước cho ô**

Cú pháp: <TH WIDTH="x" HEIGHT="y">...</TH> hoặc

<TD WIDTH="x" HEIGHT="y">...</TD>

- **Canh bảng ở giữa trang**

Cú pháp: <TABLE ALIGN="direction">...</TABLE>

Với “direction” mang 3 giá trị “center”, “left”, “right” tương ứng với bảng nằm giữa, trái, phải trên màn hình.

2.5.9. Các tag dùng tạo Form

Form là thành phần giao tiếp cơ bản giữa người duyệt Web với người tạo Web. Dữ liệu được nhập vào **Form** thông qua các hộp điều khiển (control) như Textbox, Checkbox, Radio button, push button, dropdown listbox...

Cú pháp: <FORM>...</FORM>

- **Tạo TextBox (hộp nhập liệu)**

TextBox dùng để chứa một dòng văn bản tùy ý do người dùng nhập vào. Để tạo một **Textbox** ta chỉ định thuộc tính **TYPE="text"** theo cú pháp sau

Cú pháp: <INPUT TYPE="text" NAME="name" VALUE="value" SIZE="n" MAXLENGTH="n">

Trong đó “name” là chuỗi ký tự nhận diện dữ liệu nhập vào, “value” là dữ liệu đầu tiên hiển thị trong **Textbox** và được gửi đến máy chủ khi người duyệt không gõ thông tin gì khác, **SIZE** định kích thước của **TextBox**, **MAXLENGTH** giới hạn số ký tự nhập vào **Textbox**.

Khi muốn chỉ định dữ liệu nhập vào **Textbox** dưới dạng mật khẩu (Các ký tự nhập vào không được hiển thị mà thay vào đó là các dấu ****) ta thay thuộc tính **TYPE="password"**.

Ví dụ: Tạo một **Textbox** bằng đoạn HTML sau:

Mã HTML

<HTML>

<HEAD>

```

<TITLE>Tạo TextBox</TITLE>
</HEAD>
<BODY>
<FORM>
<B>Ten KH:</B>
<INPUT TYPE="Text" NAME="TenKH" SIZE=20 MAXLENGTH="255">
<BR>
<B>Mat ma:</B>
<INPUT TYPE="password" NAME="matma" SIZE=8>
</FORM>
</BODY>
</HTML>

```

Kết quả hiển thị trên trình duyệt

- **Tạo Texbox với vùng văn bản chứa nhiều dòng**

Cú pháp: <TEXTAREA NAME="name" ROWS="n" COLS="m" WRAP></TEXTAREA>

Trong đó “name” là chuỗi ký tự nhận diện dữ liệu nhập và, “n” là chiều cao của vùng văn bản tính bằng dòng (mặc định n=4), “m” là chiều rộng của vùng văn bản tính bằng ký tự (giá trị mặc định m=40). Dùng thuộc tính **WRAP** để xuống dòng khi đựng lề.

- **Tạo nút Radio button (nút chọn Radio)**

Radio là loại nút chỉ được chọn một trong số đó, không thể chọn lựa hai nút cùng lúc. Để tạo một nút chọn **Radio** ta chỉ định thuộc tính **TYPE="radio"** như cú pháp sau:

Cú pháp: <INPUT TYPE="radio" NAME="name" VALUE="data" CHECKED>

Trong đó “name” vừa nhận diện dữ liệu vừa liên kết các nút **Radio** trong một nhóm lại với nhau, các nút **Radio** trong cùng một nhóm thì có thuộc tính **NAME** giống nhau, đảm bảo một thời điểm chỉ có một nút được chọn. “Data” là văn bản gửi đến máy chủ khi một nút **Radio** được chọn. **CHECKED** nút đó được chọn.

- **Tạo ô chọn (checkbox)**

Khác với nút **Radio**, **Checkbox** cho người chọn nhiều nút trong cùng một nhóm. Giống như nút **Radio**, **Checkbox** liên kết với nhau thông qua thuộc tính **NAME**. Ta chỉ định thuộc tính **TYPE="checkbox"** khi tạo nút chọn **Checkbox**

Cú pháp: <INPUT TYPE="checkbox" NAME="set" VALUE="value" CHECKED>

Trong đó chức năng của các thuộc tính giống như chức năng của nút chọn **Radio**.

Ví dụ:

Mã HTML

```

<HTML>
<HEAD><TITLE>Tạo TextBox</TITLE></HEAD>
<BODY>
<FORM>Trình do:
<INPUT TYPE="radio" NAME="nn" VALUE="R1" CHECKED>Đại học
<INPUT TYPE="radio" NAME="nn" VALUE="R2">Cao đẳng

```



```
<OPTION VALUE= "vang" SELECTED>Mau vang</OPTION>
<OPTION VALUE= "xanh" SELECTED>Mau do</OPTION>
<OPTION VALUE= "do" SELECTED>Mau xanh</OPTION>
</SELECT>
</FORM>
</HTML>
```

Kết quả hiển thị trên trình duyệt Web

Để tạo một ListBox cho phép hiển thị, chọn nhiều mục ta dùng thêm thuộc tính **MULTIPLE** với **SIZE="n"**, n là số mục **<SELECT NAME="MAU" MULTIPLE SIZE="2">**

Khi đó kết quả trên trình duyệt là:

2.6. Các tag tạo Frame (khung)

Frame là thành phần cơ bản của trang Web, một trang Web có thể được chia thành nhiều **frame**, mỗi **frame** sẽ chứa một trang Web riêng.

2.6.1. Tạo Frame có dạng hàng

Để chia một trang Web thành nhiều Frame ta dùng tag **<FRAMESET>** đặt ngay sau tag **</HEAD>**.

Cú pháp:

```
<FRAMESET ROWS="a,*,b">
```

```
<FRAME NAME="đặt tên frame" SRC="tên tập tin sẽ hiển thị ở Frame này">
```

```
</FRAMESET>
```

Trong đó:

- a: Chiều cao của **Frame** đầu tiên
- *: Chiều cao của **Frame** thứ hai là khoảng trống còn lại.
- b: Chiều cao của **Frame** thứ 3

Nếu ta sử dụng nhiều dấu "*" thì không gian còn lại của cửa sổ sẽ được chia đều cho các Frame có dấu "*".

Ví dụ:

Mã HTML

```
<HTML>
<HEAD><TITLE>Tạo Frame</TITLE></HEAD>
<FRAMESET ROWS="100,*,*">
<FRAME NAME="tren" SRC="khungtren.htm">
<FRAME NAME="tren" SRC="khunggiua.htm">
<FRAME NAME="tren" SRC="khungcuoi.htm">
</FRAMESET>
</HTML>
```

kết quả hiển thị trên màn hình

Thêm thanh cuộn cho Frame ta dùng thuộc tính **SCROLLING** = **yes/no** trong tag **FRAME** tương ứng để **ẩn/hiện** thanh cuộn. Nếu không có thuộc tính **SCROLLING** thì thanh cuộn sẽ xuất hiện khi cần thiết.

Thêm thuộc tính **NORESIZE** vào tag **FRAME** cho Frame tương ứng để không cho người dùng thay đổi kích thước của Frame đó.

Thêm thuộc tính **FRAMEBORDER**=**"yes/no/0"** vào tag **<FRAME>** để hiện/ẩn khung viền cho Frame. Chọn **"yes/0"** cho trình duyệt Internet Explorer và **"yes/no"** cho trình duyệt Netscape.

Thêm thuộc tính **Border** vào tag **<FRAME>** để định độ dày mỏng viền của Frame.

Thêm **Framespacing**=**"n"** vào tag **<FRAME>** để định khoảng cách từ viền khung tới nội dung trong khung.

2.6.2. Tạo Frame có dạng cột

Cú pháp: **<FRAMESET COLS="a,*b">**

<FRAME NAME="đặt tên frame" SRC="tên tập tin sẽ được hiển thị ở Frame này">

</FRAMESET>

ý nghĩa của các thuộc tính tương tự tạo Frame có dạng hàng.

2.6.3. Kết hợp tạo Frame vừa dạng hàng vừa có dạng cột

Trong một trang Web việc tạo Frame thường được kết hợp giữa tạo Frame có dạng hàng và Frame có dạng cột.

Ví dụ: Trong ví dụ này ta chia trang Web thành 4 Frame: 1 Frame trên, 1 Frame trái, 1 phải và 1 Frame dưới.

Mã HTML

```
<HTML>
<HEAD><TITLE>Danh sach cac mon hoc</TITLE></HEAD>
<FRAMESET ROWS="60,*,40">
<FRAME NAME="tren" SRC="tren.htm">
<FRAMESET COLS="150,*">
<FRAME NAME="trai" SRC="trai.htm">
<FRAME NAME="phai" SRC="phai.htm">
</FRAMESET>
<FRAME NAME="duoi" SRC="duoi.htm">
</FRAMESET>
</HTML>
```

Kết quả hiển thị trên trình duyệt

2.7. Các hiệu ứng Dynamic HTML (DHTML)

2.7.1. Tạo chuỗi ký tự chuyển động

Cú pháp: **<MARQUEE></MARQUEE>**

Các thuộc tính:

- **BEHAVIOR**=type, type có thể là: scroll để chuỗi ký tự bắt đầu từ một bên màn hình và biến mất ở bên kia, slide để chuỗi bắt đầu tại một bên

màn hình và dừng lại ở bên kia, alternate để chuỗi bắt đầu bên này màn hình và chuyển động ngược lại khi đến bên kia.

- **DIRECTION**="left/right": định hướng bắt đầu chạy.
- **LOOP**=n: số lần chuyển động ngang qua màn hình, nếu **LOOP**=infinite thì chuỗi sẽ xuất hiện liên tục.
- **VSCROLLAMOUNT**=n : tốc độ chuyển động.
- **SCROLLDELAY**=n: thời gian ngừng sau một vòng chạy.

Nếu không có các thuộc tính trên thì chuyển động lặp đi lặp lại từ lề phải sang lề trái với tốc độ=6 (pixel) và thời gian dừng=90 giây.









Ví dụ: Ta xét đoạn chương trình sau:

Mã HTML
<pre><HTML> <HEAD> <TITLE>Chuoi ky tu chay </TITLE> </HEAD> <p></p> <MARQUEE BEHAVIOR=scroll DIRECTION="left" LOOP=infinite SCROLLDELAY="60" BGCOLOR="turquoise">L ớp chuyên viên công nghệ thông tin </MARQUEE> </HTML></pre>
Kết quả hiển thị trên trình duyệt

2.7.2. Thay đổi hình dạng chuột khi đi qua một ô trong bảng

Cú pháp: <TD STYLE="cursor: thuộc tính của cursor">

Các thuộc tính và các dạng cursor tương ứng.

Auto	Hand	Move	Text	Wait	Help	Default	Crosshair
							
Beam_r.cur	Harrow.cur	Move_im.cur	Beam_r.cur	Busy_m.cur	Help_l.cur	Larrow.cur	Lcross.cur

Ví dụ: Đoạn chương trình sau mô phỏng các dạng Cursor

<pre><HTML> <HEAD><TITLE>Cursor</TITLE></HEAD> <TABLE BORDER="1"> <TR> <TD STYLE="cursor:auto">auto <TD STYLE="cursor:hand">hand <TD STYLE="cursor:move">move <TD STYLE="cursor:text">text <TD STYLE="cursor:wait">wait <TD STYLE="cursor:help">help <TD STYLE="cursor:default">default</pre>

```
<TD STYLE="cursor:crosshair">crosshair
</TABLE>
</HTML>
```

Bài tập chương 2

Bài 1: Thiết kế trang chủ chương trình đào tạo chuyên viên (<http://cv.citd.edu.vn>), trang gồm 3 Frame như hình sau, toàn bộ trang sử dụng font chữ là “Times New Roman”. Frame trên cùng gồm có một hình bên trái, bên phải là các dòng chữ chạy. Frame bên trái là các mục cần đưa tin. Frame giữa dùng để hiển thị nội dung tin.

Bài 2: Thiết kế trang Web Giáo dục

(http://www.saigonnet.vn/giaoduc/thong_tin_tuyen_sinh.htm)

Bài 3: Thiết kế trang Web thường truy cập có địa chỉ sau:

- <http://citd.vn.edu.vn> (trang chủ trung tâm tin học TP.HCM)
- <http://www.vnexpress.net>
- <http://www.saigonnet.vn>
- <http://www.vnn.vn>
- <http://yahoo.com>
- <http://www.hcmuns.edu.vn>
- <http://www.google.com>

Ghi chú:

- Để xem mã **HTML** của trang Web hiện hành ta vào Menu sau đó chọn View->Source.
- Để lưu trang Web hiện hành ta vào menu sau đó chọn File->Save as

Chương 3: GIỚI THIỆU NGÔN NGỮ SCRIPT VBSCRIPT VÀ JAVASCRIPT

1. Giới thiệu ngôn ngữ vbscript và JavaScript

VBscript và **JavaScript** là ngôn ngữ lập ra để chạy được trên trình duyệt, các đoạn chương trình viết bằng ngôn ngữ này được nhúng vào các trang HTML. Các đoạn chương trình này có khả năng:

- Được thực thi khi một sự kiện nào đó trên trang Web xảy ra như: mouseclicked, mouseover,...
- Xử lý các thành phần trên trang Web như: thay đổi màu chữ, font chữ, thay đổi ảnh,...

Cú pháp của **VBScript** gần giống với cú pháp ngôn ngữ lập trình **VisualBasic** và được Microsoft phát triển, trong khi cú pháp của JavaScript gần giống với cú pháp của ngôn ngữ lập trình C và được NetsCape phát triển.

VBScript không phân biệt chữ hoa và chữ thường trong khi JavaScript thì lại phân biệt chữ hoa và chữ thường.

JavaScript được hỗ trợ trên hầu hết các trình duyệt, còn **VBScript** chỉ được hỗ trợ tốt nhất ở trình duyệt Internet Explorer.

Cũng giống như các ngôn ngữ lập trình khác các kiểu dữ liệu thông dụng được dùng VBScript và JavaScript là: kiểu số, kiểu chuỗi, kiểu luận lý,...Tuy nhiên cách

định nghĩa các kiểu dữ liệu giữa **VBScript** và **JavaScript** có sự khác nhau. Cách khai báo các hàm cũng khác nhau. Cách sử dụng các hàm thư viện có sẵn cũng khác nhau.

2. ngôn ngữ vbscript

2.1. Chú thích một dòng lệnh

Chú thích trong **VBScript** tương tự như **Visual Basic** bắt đầu bằng ký tự nháy đơn ('). Dấu chú thích chỉ có tác dụng trên một dòng. Dấu chú thích làm cho các dòng lệnh rõ ràng và dễ hiểu đối với người thiết kế chương trình. Khi thực thi, trình biên dịch bỏ qua dòng ghi chú này.

2.2. Cách khai báo biến, hằng, mảng

2.2.1. Khai báo biến

Dùng từ khóa **Dim** để khai báo biến, biến trong ngôn ngữ **VBScript** không cần chỉ định kiểu như trong ngôn ngữ lập trình cấu trúc. Các biến không cấu trúc được xem là biến vô hướng (**variant**) có thể chứa và tự chuyển đổi hầu hết các kiểu dữ liệu.

Cú pháp: **Dim** tên_biến1, tên_biến2, tên_biến3,...

Các biến được cách nhau bởi dấu phẩy “,”.

Tuy nhiên trong **VBScript** không nhất thiết phải khai báo biến trước khi sử dụng. Để yêu cầu các biến phải được khai báo trước khi sử dụng ta dùng lệnh “Option Explicit” đặt trước lệnh đầu tiên của đoạn chương trình.

