Công nghệ Web và Ứng dụng

Biên tập bởi:

Khoa CNTT ĐHSP KT Hưng Yên

Công nghệ Web và Ứng dụng

Biên tập bởi:

Khoa CNTT ĐHSP KT Hưng Yên

Các tác giả:

Khoa CNTT ĐHSP KT Hưng Yên

Phiên bản trực tuyến:
http://voer.edu.vn/c/bac5238b

MỤC LỤC

Bài 1 : Giới thiệu về công nghệ Web

Giới thiệu về môn học

Khái niệm về website được hình thành từ thập niên 90 khi mà các trình duyệt đi vào giai đoạn hoàn thiện và phát triển mạnh mẽ như ngày nay. Không có lí do nào khiến chúng ta còn hoài nghi về lợi ích mà công nghệ web mang lại cho chúng ta cũng như cho sự phát triển của thế giới giai đoạn hiện tại và trong tương lai.

Liệu đã sắp đến lúc công nghệ web có thể hoàn toàn thay thế cho các ứng dụng desktop hiện tại. Mới đây cộng đồng mã nguồn mở đã và đang hi vọng sẽ đưa ứng dụng web đến gần với ứng dụng desktop hơn. Giả sử mọi mong muốn của họ đều trở thành sự thực khi mà ứng dụng web đã có đủ sức mạnh để thay thế cho ứng dụng Desktop khi đó công việc của chúng ta sẽ trở nên đơn giản hơn rất nhiều. Bạn thử tưởng tượng nhé thay vì một công ty có hàng trăm máy tính và phải cài hàng trăm ứng dụng Desktop giống nhau thay vào đó chúng ta có thể cài duy nhất một ứng dụng web để mọi người cùng dùng không những tại công ty mà họ còn có khả năng làm việc tại nhà. Mọi trao đổi sẽ được diễn ra trên môi trường web, những công việc đó ngày nay đã thực hiện được dựa trên nền tảng công nghệ web và các phần mềm kết nối trên môi trường internet xong điều này là chưa phổ biến.

Chúng ta hoàn toàn có thể tin tưởng vào một tương lai phát triển mạnh mẽ của công nghệ web. Mới đây một số ý tưởng cho rằng tại sao chúng ta phải cài đặt hệ điều hành trong khi nếu chúng ta khởi động từ máy tính và một trình duyệt web tương đương với một hệ điều hành sẽ được chúng ta sử dụng để thay thế cho các hệ điều hành hiện nay. Mặc dù ý tưởng đó chưa được thành sự thực xong nó cho thấy khả năng ứng dụng rộng rãi của công nghệ web trong tương lai mà con người muốn hướng tới.

Chính vì những lợi ích đó trong module này chúng ta sẽ đi vào tìm hiểu chi tiết những nguyên lý cơ bản để xây dựng các ứng dụng trên nền tảng công nghệ web. Qua đó chúng ta có thể xây dựng các website căn bản cho riêng mình và phát triển nó để sau khi ra trường chúng ta có những kiến thức nhất định trong lĩnh vực xây dựng website cho các doanh nghiệp trong tương lai.

Module này cung cấp các kiến thức căn bản về HTML (Ngôn ngữ được sử dụng phổ biến trên các trình duyệt web như một thành phần không thể thiếu), CSS (Cascading style sheet công cụ xây dựng giao diện cho các website), Javascript (Công cụ hỗ trợ trong việc tạo hiệu ứng và các bài toán phía trình duyệt), ASP.NET (Ngôn ngữ xây dựng website động tiên tiến vào bậc nhất hiện nay….) ngoài ra chúng ta còn được bắt tay xây dựng từng khâu trong việc xây dựng một website hoàn chỉnh.

Lịch sử Phát triển công nghệ web

Lịch sử ra đời của công nghệ web

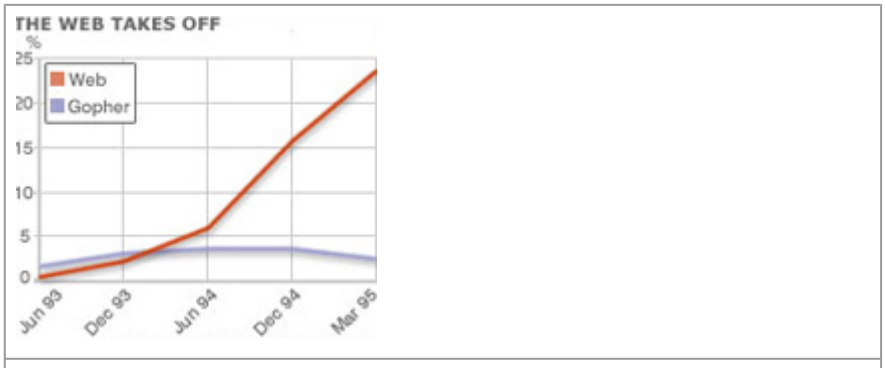
6/8/1991 là một ngày đáng nhớ bởi đó là thời điểm mã máy tính "non nớt" cho www được đăng trên alt.hypertext để mọi người có thể tải và tìm hiểu nó. Cũng bắt đầu từ hôm ấy công nghệ web được thế giới biết đến.

Jeff Groff, người cùng tham gia viết mã với Berners-Lee, cho biết ý tưởng tạo dựng web thực ra được hình thành rất đơn giản: "Chúng tôi luôn nghĩ rằng người sử dụng không cần phải xoay sở với những vấn đề kỹ thuật phức tạp". Web giống như một tấm khăn trải giường với nhiệm vụ cố che phủ sự rắc rối của những dữ liệu được lưu hành trên Internet.

Paul Kunz, nhà khoa học đã thiết lập máy chủ web đầu tiên ở châu Âu vào tháng 12/1991, cho biết đầu thập niên 90, máy tính giống như những "ốc đảo" thông tin. Một lần đăng nhập chỉ có thể truy cập tài nguyên của một hệ thống. Chuyển sang máy tính khác đồng nghĩa với việc họ phải đăng nhập thêm lần nữa và phải sử dụng những bộ lệnh khác nhau để truy xuất dữ liệu.

Web đã lôi kéo sự chú ý của Kunz khi ông chứng kiến Berners-Lee trình diễn khả năng hoạt động của web trên hệ thống IBM. Sau đó, Kunz đã thiết lập máy chủ web, cho phép các chuyên gia vật lý rà soát hơn 200.000 dữ liệu dễ dàng hơn bao giờ hết.

Tuy nhiên, dù các nhà vật lý đã bị web quyến rũ, đa số mọi người lại không nhận biết được khả năng tiềm ẩm của nó. Kunz cho rằng điều này là do nhiều tổ chức cũng đang thực hiện ý tưởng tương tự. Công nghệ nổi tiếng nhất khi đó là Gopher của Đại học Minnesota (Mỹ), cũng với tham vọng hóa giải sự phức tạp của những máy tính kết nối Internet. Gopher được ra mắt vào mùa xuân năm 1991 và lưu thông Gopher cao hơn hẳn so với lưu thông web trong vài năm tiếp theo.



Sự phát triển của công nghệ web so với Gopher. Nguồn: MIT

Trong thời gian đó, Berners-Lee, Jeff Groff và đồng nghiệp cũng tích cực giới thiệu phát minh của họ tại các hội thảo, cuộc gặp gỡ...

Dự án www chỉ thực sự thăng hoa khi chuyên gia Marc Andreessen thuộc Đại học Illinois (Mỹ) giới thiệu trình duyệt web máy tính đầu tiên vào tháng 4/1993. Trình duyệt Mosaic đã quá thành công và một số tính năng vẫn được coi là quy ước trong công nghệ web ngày nay. Cũng vào năm 1993, Đại học Minnesota thu phí Gopher khiến người ta bắt đầu phải tìm đến các giải pháp thay thế.

Ngoài ra, theo Ed Vielmetti, nhà nghiên cứu thuộc Đại học Michigan, ngay từ những năm đầu, web đã chứng minh được tính hữu ích với người sử dụng thông thường. Mọi người có thể sử dụng các trang web để tự bộc lộ mình, điều mà những công nghệ khác không cho phép (hình thức mới hiện nay của nó chính là blog).

Cuối năm 1994, lưu thông web rốt cuộc cũng vượt qua Gopher và từ đó chưa bao giờ bị tụt lại. Hiện nay, gần 100 triệu website đã xuất hiện và người ta gần như đồng nhất công nghệ web với Net.

Kunz cho biết ý tưởng hình thành www là để tạo điều kiện cho mọi người vừa đọc vừa đóng góp nội dung. Những công cụ mới như site chia sẻ ảnh, mạng xã hội, blog, các trang wiki... đang dần hoàn thành lời hứa ban đầu của nhóm chuyên gia phát triển web.

Và như thế, theo Kunz, web bây giờ mới chỉ bắt đầu những bước đi đầu tiên

Các giai đoạn phát triển của công nghệ web

Kh Ban đầu, các trang Web là tĩnh; người dùng gửi yêu cầu một tài nguyên nào đó, và server sẽ trả về tài nguyên đó. Các trang Web không có gì hơn là một văn bản được định dạng và phân tán. Đối với các trình duyệt, thì các trang Web tĩnh không phải là các vấn đề khó khăn, và trang Web lúc đầu chỉ để thông tin về các sự kiện, địa chỉ, hay lịch làm việc qua Internet mà thôi, chưa có sự tương tác qua các trang Web. Năm 1990, Tim Berners-Lee, tại CERN, đã sáng chế ra HTML (Hyper Text Markup Language), ngôn ngữ đánh dấu siêu văn bản. HTML rất đơn giản và dễ dùng, và nó trở thành một ngôn ngữ rất phổ biến và cơ bản.

Tuy nhiên, không lâu sau đó, nhu cầu về các trang Web động, có sự tương tác ngày một tăng, chính vì thế sự ra đời các công nghệ Web động là một điều tất yếu. Sau đây là một số công nghệ Web động cơ bản:

CGI

Giải pháp đầu tiên để làm các trang Web động là Common Gateway Interface (CGI). CGI cho phép tạo các chương trình chạy khi người dùng gửi các yêu cầu. Giả sử khi cần

hiển thị các các mục để bán trên Web site – với một CGI script ta có thể truy nhập cơ sở dữ liệu sản phẩm và hiển thị kết quả. Sử dụng các form HTML đơn giản và các CGI script, có thể tạo các "cửa hàng" ảo cho phép bán sản phẩm cho khách hàng qua một trình duyệt. CGI script có thể được viết bằng một số ngôn ngữ từ Perl cho đến Visual Basic.Tuy nhiên, CGI không phải là cách an toàn cho các trang Web động. Với CGI, người khác có thể chạy chương trình trên hệ thống. Vì thế có thể chạy các chương trình không mong muốn gây tổn hại hệ thống. Nhưng dù vậy, cho đến hôm nay thì CGI vẫn còn được sử dụng.

Applet

Tháng 5/1995, John Gage của hãng Sun và Andressen (nay thuộc Netscape Communications Corporation) đã công bố một ngôn ngữ lập trình mới có tên Java. Netscape Navigator đã hỗ trợ ngôn ngữ mới này, và một con đường mới cho các trang Web động được mở ra, kỷ nguyên của applet bắt đầu.

Applet cho phép các nhà phát triển viết các ứng dụng nhỏ nhúng vào trang Web. Khi người dùng sử dụng một trình duyệt hỗ trợ Java, họ có thể chạy các applet trong trình duyệt trên nền máy ảo Java Virtual Machine (JVM). Dù rằng applet làm được nhiều điều song nó cũng có một số nhược điểm: thường bị chặn bởi việc đọc và ghi các file hệ thống, không thể tải các thư viện, hoặc đôi khi không thể thực thi trên phía client. Bù lại những hạn chế trên, applet được chạy trên một mô hình bảo mật kiểu sandbox bảo vệ người dùng khỏi các đoạn mã nguy hiểm.

Có những lúc applet được sử dụng rất nhiều, nhưng nó cũng có những vấn đề nảy sinh: đó là sự phụ thuộc vào máy ảo Java JVM, các applet chỉ thực thi khi có môi trường thích hợp được cài đặt phía client, hơn nữa tốc độ của các applet là tương đối chậm vì thế applet không phải là giải pháp tối ưu cho Web động.

JavaScript

Cùng thời gian này, Netscape đã tạo ra một ngôn ngữ kịch bản gọi là JavaScript. JavaScript được thiết kế để việc phát triển dễ dàng hơn cho các nhà thiết kế Web và các lập trình viên không thành thạo Java. (Microsoft cũng có một ngôn ngữ kịch bản gọi là VBScript). JavaScript ngay lập tức trở thành một phương pháp hiệu quả để tạo ra các trang Web động.

Việc người ta coi các trang như là một đối tượng đã làm nảy sinh một khái niệm mới gọi là Document Object Model (DOM). Lúc đầu thì JavaScript và DOM có một sự kết hợp chặt chẽ nhưng sau đó chúng được phân tách. DOM hoàn toàn là cách biểu diễn hướng đối tượng của trang Web và nó có thể được sửa đổi với các ngôn ngữ kịch bản bất kỳ như JavaScript hay VBScript.

Tổ chức World Wide Web Consortium (W3C) đã chuẩn hóa DOM, trong khi European Computer Manufacturers Association (ECMA) phê duyệt JavaScript dưới dạng đặc tả ECMAScript.

JSP/Servlet, ASP và PHP

Cùng với Java, Sun đồng thời đưa ra một công nghệ mới gọi là servlet. Các đoạn mã Java sẽ không chạy phía client như với applet; chúng sẽ được chạy trên một ứng dụng phía server. Servlet cũng đồng thời phục vụ các CGI script. Servlet là một bước tiến lớn, nó đưa ra một thư viện hàm API trên Java và một thư viện hoàn chỉnh để thao tác trên giao thức HTTP.

JavaServer Page (JSP) là một công nghệ lập trình Web của Sun, cùng với nó là một công nghệ khác của Microsoft - Active Server Pages (ASP), JSP là công nghệ đòi hỏi một trình chủ hiểu được Java.

Microsoft đã nghiên cứu các nhược điểm của servlet và tạo ra ASP dễ dàng hơn để thiết kế các trang web động. Microsoft thêm các bộ công cụ rất mạnh và sự tích hợp rất hoàn hảo với các Web server. JSP và ASP có những nét tương đương vì chúng đều được thiết kế để phân tách qua trình xử lí khỏi quá trình biểu diễn. Có sự khác biệt về kỹ thuật, song cả hai đều cho phép các nhà thiết kế Web tập trung vào cách bố trí (layout) trong khi các nhà phát triển phần mềm thì tập trung vào các kỹ thuật lập trình logic. Tất nhiên Microsoft và Sun không độc quyền ở các giải pháp phía server. Còn có các công nghệ khác, trong đó phải kể đến là PHP (Hypertext Preprocessor) cho tới Cold Fusion. Các công nghệ này cung cấp các bộ công cụ rất mạnh cho các nhà phát triển.

Flash

Năm 1996, FutureWave đã đưa ra sản phẩm FutureSplash Animator. Sau đó FutureWave thuộc sở hữu của Macromedia, và công ty này đưa ra sản phẩm Flash. Flash cho phép các nhà thiết kế tạo các ứng dụng hoạt họa và linh động. Flash không đòi hỏi các kỹ năng lập trình cao cấp và rất dễ học. Cũng giống như các nhiều giải pháp khác Flash yêu cầu phần mềm phía client. Chẳng hạn như gói Shockwave Player plug-in có thể được tích hợp trong một số hệ điều hành hay trình duyệt.

DHTML

Khi Microsoft và Netscape đưa ra các version 4 của các trình duyệt của họ, thì các nhà phát triển Web có một lựa chọn mới: Dynamic HTML (DHTML). DHTML không phải là một chuẩn của W3C; nó giống một bộ công cụ thương mại hơn. Trong thực tế nó là một tập hợp gồm HTML, Cascading Style Sheets (CSS), JavaScript, và DOM. Tập hợp các công nghệ trên cho phép các nhà pháp triển sửa đổi nội dung và cấu trúc của một trang Web một cách nhanh chóng. Tuy nhiên, DHTML yêu cầu sự hỗ trợ từ các trình

duyệt. Mặc dù cả Internet Explorer và Netscape hỗ trợ DHTML, nhưng các thể hiện của chúng là khác nhau, các nhà phát triển cần phải biết được loại trình duyệt nào mà phía client dùng. DHTML thật sự là một bước tiến mới, nhưng nó vẫn cần một sự qui chuẩn để phát triển. Hiện nay DHTML vẫn đang trên con đường phát triển mạnh.

XML

Kể từ khi ra đời vào giữa năm 1990, eXtensible Markup Language (XML) của W3C dẫn xuất của SGML đã trở nên rất phổ biến. XML có mặt ở khắp nơi, Microsoft Office 12 cũng sẽ hỗ trợ định dạng file XML.

Ngày nay chúng ta có rất nhiều dạng dẫn xuất của XML cho các ứng dụng Web (tất nhiên là có cả XHTML): XUL của Mozilla; XAMJ, một sản phẩm mã nguồn mở trên nền Java; MXML từ Macromedia; và XAML của Microsoft.

Khái quát về công nghệ web

Khái niệm về website

Bạn có thể hiểu website tương tự như quảng cáo trên các trang vàng, nhưng có điểm khác ở chỗ nó cho phép người truy cập có thể trực tiếp thực hiện nhiều việc trên website như giao tiếp, trao đổi thông tin với người chủ website và với những người truy cập khác, tìm kiếm, mua bán vv...chứ không phải chỉ xem như quảng cáo thông thường. Hàng triệu người trên khắp thế giới có thể truy cập website- nhìn thấy nó chứ không giới hạn trong phạm vi lãnh thổ nào cả. Đối với một doanh nghiệp, Website là một cửa hàng ảo với hàng hoá và dịch vụ có thể được giới thiệu và rao bán trên thị trường toàn cầu. Cửa hàng đó mở cửa 24 giờ một ngày, 7 ngày một tuần, quanh năm, cho phép khách hàng của bạn tìm kiếm thông tin, xem, mua sản phẩm và dịch vụ của bạn bất cứ lúc nào họ muốn.

Khi bạn xem thông tin trên một trang Web thì trang Web đó đến từ một Website, có thể là một Website đến từ Mỹ, từ Việt Nam, hay bất cứ nơi nào khác trên thế giới. Website sẽ tên và chính là địa chỉ mà bạn đã gọi nó ra tên đó người ta gọi là tên miền hay domain name. Thường các Website được sở hữu bởi một cá nhân hoặc tổ chức nào đó.

Website là một văn phòng ảo của doanh nghiệp trên mạng Internet. Website bao gồm toàn bộ thông tin, dữ liệu, hình ảnh về các sản phẩm, dịch vụ và hoạt động sản xuất kinh doanh mà doanh nghiệp muốn truyền đạt tới người truy cập Internet. Với vai trò quan trọng như vậy, có thể coi Website chính là bộ mặt của Công ty, là nơi để đón tiếp và giao dịch với các khách hàng trên mạng. Website không chỉ đơn thuần là nơi cung cấp thông tin cho người xem, cho các khách hàng và đối tác kinh doanh của doanh nghiệp, nó còn phải phản ánh được những nét đặc trưng của doanh nghiệp, đảm bảo tính thẩm mỹ cao, tiện lợi, dễ sử dụng và đặc biệt phải có sức lôi cuốn người sử dụng để thuyết phục họ trở thành khách hàng của doanh nghiệp.

Các yêu cầu tối thiểu của một Webiste

Đối với một doanh nghiệp trong đời thường, để thành lập và hoạt động, doanh nghiệp đó phải đáp ứng được tối thiểu 3 yếu tố căn bản sau:

- Tên doanh nghiệp
- Trụ sở hoạt động của doanh nghiệp
- Các yếu tố vật chất kỹ thuật,máy móc và con người

Nếu ta tạm coi Website như 1 doanh nghiệp trong đời thường, thì để thiết lập và đưa vào hoạt động 1 Website cũng phải đáp ứng được tối thiểu 3 yếu tố cơ bản như doanh nghiệp là:

- Tên Website (hay còn gọi là Tên miền ảo hoặc Domain name) tương ứng với Tên doanh nghiệp trong đời thường.
- Web Hosting (hay còn gọi là nơi lưu giữ trên máy chủ Internet) tương ứng với Trụ sở doanh nghiệp trong đời thường.
- Các trang Web tương ứng với yếu tố vật chất kỹ thuật, máy móc của doanh nghiệp trong đời thường và con người để quản lý và vận hành Website đó.

Những thành phần cấu tạo nên một website

Ngôn ngữ siêu văn bản

HTML (HyperText Markup Language) là ngôn ngữ đánh dấu siêu văn bản được thiết kế ra để tạo nên các trang web. Tập tin (File) HTML là một văn bản có chứa các thẻ đánh dấu (markup tags), các thẻ đánh dấu này giúp các trình duyệt Web hiểu được cách trình bày và hiển thị trang Web. File HTML có phần mở rộng (Extension) là htm hay html và có thể được tạo ra bằng bất cứ chương trình xử lý văn bản đơn giản nào.

Trong File HTML các phần tử (Element) được đánh dấu bằng các thẻ HTML. Các thẻ này được bao bởi dấu < và dấu >. Thông thường các thẻ HTML được dùng theo một cặp **<tên thẻ>** (thẻ bắt đầu) và **</tên thẻ>** (thẻ kết thúc), văn bản nằm giữa cặp thẻ này là nội dung của phần tử. Các thẻ HTML không phân biệt chữ hoa và chữ thường, có nghĩa là các kiểu chữ đều được xem như nhau.

Ngôn ngữ kịch bản

Ngôn ngữ kịch bản (script): Là loại ngôn ngữ dùng để nâng cao hiệu quả và tính năng của trang web.

Có hai loại:

- Chạy trên máy server gọi là server script
- Chạy trên máy client (máy duyệt web) còn gọi là client script

Các thành phần stypesheet

Sử dụng style sheet giúp cho người sọan thảo trang web dễ dàng hơn trong việc thiết kế và hiệu chỉnh các trang web đồng thời đảm bảo tính nhất quán trong trình bày của website.

Một style là một mẫu định dạng (template) của các HTML tag. Khái niệm style sheet trong trang web rất tương tự với các khái niệm templates trong MSWORD. bạn có thể thay đổi sự trình bày của một văn bản Word thông qua việc thay đổi các style trong văn bản này. Một cách tương tự , bạn có thể thay đổi sự trình bày của trang web bằng cách hay đổi các style sheet được gán cho các tag HTML.

Ngôn ngữ xử lý dữ liệu phía server

Perl

Perl (Practical Extraction and Report Language) là ngôn ngữ kịch bản mã nguồn mở có cú pháp tương tự như C. Perl chạy ở chế độ nền phía server tạo nội dung web theo cách thức ẩn đối với người xem. Năm 1987, lrry Wall xây dựng và phát triển Perl như là phiên bản cải tiến của awk với mục đích thực hiện tự động một số tác vụ quản trị hệ thống Unix (awk là một chương trình xử lý văn bản của Unix) . Sau đó Perl được phát triển dần và phổ biến với nhiều dạng ứng dụng. Perl có tính đối tượng, nhờ vậy việc bổ sung thư viện mới rất dễ dàng. Nó thích hợp cho cả ứng dụng web phức tạp lẫn các tác vụ xử lý dữ liệu đơn giản.

Đã có thời tất cả những việc tạo trang web động có lập trình đều dùng Perl, trước khi có những công nghệ khác như ASP, JSP, PHP... và hiện Perl vẫn là một trong những ngôn ngữ được dùng phổ biến nhất để viết các ứng dụng web.

Hầu hết các máy chủ web hiện nay đều hỗ trợ Perl. Apache có hỗ trợ miễn phí tích hợp trình dịch Perl mod-perl. Điều này không chỉ giúp tăng tốc mã lệnh Perl mà còn cải thiện hiệu suất nhờ mod-perl lưu lại các mã lệnh biên dịch trong bộ nhớ. Mod_perl kết hợp với một số công nghệ khác cho phép xây dựng các website cao cấp, nổi bật nhất là hai công nghệ HTML::Embperl và HTML::Mason.

Các giải pháp thương mại của Activestate và Binary Evolution cũng giúp tăng tốc Perl theo cách thức tương tự như mod-perl. PerlEx của ActiveState tăng tốc chương trình CGI/PERL cho IIS trong khi sản phẩm của Binary Evolution dùng cho Netscape, Apache, và IIS trên cả nền Windows và Unix.

Có rất nhiều tài liệu trên Net về Perl cũng như nhiều thư viện chương trình tiện ích miễn phí. Bạn có thể tìm thấy những mô đun viết sẵn ở website CPAN (Comprehensive Perl Archive Network - www.cpan.org). Ngôn ngữ Perl hiện đang phát triển đi xa hơn thiết kế ban đầu của nó.

ASP

Công nghệ Microsoft Active Server pages (ASP) đi cùng với Microsoft Intemet Information Server (IIS). ASP hỗ trợ nhiều ngôn ngữ kịch bản như PerlScript, JScript và VBScript. PerlScript dựa trên ngôn ngữ Perl, JScript dựa trên ngôn ngữ JavaScript, nhưng ngôn ngữ mặc định của ASP là VBScript, một ngôn ngữ kịch bản dễ học, là tập con của ngôn ngữ Visual Basic - một trong những ngôn ngữ lập trình phổ biến nhất hiện nay. Công nghệ ASP cho phép trộn nội dung HTML tĩnh với mã lệnh kịch bản thực thi ở môi trường server để tạo ra kết quả động.

Ưu điểm nổi bật nhất của ASP là khả năng dùng thành phần COM và ADO (Activex Data Object), nhưng cũng chính khả năng này làm cho chương trình ASP phức tạp

và khó viết hơn. Khi cần phát triển, bạn có thể tạo thành phần COM riêng. Microsoft khuyến cáo xây dựng thành phần COM để xử lý mức luận lý. Thành phần COM có thể phát triển bằng Visual Basic, Visual C++ hay Java.

Tuy nhiên, với môi trường máy chủ dùng chung, thường các công ty cung cấp dịch vụ máy chủ giới hạn chỉ cho phép bạn dùng VBScript.

Trở ngại lớn nhất của ASP là chỉ dùng trên IIS chạy trên máy chủ Win32. Có một số sản phẩm của các hãng thứ ba cho phép ASP chạy trên môi trường và máy chủ web khác như các sản phẩm thương mại InstantASP của Halcyon, Chili!Soft của Chili!Soft và sản phẩm miễn phí OpenASP của ActiveScripting.org. Có hai phiên bản Perl cho ASP: phiên bản Unix (Apache::ASP) dùng với Apache và phiên bản Windows PerlScript của hãng ActiveState.

Việc cài đặt môi trường máy chủ hỗ trợ ASP rất đơn giản, IIS mặc định hỗ trợ sẵn ASP. Personal Web Server cung cấp môi trường chạy ASP cho Windows 95, 98. Công cụ Visual Interdev rất mạnh, giúp tạo trang ASP đơn giản và nhanh chóng. Có rất nhiều website, sách và mã nguồn miễn phí cho ASP. Đây là một lợi thế.

ASP.NET (ASP+) là bước phát triển mới của công nghệ ASP dùng với nền tảng NET. Ngôn ngữ chính dùng để phát triển trang ASP.NET (.aspx) là VB.NET, C#. Ngoài ra ASP.NET còn hỗ trợ nhiều ngôn ngữ khác như JScript.NET, Smalltalk.NET, Cobol.NET, Perl.NET...

JSP

Java Server Pages (JSP) là sự mở rộng của công nghệ JavaServlet, một thành phần trong chuẩn J2EE của Sun. Với JavaServlet, bạn phải xử lý đầu vào HTTP và đầu ra HTML trong lớp Java, bạn cần có kiến thức lập trình để xây dựng các ứng dụng phức tạp. Với JSP bạn có thể tách riêng lớp hiển thị HTML ra khỏi lớp Java xử lý nghiệp vụ phức tạp. Điều này có nghĩa là người phát triển có kinh nghiệm ngôn ngữ kịch bản, hay thậm chí người thiết kế web có thể viết mã lệnh hiển thị đơn giản, trong khi người phát triển có kiến thức về Java tập trung viết JavaServlet hay JavaBean đề giải quyết nghiệp vụ phức tạp.

Tương tự ASP, JSP cũng thực hiện phép trộn nội dung HTML tĩnh với mã lệnh kịch bản thực thi ở môi trường server để tạo ra kết quả động. JSP dùng ngôn ngữ kịch bản mặc định là Java; tuy nhiên theo đặc tả kỹ thuật thì cũng có thể dùng các ngôn ngữ khác. JSP có ưu điểm so với ASP là sau lần thực thi đầu tiên thì mã biên dịch (Servlet) của trang JSP được lưu lại trong bộ nhớ của máy chủ web và sẵn sàng đáp ứng cho các yêu cầu truy cập sau đó (trang ASP/VBSCRIPT hay Asp/jscript phải được dịch lại với mỗi yêu cầu). Lợi thế của JSP là sử dụng được toàn bộ sức mạnh của ngôn ngữ Java với các

tính năng khả chuyển, chạy được trên nhiều nền tảng hệ thống và máy chủ web, mã lệnh hướng đối tượng, bảo mật an toàn...

Hiện chưa có nhiều dịch vụ đặt web hỗ trợ JSP. Tuy Java miễn phí nhưng các côngcụ phát triển Java và phần mềm máy chủ Java khá đắt. Các công cụ phát triển trang JSP tốt là Borland Jbuilder, IBM WebSphere Studio. Một số phần mềm máy chủ Java hỗ trợ JSP miễn phí như Tomcat, JONAS.

Server-side JavaScript (SSJS) là sự mở rộng của JavaScript, ngôn ngữ kịch bản phổ biến chạy ở trình duyệt máy khách có cú pháp giống như C, mặc dù có tên gọi tương tự nhưng nó không phải là Java. SSJS có các tính năng tích hợp hỗ trợ cơ sở dữ liệu và email, quản lý phiên làm việc và khả năng liên tác với các lớp Java dùng công nghệ Livewire của Netscape. SSJS chỉ chạy trên máy chủ web Netscape.

PHP

Năm 1995 , Rasmus Lerdorf tạo ra PHP nhằm giải quyết việc viết lặp đi lặp lại cùng đoạn mã khi tạo các trang home (vì vậy PHP được viết tắt từ Personal Home Page). Ban đầu, tác giả chỉ có ý định tạo bộ phân giải đơn giản để thay thế các thẻ lệnh trong file HTML bằng các đoạn mã lệnh viết bằng C. Dự án này đã được phát triển thành ngôn ngữ kịch bản mã nguồn mở PHP - Hypertext Preprocessor được cài đặt cho khoảng 20% máy chủ web trên Internet, theo số liệu khảo sát của công ty Netcraft.

PHP tương tự JSP và ASP với tập thẻ lệnh dùng cho trang HTML. Điểm đặc biệt là PHP được phát triển hoàn toàn cho nền tảng web, chính vì vậy mà các ứng dụng viết bằng PHP rất ngắn gọn so với VBScript hay JSP. Đây cũng chính là điểm mạnh của PHP so với Perl.

Cú pháp PHP mượn từ nhiều ngôn ngữ khác như C, Java, Perl... PHP có thể giao tiếp với nhiều hệ CSDL như Sybase, Oracle, Informix, Postgres và cả Microsoft SQL. Không chỉ có khả năng thao tác CSDL, PHP còn có nhiều khả năng khác như IMAP, SNMP, LDAP, XML... PHP chạy trên hầu hết các nền tảng hệ thống. Trình máy chủ phân giải mã lệnh PHP có thể tải về miễn phí từ trang web chính thức của PHP. Có lẽ yếu tố hấp dẫn nhất của PHP là nó hoàn toàn miễn phí. Với máy tính cấu hình vừa phải chạy Linux, cài đặt Apache, PHP và MYSQL, bạn sẽ có máy chủ có thể phục vụ được nhiều ứng dụng web tương đối. Toàn bộ chi phí hầu như chỉ là thời gian bạn bỏ ra để cài đặt các phần mềm. PHP được xem là một thay thế cho Perl. PHP không thể làm được nhiều như Perl, thế nhưng chính sự hạn chế này làm cho PHP dễ học và dễ dùng. Nhiều nhà phát triển dùng kết hợp cả hai: Perl dùng cho những tác vụ chạy bên dưới còn PHP dùng cho việc xử lý bề mặt. Komodo của Active State Corp là công cụ miễn phí dùng để phát triển trang PHP.

ASP.NET

Có thể nói là thế hệ sau của ASP nhưng hỗ trợ tối đa mong muốn của lập trình viên và khắc phục được những điểm còn hạn chết trong ASP, chính vì vậy ASP.NET đang dần khẳng định là một trong những lựa chọn số một của người xây dựng website.

Mặc dù ra đời muộn cùng với sự ra đời của Net framework nhưng ASP.NET đã chứng tỏ mình là công cụ mạnh mẽ và hỗ trợ lập trình hết sức linh hoạt và là sự lựa chọn hàng đầu của các lập trình viên. Ngoài khả năng thao tác dễ dàng trong thiết kế giao diện ASP.NET còn là công nghệ hỗ trợ nhiều ngôn ngữ lập trình đặc biệt trong số đó là VB.NET, C#,…. Là những ngôn ngữ đang rất được ưa chuộng bởi các lập trình viên hiện nay.

Việc kết nối CSDL dễ dàng cũng là một trong những ưu điểm nổi bật của ASP.NET với sự hỗ trợ của ADO.NET và mới đây nữa là sự ra đời của LinQ với khả năng kết nối CSDL dễ dàng ASP.NET đang dần hoàn thiện để trở thành công nghệ thống trị trong thế giới website.

Bài 2 : Thiết kế Website với HTML

Giới thiệu về HTML

Ngày nay Internet đã phát triển đến mức nó trở thành một phần không thể thiếu được của cuộc sống hiện đại. Các nguồn thông tin được cung cấp một cách nhanh chóng và chính xác trên những Website. Trên Internet, quả thực có rất nhiều Website chú trọng đến ngôn ngữ HTML, cũng có rất nhiều Web site dành cho mục đích thương mại, nghệ thuật. Thế nhưng lại có ít nguồn thông tin đề cập đến việc thiết kế một trang Web, một Website, thiết kế đồ hoạ, giao diện người sử dụng hay những kiến thức về cách thức tổ chức thông tin. Môn học chuyên đề Web sẽ cung cấp một cái nhìn tổng thể trong việc xây dựng và phát triển Website đồng thời cung cấp một số giải pháp được lựa chọn trong thời đại hiện nay khi nói đến Website.

Khái niệm

HTML (HyperText Markup Language) – Ngôn ngữ đánh dấu siêu văn bản được sử dụng để tạo các tài liệu có thể truy cập trên mạng. Tài liệu HTML được tạo nhờ dùng các thẻ và các phần tử của HTML. File được lưu trên máy chủ dịch vụ web với phần mở rộng “.htm” hoặc “.html”.

Ngày nay với sự phát triển của công nghệ web việc ứng dụng các công cụ HTML càng trở nên quan trọng trong sự phát triển của ngành công nghệ web hiện đại.

Ví dụ về tạo file HTML

Nếu bạn sử dụng Window hãy mở Notepad, nếu bạn sử dụng Mac mở ứng dụng Simple Text. Với OSX bạn mở TextEdit và thay đổi lựa chọn sau: Select (trong cửa sổ preference) > Plain Text thay vì Rich Text và chọn "Ignore rich text commands in HTML files". Việc này rất quan trọng bởi vì nếu bạn không làm vậy thì code HTML có thể không đúng.

Sau đó bạn gõ vào những dòng sau:

<html><head><title>Trường đại học sư phạm kỹ thuật Hưng Yên</title>

</head><body>www.utehy.vn and www.utehy.edu.vn. UTHYE and web design resources

</body></html>

Lưu file lại với tên là "mypage.htm" vào desktop cũng được. Sau đó bạn đóng trình soạn thảo Notepad hoặc Simple Text lại và tìm đến file mypage.htm ở desktop rồi nhấp đúp vào trình duyệt sẽ hiển thị nội dung của trang.

Giải thích ví dụ:

Thẻ đầu tiên trong tài liệu HTML là <html>. Thẻ này nói cho trình duyệt biết đây là điểm khởi đầu của một tài liệu HTML. Thẻ cuối cùng của tài liệu là </html>, thẻ này nói cho trình duyệt biết đây là điểm kết thúc của văn bản.

Đoạn chữ nằm giữa hai thẻ <head> và </head> là thông tin của header. Thông tin header sẽ không được hiển thị trên cửa sổ trình duyệt.

Đoạn chữ nằm giữa cặp thẻ <title> là tiêu đề của văn bản. Dòng tiêu đề này sẽ xuất hiện ở thanh trạng thái của trình duyệt web.

Đoạn chữ nằm giữa hai thẻ <body> là những gì nó sẽ thể hiện trên trình duyệt của bạn.

Đoạn chữ nằm giữa hai thẻ và sẽ xuất hiện dưới dạng đậm.

Sử dụng file với HTM hay HTML

Một số bạn thường gặp rắc rối khi phân biệt cũng như giải thích tại sao trang HTM và HTML lại cho ra cùng một kết quả vậy nguyên nhân của việc này là gì?

Khi bạn lưu một văn bản dưới dạng HTML, bạn có thể sử dụng cả hai dạng là .htm và .html. Chúng ta đã sử dụng dạng .htm trong ví dụ trên. Lý do này bắt nguồn từ nguyên nhân ngày trước là có những phần mềm chỉ cho phép phần mở rộng có tối đa là 3 chữ cái. Với những phần mềm mới hiện nay chúng ta nghĩ sẽ tốt hơn nếu bạn lưu lại với phần mở rộng là .html

Một chú ý khi sử dụng trình soạn thảo HTML:

Bạn có thể dễ dàng chỉnh sửa một tài liệu HTML bằng cách sử dụng WYSIWYG (what you see is what you get = thấy gì có đó) như là Frontpage, Claris Homepage, Dream weaver hoặc Adobe PageMill thay vì bạn phải tự viết những cặp thẻ từ đầu đến cuối. Nhưng nếu bạn muốn trở thành một web master đầy kỹ thuật, tôi khuyên bạn nên sử dụng những trình soạn thảo text đơn giản để học và làm quen với cấu trúc câu lệnh của HTML.

Thành phần của HTML

Các dạng thẻ HTML

Thẻ HTML dùng để viết lên những thành tố HTML

Thẻ HTML được bao quanh bởi hai dấu lớn hơn < và > nhỏ hơn.

Những thẻ HTML thường có một cặp giống như và

Thẻ thứ nhất là thẻ mở đầu và thẻ thứ hai là thẻ kết thúc.

Dòng chữ ở giữa hai thẻ bắt đầu và kết thúc là nội dung.

Những thẻ HTML không phân biệt in hoa và viết thường, ví dụ dạng và <B> đều như nhau.

Thành phần HTML

Nhớ lại ví dụ ở trên của chúng ta về HTML

<html><head><title>Trường đại học sư phạm kỹ thuật Hưng Yên</title>

</head><body>

www.utehy.vn and www.utehy.edu.vn. UTHYE and web design resources

</body></html>

UTEHY and web design resources Thành phần của HTML bắt đầu với thẻ: Nội dung của nó là: UTHYE and web design resources Thành phần của HTML kết thúc với thẻ: Mục đích của thẻ là để xác định một thành phần của HTML phải được thể hiện dưới dạng in đậmĐây cũng là một thành phần của HTML:

<body>

www.utehy.vn and www.utehy.edu.vn.

 UTHYE and web design resources

</body>

Phần này bắt đầu bằng thẻ bắt đầu <body> và kết thúc bằng thẻ kết thúc </body>. Mục đích của thẻ <body> là xác định thành phần của HTML bao gồm nội dung của tài liệu.

Các thuộc tính của thẻ HTML

Những thẻ HTML đều có những thuộc tính riêng. Những thuộc tính này cung cấp thông tin về thành phần HTML của trang web. Tag này xác định thành phần thân của trang HTML: <body>. Với một thuộc tính thêm vào là bgcolor, bạn có thể báo cho trình duyệt biết rằng màu nền của trang này là màu đỏ, giống như sau: <body bgcolor="red"> hoặc <body bgcolor="#E6E6E6"> (#E6E6E6 là giá trị hex của màu) Thẻ này sẽ xác định dạng bảng HTML: <table> với một thuộc tính đường viền (border), bạn có thể báo cho trình duyệt biết rằng bảng sẽ không có đường viền: <table border="0"> Thuộc tính luôn luôn đi kèm một cặp như name/value: name="value" (tên="giá trị") thuộc tính luôn luôn được thêm vào thẻ mở đầu của thành phần HTML.