Ví dụ: **Dim** a
 a = 3

Ghi chú:

- Biến không phân biệt chữ hoa/thường
- Chiều dài tên biến không vượt quá 255 ký tự.
- Tên biến phải bắt đầu bằng 1 ký tự chữ cái và biến không phép chứa dấu chấm “.”

2.2.2. Khai báo hằng

Hằng được định nghĩa bằng từ khoá **Const**. Chỉ có thể sử dụng giá trị của hằng chứ không thể thay đổi nội dung hằng.

Ví dụ: **Const** ten = “Nguyen Van Tuan”

2.2.3. Khai báo mảng

- **Mảng một chiều**

Dim Ten_mang (kích thước của mảng)

Số phần tử tối đa của mảng trên= Kích thước của mảng +1.

Chỉ số của phần tử đầu tiên của mảng bằng 0, để truy xuất đến phần tử có chỉ số “i” ta dùng **Ten_mang(i)**;

Ví dụ: **Dim** A(20); thì mảng A có thể chứa tối đa 21 phần tử

- **Mảng 2 chiều**

Dim Ten_mang(dòng, cột)

Chỉ số của phần tử ở dòng đầu tiên và cột đầu tiên là (0,0).

Để truy xuất phần đến phần tử có chỉ số dòng i, chỉ số cột j ta dùng **B(i,j)**.

Ví dụ: **Dim** B(5,10); Mảng B có thể chứa 6 dòng và 11 cột.

Trong **VBScript** ta muốn khai báo một mảng động thì khi khai báo mảng ta không định rõ kích thước cho mảng. Tức kích thước của mảng có thể thay đổi trong quá trình thao tác, dùng hàm **ReDim** để thay đổi kích thước của mảng động.

Trong **VBScript** có thể khai báo một mảng có 60 chiều.

2.3. Các kiểu dữ liệu

Trong **VBScript** chỉ có một kiểu dữ liệu duy nhất là **Variant**. Đây là kiểu dữ liệu có thể chứa các loại dữ liệu từ kiểu chuỗi, kiểu số cho đến các loại dữ liệu có cấu trúc như kiểu bản ghi (record). Kiểu dữ liệu này cũng là kiểu dữ liệu trả về của các hàm và các thủ tục được viết bằng ngôn ngữ **VBScript**.

Tùy theo ngữ cảnh sử dụng mà một biến **Variant** mang giá trị là kiểu số, kiểu chuỗi (hay bất kỳ kiểu dữ liệu nào khác).

Ví dụ: Trong biểu thức $a=b+1997$, thì biến a và biến b mang kiểu dữ liệu là kiểu số. Trong biểu thức $a=b+"1997"$ thì biến a và biến b có kiểu dữ liệu là kiểu chuỗi.

Các kiểu dữ liệu mà một biến **Variant** có thể lưu trữ.

Các kiểu dữ liệu	ý nghĩa
Boolean	kiểu luận lý True hoặc False
Byte	Số nguyên có giá trị từ 0 đến 255
Integer	Số nguyên có giá trị từ -32768 đến 32768
Currency	Từ -922,337,203,685,477,5808 đến 922,337,203,685,477,5807
Long	Số nguyên từ -2,147,483,648 đến 2,147,483,647
Single	Số thực, có giá trị từ 3.402823E38 đến 1.401298E-45 cho các số âm, từ 1.401298E-45 đến 3.402823E38 cho số dương
Double	Số thực, có giá trị từ 1.79769313486232E308 đến 4.94065645841247E-324 cho các số âm, 4.94065645841247E-324 đến 1.79769313486232E308
Date(time)	chứa giá trị ngày từ 01.01.100 đến 31.12.9999
String	Chuỗi ký tự có thể chứa 2 tỉ ký tự
Empty	dữ liệu chưa được khởi tạo
Null	Null
Object	Chứa đối tượng trên Form như hộp văn bản, nhãn, nút nhấn,...
Error	chứa mã lỗi

- Để chuyển đổi dữ liệu này sang kiểu dữ liệu khác ta dùng các hàm thư viện **Cbyte** (kiểu byte), **Cdate** (kiểu ngày), **CInt** (integer), **CStr** (string), **SBool** (bool), **CDbl** (double), **CLng** (long), **CSng** (Single)
- Muốn biết kiểu dữ liệu mà một biến có kiểu **Variant** đang lưu trữ ta dùng hàm thư viện **VarType**.

2.4. Các toán tử cơ sở

- **Toán tử gán (=)**

Tên_biến = Biểu thức.

Với các biến có kiểu dữ liệu tổng quát, để gán giá trị cho biến chúng ta dùng, ta phải dùng lệnh **Set** như sau:

Set tên_biến = Biểu thức

- **Toán tử tính toán**

+(cộng), -(trừ), *(nhân), \ (chia lấy phần nguyên), /(chia làm tròn), ^ (lũy thừa), mod (chia lấy phần dư)

- **Toán tử nối chuỗi**

Dùng &: Ví dụ: S = “Dai” & “ ” & “hoc”

Dùng + : Ví dụ: S = “Dai” + “ ” + “hoc”

- **Toán tử so sánh**

= (bằng), > (lớn hơn), >= (lớn hơn hay bằng), < (nhỏ hơn), <= (nhỏ hơn hay bằng), <> (khác).

Kết quả của một biểu thức so sánh sẽ thuộc về kiểu lý luận (True/false), khi cần nối các biểu thức so sánh với nhau ta dùng toán tử luận lý **And, Or**

2.5. Các lệnh xử lý điều kiện rẽ nhánh

Bạn có thể sử dụng các lệnh rẽ nhánh **if...then, if...then...else** hoặc **Select case** để ra điều kiện rẽ nhánh dựa trên các biểu thức so sánh.

2.5.1. Cấu trúc If...Then

Dùng để xử lý lệnh khi biểu thức so sánh của **If** trả về giá trị **True**

Ví dụ:

```
Dim myDate
myDate = #2/12/2000#
if myDate < Now Then myDate = Now
end if
```

2.5.2. Cấu trúc If...Then...Else

Mở rộng hơn cấu trúc **If...Else**, khi biểu thức so sánh của **If** trả về giá trị **True** thì khối lệnh sau mệnh đề **Then** sẽ được thực hiện. Ngược lại biểu thức so sánh của **If** trả về giá trị **False** thì khối lệnh sau **Else** sẽ được thực hiện.

Ví dụ:

```
Dim myValue
myValue = 13
if myValue mod 2 = 0 Then
    document.Write(myValue)
    document.Write(“la so chan”)
else
    document.Write(myValue)
    document.Write(“la so le”)
end if
```

2.5.3. Cấu trúc Select Case

Cấu trúc **Select Case** cho phép lựa chọn nhiều trường hợp để ra quyết định thực thi. Theo cú pháp sau:

```
Select case <tên biến>
    Case <giá trị 1>
        Khối lệnh 1
    Case <giá trị 2>
        Khối lệnh 2
    ...
    Case Else
        Khối lệnh i
```

End Select

Ví dụ:

```
<HTML>
<HEAD><TITLE>Select case</TITLE></HEAD>
<BODY>
<SCRIPT LANGUAGE="VBScript">
Dim Thang
Thang = 13
document.Write("Thang")
document.Write(Thang)
SELECT CASE Thang
CASE 2: document.Write("co 28 ngay")
CASE 1,3,5,7,8,10,12 : document.Write("co 31 ngay")
CASE 2: document.Write("co 28 ngay")
CASE 4,6,9,11 : document.Write("co 30 ngay")
CASE ELSE document.Write("khong hop le")
END SELECT
</SCRIPT></BODY></HTML>
```

Mệnh đề Case Else trong cú pháp Select case dùng trong trường hợp tất cả các phép so khớp của mệnh đề Case không xảy ra.

2.6. Cấu trúc lặp

Tương tự các ngôn ngữ lập trình khác, **VBScript** cung cấp các lệnh lặp dựa trên điều kiện. Dùng cấu trúc **Exit for**, **Exit do**, **Exit While** để thoát khỏi cấu trúc lặp tương ứng.

2.6.1. Cấu trúc Do...Loop

Cấu trúc **Do...Loop** lặp trong khi điều kiện kiểm tra của **Loop** còn đúng. Có 4 cấu trúc lặp theo cú pháp sau:

- **Do While** <biểu thức điều kiện>
Khởi lệnh
Loop
- **Do**
Khởi lệnh
Loop While <biểu thức điều kiện>
- **Do**
Khởi lệnh
Loop Until <biểu thức điều kiện>

Ví dụ: Tính giá trị N!

```
Dim N, i, S
N=3
S=1
i=1
DO WHILE (i<=n)
S=S*i
i=i+1
```

2.6.2. Cấu trúc While...Wend

Lặp trong khi điều kiện kiểm tra của **While** còn đúng. Theo cú pháp sau:

While <biểu thức điều kiện>

Khối lệnh

Wend

Ví dụ: Tính tổng $S = 1+2+3+5+...+N$

Dim i,S,N

N=5

i=1

S=0

WHILE (i<=N)

S=S+i

i=i+2

Wend

2.6.3. Cấu trúc For...Next

Cấu trúc **For...Next** lặp với số lần lặp xác định, cấu trúc này có 2 dạng sau:

- **For** <biến chạy = chỉ số đầu> **To** <chỉ số cuối>

Khối lệnh

Next

- **For** <biến = chỉ số đầu> **To** <chỉ số cuối> Step <bước nhảy>

Khối lệnh

Next

Ví dụ: Tính tổng $S = 1+2+3+4+...+N$

Dim d

‘ Tao doi tuong DictionarySet

d = CreateObject(“Scripting.Dictionary”)

‘ Luu cac phan tu vao tap hop

d.Add “0”, ”Athens”

d.Add “1”, “Belgrade”

d.Add “2”, “Cairo”

‘ Duyet va in cac phan tu trong tap hop

For Each I in d

document.Write(D.Item(I))

Next

2.6.4. Cấu trúc For each...Next

Cấu trúc **For each...Next** lặp với mỗi phần tử trong tập

Ví dụ:

Dim d

‘Tạo doi tuong DictionarySet

d=CreateObject(“Scripting.Dictionary”)

‘Luu cac phan tu vao tap hop

d.Add “0”, ”Athens”


```
d.Add "1", "Belgrade"  
d.Add "2", "Cairo"  
' Duyệt và in các phần tử trong tập hợp  
For each i in d  
document.Write(D.Item(i))
```

2.7. Khai báo hàm và thủ tục

2.7.1. Khai báo hàm

Function Tên_hàm (các tham số)
Khởi lệnh

End Function

Ví dụ: Xây dựng hàm tính diện tích hình tròn khi biết bán kính

```
Function DienTich(R)  
Dim S  
S = 3.14*R*R  
DienTich=S  
document.Write(S)  
End Function
```

2.7.2. Khai báo thủ tục

Sub Tên_thủ_tục(các tham số)
khởi lệnh

End Sub

Ví dụ: Xây dựng thủ tục gọi sử dụng hàm DienTich đã viết trên

```
Sub SuDung()  
DienTich(5)  
End Sub
```

Chương 3. Ngôn ngữ JAVASCRIPT

3.1. Chú thích một hay nhiều dòng lệnh

Dòng lệnh được kết thúc bằng dấu chấm phẩy “;” ở cuối dòng. Tập hợp các dòng lệnh nằm trong hai dấu ngoặc đơn {} gọi là tập lệnh.

Muốn chú thích một dòng lệnh ta dùng dấu “//” đặt trước dòng lệnh muốn chú thích. Muốn chú thích nhiều dòng lệnh thì ta đặt các dòng lệnh cần chú thích giữa hai dấu “/*” và “*/”.

3.2. Cách khai báo biến, mảng

3.2.1 Cách khai báo biến

Dùng từ khoá **var** để khai báo biến. Biến trong JavaScript không cần định rõ kiểu dữ liệu của biến lúc khai báo. Tuy nhiên, khi gán giá trị cho biến, **JavaScript** phân biệt kiểu của các giá trị mà bạn gán. Trong **JavaScript** bắt buộc phải khai báo biến trước khi sử dụng.

```
var tên_biến1 = tri1, tên_biến2 = tri2, ...;
```

Chú ý:

- + Biến có phân biệt chữ hoa/thường.
- + Biến phải bắt đầu bằng ký tự chữ cái.
- + Biến không cho phép có khoảng trắng, không cho phép có dấu gạch ngang.

3.2.2. Khai báo mảng

- **Mảng một chiều**

```
var A = new Array(10)
```

Mảng **A** nói trên có 10 phần tử, và chỉ số phần tử đầu tiên của mảng bắt đầu 0, muốn truy xuất đến phần tử có chỉ số *i*, ta dùng **A[i]**.

- **Mảng 2 chiều**

Khai báo **A** là mảng 2 chiều có 10 dòng, 20 cột.

```
var A = new Array(10), i = 0;
```

```
for (i = 0; i < 10; i++)
```

```
    A[i] = new Array(20);
```

Để truy xuất đến phần tử có chỉ số dòng *i*, chỉ số cột *j* ta dùng **A[i][j]**.

3.3. Các kiểu dữ liệu trong JavaScript

Trong JavaScript thường sử dụng các kiểu dữ liệu sau:

3.3.1. Dữ liệu kiểu số

Kiểu số có hai loại thông dụng là kiểu số nguyên và kiểu số thực.

Ví dụ: `var a = 10, b = 100, ten = “Nguyen Van Ba”`

Các phép toán trên kiểu số

- `+`, `+=`, `-`, `-=`, `*`, `*=`, `/`, `/=`, `%` (chia lấy phần dư), `++` (phép tăng một đơn vị), `--` (phép giảm một đơn vị).
- Các phép so sánh: `<` (nhỏ), `<=` (nhỏ hơn hay bằng), `>` (lớn), `>=` (lớn hơn hay bằng), `=` (bằng), `!=` (khác).

3.3.2. Kiểu ký tự

Các ký tự được nằm giữa 2 nháy đơn. Ngoài ra còn có các ký tự đặc biệt sau đây:

Ký tự	ý nghĩa
\n	Xuống dòng mới
\t	Ký tự tab
\r	Về đầu dòng
\b	Ký tự khoảng trắng

Ví dụ: var ch= “A”, c= “B”;

Các phép toán trên ký tự

- +, += (cộng 2 ký tự)
- Phép toán so sánh : >, >=, <, <=, ==, !=

3.3.3. Kiểu chuỗi

Chuỗi là tập hợp các ký tự.

Một hằng chuỗi được nằm giữa hai dấu nháy đôi “

Ví dụ: var hoten= “Le Van Tam”;

Các phép toán trên chuỗi

- Phép nối chuỗi: +, +=
- Phép so sánh: <, <=, >, >=, ==, !=

3.3.4. Kiểu luận lý

Một biến có kiểu luận lý tồn tại 1 trong 2 trạng thái: true,false.

Ví dụ: var t = true, f = false;

Các phép toán trên kiểu luận lý

- Phép so sánh: <, <=, >, >=, ==, !=
- Phép logic: && (và), || (hoặc), ! (phủ định).

3.3.5. Kiểu ngày

Mô tả thông tin về: Ngày, Tháng, Năm, giờ, phút, giây của hệ thống.

Ví dụ: var now = new Date();

Các hàm lấy ngày giờ trong đối tượng Date như sau:

Tên hàm	Mô tả
getDate()	Ngày: 1..31
getDay()	Ngày trong tuần: 0 (chủ nhật), 1 (thứ 2)
getHours	Giờ: 0..23
getMinutes	Phút: 0..59
getMonth	Tháng: 0 (tháng 1)...11 (tháng 12)
getSeconds	Giây:0..59
getTime	Giờ theo mili giây
getFullYear	Năm

Ví dụ: Ví dụ lấy ngày hệ thống và hiển thị lên trình duyệt

Mã HTML
<pre><HTML> <HEAD> <TITLE>Outputting Text</TITLE> </HEAD> <BODY></pre>

```

<SCRIPT LANGUAGE="javascript">
<!--
var now = new Date();
var ngay = "";
ngay="hôm nay là ngày"+ now.getDate();
ngay+="tháng"+ now.getMonth();
ngay+="năm"+ now.getYear();
document.Write(ngay);
-->
</SCRIPT>
</BODY>
</HTML>

```

Ngoài các kiểu dữ liệu thông dụng trên còn có các kiểu dữ liệu **object**, **null**.