Dấu ngoặc kép, "red" hoặc 'red'

Giá trị thuộc tính nên được đặt trong dấu trích dẫn " và ". Kiểu ngoặc kép như vậy thì phổ biến hơn, tuy nhiên kiểu đơn như ' và ' cũng có thể được dùng. Ví dụ trong một vài trường hợp đặc biệt hiếm, ví dụ như giá trị thuộc tính đã mang dấu ngoặc kép rồi, thì việc sử dụng ngoặc đơn là cần thiết. Ví dụ: name='ban"tay"den'

Các loại thẻ cơ bản HTML

Những thẻ quan trọng nhất trong HTML là những thẻ xác định Heading, đoạn văn và xuống dòng.

HTML head

```
<html>

<head>

<title>The title is not displayed</title>

</head>

<body>

<p>This text is displayed</p>

</body>
```

</html>

Thông tin tiêu đề ở trong phần head thì không được hiển thị trong cửa sổ trình duyệt.

Đường link tài liệu:

<html>

<head>

<base target="_blank">

</head>

<body>

This link

Đoạn này sẽ cho ra một cửa sổ mới bởi vì nó được gắn thuộc tính target là "_blank".

This link

Chỉ thực hiện chuyển cửa sổ hiện hành đến trang cần đến.

</body>

</html>

Cách sử dụng thẻ <base> để làm cho tất cả các đường link trên một trang mở ở một cửa sổ mới.

Thành phần của head

Thành phần của head bao gồm những thông tin chung, hay còn được gọi là meta-information về tài liệu. Meta có nghĩa là "thông tin về". Bạn có thể nói rằng meta-data có nghĩa rằng thông tin về data, hoặc meta-infomation có nghĩa rằng thông thin về thông tin.

Thông tin trong thành phần head.

Theo như tiêu chuẩn của HTML, chỉ một vài thẻ được chính thức đặt trong phần head đó là: base>, <link>, <meta>, <title>, <style>, and <script>.

Đây là một ví dụ hợp quy định

<head> This is some text

</head>

Trong trương fhợp này trình duyệt có hai lựa chọn:

Hiển thị chữ bởi vì nó nằm trong thành phần đoạn văn là và

Ẩn chữ bởi vì nó nằm trong thành phần Head.

Nếu bạn muốn thêm một thành phần HTML là <h1> hoặc vào phần head như trên, hầu hết các trình duyệt sẽ hiển thị chúng thậm chí như thế là không hợp lệ. Trình duyệt có nên bỏ qua những lỗi như thế không? chúng tôi không nghĩ thế! nhưng người ta lại cho thế là đúng!

Thẻ Head

Thẻ	Miêu tả đặc tính
<head>	Xác định thông tin về tài liệu
<title>	Xác định tiêu đề của tài liệu
<base>	Xác định địa chỉ URL cơ bản cho tất cả các đường liên kết trên trang
<link>	Xác định tài nguyên để liên kết đến
<meta>	Xác định meta-information
<!DOCTYPE>	Xác định loại tài liệu. Thẻ này đi trước thẻ <html>

HTML Meta

<html>

```
<head>
<meta name="author"
content="Jan Egil Refsnes">
<meta name="revised"
content="Jan Egil Refsnes,6/10/99">
<meta name="generator"
content="Microsoft FrontPage 4.0">
</head>
<body>
<p>
The meta attributes of this document identify the author and the editor software.
</p>
</body>
</html>
```

Thông tin bên trong thành phần meta miêu tả về tài liệu.

```
<html>
<head>
<meta name="description"
content="HTML examples">
<meta name="keywords"
content="HTML, DHTML, CSS, XML, XHTML, JavaScript, VBScript">
</head>
```

<body>

The meta attributes of this document describe the document and its keywords.

</body>

</html>

Thông tin bên trong thành phần meta miêu tả từ khoá của tài liệu.

Tái định hướng người dùng

<html>

<head>

<meta http-equiv="Refresh"

content="5;url=http://www.utehy.edu.vn">

</head>

<body>

Sorry! We have moved! The new URL is:

http://www.utehy.edu.vn

You will be redirected to the new address in five seconds.

If you see this message for more than 5 seconds, please click on the link above!

</p>

</body>

</html>

Cách tái định hướng người dùng nếu bạn thay đổi tên miền.

Thành phần của meta

Như đã được giải thích ở chương trước, thành phần của head bao gồm những thông tin chung về tài liệu. HTML cũng có một thành phần meta và nằm trong phần head. Mục đích của thành phần meta là cung cấp meta-information về tài liệu.

Hầu hết thành phần meta được sử dụng để cung cấp thông tin liên quan đến trình duyệt hoặc những công cụ tìm kiếm như được miêu tả nội dung của tài liệu.

Một vài công cụ tìm kiếm trên mạng sử dụng tên và nội dung của thẻ meta trong trang index của bạn.

<meta name="description" content="Trang web chuyên về Photoshop, thiết kế web">

Mục đích chính của thuộc tính tên và nội dung là để miêu tả nội dung của trang. Tuy nhiên, bởi vì quá nhiều người đã lạm dụng thẻ meta để spam bằng cách lập đi lập lại từ khoá để cho trang web của họ có rank cao hơn, cho nên một vài công cụ tìm kiếm đã hoàn toàn không sử dụng thẻ meta nữa.

Đôi khi bạn có thể thấy thuộc tính thẻ meta là unknown như sau:

<meta name="security" content="low">

Nếu bạn có thấy thì cũng cứ chấp nhận nó bởi vì nó là một cái gì đó duy nhất cho trang hoặc cho tác giả của trang đó và nó có thể chẳng có gì liên quan đến bạn hết.

Xác định thông tin trên internet (URL = Uniform Resource Locator) HTML.

Khi bạn nhấp chuột vào link trong một tài liệu HTML như thế này: Trang Trước, một thẻ gạch chân <a> sẽ chỉ đến một nơi (một địa chỉ) trên Web với một giá trị thuộc tính href như: Trang Trước

Liên kết Trang Trước ở ví dụ trên là đường liên kết có liên quan đến một trang web mà bạn đang lướt ở đó, và trình duyệt của bạn sẽ tạo ra một địa chỉ web đầy đủ như sau: http://www.utehy.vn/html/trangtruoc.htm để truy cập trang đó.

Uniform Resource Locator (URL)

URL được dùng để xác định địa chỉ của một tài liệu (hoặc dữ liệu khác) trên World Wide Web. Một địa chỉ đầy đủ sẽ như sau: http://www.utehy.vn/html/trangtruoc.htm và tuân theo cú pháp sau:

Scheme://host.domain:port/path/filename

Scheme: Là một trong các giao thức Internet, gồm http, ftp, gopher, news (USENET news), nntp (Network News Transfer Protocol), Telnet và WAIS (Wide Area Information Servers), và những giao thức khác. Đija chỉ dưới đây ừung giao thức http:http://www.utehy.vn/html/html_basic.htm

Domain: xác định tên miền của trang web trên Internet ví dụ như utehy.com

Host xác định tên miền của host. Nếu được bỏ qua, thì mặc định của host cho http là www.

Port xác định port number tại host. Số cổng thường được bỏ qua. Số cổng mặc định của http là 80.

Path xác định đường dẫn trên server. Nếu đường dẫn được bỏ qua, thì tài liệu phải được định vị tại thư mục gốc của trang web.

Filename xác định tên của tài liệu. Tên mặc định của một tài liệu có thể là default.asp hoặc index.html hoặc một cái gì đó phụ thuộc vào những cài đặt của server.

URL Scheme

Dưới đây là ví dụ của những lược đồ thông dụn nhất

Schemes	Access
file	Một tệp tin ở máy tính của bạn
ftp	Một tệp tin ở FTP server
http	Một tệp tin ở World Wide Web Server
gopher	Một tệp tin ở Gopher server

news	Usenet newsgroup
telnet	Telnet connection
WAIS	Một tệp tin ở WAIS server

Truy cập vào Newsgroup

Dòng code HTML sau:

HTML Newsgroup

tạo ra một đường liên kết đến newsgroup như là HTML Newsgroup.

Download với FTP

Dòng code HTML sau:

Download WinZip

Tạo ra một đường link download như là: Download WinZip.

Chú ý: Liên kết trên chỉ là ví dụ thôi! cho nên nó không có giá trị.

Liên kết đến hệ thống Mail

Dòng code HTML sau:

admin@utehy.com

Tạo ra một đường liên kết với hệ thống email của bạn là admin@utehy.com

Headings

Headings được định dạng với hai thẻ <h1> đến <h6>. <h1> xác định heading lớn nhất. <h6> xác định heading nhỏ nhất

<h1>Đây là heading</h1><h2>Đây là heading</h2><h3>Đây là heading</h3><h4>Đây là heading</h4>

<h5>Đây là heading</h5><h6>Đây là heading</h6>

HTML sẽ tự động thêm một dòng trắng trước và sau mỗi heading.

Đoạn văn – paragraphs

Paragraphs được định dạng bởi thẻ .

Đây là đoạn văn Đây là một đoạn văn khác

HTML sẽ tự động thêm một dòng trắng trước và sau mỗi heading.

Line Breaks - xuống dòng

Thẻ
 được sử dụng khi bạn muốn kết thúc một dòng nhưng lại không muốn bắt đầu một đoạn văn khác. Thẻ <br> sẽ tạo ra một lần xuống dòng khi bạn viết nó.

Đây <br> là một đo
ạn văn với thẻ xuống hàng

Thẻ <br> là một thẻ trống, nó không cần thẻ đóng dạng </br>

Lời chú thích trong HTML

Thẻ chú thích được sử dụng để thêm lời chú thích trong mã nguồn của HTML. Một dòng chú thích sẽ được bỏ qua bởi trình duyệt. Bạn có thể sử dụng chú thích để giải thích về code của bạn, để sau này bạn có phải quay lại chỉnh sửa gì thì cũng dễ nhớ hơn.

<!-- Chú thích ở trong này -->

Bạn cần một dấu chấm than ! ngay sau dấu nhỏ hơn nhưng không cần ở dấu lớn hơn.

Các loại thẻ nâng cao trong HTML

HTML layout

Rất nhiều trang web bạn có thể thấy rằng trang của họ được thiết kế dạng cột giống như tờ báo vậy, đó là họ sử dụng cột của HTML.

Việc thông dụng nhất khi dùng HTML là sử dụng bảng HTML để định dạng một giao diện của một trang HTML.Một phần của trang bạn đang đọc này được định dạng bởi hai cột như dạng cột của báo.Bạn có thể nhận ra rằng nhưng gì bạn đang đọc được chia ra làm hai cột bên trái và bên phải.Dòng chữ bên này ở cột bên trái	Code HTML <table> được sử dụng để chia một phần của trang thành hai cột.Tôi cố tình để đường viền cho bạn dễ nhận thấy đây là cấu trúc bảng.Dù cho bạn có viết bao nhiêu chữ đi chăng nữa, thì nõ vẫn ngoan ngoãn nằm trong cấu trúc bảng bạn tạo ra nó.

Vẫn layout cũ nhưng với màu được thêm vào

Việc thông dụng nhất khi dùng HTML là sử dụng bảng HTML để định dạng một giao diện của một trang HTML.Một phần của trang bạn đang đọc này được định dạng bởi hai cột như dạng cột của báo.Bạn có thể nhận ra rằng nhưng gì bạn đang đọc được chia ra làm hai cột bên trái và bên phải.Dòng chữ bên này ở cột bên trái	Code HTML <table> được sử dụng để chia một phần của trang thành hai cột.Tôi cố tình để đường viền cho bạn dễ nhận thấy đây là cấu trúc bảng.Dù cho bạn có viết bao nhiêu chữ đi chăng nữa, thì nõ vẫn ngoan ngoãn nằm trong cấu trúc bảng bạn tạo ra nó.

HTML Fonts

Việc sử dụng thẻ <font> trong HTML không được tán thành và sẽ bị loại bỏ ở những phiên bản HTML mới hơn. Dù cho rất nhiều người dùng nó, nhưng bạn nên cố gắng tránh dùng nó và dùng style để thay thế.

Thẻ HTML <font>

Với một dòng code HTML như sau, bạn có thể cụ thể kích cỡ và loại font cho trình duyệt

```
<p><font size="2" face="Verdana">

This is a paragraph.
```


This is another paragraph.

Thuộc tính của font

Thuộc tính	Ví dụ	Mục đích
size="number"	size="2"	Xác định kích cỡ
size="+number"	size="+1"	Tăng kích cỡ font
size="-number"	size="-1"	Giảm kích cỡ font
face="face-name"	face="Times"	Xác định tên font
color="color-value"	color="#eeff00"	Xác định màu chữ
color="color-name"	color="red"	Xác định màu chữ

Frames

Với frame bạn có thể hiển thị nhiều trang web trên cùng một cửa sổ trình duyệt. Mỗi một tài liệu HTML được gọi là một frame, và mỗi frame đều độc lập với những frame khác.

Những nhược điểm khi sử dụng frame:

Người làm web phải theo dõi nhiều tài liệu HTML

Khó có thể in ấn toàn bộ trang web.

Thẻ frameset

Thẻ <frameset> xác định bạn sẽ chia cửa sổ trình duyệt thành những frame như thế nào. Môi một frame xác định một tập hợp các hàng hoặc cột. Giá trị của hàng hoặc cột chỉ ra diện tích của màn hình mà frame đó sẽ chiếm.

Thẻ Frame

Thẻ <frame> xác định tài liệu HTML nào sẽ được chèn vào mỗi frame.Ở ví dụ dưới đây bạn có một frameset với hai cột. Cột thứ nhất được thiết lập là chiếm 25% độ rộng của cửa sổ trình duyệt. Cột thứ hai được thiết lập sẽ chiếm 75% độ rộng của cửa sổ trình

duyệt. Tài liệu html tên là "frame_a.htm" được chèn vào cột thứ nhất, và "frame_b.htm" được chèn vào cột thứ hai.

<frameset cols="25%,75%"><frame src="frame_a.htm"><frame src="frame_b.htm"></frameset>

Chú ý:

Nếu frame của bạn có đường viền thì người dùng có thể định lại kích thước bằng cách kéo nó. Để tránh việc này bạn có thể thêm noresize="noresize" vào thẻ <frame>

Thêm thẻ <noframes> cho trình duyệt không hỗ trợ frame.

Navigation frame

Ví dụ này hướng dẫn bạn cách tạo ra một mục lục (navigation) bằng frame. Frame navigation chứa một danh sách những đường link và đích là ở frame thứ hai. Tài liệu "tryhtml_contents.htm" chứa 3 đường link và code của đường link như sau

Frame a
Frame b
Frame c

Thẻ frame

Tag	Mô Tả
<frameset>	Kiểu a set of frames
<frame>	Kiểu a sub window (a frame)
<noframes>	Kiểu a noframe section for browsers that do not handle frames
<iframe>	Kiểu an inline sub window (frame)

Bảng HTML

Bảng được định dạng bởi thẻ <table>. Một bảng được chia ra làm nhiều hàng với thẻ <tr>, môi hàng được chia ra làm nhiều cột dữ liệu với thẻ <td>. Cbữ td là chữ viết tắt của "table data", là nội dung của cột dữ liệu. Một cột dữ liệu có thể bao gồm chữ, hình ảnh, danh sách, đoạn vắn, form và bảng vv…

<table border="1">

<tr>

<td>row 1, cell 1</td>

<td>row 1, cell 2</td>

</tr>

<tr>

<td>row 2, cell 1</td>

<td>row 2, cell 2</td>

</tr>

</table>

Đoạn code trên sẽ hiển thị như thế này trong cửa sổ trình duyệt

row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

Bảng và thuộc tính đường biên

Nếu bạn không thiết lập thuộc tính đường biên cho bảng thì bảng của bạn sẽ được hiển thị mà không có đường biên. Đôi khi nó có thể hữu dụng nhưng thường thì bạn muốn bảng của bạn có đường biên.

Để hiển thị đường biên của một bảng, bạn phải sử dụng thuộc tính đường biên.

<table border="1"><tr><td>Row 1, cell 1</td><td>Row 1, cell 2</td></tr></table>

Heading trong bảng

Heading trong một bảng được xác định bằng thẻ <th>

<table border="1"><tr><th>Heading</th><th>Another Heading</th></tr><tr><td>row 1, cell 1</td><td>row 1, cell 2</td></tr>

<tr><td>row 2, cell 1</td><td>row 2, cell 2</td></tr></table>

Nó sẽ hiển thị như thế này ở trên trình duyệt

Heading	Another Heading

row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

Cột trống trong bảng

Cột trống không có nội dung thì không được hiển thị tốt lắm ở hầu hết các trình duyệt.

<table border="1"><tr><td>row 1, cell 1</td><td>row 1, cell 2</td></tr><tr><td>row 2, cell 1</td><td></td></tr></table>

Nó sẽ có dạng thế này trên trình duyệt

row 1, cell 1	row 1, cell 2
row 2, cell 1	

Bạn chú ý rằng đường biên bao quanh cột trống bị mất (nhưng trong trình duyệt Mozilla Firefox nó sẽ hiển thị đường biên)

Để tránh điều này xảy ra, bạn thêm một non-breaking space ( ) vào cột trống đó, để làm cho đường biên của nó được hiện thỉ.

<table border="1"><tr>

<td>row 1, cell 1</td><td>row 1, cell 2</td></tr><tr><td>row 2, cell 1</td><td> </td></tr></table>

Nó sẽ hiển thị như sau ở trình duyệt

row 1, cell 1	row 1, cell 2
row 2, cell 1	

Chú ý:

Loại thẻ <thead>,<tbody> and <tfoot> ít khi được sử dụng, bởi vì nó không phải là tính năng được.

Thẻ table

Tag	Mô Tả

<table>	Vẽ bảng
<th>	hàng đầu của bảng
<tr>	hàng trong bảng
<td>	ô trong hàng
<caption>	nhãn của bảng
<colgroup>	nhóm các cột
<col>	Định các thuộc tính của cột
<thead>	Hàng Đầu bảng
<tbody>	Thân của bảng
<tfoot>	Hàng cuối bảng

Forms và trường nhập liệu

HTML form được sử dụng để chọn những dữ liệu nhập vào khác nhau của người dùng.

Form

Một form là một vùng mà nó bao gồm những thành phần của form. Thành phần của form là những thành phần cho phép người dùng có thể điền thông tin như là trường chữ, menu thả xuống, nút radio, và các hộp kiểm vào một form.

Một form được xác định bởi thẻ <form>

<form><input><input></form>

Nhập liệu

Thẻ form được sử dụng nhiều nhất là thẻ <input>. Loại dữ liệu nhập vào sẽ được xác định bởi thuộc tính của nó. Những trường nhập liệu được sử dụng nhiều nhất được giải thích ở dưới đây.

Text field

Text field được sử dụng khi bạn muốn người dùng đánh chữ, số v.v.. vào một form.

<form>First name:

<input type="text" name="firstname">

Last name:

<input type="text" name="lastname">

</form>

Nó sẽ xuất hiện như sau trong trình duyệt

First name:

Last name:

Bạn chú ý rằng bản thân của form thì bị ẩn đi. Hơn nữa trên hầu hết các trình duyệt trường text được mặc định là 20 ký tự.

Nút radio

Nút radio được sử dụng khi bạn muốn người dùng chọn một trong những lựa chọn bạn đưa ra.

<form><input type="radio" name="sex" value="male"> Male

<input type="radio" name="sex" value="female"> Female

</form>

Nó sẽ xuất hiện như sau trên trình duyệt

Male

Female

Chú ý rằng chỉ có một lựa chọn có thể được chọn.

Hộp kiểm

Hộp kiểm được sử dụng khi bạn muốn người chọn có thể chọn nhiều lựa chọn hơn.

<form><input type="checkbox" name="bike">

I have a bike

<input type="checkbox" name="car">

I have a car

</form>

Nó sẽ như sau trong trình duyệt

I have a bike

I have a car

Thuộc tính hoạt động của form và nút Submit.

Khi người dùng nhấp chuột vào nút "submit", nội dung của form đó sẽ được gửi đến một tệp tin khác. Thuộct ính hoạt động của form xác định tên của file mà nó sẽ gửi nội dung đến. Tệp tin đó được xác định trong thuộc tính hoạt động của form và thường thì nó sẽ có những hành động với dữ liệu nó nhận được.

<form name="input" action="html_form_action.asp"

method="get">Username:

<input type="text" name="user"<input type="submit" value="Submit">

</form>

Trong trình duyệt nó nhìn như sau

Username:

Khi bạn gõ tên bạn vào trường chữ ở trên và nhấp vào nút Submit, bạn sẽ gửi thông tin đó vào một trang gọi là "html_form_action.asp". Trang đó sẽ chỉ cho bạn thấy dữ liệu nhận được.

Thẻ của form

Tag	Mô Tả
<form>	Kiểu form để nhập thông tin
<input>	Một ô nhập liệu
<textarea>	Vùng nhập liệu có nhiều hàng
<label>	Nhãn
<fieldset>	Nhóm các vùng nhập với nhau

<legend>	nhản của 1 fieldset
<select>	Danh sách chọn
<optgroup>	nhóm các phần tử trong danh sách chọn
<option>	1 phần tử trong danh sách chọn
<button>	Nút bấm
<isindex>	Hết hỗ trợ. Dùng <input> thay thế

Bài 3 : Cascading style sheets

Cơ bản về cascading style sheets

Khái niệm về Cascading style sheets (CSS)

CSS (Cascading Style Sheets) là một ngôn ngữ quy định cách trình bày cho các tài liệu viết bằng HTML, XHTML, XML, SVG, hay UML,…

Nếu bạn đã từng học qua HTML thì cũng biết HTML cũng hỗ trợ một số thuộc tính định dạng cơ bản cho text, picture, table, … nhưng nó không thật sự phong phú và chính xác như nhau trên mọi hệ thống. CSS cung cấp cho bạn hàng trăm thuộc tính trình bày dành cho các đối tượng với sự sáng tạo cao trong kết hợp các thuộc tính giúp mang lại hiệu quả. Ngoài ra, hiện tại CSS đã được hỗ trợ bởi tất cả các trình duyệt, nên bạn hoàn toàn có thể tự tin trang web của mình có thể hiển thị hầu như "như nhau" dù trên một hệ thống sử dụng Windows, Linux hay trên một máy Mac miễn là bạn đang sử dụng một phiên bản trình duyệt mới nhất.

Sử dụng các mã định dạng trực tiếp trong HTML tốn hao nhiều thời gian thiết kế cũng như dung lượng lưu trữ trên đĩa cứng. Trong khi đó CSS đưa ra phương thức "tờ mẫu ngoại" giúp áp dụng một khuôn mẫu chuẩn từ một file CSS ở ngoài. Nó thật sự có hiệu quả đồng bộ khi bạn tạo một website có hàng trăm trang hay cả khi bạn muốn thay đổi một thuộc tính trình bày nào đó. Hãy thử tưởng tượng bạn có một website với hàng trăm trang và bạn muốn thay đổi font chữ hay màu chữ cho một thành phần nào đó. Đó thật sự sẽ là một công việc buồn chán và tốn nhiều thời gian. Nhưng với việc sử dụng CSS việc đó là hoàn toàn đơn giản cũng như là bạn có một trò ma thuật nào đó.

Ngoài ra, CSS còn cho phép bạn áp đặt những kiểu trình bày thích hợp hơn cho các phương tiện khác nhau như màn hình máy tính, máy in, điện thoại,…

CSS được cập nhật liên tục mang lại các trình bày phức tạp và tinh vi hơn.

Cách sử dụng Style

Cú pháp CSS

Để tìm hiểu cú pháp CSS chúng ta hãy thử xem một ví dụ sau.

Ví dụ: Để định màu nền cho một trang web là xanh nhạt (light cyan) chúng ta dùng code sau:

Trong HTML: <body bgcolor="#00BFF3">

Trong CSS: body { background-color:#00BFF3; }

Nhìn qua ví dụ trên ít nhiều chúng ta cũng thấy được mối tương đồng giữa các thuộc tính trong HTML và CSS cho nên nếu bạn đã học qua HTML thì cũng sẽ rất dễ dàng tiếp thu CSS. Đó là một chút lợi thế của câu chuyện hành trình mà Pearl đã nói ở bài trước. Nhưng không sao cả, bây giờ hãy nhìn vào ví dụ của chúng ta và các bạn xem nó có giống với cấu trúc sau không nhé.

Cú pháp CSS cơ bản:

Selector { property:value; }

Trong đó:

+ Selector:Các đối tượng mà chúng ta sẽ áp dụng các thuộc tính trình bày. Nó là các tag HTML, class hay id (chúng ta sẽ học về 2 thành phần này ở bài học sau). Ví dụ: body, h2, p, img, #title, #content, .username,…

Trong CSS ngoài viết tên selector theo tên tag, class, id. Chúng ta còn có thể viết tên selector theo phân cấp như để chỉ các ảnh ở trong #entry, chúng ta viết selector là #entry img, như vậy thì các thuộc tính chỉ định sẽ chỉ áp dụng riêng cho các ảnh nằm trong #entry.

Khi viết tên cho class, đôi khi sẽ có nhiều thành phần có cùng class đó, ví dụ như thẻ img và thẻ a cùng có class tên vistors nhưng đây lại là hai đối tượng khác nhau, 1 cái là ảnh của người thăm, 1 cái là liên kết tới trang người thăm. Nên nếu khi viết CSS ta ghi là .visitors { width:50 } thì sẽ ảnh hưởng tới cả hai thành phần. Nên trong trường hợp này, nếu bạn có ý dùng CSS đó chỉ riêng phần ảnh thì chỉ nên ghi là img .visitors thôi.

Một lối viết tên selector nữa đó là dựa trên tên các thuộc tính có trong HTML. Ví dụ trong HTML ta có đoạn mã như vầy: <input name="Search" type="Text" value="Key Word">. Để áp dụng thuộc tính CSS cho riêng ô tìm kiếm này chúng ta sẽ dùng selector input[name="Search"].

Ngoài việc viết tên selector cụ thể, chúng ta cũng có thể dùng một selector đại diện như * { color:red } sẽ tác động đến tất cả các thành phần có trên trang web làm cho chúng có text màu đỏ.

+ Property:Chính là các thuộc tính quy định cách trình bày. Ví dụ: background- color, font-family, color, padding, margin,…

Mỗi thuộc tính CSS phải được gán một giá trị. Nếu có nhiều hơn một thuộc tính cho một selector thì chúng ta phải dùng một dấu ; (chấm phẩy) để phân cách các thuộc tính. Tất cả các thuộc tính trong một selector sẽ được đặt trong một cặp ngoặc nhọn sau selector.

Ví dụ:

```
body { background:#FFF; color:#FF0000; font-size:14pt }
```

Để dễ đọc hơn, bạn nên viết mỗi thuộc tính CSS ở một dòng. Tuy nhiên, nó sẽ làm tăng dung lượng lưu trữ CSS của bạn.

Ví dụ:

```
body {

background:#FFF; bolor:#FF0000; font-size:14pt

}
```

Đối với một trang web có nhiều thành phần có cùng một số thuộc tính, chúng ta có thể thực hiện gom gọn lại như sau:

```
h1 { color:#0000FF;

text-transform:uppercase }

h2 {

color:#0000FF;

text-transform:uppercase;

}

h3 {

color:#0000FF;

text-transform:uppercase;

}
```

Value:Giá trị của thuộc tính. Ví dụ: như ví dụ trên value chính là #FFF dùng để định màu trắng cho nền trang.

Đối với một giá trị có khoảng trắng, bạn nên đặt tất cả trong một dấu ngoặc kép. Ví dụ: font-family:”Times New Roman”

Đối với các giá trị là đơn vị đo, không nên đặt một khoảng cách giữa số đo với đơn vị của nó. Ví dụ: width:100 px. Nó sẽ làm CSS của bạn bị vô hiệu trên Mozilla/Firefox hay Netscape.

Chú thích trong CSS:

Cũng như nhiều ngôn ngữ web khác. Trong CSS, chúng ta cũng có thể viết chú thích cho các đoạn code để dễ dàng tìm, sửa chữa trong những lần cập nhật sau. Chú thích trong CSS được viết như sau /* Nội dung chú thích */

Ví dụ:

```
/* Màu chữ cho trang web */

body {

color:red

}
```

Đơn vị CSS:

Trong CSS2 hỗ trợ các loại đơn vị là đơn vị đo chiều dài và đơn vị đo góc, thời gian, cường độ âm thanh và màu sắc. Tuy nhiên, sử dụng phổ biến nhất vẫn là đơn vị đo chiều dài và màu sắc. Sau đây là bảng liệt kê các đơn vị chiều dài và màu sắc dùng trong CSS.

Đơn vị chiều dài

Đơn vị	Mô tả	Đơn vị	Mô tả
%	Phần trăm		
in	Inch(1 inch=2.54 cm)		
cm	Centimeter		
mm	Millimeter		
ex	1 ex bằng chiều cao của chữ x in thường của font hiện hành. Do đó, đơn vị này không những phụ thuộc trên kích cỡ font chữ mà còn		

	phụ thuộc loại font chữ vì cùng 1 cỡ 14px nhưng chiều cao chữ x của font Times và font Tohama là khác nhau.
em	1 em tương đương kích thước font hiện hành, nếu font hiện hành có kích cỡ 14px thì 1 em = 14 px. Đây là một đơn vị rất hữu ích trong việc hiển thị trang web.
pt	Point (1 pt = 1/72 inch)
pc	Pica (1 pc = 12 pt)
px	Pixels (điểm ảnh trên màn hình máy tính)

Đơn vị màu sắc

Đơn vị	Mô tả
Color-name	Tên màu tiếng Anh. Ví dụ: black, white, red, green, blue, cyan, magenta,…
RGB (r,g,b)	Màu RGB với 3 giá trị R, G, B có trị từ 0 – 255 kết hợp với nhau tạo ra vô số màu.
RGB (%r,%g,%b)	Màu RGB với 3 giá trị R, G, B có trị từ 0 – 100% kết hợp.
Hexadecimal RGB	Mã màu RGB dạng hệ thập lục. Ví dụ: #FFFFFF: trắng, #000000: đen, #FF00FF: đỏ tươi.

Các loại Style trong ứng dụng website

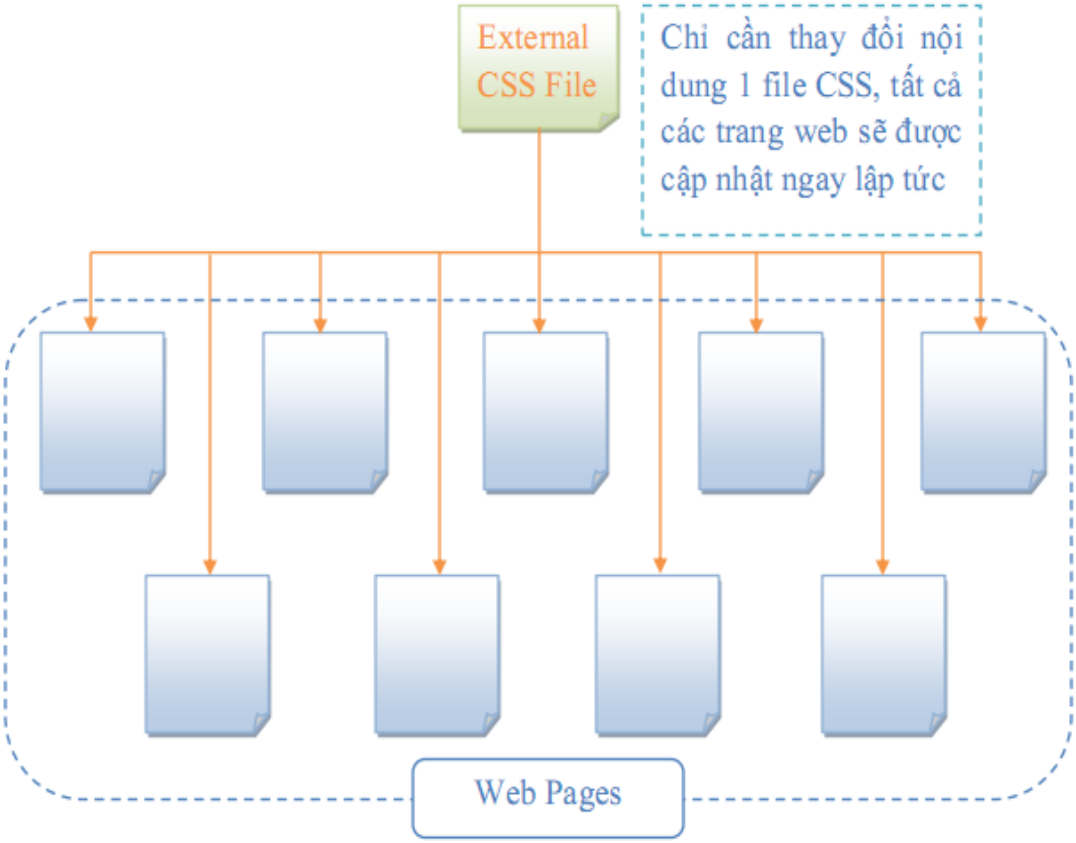
Ở trên chúng ta đã tìm hiểu về cú pháp viết CSS, nhưng còn đặt nó ở đâu trong tài liệu HTML? Trong phần này, Pearl xin giới thiệu với các bạn về vấn đề này.

Chúng ta có ba cách khác nhau để nhúng CSS vào trong một tài liệu HTML

Style Sheet "ngoại" (External Style)

Tương tự như cách 2 nhưng thay vì đặt tất cả các mã CSS trong thẻ style chúng ta sẽ đưa chúng vào trong một file CSS (có phần mở rộng .css) bên ngoài và liên kết nó vào trang web bằng thuộc tính href trong thẻ link.

Đây là cách làm được khuyến cáo, nó đặc biệt hữu ích cho việc đồng bộ hay bảo trì một website lớn sử dụng cùng một kiểu mẫu. Các ví dụ trong sách này cũng được trình bày theo kiểu này.



Nào bây giờ chúng ta hãy mở Notepad lên và thử thực hiện theo ví dụ sau:

Đầu tiên chúng ta sẽ tạo ra một file vidu.html có nội dung như sau:

```
<html>

<head>

<title>Ví dụ</title>

<link rel=”stylesheet” type=”text/css” href=”style.css” />

</head>

<body>

<p>^_^ Welcome To WallPearl’s Blog ^_^</p>

</body>

</html>
```

Sau đó hãy tạo một file style.css với nội dung:

```
body {

background-color:#FFF

}

p {

color:#00FF00

}
```

Hãy đặt 2 file này vào cùng một thư mục, mở file vidu.html trong trình duyệt của bạn và xem thành quả.

Lưu ý:

Để lưu 1 file với 1 đuôi khác .txt trong Notepad chúng ta chọn Save as type là All Files. Có thể chọn Encoding là UTF-8, nếu bạn chú thích CSS bằng tiếng Việt.

Trong CSS chúng ta còn có thể sử dụng thuộc tính @import để nhập một file CSS vào CSS hiện hành. Cú pháp: @import url(link)

Style sheet "nội" (Internal Style)

Thật ra nếu nhìn kỹ chúng ta cũng nhận ra đây chỉ là một phương cách thay thế cách thứ nhất bằng cách rút tất cả các thuộc tính CSS vào trong thẻ style (để tiện cho công tác bảo trì, sửa chữa ấy mà).

Cũng ví dụ làm trang web có màu nền trắng, đoạn văn bản chữ xanh lá, chúng ta sẽ thể hiện như sau:

```
<html>

<head>

<title>Ví dụ</title>

<style type=”text/css”>

body { background-color:#FFF }
```

```
p { color:#00FF00 }

</style>

</head>

<body>

<p>^_^ Welcome To WallPearl's Blog ^_^</p>

</body>

</html>
```

Lưu ý:Thẻ style nên đặt trong thẻ head.

Đối với những trình duyệt cũ, không thể nhận ra thẻ <style>. Theo mặc định, thì khi một trình duyệt không nhận ra một thẻ thì nó sẽ hiện ra phần nội dung chứa trong thẻ. Như ở ví dụ trên, nếu trình duyệt không hỗ trợ thẻ style thì 2 dòng CSS: body {background-color:#FFF } p { color:#00FF00 } sẽ hiện ra trên trình duyệt. Để tránh tình trạng này, bạn nên đưa vào thêm dấu <!-- ở trước và --> ở sau khối code CSS. Như ví dụ trên sẽ viết lại là:

```
<style type="text/css">

<!-- body { background-color:#FFF }

p { color:#00FF00 } -->

</style>
```

Style sheet "địa phương" (Inline Style)

Đây là một phương pháp nguyên thủy nhất để nhúng CSS vào một tài liệu HTML bằng cách nhúng vào từng thẻ HTML muốn áp dụng. Và dĩ nhiên trong trường hợp này chúng ta sẽ không cần selector trong cú pháp.

Lưu ý: Nếu bạn muốn áp dụng nhiều thuộc tính cho nhiều thẻ HTML khác nhau thì không nên dùng cách này.

Ở ví dụ sau chúng ta sẽ tiến hành định nền màu trắng cho trang và màu chữ xanh lá cho đoạn văn bản như sau:

```
<html>
```

```
<head>

<title>Ví dụ</title>

</head>

<body style="background-color=#FFF;">

<p style="color:green">^_^ Welcome To WallPearl's Blog ^_^</p>

</body>

</html>
```

Sự ưu tiên:

Trước khi thực thi CSS cho một trang web. Trình duyệt sẽ đọc toàn bộ CSS mà trang web có thể được áp dụng, bao gồm: CSS mặc định của trình duyệt, file CSS bên ngoài liên kết vào trang web, CSS nhúng trong thẻ <style> và các CSS nội tuyến. Sau đó, trình duyệt sẽ tổng hợp toàn bộ CSS này vào một CSS ảo, và nếu có các thuộc tính CSS giống nhau thì thuộc tính CSS nào nằm sau sẽ được ưu tiên sử dụng (cái này cũng giống như chương trình "Ai Là Triệu Phú" trên truyền hình vậy, chỉ câu trả lời sau cùng mới được chấp nhận (smile)). Theo nguyên tắc đó trình duyệt của bạn sẽ ưu tiên cho các CSS nội tuyến > CSS bên trong > CSS bên ngoài > CSS mặc định của trình duyệt.

Ví dụ:

Trong một trang web có liên kết tới file style.css có nội dung như sau:

```
p {

color:#333; text-align:left; width:500px

}
```

Trong thẻ <style> giữa thẻ <head> cũng có một đoạn CSS liên quan:

```
p {

background-color:#FF00FF; text-align:right; width:100%;

height:150px

}
```

Trong phần nội dung trang web đó cũng có sử dụng CSS nội tuyến:

<p style="height:200px; text-align:center; border:1px solid #FF0000; color:#000" } Khi thực thi CSS trình duyệt sẽ đọc hết tất cả các nguồn chứa style rồi sẽ tổng hợp lại vào một CSS ảo và nếu có sự trùng lắp các thuộc tính CSS thì nó sẽ lấy thuộc

Tính CSS có mức ưu tiên cao hơn. Như ví dụ trên chúng ta sẽ thấy CSS cuối cùng mà phần tử p nhận được là:

p {

background-color:#FF00FF;

width:100%; height:200px; text-align:center;

border:1px solid #FF0000;

color:#000

}

Vậy có cách nào để thay đổi độ ưu tiên cho một thuộc tính nào đó? Thật ra thì trong CSS đã có sẵn một thuộc tính giúp chúng ta thực hiện điều này, đó chính là thuộc tính !important. Chỉ cần bạn đặt thuộc tính này sau một thuộc tính nào đó theo cú pháp selector { property:value !important } thì trình duyệt sẽ hiểu đây là một thuộc tính được ưu tiên. Bây giờ, chúng ta cùng xét lại ví dụ trên nhưng có đặt thêm một số thuộc tính !important vào xem kết quả như thế nào nhé.

p {

}

p {

}

width:500px;

text-align:left !important;

color:#333 !important

background-color:#FF00FF;

width:100%; height:150px !important; text-align:right;

<p style="text-align:center; height:200px; border:1px solid #FF0000; color:#000" }

Phần CSS sẽ tác động lên thuộc tính p là:

p {

background-color:#FF0000;

width:100%; height:150px !important; text-align:left !important;

border:1px solid #FF0000;

color:#333 !important

}

Lưu ý: Cùng một thuộc tính cho một selector thì nếu cả hai thuộc tính đều đặt !important thì cái sau được lấy.