3.4. Các lệnh xử lý điều kiện rẽ nhánh

3.4.1. Cấu trúc if

Dùng để xử lý lệnh khi biểu thức của **if** trả về giá trị **true**

if (biểu thức điều kiện)

 Khối lệnh;

3.4.2. Cấu trúc if...else

if (biểu thức điều kiện)

 Khối lệnh 1;

else

 Khối lệnh 2;

Ví dụ:

if(a%2==0)

 document.Write(a, "là số chẵn");

else

 document.Write(a, "là số lẻ");

3.4.3. Cấu trúc switch...case

Cho phép thực hiện nhiều lựa chọn để ra quyết định thực thi.

switch(biến)

{

case giá trị 1:

 Khối lệnh 1;

break;

case giá trị 2:

 Khối lệnh 2;

break;

 ...

default:

 Khối lệnh n;

break;

}

3.5. Cấu trúc lặp

Dùng câu lệnh **break** để thoát khỏi cấu trúc lặp khi cần.

3.5.1. Cấu trúc for

for (*biểu thức khởi tạo; biểu thức điều kiện; biểu thức thay đổi*)

Khởi lệnh;

Khởi lệnh được thực hiện khi *biểu thức điều kiện* còn đúng.

Ví dụ: **for** (i=0;i<10;i++)

s+=2*i;

3.5.2. Cấu trúc While

While (*biểu thức điều kiện*)

Khởi lệnh;

Khởi lệnh được thực hiện khi biểu thức trong **While** còn đúng.

Ví dụ: i=0;

While(i<20)

{

s+=i;

i++;

}

3.5.3. Cấu trúc do...While

do

Khởi lệnh;

While(*biểu thức điều kiện*);

Khởi lệnh được thực hiện trước sau đó kiểm tra *biểu thức điều kiện* nếu còn đúng thì quay lên thực hiện khởi lệnh.

Ví dụ:

i=0;

do

{

s+=i;

i++;

} **While**(i<20);

3.6. Khai báo hàm

Dùng từ khoá **function** để khai báo hàm. Muốn trả về giá trị của hàm ta dùng từ khoá **return**.

function tên_hàm(*danh sách các tham số*)

{

Khởi lệnh;

}

Ví dụ:

function Add(x,y)

{

return(x+y);

}

var t;

document.**Write**(t)

Chương 4. sử dụng vbscript và javascript trong trang web

4.1. Chèn đoạn VBScript/JavaScript vào trang HTML

Các đoạn Script nằm giữa tag **<Script>** và **</Script>**, trong đó ghi rõ ngôn ngữ sử dụng để viết mã chương trình. Các đoạn **Script** được đặt giữa cặp tag **<HEAD>** và **</HEAD>** hay đặt giữa cặp tag **<BODY>** và **</BODY>**. Tuy nhiên, nếu đặt giữa tag **<HEAD>** thì các đoạn mã này đã được đọc và thông dịch trước các thành phần nằm trong tag **<BODY>**.

Dùng tag **<!--** và **-->** để báo cho trình duyệt không hiển thị các đoạn mã bên trong nếu nó không hiểu tag **<SCRIPT>**.

Ví dụ: Trong ví dụ sau ta viết một hàm tính diện tích hình vuông (viết bằng VBScript), một thủ tục tính diện tích hình chữ nhật (viết bằng JavaScript).

Mã HTML
<pre><HTML> <HEAD><TITLE>Tính diện tích </TITLE> <SCRIPT LANGUAGE="VBSCRIPT"> <!-- Function DienTichHinhVuong(a) dim tích tích=a*a DienTichHinhVuong=tích End Function Dim dt dt=DienTichHinhVuong(5) document.Write("Diện tích hình vuông=") document.Write(dt) --> </SCRIPT> </HEAD> <BODY>
 <SCRIPT LANGUAGE="JAVASCRIPT"> <!-- Function DienTichTron(R) { var dt; dt=3.14*R*R; return dt; } var D; D=DienTichTron(5); document.Write("Diện tích hình tròn=",D); --></pre>

</SCRIPT>
</BODY></HTML>
kết quả hiển thị trên trình duyệt
Dien tích hình vuong=25 Dien tích hình tron=78.5

4.2. Xuất/nhập dữ liệu trong VBScript và JavaScript

4.2.1. Xuất dữ liệu

Đối tượng **document** là đối tượng đại diện cho trang Web hiện hành. Còn đối tượng **window** thì đại diện cho cửa sổ mà trong đó trang web hiển thị.

Để xuất dữ liệu ra trang Web ta dùng hàm **write** và hàm **writeln** của đối tượng **document** theo cú pháp.

document.write("chuỗi cần hiển thị")

document.writeln("chuỗi cần hiển thị");

Chúng ta có thể dùng các tag HTML để xuất dữ liệu trong các đoạn **Script**

Ví dụ: **document.write**("<H2> Hello Script</H2>")

Để **writeln** (xuất dữ liệu và về đầu dòng mới) có tác dụng ta phải dùng kèm theo tag **<PRE>** và **</PRE>** đứng trước và sau đoạn **Script**.

Ví dụ:

<BODY>
<PRE>
<SCRIPT LANGUAGE="vbscript">
<!--
document.writeln("<H2>Hello JavaScript</H2>")
document.write("<H3>Hello VBScript</H3>")
-->
</SCRIPT>
</PRE>
</BODY>

Để hiển thị các hộp thông báo ta dùng hàm **confirm** và hàm **alert** của đối tượng **window**. Theo cú pháp sau:

window.alert("chuỗi cần hiển thị")

window.confirm("chuỗi cần hiển thị")

Đối với **VBScript** để hiển thị hộp thông báo ta dùng hàm **MsgBox**("chuỗi cần thông báo");

4.2.2. Nhập dữ liệu

Dùng hàm **prompt** của đối tượng **window** để nhập giá trị cho biến theo cấu trúc sau:

Biến = **window.prompt**("chuỗi thông báo", "trị mặc nhiên")

Biến = **InputBox**("chuỗi thông báo")

5. xử lý các sự kiện khi tương tác với các thành phần trên trang web

Sự kiện được phát sinh khi ta kích hoạt (**onClick**, **onmouseover**, **onmouseout**...) các thành phần trên trang web như các nút điều khiển **Button**, **hyperlink**, **ListBox**...

Để xử lý các sự kiện này ta tạo *hàm xử lý sự kiện* và gán *hàm xử lý sự kiện* đó cho *tên sự kiện*

<tên sự kiện>=<hàm xử lý sự kiện>

Ví dụ 1: Tạo 2 nút bấm (OK, Cancel), thủ tục xử lý sự kiện khi Click nút OK đ ược viết bằng **VBScript**, hàm xử lý sự kiện khi nhấn nút Cancel đ ược viết bằng ngôn ngữ **JavaScript**.

Mã HTML
<pre> <HTML> <HEAD><TITLE>Click Button </TITLE> <SCRIPT LANGUAGE="VBScript"> <!-- Sub SubOk() window.alert("Ban da bam nut OK") End Sub --> </SCRIPT> <SCRIPT LANGUAGE="JavaScript"> <!-- Function SubCancel() { window.alert("Ban da bam nut Cancel"); } --> </SCRIPT> </HEAD> <BODY> <INPUT TYPE=BUTTON NAME = "Ok" VALUE="OK" LANGUAGE="VBScript" onclick="SubOk()"> <INPUT TYPE=BUTTON NAME = "Cancel" VALUE="Cancel" LANGUAGE="JavaScript" onclick="SubCancel();"> </BODY> </HTML> </pre>
kết quả hiển thị trên trình duyệt

Ví dụ 2: Kiểm tra tính hợp lệ của dữ liệu nhập từ Form. Nếu thiếu th ì thông báo cho người dùng biết, ngược lại thông báo câu “dữ liệu đã đầy đủ”.

Mã HTML
<pre> <HTML> <HEAD> <TITLE>KIEM TRA DU LIEU NHAP </TITLE> </HEAD> <BODY> <SCRIPT LANGUAGE="JavaScript"> function KiemTraForm() </pre>


```

{
<!--lay doi tuong form, dienthongtin la ten Form
FormObj = document.dienthongtin;
if(FormObj.Ten.value=="")
{
    alert("Chua nhap Ten");
    FormObj.Ten.forcus();
    return false;
}
else if(FormObj.Tuoi.value=="")
{
    alert("Chua nhap Tuoi");
    FormObj.Tuoi.forcus();
    return false;
}
else if(FormObj.NgaySinh.value=="")
{
    alert("Chua nhap Ngay sinh");
    FormObj.NgaySinh.forcus();
    return false;
}
    alert("Thong tin day du");
}
</SCRIPT>
<FORM NAME="dienthongtin" method = POST>
Nhap ten
<INPUT TYPE=TEXT NAME = "Ten" SIZE=30><BR>
Nhap tuoi
<INPUT TYPE=TEXT NAME = "Tuoi" SIZE=30><BR>
Nhap ngay sinh nhat
<INPUT TYPE=TEXT NAME = "NgaySinh" S IZE=20 ><BR>
<INPUT TYPE=BUTTON NAME="btnSubmit" value="Chap nhan"
onClick="KiemTraForm()">&nbsp; &nbsp; &nbsp;
<INPUT type="RESET" NAME = "btnReset" value="Tu choi">
</FORM>
</BODY>
</HTML>

```

Kết quả hiển thị trên màn hình

Bài tập chương 3

Bài 1: Thiết kế **Form** nhập liệu như hình sau, khi nhấn chọn nút **chấp nhận** thì phải kiểm tra tính đầy đủ và hợp lệ của dữ liệu. Nếu thông tin nào không có hoặc bị sai thì

yêu cầu người dùng bỏ xung. Nếu người dùng nhấn chọn nút **không chấp nhận** thì làm rỗng tất cả các thông tin trên Form để chuẩn bị cho lần đăng ký kế tiếp.

Bài 2: Tương tự như bài 1, chúng ta thiết kế Form đặt hàng mua áo thun qua mạng như sau:

Khi khách hàng nhấn chọn nút **chấp nhận** thì phải kiểm tra tính đầy đủ và hợp lệ của dữ liệu. Nếu dữ liệu đúng thì thông báo câu “**Bạn đã đặt hàng thành công**”, nếu không hợp lệ thì yêu cầu người mua hàng điền thông tin lại cho hợp lệ. Trong trường hợp người đặt hàng nhấn chọn nút **bỏ qua** thì phải làm rỗng các thông tin trên Form để chuẩn bị cho lần đặt mua hàng kế tiếp.

Bài 3: Chúng ta thiết kế trang web đăng ký mail của Yahoo (<http://www.mail.yahoo.com>) khi nhấn vào nút Submit this Form thì phải kiểm tra tính đầy đủ và hợp lệ của dữ liệu. Nếu không đúng hay thiếu thì yêu cầu người dùng nhập lại.

Chương 4: lập trình web động với ngôn ngữ lập trình asp

1.giới thiệu về asp

ASp (Active Server Page) ngôn ngữ lập trình ứng dụng được chạy bên phía **Server**. Một trang **ASP** có các đặc điểm sau:

- Một trang **ASP** được lưu với phần mở rộng “**.asp**”
- Các ứng dụng **ASP** dễ viết, dễ sửa đổi.
- Cung cấp chế độ bảo mật tốt vì các mã code trong trang **ASP** người duyệt web không thể thấy được.
- Ngôn ngữ Script được dùng thông dụng nhất trong trang **ASP** là **VBScript**.
- Được hỗ trợ bởi trình chủ Web Server **IIS** (Internet information server) và **Personal Web Server (PWS)** là trình chủ web được dùng trên win98). Tuy nhiên, **IIS** là thông dụng nhất.

Ngoài **ASP** còn có một số ngôn ngữ lập trình web động như **JSP**, **PHP**...

Như vậy để thực thi một trang **ASP** ta cần cài trình chủ Web Server **IIS**.

Để cài **IIS**, sau khi đưa đĩa Win2K/WinXP vào ta chọn menu **Start-> Settings-> Control Panel-> Add/Remove Programs-> Add/Remove windows components -> chọn Internet Information Servers -> Next**

2. nạp một ứng dụng web lên trình chủ 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:

- **Bước 1:** Mở trình chủ web **IIS** bằng cách vào menu **Start -> Settings -> Control Panel -> Administrative Tools -> Internet Services Manager**.
- **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**.
 - 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.

- Đặt quyền cho thư mục ảo như trong hình 4.4. 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.
- 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.

- **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 4.5 ta vào mục **Document**. Bạn có thể xóa hoặc thêm một trang mặc định vào danh sách bằng cách chọn **Add** hay **Remove**.

- **Bước 4:** Để xem trang web ta mở trình duyệt web **Internet Explorer** -> gõ địa chỉ của 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>.

3. các khái niệm cơ bản về asp

3.1. Thành phần cơ bản của một trang ASP

Một trang ASP thông thường có 4 phần:

- Dữ liệu văn bản
- Các tag HTML
- Các đoạn mã chạy phía Client nằm trong đoạn tag <SCRIPT></SCRIPT>
- Các đoạn mã ASP được chạy phía Server nằm trong tag <% và %>

Như vậy một trang ASP là một trang HTML được nhúng thêm phần xử lý viết bằng mã HTML.

Ta có thể sử dụng nhiều ngôn ngữ Script khác nhau để viết trang ASP, vì thế ta phải chỉ định Script nào được sử dụng trong trang, bằng khai báo <%LANGUAGE=ScriptLanguage%> ở đầu trang. Ví dụ: Khai báo <%LANGUAGE=VBScript%> ở đầu trang để sử dụng ngôn ngữ VBScript.

Ví dụ: Trang ASP sau là sự kết hợp giữa các đoạn ASP, HTML và JavaScript. Đoạn chương trình sau xuất hiện ra màn hình câu thông báo “Good Morning” khi thời gian hệ thống ở thời điểm buổi sáng, còn thời gian buổi chiều thì in ra câu “Hello”

```
<%@language=VBScript%>
<HTML>
<BODY>
<%
    Dim dtmHour
    dtmHour=Hour(Now())
    if dtmHour<12 Then
```

```
%>
<B>Good Morning!</B>
<%
Else
%>
<B>Hello!</B>
<%
End if
%>
</BODY>
</HTML>
```

3.2. Nhập/Xuất dữ liệu

Đề 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**.

Ví dụ:

- Xuất chuỗi: Response.write"Learn ASP"
- Xuất hằng kiểu số: Response.write 5
- Xuất giá trị của biến a: Response.write a

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 4.1 và 4.2

3.3. Hoạt động của ASP

Khi một trang ASP được trình duyệt Web yêu cầu, trước tiên Web Server sẽ duyệt toàn bộ trang ASP này và chỉ thực hiện những câu lệnh kịch bản ASP, kết quả là một trang thuần HTML sẽ được đưa ra browser. Việc đưa ra kết quả cho browser lần lượt hay sau khi dịch xong tất cả các kịch bản là do người tạo lập trang Web qui định. Người dùng sẽ không thấy được các lệnh kịch bản của ASP bởi vì nó đã được server thực thi xong rồi gửi kết quả về cho browser dưới dạng trang HTML.