Các loại Style trong ứng dụng website

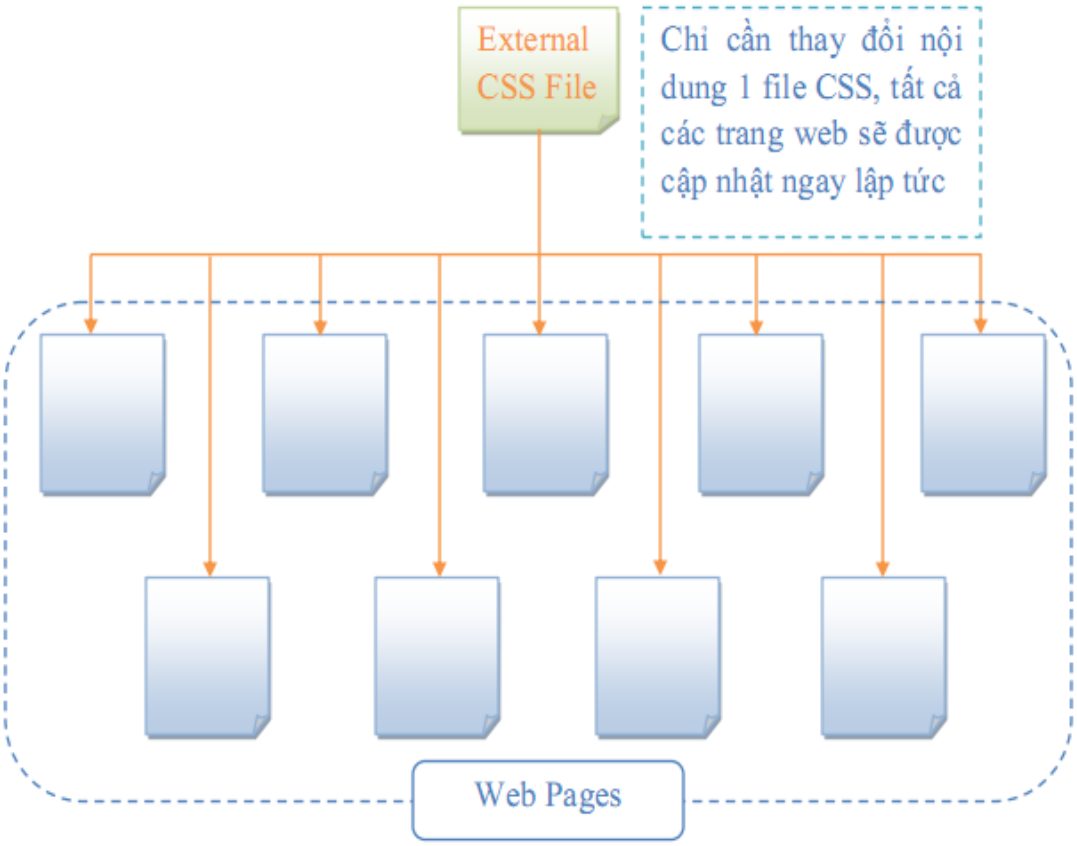
Ở trên chúng ta đã tìm hiểu về cú pháp viết CSS, nhưng còn đặt nó ở đâu trong tài liệu HTML? Trong phần này, Pearl xin giới thiệu với các bạn về vấn đề này.

Chúng ta có ba cách khác nhau để nhúng CSS vào trong một tài liệu HTML

Style Sheet "ngoại" (External Style)

Tương tự như cách 2 nhưng thay vì đặt tất cả các mã CSS trong thẻ style chúng ta sẽ đưa chúng vào trong một file CSS (có phần mở rộng .css) bên ngoài và liên kết nó vào trang web bằng thuộc tính href trong thẻ link.

Đây là cách làm được khuyến cáo, nó đặc biệt hữu ích cho việc đồng bộ hay bảo trì một website lớn sử dụng cùng một kiểu mẫu. Các ví dụ trong sách này cũng được trình bày theo kiểu này.



Nào bây giờ chúng ta hãy mở Notepad lên và thử thực hiện theo ví dụ sau:

Đầu tiên chúng ta sẽ tạo ra một file vidu.html có nội dung như sau:

```
<html>

<head>

<title>Ví dụ</title>

<link rel="stylesheet" type="text/css" href="style.css" />

</head>

<body>

<p>^_^ Welcome To WallPearl's Blog ^_^</p>

</body>

</html>
```

Sau đó hãy tạo một file style.css với nội dung:

```
body {

background-color:#FFF

}

p {

color:#00FF00

}
```

Hãy đặt 2 file này vào cùng một thư mục, mở file vidu.html trong trình duyệt của bạn và xem thành quả.

Lưu ý:

Để lưu 1 file với 1 đuôi khác .txt trong Notepad chúng ta chọn Save as type là All Files. Có thể chọn Encoding là UTF-8, nếu bạn chú thích CSS bằng tiếng Việt.

Trong CSS chúng ta còn có thể sử dụng thuộc tính @import để nhập một file CSS vào CSS hiện hành. Cú pháp: @import url(link)

Style sheet "nội" (Internal Style)

Thật ra nếu nhìn kỹ chúng ta cũng nhận ra đây chỉ là một phương cách thay thế cách thứ nhất bằng cách rút tất cả các thuộc tính CSS vào trong thẻ style (để tiện cho công tác bảo trì, sửa chữa ấy mà).

Cũng ví dụ làm trang web có màu nền trắng, đoạn văn bản chữ xanh lá, chúng ta sẽ thể hiện như sau:

```
<html>

<head>

<title>Ví dụ</title>

<style type=”text/css”>

body { background-color:#FFF }

p { color:#00FF00 }

</style>

</head>

<body>

<p>^_^ Welcome To WallPearl’s Blog ^_^</p>

</body>

</html>
```

Lưu ý:Thẻ style nên đặt trong thẻ head.

Đối với những trình duyệt cũ, không thể nhận ra thẻ <style>. Theo mặc định, thì khi một trình duyệt không nhận ra một thẻ thì nó sẽ hiện ra phần nội dung chứa trong thẻ. Như ở ví dụ trên, nếu trình duyệt không hỗ trợ thẻ style thì 2 dòng CSS: body {background-color:#FFF } p { color:#00FF00 } sẽ hiện ra trên trình duyệt. Để tránh tình trạng này, bạn nên đưa vào thêm dấu <!-- ở trước và --> ở sau khối code CSS. Như ví dụ trên sẽ viết lại là:

```
<style type=”text/css”>
```

```
<!-- body { background-color:#FFF }

p { color:#00FF00 } -->

</style>
```

Style sheet "địa phương" (Inline Style)

Đây là một phương pháp nguyên thủy nhất để nhúng CSS vào một tài liệu HTML bằng cách nhúng vào từng thẻ HTML muốn áp dụng. Và dĩ nhiên trong trường hợp này chúng ta sẽ không cần selector trong cú pháp.

Lưu ý: Nếu bạn muốn áp dụng nhiều thuộc tính cho nhiều thẻ HTML khác nhau thì không nên dùng cách này.

Ở ví dụ sau chúng ta sẽ tiến hành định nền màu trắng cho trang và màu chữ xanh lá cho đoạn văn bản như sau:

```
<html>

<head>

<title>Ví dụ</title>

</head>

<body style=”background-color=#FFF;”>

<p style=”color:green”>^_^ Welcome To WallPearl’s Blog ^_^</p>

</body>

</html>
```

Sự ưu tiên:

Trước khi thực thi CSS cho một trang web. Trình duyệt sẽ đọc toàn bộ CSS mà trang web có thể được áp dụng, bao gồm: CSS mặc định của trình duyệt, file CSS bên ngoài liên kết vào trang web, CSS nhúng trong thẻ <style> và các CSS nội tuyến. Sau đó, trình duyệt sẽ tổng hợp toàn bộ CSS này vào một CSS ảo, và nếu có các thuộc tính CSS giống nhau thì thuộc tính CSS nào nằm sau sẽ được ưu tiên sử dụng (cái này cũng giống như chương trình “Ai Là Triệu Phú” trên truyền hình vậy, chỉ câu trả lời sau cùng mới được chấp nhận (smile)). Theo nguyên tắc đó trình duyệt của bạn sẽ ưu tiên cho các CSS nội tuyến > CSS bên trong > CSS bên ngoài > CSS mặc định của trình duyệt.

Ví dụ:

Trong một trang web có liên kết tới file style.css có nội dung như sau:

p {

color:#333; text-align:left; width:500px

}

Trong thẻ <style> giữa thẻ <head> cũng có một đoạn CSS liên quan:

p {

background-color:#FF00FF; text-align:right; width:100%;

height:150px

}

Trong phần nội dung trang web đó cũng có sử dụng CSS nội tuyến:

<p style="height:200px; text-align:center; border:1px solid #FF0000; color:#000" } Khi thực thi CSS trình duyệt sẽ đọc hết tất cả các nguồn chứa style rồi sẽ tổng hợp lại vào một CSS ảo và nếu có sự trùng lắp các thuộc tính CSS thì nó sẽ lấy thuộc

Tính CSS có mức ưu tiên cao hơn. Như ví dụ trên chúng ta sẽ thấy CSS cuối cùng mà phần tử p nhận được là:

p {

background-color:#FF00FF;

width:100%; height:200px; text-align:center;

border:1px solid #FF0000;

color:#000

}

Vậy có cách nào để thay đổi độ ưu tiên cho một thuộc tính nào đó? Thật ra thì trong CSS đã có sẵn một thuộc tính giúp chúng ta thực hiện điều này, đó chính là thuộc tính !important. Chỉ cần bạn đặt thuộc tính này sau một thuộc tính nào đó theo cú pháp

selector { property:value !important } thì trình duyệt sẽ hiểu đây là một thuộc tính được ưu tiên. Bây giờ, chúng ta cùng xét lại ví dụ trên nhưng có đặt thêm một số thuộc tính !important vào xem kết quả như thế nào nhé.

p {

}

p {

}

width:500px;

text-align:left !important;

color:#333 !important

background-color:#FF00FF;

width:100%; height:150px !important; text-align:right;

<p style="text-align:center; height:200px; border:1px solid #FF0000; color:#000" }

Phần CSS sẽ tác động lên thuộc tính p là:

p {

background-color:#FF0000;

width:100%; height:150px !important; text-align:left !important;

border:1px solid #FF0000;

color:#333 !important

}

Lưu ý: Cùng một thuộc tính cho một selector thì nếu cả hai thuộc tính đều đặt !important thì cái sau được lấy.

Bài 4 : Ngôn ngữ kịch bản Javascript

Giới thiệu về javascript

Giới thiệu

JavaScript is là 1 ngôn ngữ dạng script thường được sử dụng cho việc lập trình web ở phía client. Nó tuân theo chuẩn ECMAScript.

Nó là 1 ngông ngữ linh động, cú pháp dễ sử dụng như các ngôn ngữ khác và dễ dàng lập trình.

JavaScript không hề liên quan tới ngôn ngữ lập trình java, được hầu hết các trình duyệt ngày nay hỗ trợ.

Với javascript, ứng dụng web của bạn sẽ trở nên vô cùng sinh động, mang tính trực quan và tương tác cao.

JavaScript, theo phiên bản hiện hành, là một ngôn ngữ lập trình kịch bản dựa trên đối tượng được phát triển từ các ý niệm nguyên mẫu. Ngôn ngữ này được dùng rộng rãi cho các trang web, nhưng cũng được dùng để tạo khả năng viết script sử dụng các đối tượng nằm sẵn trong các ứng dụng. Nó vốn được phát triển bởi Brendan Eich tại Hãng truyền thông Netscape với cái tên đầu tiên Mocha, rồi sau đó đổi tên thành LiveScript, và cuối cùng thành JavaScript. Giống Java, JavaScript có cú pháp tương tự C, nhưng nó gần với Self hơn Java. .js là phần mở rộng thường được dùng cho tập tin mã nguồn JavaScript.

Phiên bản mới nhất của JavaScript là phiên bản 1.5, tương ứng với ECMA-262 bản 3. ECMAScript là phiên bản chuẩn hóa của JavaScript. Trình duyệt Mozilla phiên bản 1.8 beta 1 có hỗ trợ không đầy đủ cho E4X - phần mở rộng cho JavaScript hỗ trợ làm việc với XML, được chuẩn hóa trong ECMA-357.

JavaScript là một ngôn ngữ lập trình dựa trên nguyên mẫu với cú pháp phát triển từ C. Giống như C, JavaScript có khái niệm từ khóa, do đó, JavaScript gần như không thể được mở rộng.

Cũng giống như C, JavaScript không có bộ xử lý xuất/nhập (input/output) riêng. Trong khi C sử dụng thư viện xuất/nhập chuẩn, JavaScript dựa vào phần mềm ngôn ngữ được gắn vào để thực hiện xuất/nhập.

Trên trình duyệt, rất nhiều trang web sử dụng JavaScript để thiết kế trang web động và một số hiệu ứng hình ảnh thông qua DOM. JavaScript được dùng để thực hiện một số

tác vụ không thể thực hiện được với chỉ HTML như kiểm tra thông tin nhập vào, tự động thay đổi hình ảnh,... Ở Việt Nam, JavaScript còn được ứng dụng để làm bộ gõ tiếng Việt giống như bộ gõ hiện đang sử dụng trên trang Wikipedia tiếng Việt. Tuy nhiên, mỗi trình duyệt áp dụng JavaScript khác nhau và không tuân theo chuẩn W3C DOM, do đó trong rất nhiều trường hợp lập trình viên phải viết nhiều phiên bản của cùng một đoạn mã nguồn để có thể hoạt động trên nhiều trình duyệt. Một số công nghệ nổi bật dòng JavaScript để tương tác với DOM bao gồm DHTML, Ajax và SPA.

Bên ngoài trình duyệt, JavaScript có thể được sử dụng trong tập tin PDF của Adobe Acrobat và Adobe Reader. Điều khiển Dashboard trên hệ điều hành Mac OS X phiên bản 10.4 cũng có sử dụng JavaScript. Công nghệ kịch bản linh động (active scripting) của Microsoft có hỗ trợ ngôn ngữ JScript làm một ngôn ngữ kịch bản dùng cho hệ điều hành. JScript .NET là một ngôn ngữ tương thích với CLI gần giống JScript nhưng có thêm nhiều tính năng lập trình hướng đối tượng.

Mỗi ứng dụng này đều cung cấp mô hình đối tượng riêng cho phép tương tác với môi trường chủ, với phần lõi là ngôn ngữ lập trình JavaScript gần như giống nhau.

Lịch sử phát triển

Cùng thời điểm Netscape bắt đầu sử dụng công nghệ Java trên trình duyệt Netscape, LiveScript đã được đổi tên thành JavaScript để được chú ý hơn bởi ngôn ngữ lập trình Java lúc đó đang được coi là một hiện tượng. JavaScript được bổ sung vào trình duyệt Netscape bắt đầu từ phiên bản 2.0b3 của trình duyệt này vào tháng 12 năm 1995. Trên thực tế, hai ngôn ngữ lập trình Java và JavaScript không có liên quan gì đến nhau, ngoại trừ việc cú pháp của cả hai ngôn ngữ cùng được phát triển dựa trên cú pháp của C. Java Script gồm 2 mảng là client-server thực hiện lệnh trên máy của end-user và web-server.

Sau thành công của JavaScript, Microsoft bắt đầu phát triển JScript, một ngôn ngữ có cùng ứng dụng và tương thích với JavaScript. JScript được bổ sung vào trình duyệt Internet Explorer bắt đầu từ Internet Explorer phiên bản 3.0 được phát hành tháng 8 năm 1996.

Đặc điểm của ngôn ngữ javascript

Nhúng mã-Cách khai báo biến

Nhúng mã javascript trong trang HTML

Bạn có thể nhúng JavaScript vào một file HTML theo một trong các cách sau đây:

- Sử dụng các câu lệnh và các hàm trong cặp thẻ **<SCRIPT>**
- Sử dụng các file nguồn JavaScript
- Sử dụng một biểu thức JavaScript làm giá trị của một thuộc tính HTML
- Sử dụng thẻ sự kiện (event handlers) trong một thẻ HTML nào đó

Trong đó, sử dụng cặp thẻ <**SCRIPT**>...</**SCRIPT**> và nhúng một file nguồn JavaScript là được sử dụng nhiều hơn cả.

Sử dụng thẻ SCRIPT

Script được đưa vào file HTML bằng cách sử dụng cặp thẻ <**SCRIPT**> và <**SCRIPT**>. Các thẻ <SCRIPT> có thể xuất hiện trong phần <HEAD> hay <BODY> của file HTML. Nếu đặt trong phần <HEAD>, nó sẽ được tải và sẵn sàng trước khi phần còn lại của văn bản được tải.

Chú ý:Ghi chú không được đặt trong cặp thẻ <- và -> như ghi chú trong file HTML. Cú pháp của JavaScript tương tự cú pháp của C nên có thể sử dụng // hay /* ... */.Thuộc tính duy nhất được định nghĩa hiện thời cho thẻ <**SCRIPT**> là "**LANGUAGE**=" dùng để xác định ngôn ngữ script được sử dụng. Có hai giá trị được định nghĩa là "JavaScript" và "VBScript". Với chương trình viết bằng JavaScript bạn sử dụng cú pháp sau :

<**SCRIPT** LANGUAGE="JavaScript">

// INSERT ALL JavaScript HERE

</SCRIPT>

Điểm khác nhau giữa cú pháp viết các ghi chú giữa HTML và JavaScript là cho phép bạn ẩn các mã JavaScript trong các ghi chú của file HTML, để các trình duyệt cũ không hỗ trợ cho JavaScript có thể đọc được nó như trong ví dụ sau đây:

<**SCRIPT** LANGUAGE="JavaScript">

<!-- From here the JavaScript code hidden

// INSERT ALL JavaScript HERE

// This is where the hidden ends -->

</**SCRIPT**>

Dòng cuối cùng của script cần có dấu // để trình duyệt không diễn dịch dòng này dưới dạng mã JavaScript. Các ví dụ trong chương này không chứa đặc điểm ẩn của JavaScript để mã có thể dễ hiểu hơn.

Sử dụng một file nguồn JavaScript

Thuộc tính *SRC* của thẻ *<SCRIPT>* cho phép bạn chỉ rõ file nguồn JavaScript được sử dụng (dùng phương pháp này hay hơn nhúng trực tiếp một đoạn lệnh JavaScript vào trang HTML).

Cú pháp:

<**SCRIPT** SRC="file_name.js">

....

</**SCRIPT**>

Chú ýKhi bạn muốn chỉ ra một xâu trích dẫn trong một xâu khác cần sử dụng dấu nháy đơn (') để phân định xâu đó. Điều này cho phép script nhận ra xâu ký tự đó.Thuộc tính này rấy hữu dụng cho việc chia sẻ các hàm dùng chung cho nhiều trang khác nhau. Các câu lệnh JavaScript nằm trong cặp thẻ *<SCRIPT>* và *</SCRIPT>* có chứa thuộc tinh *SRC* trừ khi nó có lỗi. Ví dụ bạn muốn đưa dòng lệnh sau vào giữa cặp thẻ *<SCRIPT SRC="..."*> và *</SCRIPT>:*

document.write("Không tìm thấy file JS đưa vào!");

Thuộc tính *SRC* có thể được định rõ bằng địa chỉ *URL*, các liên kết hoặc các đường dẫn tuyệt đối, ví dụ:

<SCRIPT SRC=" http://cse.com.vn ">

Các file JavaScript bên ngoài không được chứa bất kỳ thẻ HTML nào. Chúng chỉ được chứa các câu lệnh JavaScript và định nghĩa hàm.

Tên file của các hàm JavaScript bên ngoài cần có đuôi *.js*, và server sẽ phải ánh xạ đuôi *.js* đó tới kiểu MIME application/x-javascript. Đó là những gì mà server gửi trở lại phần

Header của file HTML. Để ánh xạ đuôi này vào kiểu MIME, ta thêm dòng sau vào file mime.types trong đường dẫn cấu hình của server, sau đó khởi động lại server:

type=application/x-javascript

Nếu server không ánh xạ được đuôi *.js* tới kiểu MIME application/x-javascript , Navigator sẽ tải file JavaScript được chỉ ra trong thuộc tính *SRC* về không đúng cách.

Trong ví dụ sau, hàm bar có chứa xâu "left" nằm trong một cặp dấu nháy kép: **function** bar(widthPct){

document.**write**(" <**HR** ALIGN='LEFT' WIDTH="+widthPct+"%>")

}

Bên trong trang.

Để đưa javascript vào trang HTML , chúng ta sử dụng thẻ

Chúng ta hãy xem xét ví dụ hello world sau:

HTML Code:

```
<html>

<body>

<script type="text/javascript">

document.write("Hello World!");

</script>

</body>

</html>
```

Lưu đoạn mã trên vào file dưới dạng HTML, chạy thử và nó sẽ xuất kết quả như sau:

Code:

Hello World!

Trong ví dụ trên, document là đối tượng tài liệu thể hiện cho trang HTML đang được thực thi, phương thức write() cho phép chúng ta xuất dữ liệu ra trang.

Bên ngoài trang

Để việc sử dụng javascript hiệu quả, mã javascript sẽ được đưa vào 1 file dạng js. Từ đó nó có thể sử dụng cho tất cả các trang web mà không cần phải viết lại cho từng trang.

Bạn sử dụng cú pháp như sau với thuộc tính src là đường dẫn tới file js chứa mã javascript.

HTML Code:

```
<html>

<head>

<script src="path/file.js"></script>

</head>

<body>

</body>

</html>
```

Biến và cách khai báo biến

Tên biến trong JavaScript phải bắt đầu bằng chữ hay dấu gạch dưới. Các chữ số không được sử dụng để mở đầu tên một biến nhưng có thể sử dụng sau ký tự đầu tiên.

Phạm vi của biến có thể là một trong hai kiểu sau:

Biến toàn cục: Có thể được truy cập từ bất kỳ đâu trong ứng dụng.được khai báo như sau :

```
x = 0;
```

Biến cục bộ: Chỉ được truy cập trong phạm vi chương trình mà nó khai báo. Biến cục bộ được khai báo trong một hàm với từ khoá *var*như sau:

```
var x = 0;
```

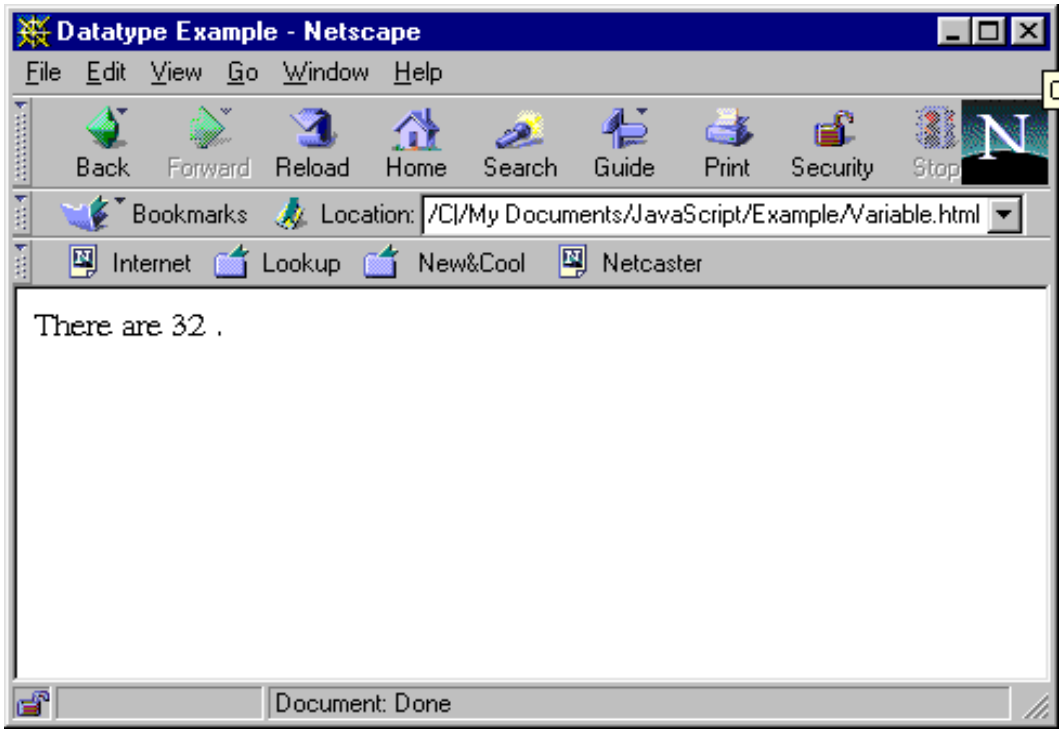
Biến toàn cục có thể sử dụng từ khoá **var**, tuy nhiên điều này không thực sự cần thiết.

Khác với C++ hay Java, JavaScript là ngôn ngữ có tính định kiểu thấp. Điều này có nghĩa là không phải chỉ ra kiểu dữ liệu khi khai báo biến. Kiểu dữ liệu được tự động chuyển thành kiểu phù hợp khi cần thiết.

Ví dụ file Variable.Html:

```
<HTML>

<HEAD>

<TITLE> Datatype Example </TITLE>

<SCRIPT LANGUAGE= "JavaScript">

var fruit='apples';

var numfruit=12;

numfruit = numfruit + 20;

var temp ="There are " + numfruit + " " + ".";

document.write(temp);

</SCRIPT>

</HEAD>

<BODY>

</BODY>

</HTML>
```

Các trình duyệt hỗ trợ JavaScript sẽ xử lý chính xác ví dụ trên và đưa ra kết quả dưới đây:



Hình 3.1: *Kết quả của xử lý dữ liệu*Trình diễn dịch JavaScript sẽ xem biến numfruit có kiểu nguyên khi cộng với 20 và có kiểu chuỗi khi kết hợp với biến temp.

Trong JavaScript, có bốn kiểu dữ liệu sau đây: *kiểu số nguyên, kiểu dấu phẩy động, kiểu logic và kiểu chuỗi.*

KIểu nguyên (Interger)

Số nguyên có thể được biểu diễn theo ba cách

Hệ cơ số 10 (hệ thập phân) - có thể biểu diễn số nguyên theo cơ số 10, chú ý rằng chữ số đầu tiên phải khác 0.

Hệ cơ số 8 (hệ bát phân) - số nguyên có thể biểu diễn dưới dạng bát phân với chữ số đầu tiên là số 0.

Hệ cơ số 16 (hệ thập lục phân) - số nguyên có thể biểu diễn dưới dạng thập lục phân với hai chữ số đầu tiên là 0x.

Kiểu dấu phẩy động (Floating Point)

Một literal có kiểu dấu phẩy động có 4 thành phần sau:

Phần nguyên thập phân.

Dấu chấm thập phân (.).

Phần dư.

Phần mũ.

Để phân biệt kiểu dấu phẩy động với kiểu số nguyên, phải có ít nhất một chữ số theo sau dấu chấm hay **E.** Ví dụ:

9.87

-0.85E4

9.87E14

.98E-3

Kiểu logic (Boolean)

Kiểu logic được sử dụng để chỉ hai điều kiện : đúng hoặc sai. Miền giá trị của kiểu này chỉ có hai giá trị

true.

false.

Kiểu chuỗi (String)

Một literal kiểu chuỗi được biểu diễn bởi không hay nhiều ký tự được đặt trong cặp dấu " ... " hay '... '. Ví dụ:

“The dog ran up the tree”

‘The dog barked’

“100”

Để biểu diễn dấu nháy kép ("), trong chuỗi sử dụng (\"), ví dụ:

document.write(“ \”This text inside quotes.\” ”);

Cú pháp khai báo biến

var test ;

test = “hi”;

var age = 22;

Với javascript, kiểu dữ liệu chúng ta không cần phải khai báo tường minh, trình biên dịch sẽ tự động hiểu biến test là kiểu chuỗi, còn age là kiểu số nguyên.

2. Các kiểu dữ liệu

Trong javascript có 2 kiểu dữ liệu : primitive type và reference type

Primitive type là những kiểu dữ liệu cơ bản của javascript như : boolean, number, string , null.

Reference type là những lớp được hỗ trợ trong javascript như : Number, String, Date , Array, Object, Function ..

Điều khiển rẽ nhánh trong javascript

Rẽ nhánh theo điều kiện với if ... else

Cú pháp if ... else dùng trong trường hợp muốn rẽ nhánh theo điều kiện. Cú pháp này tương đương với nếu x thì làm y, còn nếu không thì làm z. Các câu lệnh if ... else có thể lồng trong nhau.

Cú pháp:

```
<script language="JavaScript">

if (biểu_thức_1)

{

  khối lệnh được thực hiện nếu biểu thức 1 đúng;

}

else if (biểu_thức_2)

{

khối lệnh được thực hiện nếu biểu thức 2 đúng;

}

else
```

{

khối lệnh được thực hiện nếu cả hai biểu thức trên đều không đúng;

}

</script>

Ví dụ:

<script language="JavaScript"> var x = prompt("Nhập vào giá trị của x:"); x = parseFloat(x); if (!isNaN(x)) { if (x > 0) {

alert("x > 0"); }

else if (x == 0) {

alert("x = 0"); } else

{ alert("x < 0"); } } else { alert("giá trị bạn nhập không phải là một số"); }</script>

Đoạn mã nguồn trên mở một hộp thoại yêu cầu nhập vào một giá trị số, sau đó hiển thị thông báo số đó lớn hơn 0, bằng 0 hay nhỏ hơn 0.

Toán tử điều kiện

Toán từ điều kiện còn được biết đến với tên gọi toán tử tam phân. Cú pháp của toán tử này như sau:

<script language="JavaScript">

điều_kiện ? biểu_thức_đúng : biểu_thức_sai;

</script>

Toán tử này sẽ trả lại giá trị là kết quả của biểu_thức_đúng nếu điều_kiện có giá trị bool bằng true, ngược lại nó sẽ trả lại giá trị bằng biểu_thức_sai.

Cú pháp switch

Cú pháp switch cũng là cú pháp điều kiện như if ... else hay toán tử tam phân. Tuy nhiên, cú pháp switch thường được dùng khi chỉ cần so sánh bằng với số lượng kết quả cần kiểm tra lớn. Cách sử dụng cú pháp switch:

Code

<script language="JavaScript"> switch (biểu_thức_điều_kiện) { case kết_quả_1 : khối lệnh cần thực hiện néu biểu_thức_điều_kiện bằng kết_quả_1;

break; case kết_quả_2 : khối lệnh cần thực hiện néu biểu_thức_điều_kiện bằng kết_quả_2; break; default : khối lệnh cần thực hiện nếu biểu_thức_điều_kiện cho ra một kết quả khác; }</script>

Sau mỗi khối lệnh trong một mục kiểm tra kết quả (trừ mục default), lập trình viên cần phải thêm vào break.

Các điều khiển

Các điều khiển lặp trong javascript

Vòng lặp while

Vòng lặp while có mục đích lặp đi lặp lại một khối lệnh nhất định cho đến khi biểu thức điều kiện trả về false. Khi dùng vòng lặp while phải chú ý tạo lối thoát cho vòng lặp (làm cho biểu thức điều kiện có giá trị false), nếu không đoạn mã nguồn sẽ rơi vào vòng lặp vô hạn, là một lỗi lập trình. Vòng lặp while thường được dùng khi lập trình viên không biết chính xác cần lặp bao nhiêu lần. Cú pháp của vòng lặp while như sau:

Code

<script language="JavaScript"> while (biểu_thức_điều_kiện) { khối lệnh cần thực hiện nếu biểu_thức_điều_kiện trả về true; }</script>

Vòng lặp do ... while

Về cơ bản, vòng lặp do ... while gần như giống hệt như vòng lặp while. Tuy nhiên, trong trường hợp biểu thức điều kiện trả về false ngay từ đầu, khối lệnh trong vòng lặp while sẽ không bao giờ được thực hiện, trong khi đó, vòng lặp do ... while luôn đảm bảo khối lệnh trong vòng lặp được thực hiện ít nhất một lần. Ví dụ:

<script language="JavaScript"> while (0 > 1) { alert("while"); // Câu lệnh này sẽ không bao giờ được thực hiện } do { alert("do ... while"); // Bạn sẽ nhận được thông báo do ... while một lần duy nhất } while (0 > 1);</script>

Cú pháp của vòng lặp do ... while như sau:

Code:

<script language="JavaScript"> do

  { khối lệnh;

  } while (biểu_thức_điều_kiện);</script>

Vòng lặp for

Vòng lặp for thường được sử dụng khi cần lặp một khối lệnh mà lập trình viên biết trước sẽ cần lặp bao nhiêu lần. Cú pháp của vòng lặp for như sau:

<script language="JavaScript"> for (biểu_thức_khởi_tạo; biểu_thức_điều_kiện; biểu_thức_thay_đổi_giá_trị) { Khối lệnh cần lặp; }</script>

Khi bắt đầu vòng lặp for, lập trình viên cần khởi tạo một biến nhất định bằng biểu_thức_khởi_tạo để dùng trong biểu_thức_điều_kiện, nếu biểu_thức_điều_kiện trả về true, khối lệnh cần lặp sẽ được thực hiện, sau khi thực hiện xong khối lệnh cần lặp, biểu_thức_thay_đổi_giá_trị sẽ được thực hiện, tiếp theo, biểu_thức_điều_kiện sẽ lại được kiểm tra, cứ như vậy cho đến khi biểu_thức_điều_kiện trả về false, khi đó vòng lặp sẽ kết thúc.

Vòng lặp for ... in

Vòng lặp for ... in dùng để lặp qua tất cả các thuộc tính của một đối tượng (hay lặp qua tất cả các phần tử của một mảng). Cú pháp của vòng lặp này như sau:

<script language="JavaScript"> for (biến in đối_tượng) { khối lệnh cần thực hiện, có thể sử dụng đối_tượng[biến] để truy cập từng thuộc tính (phần tử) của đối tượng; }</script>

Đối tượng navigator- window- location-frame

Đối tượng navigator

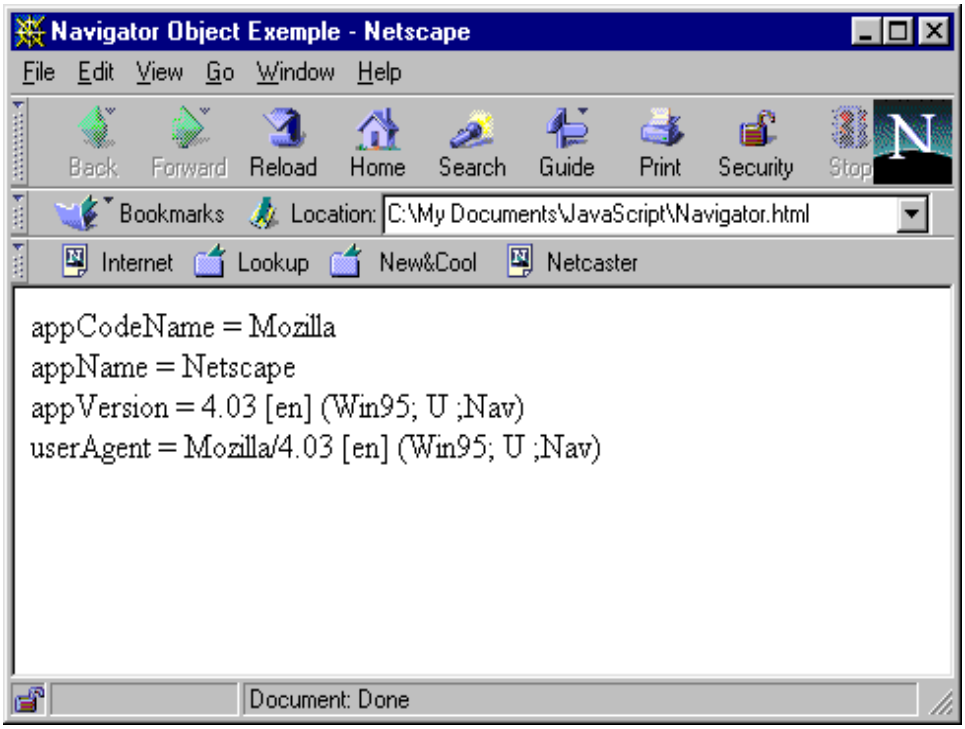
Đối tượng này được sử dụng để đạt được các thông tin về trình duyệt như số phiên bản. Đối tượng này không có phương thức hay chương trình xử lý sự kiện.

Các thuộc tính

appCodeName	Xác định tên mã nội tại của trình duyệt (Atlas).
AppName	Xác định tên trình duyệt.
AppVersion	Xác định thông tin về phiên bản của đối tượng navigator.
userAgent	Xác định header của user - agent.

Ví dụ sau sẽ hiển thị các thuộc tính của đối tượng navigator

```
<HTML>

<HEAD>

<TITLE> Navigator Object Exemple </TITLE>

<SCRIPT LANGUAGE= "JavaScript">

document.write("appCodeName = "+navigator.appCodeName + "<BR>");

document.write("appName = "+navigator.appName + "<BR>");

document.write("appVersion = "+navigator.appVersion + "<BR>");

document.write("userAgent = "+navigator.userAgent + "<BR>");

</SCRIPT>

</HEAD>

<BODY>

</BODY>

</HTML>
```



Đối tượng window

Đối tượng window như đã nói ở trên là đối tượng ở mức cao nhất. Các đối tượng document, frame, vị trí đều là thuộc tính của đối tượng window.

Các thuộc tính

- defaultStatus - Thông báo ngầm định hiển thị lên trên thanh trạng thái của cửa sổ
- Frames - Mảng xác định tất cả các frame trong cửa sổ.
- Length - Số lượng các frame trong cửa sổ cha mẹ.
- Name - Tên của cửa sổ hiện thời.
- Parent - Đối tượng cửa sổ cha mẹ
- Self - Cửa sổ hiện thời.
- Status - Được sử dụng cho thông báo tạm thời hiển thị lên trên thanh thạng thái cửa sổ. Được sử dụng để lấy hay đặt lại thông báo trạng thái và ghi đè lên defaultStatus.
- Top - Cửa sổ ở trên cùng.
- Window - Cửa sổ hiện thời.

Các phương thức

- alert ("message") -Hiển thị hộp hội thoại với chuỗi "message" và nút OK.
- clearTimeout(timeoutID) -Xóa timeout do SetTimeout đặt. SetTimeout trả lại timeoutID

- windowReference.close -Đóng cửa sổ windowReference.
- confirm("message") -Hiển thị hộp hội thoại với chuỗi "message", nút OK và nút Cancel. Trả lại giá trị True cho OK và False cho Cancel.
- [windowVar =][window]. open("URL", "windowName", ["windowFeatures"]) - Mở cửa sổ mới.
- prompt ("message" [,"defaultInput"]) - Mở một hộp hội thoại để nhận dữ liệu vào trường text.
- TimeoutID = setTimeout(expression,msec) - Đánh giá biểu thức expresion sau thời gian msec.

Ví dụ: Sử dụng tên cửa sổ khi gọi tới nó như là đích của một form submit hoặc trong một Hipertext link (thuộc tính TARGET của thẻ FORM và A).