4. các đối tượng cơ bản 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 là :

- Các đối tượng cơ bản: Application, Session, Server, Request, Response, ObjectContext.
- Các thành phần (component) xây dựng sẵn: Dictionary, FileSystemObject, AdRotator, Browser Capabilities...

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:

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 Certificate**

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 cookies (kiểu chuỗi).
- key: khoá 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 gửi 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 khoá của cookie có tồn tại không.
 - + **Path**: (chỉ ghi- write only) nếu thuộc tính này được xác lập thì chỉ cookie chỉ được gửi cho những Request của đường dẫn này, nếu không thì cookie chỉ được gửi 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 khoá.

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 gửi 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>
    Chào bạn:
<%
    Response.Write(Request.Form("Ho_Lot"))
    Response.Write(" " & Request.Form("Ten"))
%>
</BODY>
</HTML>

```

- **QueryString**

Khi người dùng yêu cầu 1 trang 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 gửi 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 gửi 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>
    Chao ban:
<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 gửi, 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(username)
AUTH_USER	Trả về tên của người dùng (username)
CERT_COOKIE	Trả về ID duy nhất của client
CONTEXT_LENGTH	Trả về kích thước của dữ liệu mà client gửi
CONTEXT_TYPE	Trả về kiểu dữ liệu
GATEWAY_INTERFACE	
HTTP_<headername>	Trả về giá trị chứa trong header <i>headername</i>
HTTP_USER_AGENT	Trả về một chuỗi mô tả browser gửi yêu cầu
LOCAL_ADDR	Trả về địa chỉ của server mà browser gửi 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
```

```
%>
```

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...

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)
```

```
%>
```

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 gửi cho client. Đối tượng **Response** sẽ đảm nhiệm việc chuyển kết quả từ server cho client.

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.

4.2.2. Thuộc tính (property) của đối tượng Response

- **Buffer:** Dùng để xác định xem kết quả tạo ra bởi trang ASP có được giữ lại trong vùng đệm hay không. Thuộc tính **Buffer** nhận 1 trong 2 giá trị là true hoặc false. Nếu nhận giá trị 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 thức **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 gửi 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 gửi 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 gửi 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 = "text/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 **Expires** là -1.

Cú pháp:

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

Ví dụ: Nếu ta muốn cho một trang sẽ hết hạn sau 24 giờ (=1440 phút) ta đặt nh ư sau:

```
<%
Response.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:

```
<%  
Response.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 1 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 đầu 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
%>
```

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 gỡ bỏ nó ra.

Trong IIS 4.0, bạn phải gọi phương thức này trước bất kỳ kết quả nào gửi 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 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 gửi tới người  
dùng</P>  
<P>Bắt đầu xoá 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 gửi đã bị xoá bởi việc 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 gửi tới browser và người dùng có thể đọc được</P>  
<%  
Response.End  
%>  
<P>Đoạn văn bản này sẽ không được gửi 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 gửi tới người dùng ngay khi gọi phương thức
```

```

Flush</P>
<P>Một số đoạn văn bản khác sẽ được gửi sau một lúc nữa!!!</P>
<%
Response.Flush
Dim i
For i = 1 to 1000
    Response.Write " "
    Response.Write " Đây là đo ạn 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 hoạ 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>
</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 gửi 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>
```

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 đề khác: Web Server không biết bạn là ai và bạn 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 gửi 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 huỷ session đó khi người dùng không kết nối nữa hoặc khi session hết hạn.

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.StaticObject(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 StaticObject
</CENTER>
<%
Dim x
For each x in Session.Contents
Response.Write (x & "<br>")
Next
%>
</BODY>
</HTML>
```

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 ngẫu nhiên 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 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>
```

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 xoá 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 chỉ xóa tất cả các phần tử ra khỏi tập **Contents**.

Cú pháp:

Session.Contents.RemoveAll()

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** vì ở đây phương thức này không còn hiệu lực.

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. Do đó đối tượng Application nên chứa các thông tin mà có thể được truy cập bởi nhiều trang trong ứng dụng (nh ư 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.

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 **StaticObject** 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>")  
%>
```

4.4.2. Các phương thức của đối tượng **Application**

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 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!
```

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**

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.

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]
```

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 huỷ khi server xử lý 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 **ASPErrors** 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 gửi bất cứ nội dung gì xuống browser.

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+1
next
```

```
tong = tong/j
%>
</BODY>
</HTML>
```

- **HTMLEncode**

Phương thức này dùng để mã hoá 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 nội 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 gửi (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 hoá một chuỗi URL.

Cú pháp:

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

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>
```

Chương 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 tinc  
%>  
</BODY>  
</HTML>
```

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

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.asp** 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 các 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("Status") + "<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>Kiểm tra file global.asa</TITLE>
</HEAD>
<BODY>
<B>
<%
Response.Write "Nội dung của 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 global.asa để có thể các trang trong ứng dụng có thể triệu gọi các hàm này.

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 xóa 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** 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.

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 huỷ bỏ thực thể đó bằng cách:

```
Set Dic = nothing
```

7.2. Các thuộc tính của đối tượng Dictionary

□ ComapareMode

Ta dùng thuộc tính **ComapareMode** 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 khóa 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>

```

□ Exits

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

Cú pháp:

```
Dictionary.Exits (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.Exits("V") = true Then
Response.Write "Khóa 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
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 xóa 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 xóa tất cả các phần tử của đối tượng **Dictionary**.

Cú pháp:

```
Dictionary.RemoveAll
```

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 dụng đối tượng này để thao tác với các tập tin, thư mục,...

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 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

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 đĩa trên máy tính.

Cú pháp:

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

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

+ name: là tên cần gắn thêm vào Path

+ newpath: 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, de s [,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 phép 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.Copy "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 xóa một hoặc nhiều tập tin. Nếu tập tin không tồn tại thì sẽ xuất hiện lỗi.

Cú pháp:

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

☐ **DeleteFile**

Phương thức này xóa 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 xóa. Nếu nhận giá trị false thì các tập tin mang thuộc tính read-only sẽ không bị xóa.

☐ **DeleteFolder**

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

Cú pháp:

`FileSystemObject.DeleteFolder(foldername[,bReadOnly])`

☐ **DrivExits**

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

Cú pháp:

`FileSystemObject.DrivExits(drive)`

Trong đó drive là tên của ổ đĩa cần kiểm tra.

Ví dụ:

<HTML>

<BODY>

<%

Dim fso

Set fso = Server.CreateObject("Scripting.FileSystemObject")

If fso.DrivExits("C:") = true Then

 Response.Write ("Ổ đĩa C tồn tại!")

else

Response.Write("Ổ đĩa 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 tượng **Drive**.

Cú pháp:

FileSystemObject.GetDrive(Drive)

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

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

□ **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:

+ **WindowsFolder** 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 (TEM) 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 đi, 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 đi, 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
- + creat: 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.

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 ' Biến chứa đối tượng FileSystem
Dim ftxt ' Biến chứa đối tượng File
Dim sLine
Dim path
Dim uname_pass
uname = CStr(uname)
upass = CStr(upass)
uname_pass = uname & ":" & upass
CheckAccount = false ' Mặc định ban đầu
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>P lease enter passwor!</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
nvleng: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 và password thì một câu thông báo sẽ được hiện lên và bắt người dùng đăng nhập lại như sau:

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.

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")
```

9.2. Định dạng tập tin văn bản

```
REDIRECT URL
WIDTH 480
HEIGHT 100
BORDER 0
*
CITD.GIF
http://www.citd.edu.vn/
Đến với CITD
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.

9.3. Các thuộc tính của đối tượng AdRotator

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

Ví dụ:

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

9.4. Các phương thức của đối tượng AdRotator

GetAdvertisement: Phương thức này trả về nội dung HTML mà hiển thị mã quảng cáo trên trang web.

Cú pháp:

AdRotator.GetAdvertisement(path)

Trong đó path là đường dẫn tới tệp tin văn bản mô tả các mã quảng cáo.

Bài tập chương 4

Bài 1. Thiết kế và cài đặt trang web hiển thị máy tính tay với các phép toán: cộng (+), trừ (-), nhân (*), chia (/), lũy thừa (^), căn (), bình phương (x^2), nghịch đảo ($1/x$).

Bài 2. Viết mã code dùng mã số liên truy cập vào một trang web. Sau đó nhúng mã code đó vào trang web của bạn kiểm tra.

Bài 3. Tạo một câu hỏi nhúng mã số để dùng để login vào trang web.

Bài 4. Thiết kế và cài đặt trang web hiển thị lịch (calendar). Trang web phải cho phép người dùng xem lịch tháng của một năm nào đó, xem lịch của tháng trước, tháng kế tiếp.

Bài 5. Thiết kế và cài đặt trang web hiển thị hình ảnh cây thêu mã của máy chiếu.

Chương 5: giới thiệu ado và các kết nối cơ sở dữ liệu

1.giới thiệu

ADO (ActiveX Data Object) là một kỹ thuật mà Microsoft phát triển làm việc với các cơ sở dữ liệu, cung cấp các khả năng kết nối và xử lý trên cơ sở dữ liệu.

Vì cơ sở dữ liệu ADO truy cập và xử lý cơ sở dữ liệu trong trang ASP có thể chia làm các bước chính sau:

- Kết nối với cơ sở dữ liệu thông qua OLEDB hoặc ODBC
- Xây dựng câu truy vấn dữ liệu và yêu cầu thể hiện câu truy vấn thể hiện thao tác xử lý trên cơ sở dữ liệu
- Xử lý các kết quả từ câu truy vấn
- Đóng kết nối với cơ sở dữ liệu, giải phóng các tài nguyên hệ thống đã dùng.

2. Kết nối với cơ sở dữ liệu

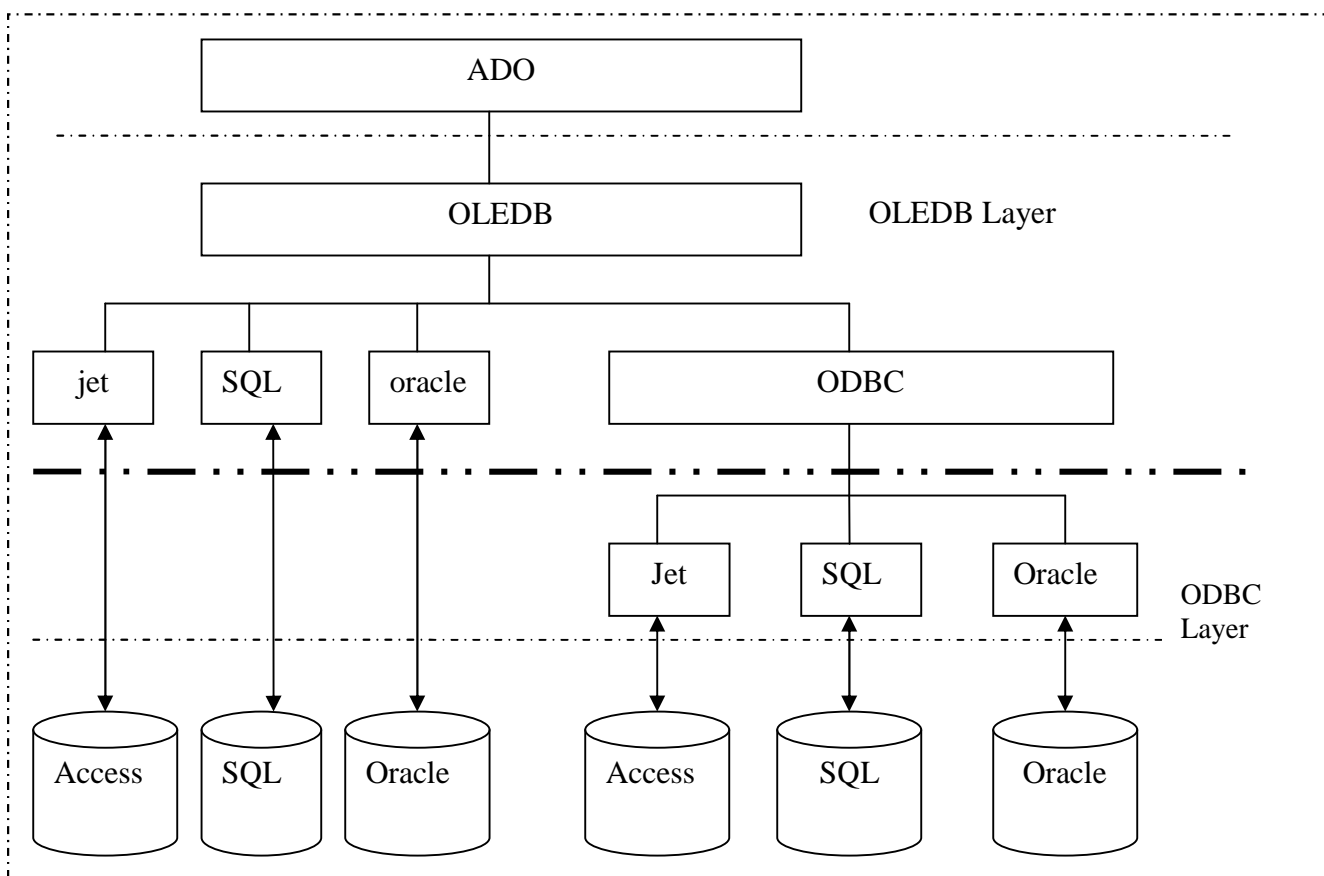
2.1. Tạo connection string thông qua OLEDB, ODBC

Có thể xử lý dữ liệu, bước đầu tiên chúng ta phải cung cấp các thông tin cần thiết để hệ thống biết đến nguồn truy cập đến cơ sở dữ liệu nào. Connection string là một chuỗi ký tự dùng lưu trữ thông tin về dữ liệu như sau:

- Thông tin về hướng đến cơ sở dữ liệu dùng trong cơ sở dữ liệu của bạn. Cơ sở dữ liệu của bạn dùng có thể MS Access, MS SQL Server hay Oracle...

- Thông tin về vị trí tệp cơ sở dữ liệu của bạn. Ví dụ nếu bạn dùng MS Access, bạn phải chỉ ra cơ sở dữ liệu của bạn nằm ở đâu trong tệp tin .mdb nào.

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



- Bảng sau liệt kê các OLE DB connection string cho một số hệ quản trị cơ sở dữ liệu thông dụng:

Data Source	OLE DB Connection String
Microsoft Access	provider=Microsoft.Jet.OLEDB.4.0; Data Source= <input type="text"/> tin.mdb
Microsoft SQL Server	Provider=SQLOLEDB.1; Data Source= <input type="text"/> database trên server
Oracle	Provider=MSDAORA.1; Data Source= <input type="text"/> database trên server
Microsoft Indexing Service	Provider=MSIDXS.1; Data Source= <input type="text"/>

- Bảng sau liệt kê các ODBC connection string cho một số hệ quản trị cơ sở dữ liệu thông dụng:

Data Source Driver	ODBC Connection String
Microsoft Access	Driver={Microsoft Access Driver (*.mdb)}; DBQ= <input type="text"/> tin.mdb
Microsoft SQL Server	Driver={SQL Server};SERVER= <input type="text"/> database trên server
Oracle	Driver=MSDAORA.1; Data

	Source= ng d n n database trên server
Microsoft Exel	Driver={Microsoft Exel Driver(*.xls)};DBQ= ng d n n t p tin.xls; DriverID=26
Microsoft Exel 97	Driver={Microsoft Exel Driver(*.xls)};DBQ= ng d n n t p tin.xls; DriverID=790
Paradox	Driver={Microsoft Paradox Driver(*.db)};DBQ= ng d n n t p tin.db; DriverID=26
Text	Driver={Microsoft Text Driver(*.txt;*.csv)};DBQ= ng d n n t p tin.xls; DefaultDir= ng d n n *.txt
Microsoft Visual FoxPro (with a database container)	Driver={Microsoft Visual FoxPro Driver}; SourceType=DBC;Sourcedb= ng d n n t p tin.dbc
Microsoft Visual FoxPro (without a database container)	Driver={Microsoft Visual FoxPro Driver}; SourceType=DBF;SourceDb= ng d n n t p tin.dbf

2.2. Tạo connection string thông qua DSN

- Ta có thể xây dựng connection strings bằng cách tạo Data Source Name (DSN) trong ODBC. Một DSN chứa các thông tin sau:
 - Tên của DSN
 - Tập tin cơ sở dữ liệu mà nó trỏ tới
 - Chọn các driver kết nối với tập tin cơ sở dữ liệu
 - UserID và password để truy xuất data store
 - Các thông tin cần thiết khác cho việc kết nối

DSN có ba loại: User, System và File. User DSN bị giới hạn trong phạm vi người tạo. Một user đăng nhập vào mạng sẽ không thấy DSN của các user khác. System DSN được lưu trên registry và được nhìn thấy bởi tất cả người dùng trên máy cục bộ bao gồm các dịch vụ NT. File DSN chứa nội dung tương đương nhưng dưới dạng tập tin văn bản chứ không phải trong registry

-Các bước tạo DSN cho cơ sở dữ liệu Access

1. Mở cửa sổ điều khiển ODBC và chọn System DSN. Vào Control Panel-> 32 bit ODBC (window 98)

Vào Control Panel -> Administrator Tools -> Data Source (ODBC) (Windows 2000 hoặc Windows XP)

2. Nhấp vào Add để tạo DSN mới. Chọn từ danh sách driver, driver bạn cần sử dụng cho cơ sở dữ liệu của mình. ở đây là Access.

3. Đặt tên DSN

4. Chỉ ra cơ sở dữ liệu mf DSN trở tới bằng cách nhấp Select v à duyệt rồi chọn cơ sở dữ liệu.

Đối với Windows 9X, mọi người đều có thể cấu hình DSN

Đối với Windows NT/2000 (Professional hoặc Server) thì chỉ có quyền Administrator hay uỷ quyền Admin mới có thể cấu hình DSN.

Sau khi click nút Add để Add DSN, chọn Driver Microsoft Access.

Đặt tên Data Source Name và nhấn nút Select chỉ ra tập tin *.mdb chứa cơ sở dữ liệu MS Access

- Các bước tạo DSN cho cơ sở dữ liệu SQL Server

Chúng ta cần phải biết một số thông tin sau trước khi tạo DSN cho cơ sở dữ liệu SQL Server

- o Địa chỉ mạng của database server
- o Kiểu mạng (network type) của server (Name pipes, TCP/IP...)
- o Phương thức bảo mật của server (NT hay SQL Server)
- o Một tài khoản hợp lệ khi đăng nhập vào server
- o Tên của cơ sở dữ liệu trên server

Đặt tên cho data source name và chỉ ra tên Server chứa database trên SQL Server

Chỉ ra login ID và password của database cần nối kết. Có thể chỉnh lại các cấu hình các kiểu kết nối mạng đến máy server nối chứa dữ liệu bằng cách nhấn Client Configuration

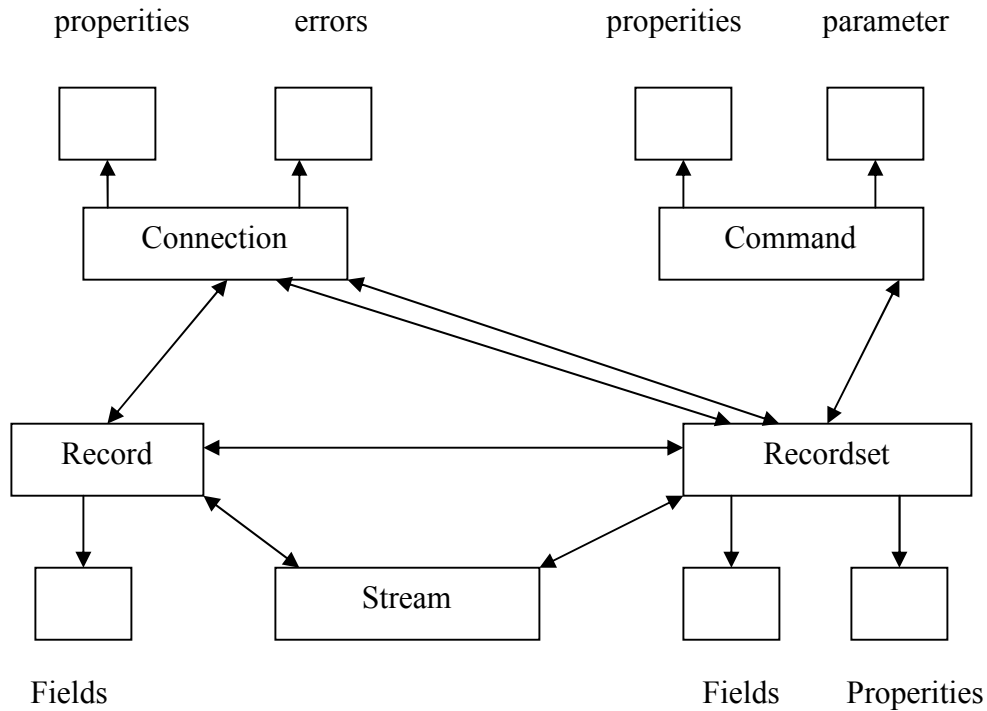
Chọn ra database cần nối kết từ list box

Sau khi đã hoàn tất các bước trước ta nhấn nút Finish. Nhấn nút Test Data Source để kiểm tra xem việc tạo DSN đã thành công hay chưa và các kết nối với cơ sở dữ liệu có thực thi được hay không.

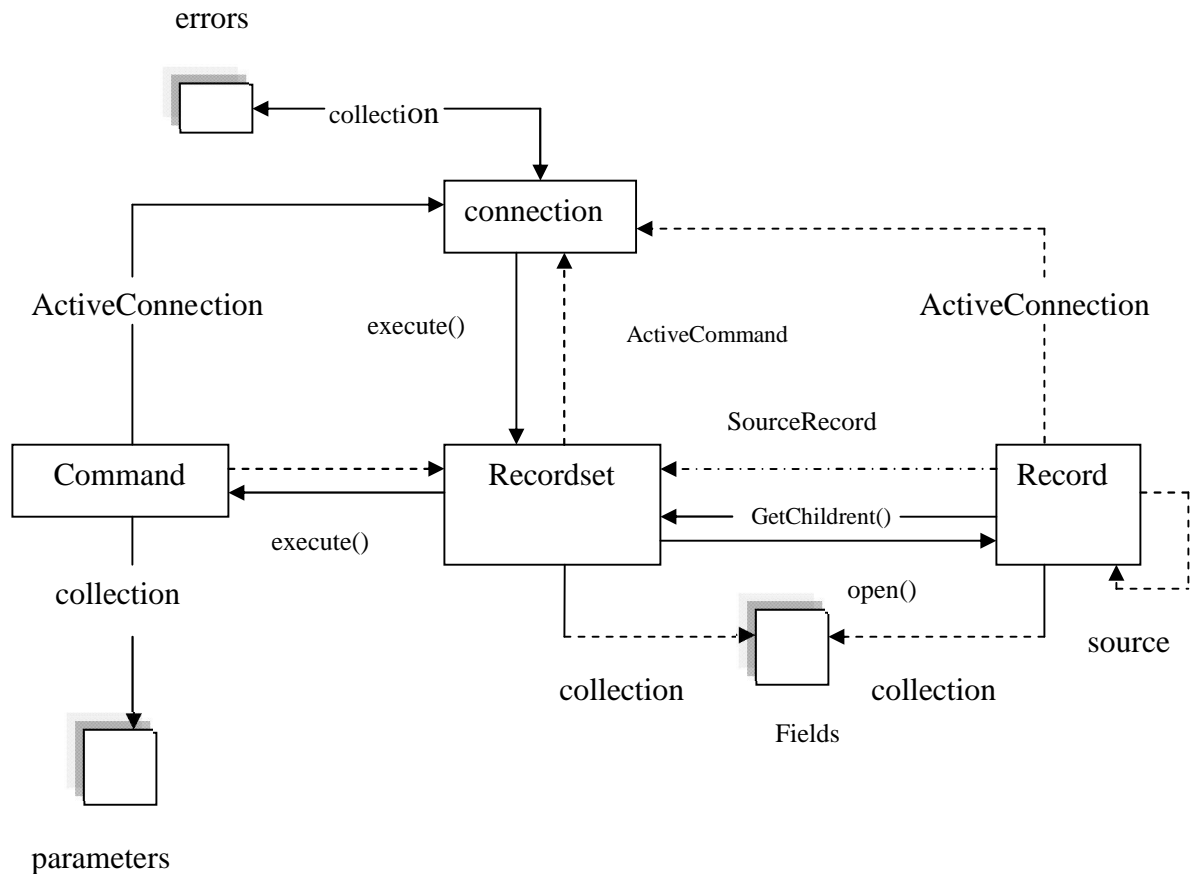
Nếu kết nối thành công ta có thông báo như sau:

3. Các đối tượng của ADO

Mô hình các đối tượng ADO và mối liên hệ đối tượng



ADO có các đối tượng Connection, Command, Recordset, Record, Stream và các tập hợp Errors Fields, Properties, Parameter như hình trên. Các đối tượng trong ADO và các mối liên hệ giữa chúng được thể hiện như hình bên dưới. Dựa trên mô hình đối tượng của ADO sẽ triển khai và mô tả một cách chi tiết các đối tượng, các thuộc tính và các phương thức chức năng của mỗi đối tượng được tạo và sử dụng cho việc kết nối, truy vấn cơ sở dữ liệu và hiển thị dữ liệu thông qua các đối tượng ADO để hiển thị các trường của mẫu tin, hay kết quả câu truy vấn lên trang web ASP.



- **Đối tượng Connection**
 Đối tượng cho phép bạn thiết lập kết nối với cơ sở dữ liệu. Qua đó, nó cung cấp các phương thức để truy vấn dữ liệu từ cơ sở dữ liệu. Nó chứa ba thông tin:
 - Cơ sở dữ liệu
 - Giao thức (driver/provider) trao đổi thông tin với cơ sở dữ liệu
 - username và Password
 - **Đối tượng Command**
 Đối tượng để thực thi 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 là các stored procedure. Đối tượng này thường dùng để thực thi các câu lệnh SQL không trả về kết quả.
 - **Đối tượng Recordset**
 Đối tượng chứa tập hợp các dữ liệu được rút ra từ cơ sở dữ liệu. Nó cho phép thay đổi dữ liệu như thêm, xóa, sửa dữ liệu, hay di chuyển dữ liệu.
 - **Đối tượng Record**
 Đối tượng Record lưu trữ một hàng (mẫu tin) trong Recordset, một tệp tin hay một tệp tin trong File System.
 - **Đối tượng Stream**
 Đối tượng để quản lý dữ liệu dạng binary, nó dùng để quản lý dữ liệu BLOB (Binary Large Object) như hình ảnh hay âm thanh.
- ### 3.1. Đối tượng Connection

3.1.1 Kết nối cơ sở dữ liệu qua đối tượng Connection

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

Có thể 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 dùng để xác định cơ sở dữ liệu.

Ví dụ: minh họa cho việc tạo kết nối cơ sở dữ liệu của MS Access

```
<%  
strDSN="Driver={Microsoft Access  
Driver (*.mdb)}; DBQ=C:\path\filename.mdb"  
' Tạo connection string  
Set Conn=server.CreateObject("ADODB.Connection")  
' Mở kết nối qua connection string  
Conn.Open strConn  
>%
```

Khi cần có một kết nối riêng cho từng các trang ta có thể thiết lập một biến cho đối tượng Connection bằng cách viết các thụt lề sau trong tệp tin global.asa

- ở mức **application**:

```
<%  
Sub Application_OnStart()  
Set objConn=Server.CreateObject("ADODB.Connection")  
End Sub  
>%
```

- ở mức **Session**:

```
<%  
Sub Session_Start()  
Set objConn=Server.CreateObject("ADODB.Connection")  
End Sub  
>%
```

3.1.2. Thao tác dữ liệu thông qua đối tượng Connection

Đối tượng Connection cung cấp phương thức Execute để thực hiện một câu lệnh truy vấn. Thao tác trên cơ sở dữ liệu trực tuyến tạo instance cho đối tượng Connection, xây dựng câu truy vấn SQL và yêu cầu gửi lệnh thực hiện thông qua phương thức Execute của đối tượng Connection.

Cú pháp:

objConn.Execute CommandText, RecordAffected, Options

CommandText: Câu lệnh SQL, tên bảng hay Stored Procedure

Options: Quy định loại CommandText

Hạng	Giá trị	Loại của CommandText
adCmdUnknown	0	Mặc định, không xác định loại CommandText
AdCmdTable	1	CommandText là câu lệnh

		SQL
adCmdTable	2	CommandText là tên bảng
adCmdStore	4	CommandText là stored procedure hay câu truy vấn

☐ Ví dụ minh họa cách hiển thị truy vấn mẫu tin vào cửa sổ dữ liệu

```
<%
Dim objConn
Set objConn=Server.CreateObject("ADODB.Connection")
objConn.Open "DSN=CITDDB"
Dim sqlText
SqlText="SELECT * FROM STUDENTS"
Set objRS=objConn.Execute(sqlText)
%>
```

☐ Ví dụ minh họa cách thêm mẫu tin vào cửa sổ dữ liệu

```
<%
Dim objConn
Set objConn=Server.CreateObject("ADODB.Connection")
objConn.Open "DSN=CITDDB"
Dim sqlText
SqlText="INSERT INTO STUDENTS (HOTEN, NAMSINH, LOP, DIACHI,
PHONE) VALUE ('Nguyen Thang','14/08/1974', 'CV=012003', '78 Truong Dinh
TPHCM', '9317442')"
```

```
Set objRS=objConn.Execute(sqlText)
%>
```

☐ Các câu lệnh DELETE, UPDATE cũng có thể hiển thị kết quả

```
<%
Dim objConn
Set objConn=Server.CreateObject("ADODB.Connection")
objConn.Open "DSN=CITDDB"
Dim sqlText
SqlText="UPDATE STUDENTS SET DIACHI='34 Truong Dinh TPHCM',
NAMSINH='07/06/1977' WHERE HOTEN='Nguyen Thang'"
Set objRS=objConn.Execute(sqlText)

%>
<%
```

```
Dim objConn
Set objConn=Server.CreateObject("ADODB.Connection")
objConn.Open "DSN=CITDDB"
Dim sqlText
SqlText="DELETE STUDENTS WHERE HOTEN='Nguyen Nam' AND
NAMSINH='07/06/1977'"
Set objRS=objConn.Execute(sqlText)
%>
```

3.2. Thao tác cơ sở dữ liệu bằng đối tượng Command

Để có thể thao tác và thay đổi cơ sở dữ liệu bằng đối tượng Command, ta phải thực hiện các bước sau trước khi sử dụng đối tượng Command:

- ☐ Khai báo và khởi tạo thể hiện (instance) của đối tượng Command
- ☐ Khởi tạo các thuộc tính đối tượng

Trong đó thuộc tính của đối tượng Command như sau:

ActiveConnection	chứa instance đối tượng Connection đã được khai báo
CommandText	chứa câu lệnh SQL hay tên bảng
CommandType	chứa các thuộc tính quy định cho CommandText
CommandTimeout	Thời gian để thực hiện câu lệnh, nếu việc thực hiện vượt quá thời gian đã định này sẽ có thông báo lỗi
Prepared	True/false. True cho phép biên dịch trước thực thi câu lệnh và false ngược lại
Execute	Thực thi câu lệnh

Giá trị của thuộc tính **CommandType**:

thuộc tính	mô tả
adCmdText	Câu lệnh SQL
adCmdTable	tên bảng
adCmdStoreProc	Store Procedure hay câu lệnh query
adCmdUnknown	giá trị mặc định

Ví dụ: minh họa sử dụng đối tượng Command