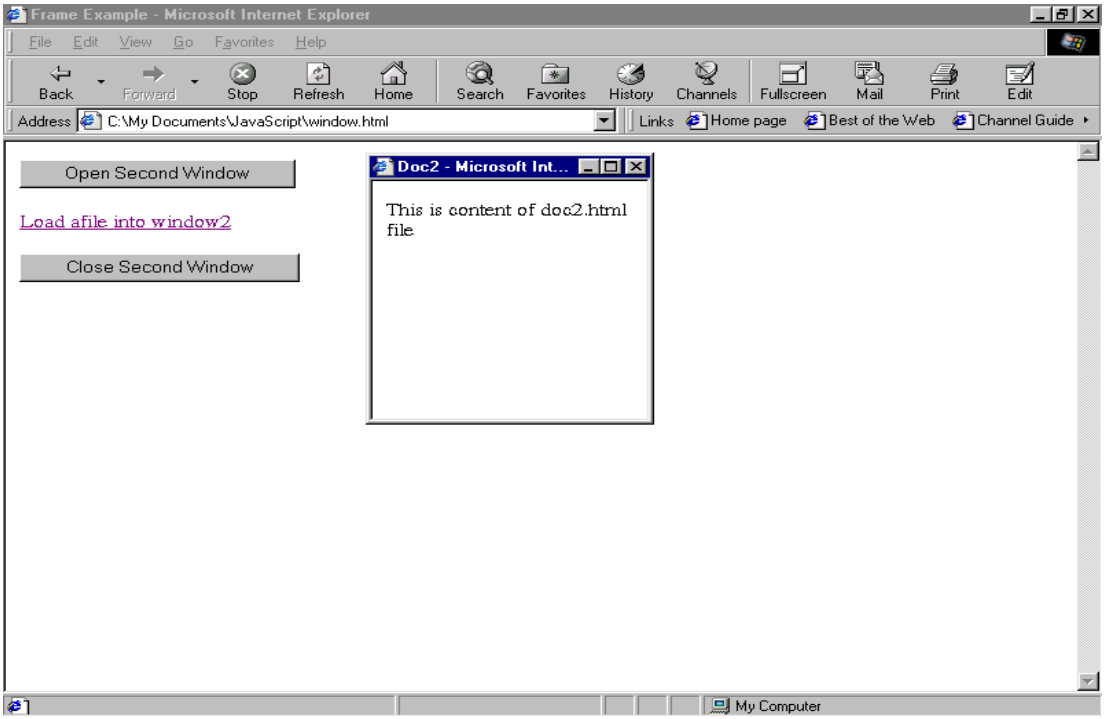
Trong ví dụ tạo ra một tới cửa sổ thứ hai, như nút thứ nhất để mở một cửa sổ rỗng, sau đó một liên kết sẽ tải file doc2.html xuống cửa sổ mới đó rồi một nút khác dùng để đóng của sổ thứ hai lại, ví dụ này lưa vào file *window.html*:

```
<HTML>

<HEAD>

<TITLE>Frame Example </TITLE>

</HEAD>

<BODY>

<FORM>

<INPUT TYPE="button" VALUE="Open Second Window" onClick="msgWindow= window.open(','window2','resizable=no,width=200,height=200')">

<P>

<A HREF="doc2.html" TARGET="window2">

Load a file into window2 </A>

</P>

<INPUT TYPE="button" VALUE="Close Second Window"

onClick="msgWindow.close()">
```

</FORM>

</BODY>

</HTML>



Hình 6.3: Minh hoạ cho đối tượng cửa sổ

Các chương trình xử lý sự kiện

- onLoad - Xuất hiện khi cửa sổ kết thúc việc tải.
- onUnLoad - Xuất hiện khi cửa sổ được loại bỏ.

Đối tượng location

Các thuộc tính của đối tượng location duy trì các thông tin về URL của document hiện thời. Đối tượng này hoàn toàn không có các phương thức và chương trình xử lý sự kiện đi kèm. *Ví dụ*:

http:// www.abc.com/ chap1/page2.html#topic3

Các thuộc tính

- hash - Tên anchor của vị trí hiện thời (ví dụ topic3).
- Host - Phần hostname:port của URL (ví dụ www.abc.com). Chú ý rằng đây thường là cổng ngầm định và ít khi được chỉ ra.

- Hostname - Tên của host và domain (ví dụ www.abc.com).
- href - Toàn bộ URL cho document hiện tại.
- Pathname - Phần đường dẫn của URL (ví dụ /chap1/page2.html).
- Port - Cổng truyền thông được sử dụng cho máy tính host, thường là cổng ngầm định.
- Protocol - Giao thức được sử dụng (cùng với dấu hai chấm) (ví dụ http:).
- Search - Câu truy vấn tìm kiếm có thể ở cuối URL cho các script CGI.

Đối tượng frame

Một cửa sổ có thể có một vài frame. Các frame có thể cuộn một cách độc lập với nhau và mỗi frame có URL riêng. frame không có các chương trình xử lý sự kiện. Sự kiện onLoad và onUnLoad là của đối tượng window.

Các thuộc tính

- frames - Mảng tất cả các frame trong cửa sổ.
- Name - Thuộc tính NAME của thẻ <FRAME>
- Length - Số lượng các frame con trong một frame.
- Parent - Cửa sổ hay frame chứa nhóm frame hiện thời.
- self - frame hiện thời.
- Window - frame hiện thời.

Các phương thức

- clearTimeout (timeoutID) - Xoá timeout do setTimeout lập. SetTimeout trả lại timeoutID.
- TimeoutID = setTimeout (expression,msec) - Đánh giá expression sau khi hết thời gian msec.

Sử dụng Frame

Tạo một frame (create)

Để tạo một frame, ta sử dụng thẻ **FRAMESET**. Mục đích của thẻ này là định nghĩa một tập các frame trong một trang.

Ví dụ1: tạo frame

<HTML>

<HEAD>

<TITLE>Frame Example </TITLE>

```
<FRAMESET ROWS="90%,10%">

<FRAMESET COLS="30%,70%">

<FRAME SRC=CATEGORY.HTM NAME="ListFrame">

<FRAME SRC=TITLES.HTM NAME="contentFrame">

</FRAMESET >

<FRAME SRC=NAVIGATOR.HTM NAME="navigateFrame">

</FRAMESET >

</HEAD>

<BODY> </BODY>

</HTML>
```

Sơ đồ sau hiển thị cấu trúc của các frame: Cả 3 frame đều trên cùng một cửa sổ cha, mặc dù 2 trong số các frame đó nằm trong một frameset khác.

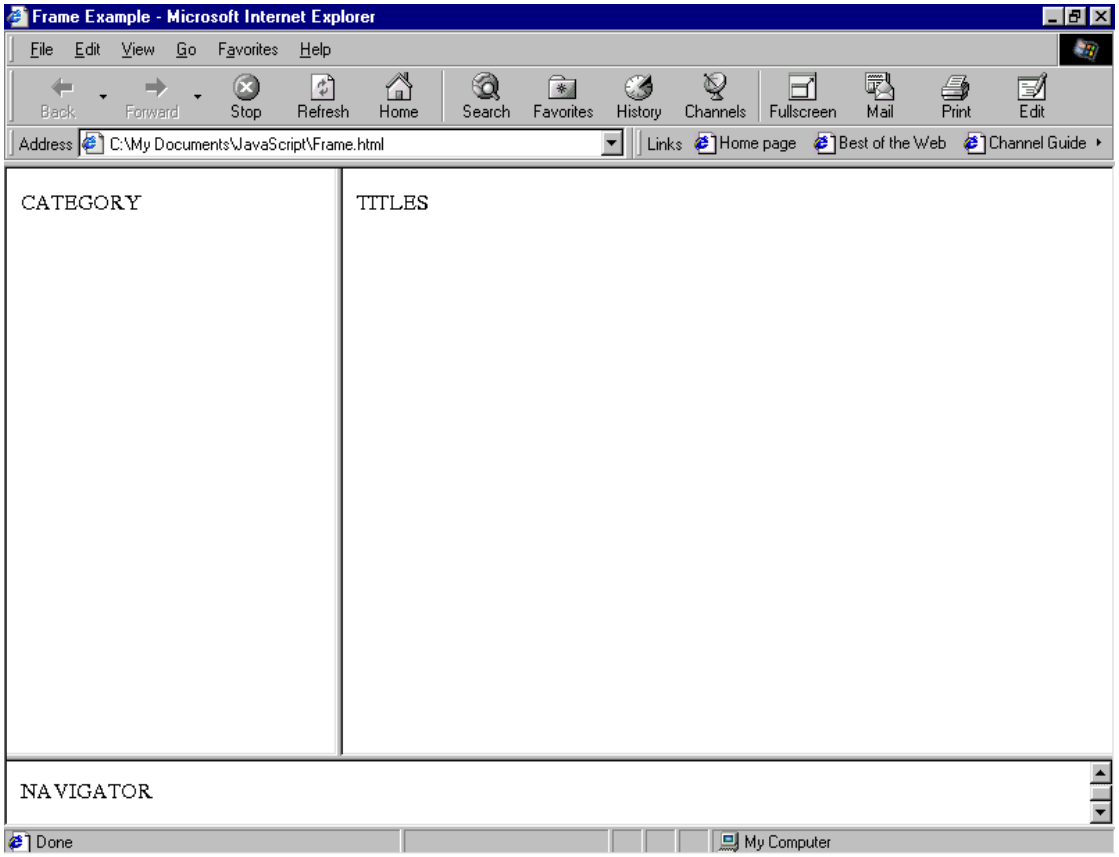
ToplistFrame (category.html)contentFrame (titles.html)navigatorFrame (navigator.html)

Bạn có thể gọi tới những frame trước đó bằng cách sử dụng thuộc tính **frames** như sau:

listFrame chính là top.frames[0]

contentFrame chính là top.frames[1]

navigatorFrame chính là top.frames[2]



Hình 6.4: Kết quả việc tạo frame trong

Ví dụ 2: Cũng giống như một sự lựa chọn, bạn có thể tạo ra một cửa sổ giống như ví dụ trước nhưng trong mỗi đỉnh của hai frame lại có một cửa sổ cha riêng từ **navigateFrame**. Mức frameset cao nhất có thể được định nghĩa như sau:

```
<HTML>

<HEAD>

<TITLE>Frame Example </TITLE>

<FRAMESET ROWS="90%,10%">

<FRAME SRC=muske13.HTML NAME="upperFrame">

<FRAME SRC=NAVIGATOR.HTM NAME="navigateFrame">

</FRAMESET >

</HEAD>

<BODY>
```

```
</BODY>

</HTML>
```

Trong file muske13.html lại tiếp tục đặt một frameset:

```
<HTML>

<HEAD>

<TITLE>Frame Example </TITLE>

<FRAMESET COLS="30%,70%">

<FRAME SRC=CATEGORY.HTM NAME="ListFrame">

<FRAME SRC=TITLES.HTM NAME="contentFrame">

</FRAMESET >

</HEAD>

<BODY>

</BODY>

</HTML>
```

Khi đó kết quả hiển thị của ví dụ 2 giống ví dụ 1 nhưng sự phân cấp của các frames lại khác hẳn:

topupperFrame (muske13.html)navigatorFrame (navigator.html)listFrame (category.html)contentFrame (titles.html)Bạn có thể gọi tới các frame trên bằng cách sử dụng thuộc tính mảng **frames** như sau:

upperFrame chính là **top.frames[0]**

navigatorFrame chính là **top.frames[1]**

listFrame chính là **upperFrame.frames[0]**

hoặc **top.frames[0].frames[0]**

contentFrame chính là **upperFrame.frames[1]**

hoặc **top.frames[0].frames[1]**

Cập nhật một frame (update)

Bạn có thể cập nhật nội dung của một frame bằng cách sử dụng thuộc tính **location** để đặt địa chỉ URL và phải định chỉ rõ vị trí của frame trong cấu trúc.

Trong ví dụ trên, nếu bạn thêm một dòng sau vào **navigatorFrame**:

<INPUT TYPE="button" VALUE="Titles only" onClick="top.frames[0].location='artist.html'">

Thì khi nút "**Titles only**" được nhấn, file **artist.html** sẽ được tải vào **upperFrame**, và hai frame **listFrame, contentFrame** sẽ bị đóng lại như chúng chưa bao giờ tồn tại.

Đối tượng Document- Anchors- Forms- History

Đối tượng document

Đối tượng này chứa các thông tin về document hiện thời và cung cấp các phương thức để đưa thông tin ra màn hình. Đối tượng document được tạo ra bằng cặp thẻ <BODY> và </BODY>. Một số các thuộc tính gắn với thẻ <BODY>.

Các đối tượng anchor, forms, history, links là thuộc tính của đối tượng document. Không có các chương trình xử lý sự kiện cho các frame. Sự kiện onLoad và onUnLoad là cho đối tượng window.

Các thuộc tính

- alinkColor - Giống như thuộc tính ALINK.
- anchor - Mảng tất cả các anchor trong document.
- bgColor - Giống thuộc tính BGCOLOR.
- cookie - Sử dụng để xác định cookie.
- fgColor - Giống thuộc tính TEXT.
- forms - Mảng tất cả các form trong document.
- lastModified - Ngày cuối cùng văn bản được sửa.
- linkColor - Giống thuộc tính LINK.
- links - Mảng tất cả các link trong document.
- location - URL đầy đủ của văn bản.
- referrer - URL của văn bản gọi nó.
- title - Nội dung của thẻ <TITLE>.
- vlinkColor - Giống thuộc tính VLINK.

Các phương thức

- document.clear - Xoá document hiện thời.
- document.close - Đóng dòng dữ liệu vào và đưa toàn bộ dữ liệu trong bộ đệm ra màn hình.
- document.open (["mineType"]) - Mở một stream để thu thập dữ liệu vào của các phwong thức write và writeln.
- document.write(expression1 [,expression2]...[,expressionN]) - Viết biểu thức HTML lên văn bản trông một cửa sổ xác định.
- document.writeln (expression1 [,expression2] ... [,expressionN]) - Giống phương thức trên nhưng khi hết mỗi biểu thức lại xuống dòng.

Đối tượng anchors

anchor là một đoạn văn bản trong document có thể dùng làm đích cho một siêu liên kết. Nó được xác định bằng cặp thẻ <A> và </A>. Đối tượng anchor không có thuộc tính, phương thức cũng như chương trình xử lý sự kiện. Mảng anchor tham chiếu đến mỗi anchor có tên trong document. Mỗi anchor được tham chiếu bằng cách:

document.anchors [index]

Mảng anchor có một thuộc tính duy nhất là length xác định số lượng các anchor trong document, nó có thể được xác định như sau:

document.anchors.length.

Đối tượng forms

Các form được tạo ra nhờ cặp thẻ <FORM> và </FORM>. Phần lớn các thuộc tính của đối tượng form phản ánh các thuộc tính của thẻ <FORM>. Có một vài phần tử (elements) là thuộc tính của đối tượng forms.

Nếu document chứa một vài form, chúng có thể được tham chiếu qua mảng forms. Số lượng các form có thể được xác định như sau:

document.forms.length.

Mỗi một form có thể được tham chiếu như sau:

document.forms[index]

Các thuộc tính

action thuộc tính ACTION của thẻ FORM.

elements Mảng chứa tất cả các thành phần trong một form (như checkbox, trường text, danh sách lựa chọn

encoding Xâu chứa kiểu MIME được sử dụng để mã hoá nội dung của form gửi cho server.

length Số lượng các thành phần trong một form.

method Thuộc tính METHOD.

target Xâu chứa tên của cửa sổ đích khi submit form

Các phương thức

formName.submit () - Xuất dữ liệu của một form tên formName tới trang xử lý. Phương thức này mô phỏng một click vào nút submit trên form.

Các chương trình xử lý sự kiện

onSubmit - Chương trình xử lý sự kiện này được gọi khi người sử dụng chuyển dữ liệu từ form đi.

Đối tượng history

Đối tượng này được sử dụng để lưu giữ các thông tin về các URL trước được người sử dụng sử dụng. Danh sách các URL được lưu trữ theo thứ tự thời gian. Đối tượng này không có chương trình xử lý sự kiện.

Các thuộc tính

length - Số lượng các URL trong đối tượng.

Các phương thức

- history.back() - Được sử dụng để tham chiếu tới URL mới được thăm trước đây.
- history.forward() - Được sử dụng để tham chiếu tới URL kế tiếp trong danh sách. Nó sẽ không gây hiệu ứng gì nếu đã đến cuối của danh sách.
- history.go (delta | "location") - Được sử dụng để chuyển lên hay chuyển xuống delta bậc hay di chuển đến URL xác định bởi location trong danh sách. Nếu delta được sử dụng thì việc dịch chuyển lên phía trên khi delta dương và xuống phía dưới khi delta âm. nếu sử dụng location, URL gần nhất có chứa location là chuỗi con sẽ được tham chiếu.

Đối tượng links

Đối tượng link là một đoạn văn bản hay một ảnh được xem là một siêu liên kết. Các thuộc tính của đối tượng link chủ yếu xử lý về URL của các siêu liên kết. Đối tượng link cũng không có phương thức nào.

Mảng link chứa danh sách tất cả các liên kết trong document. Có thể xác định số lượng các link qua

document.links.length()

Có thể tham chiếu tới một liên kết qủa

document.links [index]

Để xác định các thuộc tính của đối tượng link, có thể sử dụng URL tương tự:

http://www.abc.com/chap1/page2.html#topic3

Các thuộc tính

- hash - Tên anchor của vị trí hiện thời (ví dụ topic3).
- Host - Phần hostname:port của URL (ví dụ www.abc.com). Chú ý rằng đây thường là cổng ngầm định và ít khi được chỉ ra.
- Hostname - Tên của host và domain (ví dụ ww.abc.com).
- href - Toàn bộ URL cho document hiện tại.
- Pathname - Phần đường dẫn của URL (ví dụ /chap1/page2.html).
- port - Cổng truyền thông được sử dụng cho máy tính host, thường là cổng ngầm định.
- Protocol - Giao thức được sử dụng (cùng với dấu hai chấm) (ví dụ http:).
- Search - Câu truy vấn tìm kiếm có thể ở cuối URL cho các script CGI.
- Target - Giống thuộc tính TARGET của <LINK>.

Các chương trình xử lý sự kiện

- onClick - Xảy ra khi người sử dụng nhấn vào link.
- onMouseOver - Xảy ra khi con chuột di chuyển qua link.

Đối tượng Math

Đối tượng Math là đối tượng nội tại trong JavaScript. Các thuộc tính của đối tượng này chứa nhiều hằng số toán học, các hàm toán học, lượng giác phổ biến. Đối tượng Math không có chương trình xử lý sự kiện.

Việc tham chiếu tới *number* trong các phương thức có thể là số hay các biểu thức được đnáh giá là số hợp lệ.

Các thuộc tính

- E - Hằng số Euler, khoảng 2,718.
- LN2 - logarit tự nhiên của 2, khoảng 0,693.
- LN10 - logarit tự nhiên của 10, khoảng 2,302.
- LOG2E - logarit cơ số 2 của e, khoảng 1,442.
- PI - Giá trị của π, khoảng 3,14159.
- SQRT1_2 - Căn bậc 2 của 0,5, khoảng 0,707.

- SQRT2 - Căn bậc 2 của 2, khoảng 1,414.

Các phương thức

- Math.abs (number) - Trả lại giá trị tuyệt đối của *number.*
- Math.acos (number) - Trả lại giá trị arc cosine (theo radian) của *number.* Giá trị của *number* phải năng giữa -1 và 1.
- Math.asin (number) - Trả lại giá trị arc sine (theo radian) của *number.* Giá trị của *number* phải năng giữa -1 và 1.
- Math.atan (number) - Trả lại giá trị arc tan (theo radian) của *number.*
- Math.ceil (number) - Trả lại số nguyên nhỏ nhất lớn hơn hoặc bằng *number.*
- Math.cos (number) - Trả lại giá trị cosine của *number.*
- Math.exp (number) - Trả lại giá trị e^ *number*, với e là hằng số Euler.
- Math.floor (number) - Trả lại số nguyên lớn nhất nhỏ hơn hoặc bằng *number.*
- Math.log (number) - Trả lại logarit tự nhiên của *number.*
- Math.max (num1,num2) - Trả lại giá trị lớn nhất giữa num1 và num2
- Math.min (num1,num2) - Trả lại giá trị nhỏ nhất giữa num1 và num2.
- Math.pos (base,exponent) - Trả lại giá trị base luỹ thừa exponent.
- Math.random (r) - Trả lại một số ngẫu nhiên giữa 0 và 1. Phwong thức này chỉ thực hiện được trên nền tảng UNIX.
- Math.round (number) - Trả lại giá trị của *number* làm tròn tới số nguyên gần nhất.
- Math.sin (number) - Trả lại sin của *number.*
- Math.sqrt (number) - Trả lại căn bậc 2 của *number.*
- Math.tan (number) - Trả lại tag của *number.*

Đối tượng Date

Đối tượng Date là đối tượng có sẵn trong JavaScript. Nó cung cấp nhiều phương thức có ích để xử lý về thời gian và ngày tháng. Đối tượng Date không có thuộc tính và chương trình xử lý sự kiện.