```
<%
strDSN="Driver={Microsoft Access Driver (*.mdb)};
DBQ=C:\path\filename.mdb"
Set Conn=server.CreateObject("ADODB.Connection")
' Khởi tạo các tạo đối tượng Command
Dim objCmd
Set ObjCmd=Server.CreateObject("ADODB.Command")
objCmd.ActiveConnection=objConn
sqlText="INSERT INTO Customers(FirstName,LastName)
Values('Truc','Nguyen')"
objCmd.CommandText=sqlText
objCmd.CommandType=adCmdText
objCmd.CommandTimeout=30 ' tính bằng giây
objCmd.Prepared=false
objCmd.Execute
%>
```

3.3. Xử lý dữ liệu thông qua đối tượng Recordset

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

	tương liên quan
Delete	xoá record hay một tập record hiện hành
Find	tìm một record thoả mãn 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	xoá đối tượng Recordset hiện hành và trả về đối tượng recordset kế tiếp
Open	mở một record
Requery	cập nhật lại dữ liệu bằng cách thực hiện lại câu lệnh truy vấn ban đầu
Resync	Refresh lại dữ liệu trong đối tượng Recordset hiện hành
Save	lưu Recordset xuống file
seek	tìm chủ 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 thức này một cách cụ thể.

1. Phương thức Open

Cú pháp:

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

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

Tham số CursorType:

Hằng số	giá trị	Chức năng
adOpenForwardOnly	0	truy vấn 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 xoá 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 khoá

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

Tham số Options:

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 thức 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 xoá mẫu tin trong Recordset

Cú pháp: objRs.Delete

hay objRs.Delete <tham số>

Tham số	Mô tả
AdAffectCurrent	xoá mẫu tin hiện hành
AdAffectGroup	xoá mẫu tin thoả điều kiện lọc

Ví dụ: Xóa sinh viên mang họ tên ‘Nguyen Thang’
 objRs.Find “HOTEN=’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  
%>
```

3.3.1. Lưu trữ dữ liệu trả về

ADO sử dụng đối tượng Recordset để lưu trữ kết quả trả về từ câu truy vấn dữ liệu SELECT. Vì kết quả trả về của một câu truy vấn SELECT có thể có nhiều mẫu tin, cho nên có thể xem Recordset như là một mảng các mẫu tin.

Thông thường có 2 cách để lấy dữ liệu từ câu truy vấn đặt vào biến Recordset

Thực hiện phương thức Execute của đối tượng Connection cho câu lệnh truy vấn và trả về kết quả cho Recordset. Ví dụ như **Set rs=Conn.Execute(strSQL)**

Tạo một thể hiện (instance) cho đối tượng Recordset và sử dụng phương thức Open, kết hợp với đối tượng Connection đã tạo

3.3.2. Hiện thị dữ liệu trả về

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

Ví dụ minh họa thể hiện dữ liệu của tất cả mẫu tin có họ tên bắt đầu là chữ ‘Nguyen’ của bảng STUDENTS

```
<%  
Dim objConn  
Set objConn=Server.CreateObject(“ADODB. Connection”)  
objConn.Open “DSN=CITDDB”  
Dim sqlText  
SqlText=”SELECT * FROM STUDENTS WHERE HOTEN LIKE ’Nguyen%’”  
Set objRS=objConn.Excute(sqlText)
```

```

‘STUDENTS(HOTEN,NAMSINH,LOP)
Response.Write “&nbsp;” “Họ tên” & “&nbsp;” & “Năm Sinh” & “&nbsp;” &
“Lớp”
Do until objRs.EOF
Response.Write i & “&nbsp;” & objRs.Fields(“HOTEN”)
& “&nbsp;” & objRs.Fields(“NAMSINH”) & “&nbsp;”
& objRs.Fields(“LOP”)
i=i+1
objRs.MoveNext
Loop
%>

```

3.4. Đối tượng Record

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

Để sử dụng đối tượng này ta phải khai báo thể hiện (instance) cho đối tượng record

Dim objRec

Set objRec=Server.CreateObject(“ADODB.Record”)

Đối tượng Record có các phương thức:

Phương thức	Mô tả
Cancel	hủy việc thực hiện trên record
Close	đóng một đối tượng Record
CopyRecord	copy một file hay thư mục
DeleteRecord	xoá một file hay thư mục
GetChildren	trả về một đối tượng Recordset, mỗi dòng của Recordset lưu trữ tập tin trong thư mục
MoveRecord	di chuyển một file hay một thư mục
Open	Mở một đối tượng Record tồn tại, tạo mới một file, hay tạo mới một thư mục

1. Phương thức Open

Sau khi tạo một instance của đối tượng Record, ta có thể dùng phương thức này để open, tạo mới một file hay tạo mới một thư mục

Cú pháp:

objRec.Open (Source, ActiveConnection,[Mode],[CreateOption],[Option],[Username],[Password])

Ví dụ:

- Source là URL chỉ định địa chỉ:


```

<%
Dim objRec
Set objRec=Server.CreateObject(“ADODB.Record”)
objRec.Open “URL=http://localhost/sinhvien”
objRec.Open “readme.txt”,objConn
%>

```
- Source là một hàng (row) hiện tại trong Recordset


```

<%

```

```

Set objRs=Server.CreateObject("ADODB.RecordSet")
Set objRs=Server.CreateObject("ADODB.Record")
objRs.Open "SINHVIEN","DSN=CITDDB"
objRec.MoveFirst
objRec.Open objRs
%>

```

2. Phương thức CopyRecord:

Phương thức CopyRecord dùng để chép một file hay một tập tin từ nơi này sang nơi khác

Cú pháp:

objRec.CopyRecord (Source, Destination, username, psword, opt, async)

- Source: Tên hay tập tin cần copy
- Destination: vị trí cần copy đến
- username: userID có quyền truy cập vào Destination
- Psword: mật khẩu
- opt: chế độ Copy. Mặc định là adCopyUnspecified. Nếu bạn muốn ghi đè có thể dùng adCopyOverWrite.
- async: nếu là true, thì sẽ thực hiện ngầm

Tương tự cho phương thức **MoveRecord**

3. Phương thức DeleteRecord:

Phương thức DeleteRecord cho phép xóa một file hay một tập tin nào đó.

Cú pháp:

objRec.DeleteRecord (Source, Async)

3.5. Đối tượng Stream

Đối tượng Stream dùng để lưu trữ luồng dữ liệu dạng text hoặc nhị phân

Trước khi sử dụng đối tượng này cần phải khai báo thể hiện (instance):

```
Set objStream = Server.CreateObject(ADODB.Stream)
```

Các phương thức của đối tượng Stream:

Phương thức	Mô tả
Close	Đóng đối tượng Stream
CopyTo	Chép một chuỗi ký tự hoặc bytes từ đối tượng Stream này sang đối tượng Stream khác
Flush	Gửi nội dung của đối tượng Stream
LoadFromFile	Lấy nội dung của một đối tượng vào đối tượng Stream
Open	Mở một đối tượng Stream từ URL hay đối tượng Record
read	Đọc số lượng bytes trong đối tượng Stream đã lưu trữ
ReadText	Đọc ký tự trong đối tượng Stream chứa nội dung là text
SaveToFile	Lưu nội dung đối tượng Stream xuống tệp tin
SetEOS	Thiết lập thuộc tính của EOS để biết vị trí hiện tại
SkipLine	Bỏ qua một dòng khi làm việc với TextStream

Write	ghi m t s i t ng bytes c a d li u nh phn vào i t ng Stream
WriteText	ghi d li u d ng Text vào i t ng Stream

4. Tập hợp errors

X lý l i là ph n quan tr ng trong vi c t o các ng d ng. B n c n ph i x lý khéo léo các l i d oán s x y ra c ng nh c n có m t c ch x lý nh ng l i không oán tr c c. i u này cho phép b n ki m soát dòng ch ng trình khi có l i x y ra, cung c p cho ng i dùng cu i nh ng thông báo l i có ý nghĩa h n và xu t cách x lý.

Mô hình i t ng ASP không h tr ch c n ng x lý l i, công vi c này do i t ng Error có s n c a ngôn ng k ch b n th c hi n nh VBScript. i t ng Error b h n ch vì nó ch h tr phát bi u “**On Error Resume Next**”

Thu c tính	Mô t
Description	Tr v ph n mô t l i
HelpContext	Tr v ID c a ch trong ph n h th ng tr giúp
HelpFile	Tr v ng d n và t p tin tr giúp
Nativeerror	Tr v mã l i t provider hay data source
Number	Tr v s hi u l i
Source	Tr v tên i t ng hay ng d ng ngu ng c l i
SQLState	Tr v 5 ký t mã l i SQL

Ví dụ: Hi n th thông báo l i khi k t n i n c s d li u b l i

```
<html>
<body>
<%
set objconn=Server.CreateObject("ADODB.Connection")
objconn.Provider="Microsoft.Jet.OLEDB.4.0"
on error resume next
objconn.Open(Server.MapPath("northwind1.mdb"))
if objconn.Errors.Count >0 then
set objErr=Server.CreateObject("ADODB.Error")
%>
<%
for each objErr in objConn.Errors
if objErr.Number<>0 then
response.Write("<p>")
response.Write("Description:")
response.Write("objErr.Description & <br />")
response.Write("Help context:")
response.Write(objErr.helpContext & "<br />")
response.Write("Help File:")
response.Write("objErr.HelpFile & "<br />")
response.Write("Native error")
response.Write(objErr.Nativeerror & "<br />")
```

```

response.Write("Error number")
response.Write(objErr.Number & "<br />")
response.Write("Error source")
response.Write(objErr.Source & "<br />")
response.Write("SQL state:")
response.Write(objErr.SQLState & "<br />")
response.Write("</p>")
End if
Next
End if
objconn.close
%>
</body>
</html>

```

5. stored procedure

Stored Procedure là tập các câu lệnh T-SQL được biên dịch trước, nó có nhiều ưu nhược điểm như sau vì vậy sử dụng các câu lệnh SQL bên trong chương trình sẽ có lợi:

- Thực hiện nhanh hơn, tiết kiệm thời gian khi thực hiện công việc. Giảm số lần truy cập các gói dữ liệu trên mạng
- Có thể dễ dàng truy cập tham số vào Stored Procedure và truy xuất dữ liệu
- Nâng cao tính bảo mật thông qua việc chỉ định quyền truy cập đối với Stored Procedure

Cú pháp:

```

CREATE PROC[EDURE] procedure_name { :number }
[ { @parameter data_type } [ VARYING ] = default ]
[ OUTPUT ] [, ...n ]
[ WITH ] { RECOMPILE | ENCRYPTION | RECOMPILE,
EMCRYPTION }
[ FOR REPLICATION ]
As sql_statement [ ...n ]

```

Ví dụ:

- Stored procedure không có tham số; tạo stored procedure hiển thị mã số, tên và địa chỉ của sinh viên

```

CREATE PROCEDURE sp_Sinhvien
AS
SELECT MASV, HOTEN, DIACHI, NAMSINH
FROM SINHVIEN

```

Bạn có thể sử dụng Query Analyser trong SQL SERVER để thực thi stored procedure trên

```
EXEC sp_Sinhvien
```

- Stored procedure có tham số: tạo stored procedure tìm kiếm thông tin của sinh viên, tham số truy cập vào là mã số sinh viên (MSSV)

```

CREATE PROCEDURE sp_TimSinhvien
(@MSSV varchar(15))
AS

```

```
SELECT MSSV, HOTEN, DIACHI, NAMSINH
FROM SINHVIEN
WHERE (MASV=@MSSV)
```

Stored Procedure với đối tượng Command:

Chương trình minh họa: stored procedure cung cấp thông qua đối tượng Command

```
<%@LANGUAGE=VBScript%>
<HTML>
<HEAD>
<TITLE>Stored procedure qua đối tượng ADO</TITLE>
</HEAD>
<BODY>
<%
Dim MSSV
MSSV='CV-01-2003'
Set objConn=Server.CreateObject("ADODB.Connection")
objConn.Open "DSN=CITDDB",sa
Set objCmd=Server.CreateObject("ADODB.Command")
Set objCmd.CommandText="sp_TimSinhvien '
"& CStr(MSSV) &"' "
Set objRS=objCmd.Execute
Response.Write("Thông tin của sinh viên có mã số: " & MSSV & "<BR>")
Response.Write("<TABLE BORDER=1>")
Do Until Not objRS.EOF
Response.Write("<TR>")
Response.Write("<TD>" & objRS("MSV") & "</TD>")
Response.Write("<TD>" & objRS("HOTEN") & "</TD>")
Response.Write("<TD>" & objRS("NAMSINH") & "</TD>")
Response.Write("<TD>" & objRS("DIACHI") & "</TD>")
Response.Write("</TR>")
objRS.MoveNext
Loop
Response.Write("</TABLE>")
objRS.Close
Set objRS=Nothing
Set objConn=Nothing
%>
</BODY>
</HTML>
```

Stored procedure với đối tượng Recordset:

Chương trình minh họa: stored procedure cung cấp thông qua đối tượng Command

```
<%@LANGUAGE=VBScript%>
<HTML>
<HEAD>
<TITLE>Stored procedure qua đối tượng ADO</TITLE>
</HEAD>
```

```

<BODY>
<%
Dim MSSV
MSSV='CV-01-2003'
Set objConn=Server.CreateObject("ADODB.Connection")
objConn.Open "DSN=CITDDB",sa
sqlText="sp_TimSinhvien ' "&CStr(MSSV) &" ' "
Set objCmd=Server.CreateObject("ADODB.Recordset")
objRs.Open sqlText, objConn
Response.Write ("Thông tin c  a sinh viên có mã s : " & MSSV & "<BR>")
Response.Write ("<TABLE BORDER=1>")
Do Until Not objRS.EOF
Response.Write("<TR>")
Response.Write("<TD>" & objRS("MSV") & "</TD>")
Response.Write("<TD>" & objRS("HOTEN") & "</TD>")
Response.Write("<TD>" & objRS("NAMSINH") & "</TD>")
Response.Write("<TD>" & objRS("DIACHI") & "</TD>")
Response.Write("</TR>")
objRS.MoveNext
Loop
Response.Write("</TABLE>")
objRS.Close
Set objRS=Nothing
Set objConn=Nothing
%>
</BODY>
</HTML>

```

6. xÂY DỰNG WEBSITE BẰNG ASP

Trong ph n này s minh ho m t s ví d vi c s d ng ADO k t n i v à thao tác d li u, và m t s bài t p

Ví dụ 1: Chúng ta mu n hi n th ch nh ng m u tin t b ng "Customers" trong database northwind có CompanyName b t u v i ch A (chú ý l u t p tin v i tên m r ng .asp):

```

<HTML>
<BODY>
<%
Set conn=Server.CreateObject("ADODB.Connection")
conn.Provider= "Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"
Set rs=Server.CreateObject("ADODB.Recordset")
sql=" SELECT Companyname, Contactname FROM
Customers
WHERE CompanyName LIKE 'A%'"
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>

```

Kết quả hiển thị như sau:

Ví dụ 2: Hiển thị danh sách khách hàng sau khi chọn là tên của khách hàng (chú ý lưu ý tên file là r ng.asp), danh sách các nước có thể Country, tên khách hàng có thể Customers trong database northwind:

Kết quả hiển thị như sau:

```

<HTML>
<BODY>
<%
Set conn=Server.CreateObject("ADODB.Connection")
conn.Provider= "Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.MapPath("northwind.mdb"))
Set rs=Server.CreateObject("ADODB.Recordset")
sql=" SELECT DISTINCT Country FROM Customers ORDER BY Country"
rs.Open sql, conn
country=request.form("country")
%>
<form method="post">
Choose Country<select name="country">
<%do until rs.EOF%>
response.write("<option>")
if rs.fields("country")=country then
response.write("selected")
end if
response.write(">")
response.write(rs.fields("Country"))

```

```

rs.MoveNext
Loop
rs.Close
set rs=Nothing%>
</select>
<input type="submit" value="Show customers">
</form>
<%
if country<>"" then
sql="SELECT Companyname, Contacname, Country FROM Customers WHERE
country=" & country & ""
set rs=Server.CreateObject("ADODB.Recordset")
rs.Open sql, conn
%>
<table width="100%" cellpadding="2" border="1">
<tr>
<th>Companyname</th>
<th>Contactname</th>
<th>Country</th>
</tr>
<%
do until rs.EOF
response.write("<tr>")
response.write("<td>" & rs.fields("companyname") & "</td>")
response.write("<td>" & rs.fields("contactname") & "</td>")
response.write("<td>" & rs.fields("country") & "</td>")
response.write("</tr>")
rs.MoveNext
loop
rs.close
conn.Close
set rs=nothing
set conn=nothing
%>
</table>
<%end if%>
</BODY>
</HTML>
Ví dụ 3: Hi n th thông tin khách hàng t b ng Customers, ta s d ng ph ng th c
QueryString c a i t ng Request l y thông s sort truy n vào, d a vào thông s
này s p x p th t theo tên tr ng c truy n vào, l u thành t p tin
demo_sort_3.asp:
<HTML>
<BODY>
<table border="1" width="100%" bgcolor="#fff5ee">

```

```

<tr>
<th align="left" bgcolor="#b0c4de">
<a href="demo_sort_3.asp?sort=companyname">Company</a>
</th>
<th align="left" bgcolor="#b0c4de">
<a href="demo_sort_3.asp?sort=contactname">Contact</a>
</th>
</tr>
<%
if request.querystring("sort")<>"" then
sort=request.querystring("sort")
else
sort="companyname"
end if
Set conn=Server.CreateObject("ADODB.Connection")
conn.Provider= "Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.MapPath("northwind.mdb"))
Set rs=Server.CreateObject("ADODB.Recordset")
sql=" SELECT Companyname, Contactname FROM Customers ORDER BY " & sort
rs.Open sql, conn

do until rs.EOF
response.write("<tr>")
for each x in rs.Fields
response.write("<td>" & x.value & "</td>")
next
rs.MoveNext
response.write("</tr>")
Loop
rs.Close
conn.Close
%>
</table>
</BODY>
</HTML>

```

Bài tập 1: Ta sẽ xây dựng mô hình ứng dụng cho việc bán hàng qua mạng, xây dựng menu sản phẩm, qua đó vận dụng các kiến thức của ADODB có thể kết nối với cơ sở dữ liệu sao cho có thể hiển thị và cung cấp các thông tin cho khách hàng. Hãy nghĩ về cách thức thiết kế menu ứng dụng web có các chức năng như sau: xem thông tin chi tiết của một mặt hàng, chuyển lại hàng và tính tiền

Hướng dẫn:

Tạo cơ sở dữ liệu gồm có hai bảng gọi là "basket" và thành phần chi tiết của mặt hàng để chi tiết mua "basket_detail"

Đây là cách thức tạo bảng bằng script cho SQL Server

```
CREATE TABLE [dbo].[basket]
```

```

([shopper_id] [varchar] (32) NOT NULL, [basket_id] [int]
IDENTITY (1000,1) NOT NULL
PRIMARY KEY CLUSTERED,
[basket_name] [varchar] (100) NULL,
[default_basket] [bit] NULL,
[date_created] [datetime] NULL,
[date_modified] [datetime] NULL) ON [PRIMARY]

```

```

CREATE TABLE [dbo].[basket_detail]
([basket_id] [int] NOT NULL
FOREIGN KEY REFERENCES basket(basket_id),
[sku] [varchar] (25) NOT NULL,
[sku_name] [varchar] (100) NOT NULL,
[sku_qty] [int] NOT NULL,
[sale_price] [money] NOT NULL,
[list_price] [money] NOT NULL,
[adjuste_price] [money] NOT NULL,
PRIMARY KEY (basket_id, sku) ON [PRIMARY]

```

Sau khi đã tạo được cơ sở dữ liệu, ta cần phải xây dựng các trang chính có giao diện như sau:

Trang hiển thị tất cả các thông tin các mặt hàng ví dụ tên mặt hàng, hình ảnh đi kèm, giá, giá giảm, mã số mặt hàng,...để cho phép khách hàng chọn lựa và đặt mua mặt hàng này cho vào giỏ. Trang này chúng ta có thể làm tĩnh hoặc động tức là ta có thể liệt kê sẵn các mặt hàng lên trang này, hoặc lấy dữ liệu từ cơ sở dữ liệu để hiển thị bằng cách tạo thêm bảng chứa sản phẩm cần bán. Từ trang này có thể xem thông tin về giỏ hàng của khách hàng bằng nhấn đến nút “your basket”.

Sau khi đã chọn mua “buy” được sản phẩm cho phép hiển thị danh sách các mặt hàng đã chọn lựa một cách chi tiết.

Với trang này cho phép khách hàng xóa “Delete” mặt hàng này nếu không chọn mua, cho phép thay đổi số lượng hàng cần mua chỉ việc nhấn nút “update”, ghi nhận các mặt hàng mà khách hàng đã mua “save basket” ghi nhận mã số giỏ hàng đã chọn và có thể tiếp tục chọn lựa mặt hàng cho giỏ hàng mới, và “checkout” để kết thúc chọn và mua hàng. Có thể cho phép tiếp tục chọn lựa các mặt hàng khác “continue shopping”.

Cập nhật các thông tin mặt hàng vào giỏ hàng hiện hành

Các hàm tiện ích trong tập tin basket_util.asp

<%

‘ Hàm thêm một mặt hàng đã chọn vào giỏ hàng của khách hàng

```

Function FcnAddBasket (ConnObj, usr_GUID, proID,
prodname,prodquantity,prodlistprice,prodsaleprice)

```

```

    dim strAddCode,rsAdd

```

```

    set rsAdd=server.CreateObject(“ADODB.RECORDSET”)

```

```

    rsAdd.Open “EXEC sproc_addToBasket” & chr(39) & usr_GUID & chr(39)
&”, “& chr(39) & proddID & chr(39) &”.”& chr(39) & prodname & chr(39) &”,”&
prodquantity &”, “& prodlistprice &”,” & prodsaleprice, ConnObj

```

```

strAddCode=rsAdd("errorcode")
FcnAddtoBasket=strAddCode
rsAdd.Close
set rsAdd=nothing
End Function

```

‘Hàm xoá toàn bộ hàng đã có trong giỏ hàng

```

Function FcnDeleteBasket
(ConnObj, usr_GUID, bID)
    dim rsDel
    set rsDel=server.CreateObject("ADODB.RECORDSET")
    rsDel.Open "EXEC sproc_DelBasket" & chr(39) & usr_GUID & chr(39) & ",
"& chr(39) & bID, ConnObj
FcnDeleteBasket=rsDel("errorcode")
rsDel.Close
set rsDel=nothing
End Function

```

‘Hàm xoá toàn bộ hàng trong giỏ hàng

```

Function FcnMakeDefBasket
(ConnObj, usr_GUID, bID)
    dim rsDef
    set rsDef=server.CreateObject("ADODB.RECORDSET")
    rsDef.Open "EXEC sproc_MakeDefaultBasket" & chr(39) & usr_GUID &
chr(39) & ", "& chr(39) & bID, ConnObj
FcnMakeDefBasket=rsDef("errorcode")
rsDef.Close
set rsDef=nothing
End Function

```

‘Hàm cập nhật vào giỏ hàng

```

Sub SubUpdateBasket(updStr)
    dim updConn
    set updConn=server.CreateObject("ADODB.Connection")
    updConn.ConnectionString=strBasketDSN
    updConn.Open
    updConn.Execute updStr
    updConn.Close
    Set updConn=nothing
End Sub

```

‘Hàm lấy mã số của giỏ hàng của khách hàng

```

Function FcnGetBasket
(ConnObj, usr_GUID)
    dim rsTemp
    set rsTem=server.CreateObject("ADODB.RECORDSET")

```

```

        rsTemp.Open "EXEC spoc_getBasket" & chr(39) & usr_GUID & chr(39) ,
ConnObj
set FcnGetBasket=rsTemp
End Function
' Hàm lưu giỏ hàng hiện tại của khách hàng
Function FcnSaveBasket
(ConnObj, usr_GUID, bName,bID)
    dim rsSave
    set rsSave=server.CreateObject("ADODB.RECORDSET")
    rsSave.Open "EXEC spoc_SaveBasket" & chr(39) & usr_GUID & chr(39) & ",
"& chr(39) & bName & chr(39) & ", " & bID, ConnObj
    FcnSaveBasket=rsSave("errorcode")
    rsSave.Close
    set rsSave=nothing
End Function
' Hàm lấy giỏ hàng trước của khách hàng nếu có
Function FcnGetSavedBaskets
(ConnObj, usr_GUID)
    dim rsTempSaved
    set rsTempSaved=server.CreateObject("ADODB.RECORDSET")
    rsTempSaved.Open "EXEC spoc_GetSavedBaskets" & chr(39) & usr_GUID
&chr(39), ConnObj
    set FcnSavedBaskets=rsTempSaved
    Set rsTempSaved=nothing
End Function
' Thủ tục hiển thị thông báo lỗi phát sinh
Sub writeError
Response.Write("<font color='red'><p>An error occurred while processing your
basket request!<br>Please contact the site Administrator<p></font>" )
End Sub
' Thủ tục hiển thị thông báo giỏ hàng đã được lưu
Sub writeSavedBasketMessage
Response.Write("<h4>Your Saved Baskets</h4><p><font color='brown'>You do not
have any saved basket at this time</font><p>")
End Sub
' Thủ tục hiển thị thông báo mặt hàng đã có trong giỏ hàng
Sub writeDuplicateEntryMessage
Response.Write"<font color='green'>You already have this product in your
basket</font>"
End Sub
' This function returns a 32 character GUID from SQL Server
Function FcnGenerateGUID(ConnObj)
dim strGUID,rsGUID
set rsGUID=server.CreateObject("ADODB.RECORDSET")
rsGUID.Open "EXEC spoc_createGUID", ConnObj

```

```

strGUID=rsGUID("guid")
FcnGenerateGUID=strGUID
rsGUID.Close
set rsGUID=nothing
End Function
' Thủ tục lấy 32 ký tự guid từ SQL và ghi giá trị cookie
Sub SubCreateNReturnGUID
    Dim myGUID
    'Call function to return a GUID
    myGuid=FcnGenerateGUID(BaskConn)
    'append created userGuid to cookie...
    Response.Cookies("shopGuid")("Guid")=myGuid
    Response.Cookies("shopGuid").expires="August 30,2020"
End Sub
%>
<%Sub BasketHeader To display Basket Header%>
<table cellspacing="1" cellpadding="1" border="0" width="500">
<tr bgcolor="gray">
<th><b><font size="-1" color="#ffffff">Item</font></b></th>
<th><b><font size="-1" color="#ffffff">Item ID</font></b></th>
<th><b><font size="-1" color="#ffffff">Qty</font></b></th>
<th><b><font size="-1" color="#ffffff">Unit<br>Price</font></b></th>
<th><b><font size="-1" color="#ffffff">Sale<br>Price</font></b></th>
<th><b><font size="-1" color="#ffffff">Total</font></b></th>
</tr>
<%End Sub%>
<% Sub BasketManager 'To Manage Saved Baskets%>
<p>
<h4>Your Saved Baskets</h4>
<form name="BaskMgr" method="POST" action="basket.asp">
<table cellspacing="2" cellpadding="1" border="0" width="500">
<th align="left" bgcolor="gray"><b><font size="-1" color="#ffffff">Basket
Name</font></b></th>
<th align="left" bgcolor="gray"><b><font size="-1" color="#ffffff">Date
Created</font></b></th>
<th>&nbsp;</th>
<th>&nbsp;</th>
<!--Loop to populate table-->
<%
do
Dim strBName,strDateCreate,intSavedBaskID
intSavedBaskID=rsSavedBaskets("basket_id")
strBName=rsSavedBaskets("basket_name")
strDateCreate=rsSavedBaskets("date_created")
%>

```

```

<tr>
<td><%=strBName%></td>
<td><%=strDateCreated%></td>
<td align="center"><a
href="basket.asp?bntMkDef=<%=intSavedBasketID%>"><b>Make
Default</b></a></td>
<td align="center"><a
href="basket.asp?bntDelSaved=<%=intSavedBasketID%>"><b>Delete</b></a></td>
</tr>
<%
rsSavedBaskets.moveNext
loop until rsSavedBaskets.eof
rsSavedBaskets.close
set rsSavedBaskets=nothing
%>
<!--End Loop-->
</table>
</form>
<%End Sub%>

```

Các store procedure dùng trong các hàm tiện ích

```

CREATE PROCEDURE spoc_createBasket
@shopperid varchar(32)
/* DESCR:Checks for existing default basket for shopper*/
/* Returns the basket ID, if non exists, creates one */
declare @basketID int
set nocount on
SELECT @basketID=basket_id FROM basket WHERE shopper_id=@shopperid AND
default_basket=1
IF (@basketID IS NULL) OR (@basketID='')
BEGIN --create and return a new basket
INSERT INTO
basket(shopper_id,default_basket,date_created,date_modified )
VALUES(@shopperid,1,getdate(),getdate())
IF @@ERROR!=0
BEGIN
SELECT @basketID=1
RETURN @basketID
END
SELECT @basketID=basket_id FROM basket WHERE shopper_id=@shopperid AND
default_basket=1
--add the default basket name (=BasketID)
UPDATE basket SET
basket_name=@basketID
WHERE shopper_id=@shopperid AND default_basket=1 RETURN @basketID
END

```



```

ELSE
BEGIN      --return existing basket
RETURN @baskID
END
set nocount off

CREATE PROCEDURE sproc_addToBasket
@shopperid varchar(32),
@sku varchar(25)
@skuname varchar(150),
@qty int,
@listprice money,
@salepricemoney
/*DESCR: Inserts new product into shopper basket tables */
/*RETURNS: interger errorcode; 0=success,1=error */
AS
DECLARE @basketID int
DECLARE @subtotal money
--calculate subtotal
SET  @subtotal=@saleprice * @qty
set nocount on
--get the default basket ID by calling sproc_createBasket
EXEC @basketID=sproc_createBasket @shopperid
--check returned value. If=1, then return error!
If @basketID=1
BEGIN
SELECT @basketID AS 'errorcode'
RETURN
END
--start the insert transaction
BEGIN TRAN InsertBasket
--do the insert into the basket_detail table
INSERT INTO
basket_detail(basket_id,sku,sku_name,sku_qty,sale_price,list_price,adjusted_price)
VALUES
(@basketID,@sku,@skuname,@qty,@saleprice,@listprice,@subtotal)
--do error checking
IF @@ERROR!=0
BEGIN
--rollback transaction!
ROLLBACK TRAN InsertBasket
--return error code
SELECT 1 AS 'errorcode'
RETURN
END

```

```

--update the basket modified time
UPDATE basket SET date_modified=getdate() WHERE shopper_id=@shopperid
AND basket_id=@basketID
--do error checking
IF @@ERROR!=0
BEGIN
--rollback transaction!
ROLLBACK TRAN InsertBasket
--return error code
SELECT 1 AS 'errorcode'
RETURN
END
--commit transaction
COMMIT TRAN InsertBasket
--return success code(0)
SELECT 0 AS 'errorcode'

```

```

CREATE PROCEDURE sproc_createGUID
/*DESCR:this sp creates a 32 character GUID */
/*RETURNS: 32 character GUID alpha -numeric */
AS
set nocount on
DECLARE @guid_val varchar(100)
SET @guid_val=REPLACE(CONVERT(varchar(100),NEWID()),' -','')
--Return the GUID value
SELECT @guid_val as 'guid'
set nocount off

```

```

CREATE PROCEDURE sproc_deleteBasket
@shopperguid varchar(32)
@basketid int
/*DESCR:Deletes shopper's entire basket content */
/* Uses a Transaction-either whole basket *****/
/* deleted or its left as is ! */
/*RETURNS: Int: 0=success,1=error */
AS
--disable the row count message
set nocount on
--begin a transaction
BEGIN TRAN BasketDeletion
--delèt contents of basket in foreign key table
DELETE FROM basket_detail
WHERE basket_id=@basketid
--do error checking
IF @@ ERROR!=0

```

```

BEGIN
--rollback the transaction!
ROLLBACK TRAN BasketDeletion
--return error code
SELECT 1 AS 'errorcode'
RETURN
END
--delete contents of basket table (PK table)
DELETE FROM basket
WHERE shopper_id=@shopperguid
AND basket_id=@basketid
--do error checking
IF @@ERROR!=0
BEGIN
--rollback the transaction!
ROLLBACK TRAN BasketDeletion
--return error code
SELECT 1 AS 'errorcode'
RETURN
END
--no errors, commit the transaction!
COMMIT TRAN BasketDeletion
--return success code
SELECT 0 AS 'errorcode'
set nocount off

CREATE PROCEDURE sproc_getBasket
@shopperid varchar(32)
/*DESCR:Returns recordset of shopper's default basket */
/*RETURNS: Returns recordset of shopper's default basket */
AS
set nocount on
--get the RS
SELECT
bd.basket_id,bd.sku,bd.sku-
name,bd.sku_qty,bd.sale_price,bd.list_price,bd.adjusted_price,bt.basket_name
FROM      basket_detail bd,basket bt
WHERE     bd.basket_id=bt.basket_id
AND       bt.default_basket=1
AND       bt.shopper_id=@shopperid
set nocount off

CREATE PROCEDURE sproc_getSavedBaskets
@shopperid varchar(32)

```

```

/* Retrieves all saved baskets by shopper*/
/* Returns recordset of all saved baskets by shopper */
AS
set nocount on
--get the RS
SELECT
bt.basket_id, bt.basket_name, bt.default_basket, bt.date_created, bt.date_modified
FROM      basket bt
WHERE      bt.shopper\_id=@shopperid
AND        bt.default_basket=0
set nocount off

CREATE PROCEDURE sproc_makeDefaultBasket
@shopperguid varchar(32)
@basketid int
/* Makes the specified basket shopper's default basket */
/* RETURNS: Int: 0=success, 1=error ****/
AS
--disable the row count message
set nocount on
--begin a transaction
BEGIN TRAN BasketDefault
--set shopper's existing default basket to 0
update basket set default_basket=0 where shopper_id=@shopperguid
--do error checking
IF @@ERROR!=0
BEGIN
--rollback the transaction!
ROLLBACK TRAN BasketDefault
--return error code
SELECT 1 AS 'errorcode'
RETURN
END
--set the new default basket
update basket set default_basket=1 where shopper_id=@shopperguid and
basket_id=@basketid
--do error checking
IF @@ERROR!=0
BEGIN
--rollback the transaction!
ROLLBACK TRAN BasketDefault
--return error code
SELECT 1 AS 'errorcode'
RETURN
END

```

-no errors,commit the transaction!

COMMIT TRAN BasketDefault

--return success code

SELECT 0 AS 'errorcode'

set nocount off

CREATE PROCEDURE sproc_saveBasket

@shopperid varchar(32)

@basketname varchar(100)

@basketid int

/* DESCR: Saves specified basket for shopper */

/* RETURNS: interger errorcode: 0=succes,1=error *****/

AS

set nocount on

UPDATE basket

SET basket_name=@basketname,default_basket=0,date_modified=getdate()

WHERE shopper_id=@shopperid

AND basket_id=@basketid

--do error checking

IF @@ERROR!=0

BEGIN

--return error code

SELECT 1 AS 'errorcode'

RETURN

END

ELSE

--return success code

SELECT 0 AS 'errorcode'

set nocount off

CREATE PROCEDURE sproc_updateBasket

@shopperid varchar(32)

@basketid int,

@sku varchar(25)

@qty int

/* DESCR: Updates shopper's default basket */

/* RETURNS:interger errorcode: 0=success, 1=error */

AS

set nocount on

--do the update

IF @qty=0

BEGIN --delete this sku from basket

DELETE FROM basket_detail WHERE basket_id=@basketid AND sku=@sku

UPDATE basket SET date_modified=getdate() WHERE shopper_id=@shopperid

AND basket_id=@basketid

```

--do error checking
IF @@ERROR!=0
BEGIN
--return error code
SELECT 2 AS 'errorcode'
RETURN
END
END
ELSE
BEGIN
UPDATE basket_detail SET sku_qty=@qty,adjusted price=(@qty*sale pric) WHERE
basket id=@basketid AND sku=@sku
UPDATE basket SET date_modified=getdate()
WHERE shopper_id=@shopperid AND basket_id=@basketid
--do error checking
IF @@ERROR!=0
BEGIN
--return error code
SELECT 1 AS 'errorcode'
RETURN
END
END
--return success code
SELECT 0 AS 'errorcode'
set nocount off

```

Bài tập 2:

Một siêu thị muốn thông báo cho các khách hàng của mình mỗi khi có mặt hàng mới về. Hãy viết một ứng dụng web hỗ trợ công việc trên bằng các chức năng sau:

- a. Cho phép người dùng điền vào thông tin của khách hàng như tên đăng nhập, mật khẩu, họ tên, địa chỉ liên lạc, điện thoại, địa chỉ email... và các chủng loại mặt hàng mà người dùng muốn thông tin khi có mặt hàng mới thuộc chủng loại này.
- b. Cho phép người dùng cập nhật lại các thông tin đã đăng ký. Để thực hiện được thao tác này, người dùng phải đăng nhập đúng với tên và mật khẩu đã đăng ký.
- c. Cho phép quản lý liệt kê danh sách các khách hàng đã đăng ký theo từng chủng loại mặt hàng.

Hướng dẫn:

- Xây dựng csdl quản lý thông tin khách hàng, mặt hàng, loại mặt hàng.
- Xây dựng các trang trong đó có các form cho phép đăng ký khách hàng, đăng nhập để cung cấp thông tin.
- Xây dựng các câu truy vấn để liệt kê danh sách khách hàng theo chủng loại mặt hàng.

Tài liệu tham khảo

1. ASP 3.0, ASP.NET
Nguyễn Phương Lan (chủ biên) của Nhà Xuất Bản Giáo Dục năm 2001
2. ASP Database (dịch và tổng hợp)
SAIGONBOOK của Nhà Xuất Bản Trẻ năm 2002
3. Xây dựng trang web động ASP
4. Building an Intranet
5. HTML & CGI Unleased
John December & Mark Ginsburg_USA:Sams.net,1996_830 tr;23 cm
6. Intranet bible
Ed Tittel & James M.Stewart_USA: IDG Books Worldwide, 1997_854 t r,23 cm

Mục lục

Chương 1: các khái niệm cơ bản	1
1. Mạng máy tính là gì?	1
2. Internet là gì?	1
3. Địa chỉ IP là gì?	2
4. Giao thức SMTP, POP3	3
5. Giao thức FTP	3
6. Giao thức HTTP	4
7. Giao thức NNTP	4
8. Giao thức Chat	4
9. URL	6
10. Hyperlink là gì?	6
11. Web Browser là gì?	7
12. Web Server là gì?	7
13. Website là gì?	7
14. World Wide Web (www) là gì?	8
15. Sự khác biệt giữa Internet và WorldWideWeb	8
16. Web page là gì?	8
Chương 2: Lập trình web với ngôn ngữ siêu văn bản HTML	10
1. khái niệm ngôn ngữ HTML	10
2. Lập trình web với ngôn ngữ html	10
2.1. Các thành phần cơ bản của HTML	12
2.2. Tạo trang Web	12
2.3. Cấu trúc của một tập tin HTML	13
2.4. Xem trang HTML bằng trình duyệt Web	13
2.5. Các tag cơ bản trong HTML	13
2.5.1. Tag chú giải	14
2.5.2. Các tag định dạng văn bản	20
2.5.3. Các tag định dạng hình ảnh	22
2.5.4. Các tag định dạng trang	23
2.5.5. Các tag tạo danh sách(list)	28
2.5.6. Tạo liên kết(link)	29
2.5.7. Một số ký tự đặc biệt trong HTML	30
2.5.8. Các tag dùng thiết kế bảng	33
2.5.9. Các tag dùng tạo Form	38
2.6. Các tag tạo Frame (khung)	38
2.6.1. Tạo Frame có dạng hàng	40
2.6.2. Tạo Frame có dạng cột	40
2.6.3. Kết hợp tạo Frame vừa dạng hàng vừa có dạng cột	41
2.7. Các hiệu ứng Dynamic HTML (DHTML)	41
2.7.1. Tạo chuỗi ký tự chuyển động	41
2.7.2. Thay đổi hình dạng chuột khi đi qua một ô trong bảng	2
Chương 3: Giới thiệu ngôn ngữ kịch bản vbscript và javascript	43
1. giới thiệu ngôn ngữ vbscript và javascript	44

2.ngôn ngữ vbscript.....	44
2.1.Chú thích một dòng lệnh	44
2.2.Cách khai báo biến, hằng, mảng.....	44
2.2.1.Khai báo biến.....	45
2.2.2.Khai báo hằng.....	45
2.2.3.Khai báo mảng.....	46
2.3.Các kiểu dữ liệu	47
2.4.Các toán tử cơ sở	48
2.5.Các lệnh xử lý điều kiện rẽ nhánh	48
2.5.1.Cấu trúc If...Then.....	48
2.5.2.Cấu trúc If...Then...Else	49
2.5.3.Cấu trúc Select Case.....	0
2.6.Cấu trúc lặp	50
2.6.1.Cấu trúc Do...Loop	51
2.6.2.Cấu trúc While...Wend	51
2.6.3.Cấu trúc For...Next	52
2.6.4.Cấu trúc For Each...Next	52
2.7.Khai báo hàm và thủ tục	52
2.7.1.Khai báo hàm	53
2.7.2.Khai báo thủ tục.....	53
3.Ngôn ngữ javascript	53
3.1.Chú thích một hay nhiều dòng lệnh.....	53
3.2.Cách khai báo biến,mảng.....	3
3.2.1.Cách khai báo biến	54
3.2.2.Khai báo mảng.....	54
3.3.Các kiểu dữ liệu trong javascript	54
3.3.1.Dữ liệu kiểu số.....	55
3.3.2.Kiểu ký tự	55
3.3.3.Kiểu chuỗi.....	55
3.3.4.Kiểu luận lý	56
3.3.5.Kiểu ngày.....	57
3.4.Các lệnh xử lý điều kiện rẽ nhánh	57
3.4.1.Cấu trúc If.....	57
3.4.2.Cấu trúc If...else.....	57
3.4.3.Cấu trúc switch...case	58
3.5.Cấu trúc lặp.....	58
3.5.1.Cấu trúc For	58
3.5.2.Cấu trúc While.....	59
3.5.3.Cấu trúc Do...While	59
3.6.Khai báo hàm	60
4.Sử dụng vbscript và javascript trong trang web	61
4.1.Chèn đoạn vbscript/javascript vào trang HTML	61
4.2.Xuất/nhập dữ liệu trong vbscript và javascript	62
4.2.1.Xuất dữ liệu	61
4.2.2.Nhập dữ liệu	62

5.Xử lý các sự kiện khi tương tác với các thành phần trên trang web	62
Chương 4:	
Lập trình web động với ngôn ngữ lập trình ASP	66
1.Giới thiệu về asp	66
2. nạp một ứng dụng web lên trình chủ iis	67
3.các khái niệm cơ bản về asp	68
3.1.Thành phần cơ bản của một trang asp	68
3.2.Nhập/xuất dữ liệu	69
3.3.Hoạt động của asp	69
4.Các đối tượng cơ bản trong asp	69
4.1.Đối tượng Request	69
4.1.1.Các tập hợp của đối tượng Request	70
4.1.2.Thuộc tính của đối tượng Request	73
4.1.3.Phương thức của đối tượng Request	73
4.2.Đối tượng Response	73
4.2.1.Các tập hợp của đối tượng Response	73
4.2.2.Thuộc tính của đối tượng Response	74
4.2.3.Phương thức của đối tượng Response	77
4.3.Đối tượng Session	81
4.3.1.Tập hợp của đối tượng Session	81
4.3.2.Các thuộc tính của đối tượng Session	82
4.3.3.Các phương thức của đối tượng Session	84
4.3.4.Các sự kiện của đối tượng Session	86
4.4.Đối tượng Application	86
4.4.1.Tập hợp của đối tượng Application	86
4.4.2.Các phương thức của đối tượng Application	87
4.4.3.Các sự kiện của đối tượng Application	88
4.5.Đối tượng Server	88
4.5.1.Các thuộc tính của đối tượng Server	88
4.5.2.Các phương thức của đối tượng Server	88
4.6.Đối tượng ASP Error	93
5.Chỉ thị #include	94
6.tập tin global.asa	95
7.đối tượng dictionary	97
7.1.Tạo đối tượng dictionary	98
7.2.Các thuộc tính của đối tượng dictionary	98
7.3.Các phương thức của đối tượng dictionary	100
8.đối tượng filesystemobject	103
8.1.Tạo đối tượng filesystemobject	103
8.2.Các thuộc tính của đối tượng filesystemobject	103
8.3.Các phương thức của đối tượng Filesystemobject	103
8.4.Ví dụ minh họa	109
9.đối tượng adrotator	112
9.1.Cách tạo đối tượng AdRotator	112
9.2.Định dạng tập tin văn bản	112

9.3.Các thuộc tính của đối tượng AdRotator	113
9.4.Các phương thức của đối tượng AdRotator.....	113
Chương 5: giới thiệu ado và các kết nối cơ sở dữ liệu	114
1.giới thiệu	114
2.kết nối với cơ sở dữ liệu.....	114
2.1.Tạo connection string thông qua OLEDB,ODBC	114
2.2.Tạo connection string thông qua DSN	116
3.các đối tượng của ADO.....	117
3.1.Đối tượng Connection	120
3.1.1.Kết nối cơ sở dữ liệu qua đối tượng Connection.....	120
3.1.2.Thao tác dữ liệu thông qua đối tượng Connection	120
3.2.Thao tác cơ sở dữ liệu bằng đối tượng Command	122
3.3.Xử lý dữ liệu thông qua đối tượng Recordset	123
3.3.1.Lưu trữ dữ liệu trả về.....	125
3.3.2.Hiển thị dữ liệu trả về	126
3.4.Đối tượng Record	126
3.5.Đối tượng Stream	128
4.tập hợp errors	128
5.stored procedure	130
6.xây dựng website bằng asp	133