Phần lớn các phương thức date đều có một đối tượng Date đi cùng. Các phương thức giới thiệu trong phần này sử dụng đối tượng Date dateVar, ví dụ:

dateVar = new Date ('August 16, 1996 20:45:04');

Các phương thức

- dateVar.getDate() - Trả lại ngày trong tháng (1-31) cho *dateVar*.
- dateVar.getDay() - Trả lại ngày trong tuần (0=chủ nhật,...6=thứ bảy) cho *dateVar*.
- dateVar.getHours() - Trả lại giờ (0-23) cho *dateVar*.

- dateVar.getMinutes() - Trả lại phút (0-59) cho *dateVar*.
- dateVar.getSeconds() - Trả lại giây (0-59) cho *dateVar*.
- dateVar.getTime() - Trả lại số lượng các mili giây từ 00:00:00 ngày 1/1/1970.
- dateVar.getTimeZoneOffset() - Trả lại độ dịch chuyểnbằng phút của giờ địa phương hiện tại so với giờ quốc tế GMT.
- dateVar.getYear() -Trả lại năm cho dateVar.
- Date.parse (dateStr) - Phân tích chuỗi *dateStr* và trả lại số lượng các mili giây tính từ 00:00:00 ngày 01/01/1970.
- dateVar.setDay(day) - Đặt ngày trong tháng là *day* cho *dateVar*.
- dateVar.setHours(hours) - Đặt giờ là *hours* cho *dateVar*.
- dateVar.setMinutes(minutes) - Đặt phút là *minutes* cho *dateVar*.
- dateVar.setMonths(months) - Đặt tháng là *months* cho *dateVar*.
- dateVar.setSeconds(seconds) - Đặt giây là s*econds* cho *dateVar*.
- dateVar.setTime(value) - Đặt thời gian là value, trong đó value biểu diễn số lượng mili giây từ 00:00:00 ngày 01/01/10970.
- dateVar.setYear(years) - Đặt năm là *years* cho *dateVar*.
- dateVar.toGMTString() - Trả lại chuỗi biểu diễn dateVar dưới dạng GMT.
- dateVar.toLocaleString()-Trả lại chuỗi biểu diễn dateVar theo khu vực thời gian hiện thời.
- Date.UTC (year, month, day [,hours] [,minutes] [,seconds]) - Trả lại số lượng mili giây từ 00:00:00 01/01/1970 GMT.

Đối tượng String

Đối tượng String là đối tượng được xây dựng nội tại trong JavaScript cung cấp nhiều phương thức thao tác trên chuỗi. Đối tượng này có thuộc tính duy nhất là độ dài (length) và không có chương trình xử lý sự kiện.

Các phương thức

Hàm	Nội dung
str.anchor (name)	Được sử dụng để tạo ra thẻ <A> (một cách động). Tham số name là thuộc tính NAME của thẻ <A>.
str.big()	Kết quả giống như thẻ <BIG> trên chuỗi *str*.
str.blink()	Kết quả giống như thẻ <BLINK> trên chuỗi *str*.
str.bold()	Kết quả giống như thẻ <BOLD> trên chuỗi *str*.
str.charAt(a)	Trả lại ký tự thứ a trong chuỗi *str*.
str.fixed()	Kết quả giống như thẻ <TT> trên chuỗi *str*.
str.fontcolor()	Kết quả giống như thẻ <FONTCOLOR = *color*>.

str.fontsize(size)	Kết quả giống như thẻ <FONTSIZE = *size*>.
str.index0f(srchStr [,index])	Trả lại vị trí trong chuỗi *str* vị trí xuất hiện đầu tiên của chuỗi *srchStr*. Chuỗi *str* được tìm từ trái sang phải. Tham số *index* có thể được sử dụng để xác định vị trí bắt đầu tìm kiếm trong chuỗi.
str.italics()	Kết quả giống như thẻ <I> trên chuỗi *str*.
str.lastIndex0f(srchStr [,index])	Trả lại vị trí trong chuỗi *str* vị trí xuất hiện cuối cùng của chuỗi *srchStr*. Chuỗi *str* được tìm từ phải sang trái. Tham số *index* có thể được sử dụng để xác định vị trí bắt đầu tìm kiếm trong chuỗi.
str.link(href)	Được sử dụng để tạo ra một kết nối HTML động cho chhuỗi *str*. Tham số *href* là URL đích của liên kết.
str.small()	Kết quả giống như thẻ <SMALL> trên chuỗi *str*.
str.strike()	Kết quả giống như thẻ <STRIKE> trên chuỗi *str*
str.sub()	Tạo ra một subscript cho chuỗi *str*, giống thẻ <SUB>.
str.substring(a,b)	Trả lại chuỗi con của *str* là các ký tự từ vị trí thứ a tới vị trí thứ b. Các ký tự được đếm từ trái sang phải bắt đầu từ 0.
str.sup()	Tạo ra superscript cho chuỗi *str*, giống thẻ <SUP>.
str.toLowerCase()	Đổi chuỗi *str* thành chữ thường.
str.toUpperCase()	Đổi chuỗi *str* thành chữ hoa.

Xây dựng các hàm và sự kiện trong javascript

Hàm (function) trong javascript

Khái niệm : Hàm là 1 tập các câu lệnh có thể được chạy bất cứ khi nào, ở đâu.Hàm được khai báo với từ khóa function , theo sau là 1 tập các đối số, cuối cùng là đoạn mã lệnh sẽ được đặt trong cặp dấu ngoặc. Cú pháp cơ bản :

```
function functionName(arg0, arg1,...,argN)

{

Statements

}
```

Ví dụ :

```
function sayHi(sName, sMessage)

{

alert("Hello " + name + "," + sMessage);

}

sayHi("Nicholas", "how are you today?");
```

Kết quả sẽ như sau:

Lưu ý: trong javascript không có khái niệm overloading

Ví dụ :

```
function doAdd(iNum)
```

```
{

alert(iNum + 100);

}

function doAdd(iNum)

{

alert(iNum + 10);

}

doAdd(10); // Nó sẽ gọi hàm số 2, kết quả 20
```

Đối tượng arguments

Javascript hỗ trợ đối tượgn arguments cho phép bạn có thể sử dụng các đối mà không cần phải biết tên.

Nó là 1 array của các đối số, bắt đầu từ chỉ số 0.

```
function sayHi()

{if (arguments[0] == “bye”)

{return;}alert(arguments[0]);}
```

Bạn có thể sử dụng thuộc tính length để biết số lượng đối số của hàm.

```
function howManyArgs()

{

alert(arguments.length);

}
```

Đối tượng Function

Trong javascript hàm được coi như là 1 biến đối tượng

var sayHi = new Function(“sName”, “sMessage”, “alert(\”Hello \” + sName + \”, \” + sMessage + \”);”);

Tương đương với :

function sayHi(sName, sMessage) {

alert(“Hello “ + sName + “,” + sMessage);

}

Trong 1 hàm, bạn có thể sử dụng các biến được khai báo bên ngoài thân hàm.

var sMessage = “Hello World!”;

function sayHelloWorld() {

alert(sMessage);

}

sayHelloWorld();

Nguồn : Wrox - Professional JavaScript For Web Developers

Hàm là một khối các câu lệnh với một danh sách một hoặc nhiều đối số (có thể không có đối số) và thường có tên (mặc dù trong JavaScript hàm không nhất thiết phải có tên). Hàm có thể trả lại một giá trị. Cú pháp của hàm như sau:

Code:

<script language="JavaScript">

function tên_hàm(đối_số_1, đối_số_2)

{ các câu lệnh cần thực hiện mỗi khi hàm được gọi;

return giá_trị_cần_trả_về;

}tên_hàm(1, 2);

// Gọi hàm tên_hàm với hai đối số 1 và 2 ứng với đối_số_1 và đối_số_2tên_hàm(1);

// Gọi hàm tên_hàm với đối_số_1 có giá trị 1, đối_số_2 có giá trị undefined</script>

Trong JavaScript, khi gọi hàm không nhất thiết phải gọi hàm với cùng số đối số như khi định nghĩa hàm, nếu số đối số ít hơn khi định nghĩa hàm, những đối số không được chuyển cho hàm sẽ mang giá trị undefined.

Các kiểu cơ bản sẽ được chuyển vào hàm theo giá trị, đối tượng sẽ được chuyển vào hàm theo tham chiếu.

Hàm là đối tượng hạng nhất trong JavaScript. Tất cả các hàm là đối tượng của nguyên mẫu Function. Hàm có thể được tạo và dùng trong phép toán gán như bất kỳ một đối tượng nào khác, và cũng có thể được dùng làm đối số cho các hàm khác. Do đó, JavaScript hỗ trợ hàm cấp độ cao. Ví dụ:

Code:

```
<script language="JavaScript">

 Array.prototype.fold = function (value, functor)

{

var result = value;

for (var i = 0; i < this.length; i++) {

        result = functor(result, this[i]);

}     return result;

}    var sum = [1,2,3,4,5,6,7,8,9,10].fold(0, function (a, b) { return a + b })

</script>
```

Đoạn mã nguồn trên sẽ trả lại kết quả là 55.

Vì hàm trong JavaScript là đối tượng, lập trình viên có thể khởi tạo hàm không tên:

Code:

```
<script language="JavaScript">

Function tenham()

{
```

thân hàm;

}</script>

Một ví dụ sử dụng hàm không tên trong JavaScript:

Code:

```
<script language="JavaScript">

document.onkeypress = function(e) {

    alert("Bạn vừa nhấn một phím trên bàn phím");

}</script>
```

Hàm trên sẽ hiển thị thông báo khi một số phím trên bàn phím có thể gây sự kiện onkeypress được nhấn.

Mặc định, tất cả các thành phần của đối tượng thuộc phạm vi công cộng (public). Trong JavaScript, không có khái niệm thành phần riêng hay thành phần được bảo vệ (private và protected), tuy nhiên những tính năng này có thể được giả lập.

Xây dựng đối tượng trong javascript

Đối tượng và thuộc tính

Như đã biết, một đối tượng trong JavaScript có các thuộc tính đi kèm với nó. Bạn có thể truy nhập đến các thuộc tính của nó bằng cách gọi :

objectName.propertyName

Cả tên đối tượng và tên thuộc tính đều nhạy cảm. Bạn định nghĩa một thuộc tính bằng cách gán cho nó một giá trị. Ví dụ, giả sử có một đối tượng tên là myCar (trong trường hợp này giả sử đối tượng này đã tồn tại sẵn sàng). Bạn có thể lấy các thuộc tính có tên make, model và year của nó như sau:

myCar.make = “Ford“

myCar.model = “Mustang“

myCar.year = 69;

Có một mảng lưu trữ tập hợp các giá trị tham chiếu tới từng biến. Thuộc tính và mảng trong JavaScript có quan hệ mật thiết với nhau, thực ra chúng chỉ khác nhau về cách giao tiếp với cùng một cấu trúc dữ liệu. Ví dụ cũng có thể truy nhập tới các thuộc tính của đối tượng myCar ở trên bằng mảng như sau:

myCar[make] = “Ford“

myCar[model] = “Mustang“

myCar[year] = 69;

Kiểu mảng này được hiểu như một mảng có khả năng liên kết bởi mỗi một phần tử trong đó đều có thể liên kết đến một giá trị xâu nào đó. Để minh hoạ việc mày được thực hiện như thế nào, hàm sau đây sẽ hiển thị các thuộc tính của một đối tượng thông qua tham số về kiểu đối tượng đó và tên đối tượng.

```
function show_props (obj, obj_name)

{

var result=””

for (i in obj)
```

```
result=result+ obj_name + “.”+ i+ “=” + obj[i] + “\n”

return result

}
```

Khi gọi hàm show_props(myCar,”myCar”) sẽ hiện lên:

```
myCar.make = Ford

myCar.model = Mustang

myCar.year = 69;
```

Tạo các đối tượng mới

Cả JavaScript client-side và server-side đều có một số đối tượng được định nghĩa trước. Tuy nhiên, bạn cũng có thể tạo ra những đối tượng của riêng bạn. Trong JavaScript 1.2, nếu bạn chỉ muốn tạo ra một đối tượng duy nhất của một kiểu đối tượng, bạn có thể tạo nó bằng cách sử dụng khởi tạo đối tượng. Hoặc nếu bạn muốn tạo ra nhiều cá thể của một kiểu đối tượng, bạn có thể tạo ra một hàm xây dựng trước, sau đó tạo ra các đối tượng có kiểu của hàm đó bằng toán tử new

Sử dụng khởi tạo đối tượng

Trong những phiên bản trước của Navigator, bạn chỉ có thể tạo ra một đối tượng bằng cách sử dụng hàm xây dựng chúng hoặc sử dụng một hàm được cung cấp bởi một vài đối tượng khác để đạt được mục đích.

Tuy nhiên, trong Navigator 4.0, bạn có thể tạo ra một đối tượng bằng cách sử dụng một khởi tạo đối tượng.Bạn sử dụng cách này khi bạn chỉ muốn tạo ra một cá thể đơn lẻ chứ không phải nhiều cá thể của đối tượng.

Cú pháp để tạo ra một đối tượng bằng cách khởi tạo đối tượng (Object Initializers):

```
objectName={property1: value1, property2: value2,

..., propertyN: valueN}
```

Trong đó **objectName** là tên của đối tượng mới, mỗi **propertyI** là một xác minh (có thể là một tên, một số hoặc một xâu ký tự) và mỗi **valueI** là một biểu thức mà giá trị của nó được gán cho **propertyI**. Có thể lựa chọn khởi tạo bằng tên đối tượng hoặc chỉ bằng các

khai báo. Nếu như bạn không cần dùng đến đối tượng đó trong mọi chỗ, bạn không cần phải gán nó cho một biến.

Nếu một đối tượng được tạo bằng cách khởi tạo đối tượng ở mức cao nhất, mỗi lần đối tượng đó xuất hiện trong các biểu thức, JavaScript sẽ đánh giá lại nó một lần. Ngoài ra, nếu sử dụng việc khởi tạo này trong một hàm thì mỗi lần gọi hàm, đối tượng sẽ được khởi tạo một lần

Giả sử bạn có câu lệnh sau:

if (condition)

x={hi: ”there.”}

Trong trường hợp này, JavaScript sẽ tạo ra một đối tượng và gắn nó vào biến x nếu biểu thức **condition** được đánh giá là đúng

Còn ví dụ sau tạo ra một đối tượng myHonda với 3 thuộc tính:

myHonda={color:”red”,wheels:4,engine:{cylinder:4,size:2.2}}

Chú ý rằng thuộc tính **engine** cũng là một đối tượng với các thuộc tính của nó

Trong Navigator 4.0, bạn cũng có thể sử dụng một khởi tạo để tạo một mảng. Cú pháp để tạo mảng bằng cách này khác với tạo đối tượng:

arrayName=[element0, element1,...,elementN]

Trong đó, **arrayName** là tên của mảng mới, và mỗi **elementI** là giá trị của phần tử ở vị trí đó của mảng. Khi bạn tạo một mảng bằng cách sử dụng phương pháp khởi tạo, thì nó sẽ coi mỗi giá trị là một phần tử trên mảng, và chiều dài của mảng chính là số các tham số.

Bạn không cần phải chỉ định rõ tất cả các phần tử trên mảng mới. Nếu bạn đặt hai dấu phẩy vào hàng, thì mảng sẽ được tạo với những chốn trống cho những phần tử chưa được định nghĩa như ví dụ dưới đây:

Nếu một mảng được tạo bằng cách khởi tạo(initializer) ở mức cao nhất, mỗi lần mảng đó xuất hiện trong các biểu thức, JavaScript sẽ đánh giá lại nó một lần. Ngoài ra, nếu sử dụng việc khởi tạo này trong một hàm thì mỗi lần gọi hàm, mảng sẽ được khởi tạo một lần

Ví dụ1: Tạo một mảng coffees với 3 phần tử và độ dài của mảng là 3:

coffees = [“French Roast”,”Columbian”,”Kona”]

Ví dụ 2: Tạo ra một mảng với 2 phần tử được khởi đầu và một phần tử rỗng:

fish = [“Lion”, ,” Surgeon”]

Với biểu thức này, **fish[0]** là **“Lion”, fish[2]** là **” Surgeon**”, và **fish[2]** chưa được định nghĩa.

Sử dụng một hàm xây dựng(Constructor Function)

Bạn có thể tạo ra đối tượng của riêng mình với hai bước sau:

- Định nghĩa kiểu của đối tượng bằng cách viết một hàm xây dựng.
- Tạo ra một cá thể của đối tượng đó bằng toán tử **new**

Để định nghĩa một kiểu đối tượng, ta phải tạo ra một hàm để chỉ định rõ tên, các thuộc tính và các cách thức của kiểu đối tượng đó. Ví dụ giả sử bạn muốn tạo một kiểu đối tượng ô tô với tên là **car**, có các thuộc tính **make, model, year** và **color**, để thực hiện việc này có thể viết một hàm như sau:

```
function car(make, model, year )

{

this.make = make

this.model = model

this.year = year

}
```

Chú ý việc sử dụng toán tử **this** để gán giá trị cho các thuộc tính của đối tượng phải thông qua các tham số của hàm.

Ví dụ, bạn có thể tạo một đối tượng mới kiểu **car** như sau:

mycar = new car(“Eagle”,”Talon TSi”,1993)

Câu lệnh này sẽ tạo ra đối tượng mycar và liên kết các giá trị được đưa vào với các thuộc tính. Khi đó giá trị của **mycar.make** là “*Eagle*”, giá trị của **mycar.model** là “*Talon TSi*”, và **mycar.year** là một số nguyên 1993....Cứ như vậy bạn có thể tạo ra nhiều đối tượng kiểu **car.**

Một đối tượng cũng có thể có những thuộc tính mà bản thân nó cũng là một đối tượng. Ví dụ bạn định nghĩa thêm một đối tượng khác là **person** như sau:

```
function person(name, age, sex)

{

this.name=name

this.age=age

this.sex=sex

}
```

Và sau đó ta tạo ra hai người mới:

```
rank = new person("Rank McKinnon",33,"M")

ken = new person("Ken John",39,"M")
```

Bây giờ bạn định nghĩa lại hàm xây dựng **car** như sau:

```
function car(make, model, year,owner )

{

this.make = make

this.model = model

this.year = year

this.owner = owner

}
```

Như vậy bạn có thể tạo đối tượng kiểu **car** mới:

```
car1 = new car("Eagle","Talon TSi",1993,rank)

car2 = new car("Nissan","300ZX",1992,ken)
```

Như vậy, thay vì phải qua một xâu ký tự hay một giá trị số khi tạo đối tượng, ta chỉ cần đưa hai đối tượng đã được tạo ở câu lệnh trên vào dòng tham số của đối tượng mới tạo. Ta cũng có thể lấy được thuộc tính của đối tượng owner bằng câu lênh sau:

car2.owner.name

Chú ý rằng bạn cũng có thể tạo ra một thuộc tính mới cho đối tượng trước khi định nghĩa nó, ví dụ:

car1.color="black"

Như vậy, thuộc tính *color* của đối tượng **car1** được gán là "*black*". Tuy nhiên, nó sẽ không gây tác động tới bất kỳ một đối tượng kiểu **car** nào khác. Nếu muốn thêm thuộc tính cho tất cả các đối tượng thì phải định nghĩa lại hàm xây dựng đối tượng.

Lập mục lục cho các thuộc tính của đối tượng

Trong Navigator 2.0, bạn có thể gọi thuộc tính của một đối tượng bằng tên thuộc tính hoặc bằng số thứ tự của nó. Tuy nhiên từ Navigator 3.0 trở đi, nếu ban đầu bạn định nghĩa một thuộc tính bằng tên của nó, bạn sẽ luôn luôn phải gọi nó bằng tên, và nếu bạn định nghĩa một thuộc tính bằng chỉ số thì bạn cũng luôn luôn phải gọi tới nó bằng chỉ số.

Điều này ứng dụng khi bạn tạo một đối tượng với những thuộc tính của chúng bằng hàm xây dựng (như ví dụ về kiểu đối tượng **car** ở phần trước) và khi bạn định nghĩa những thuộc tính của riêng một đối tượng (như mycar.color="red"). Vì vậy nếu bạn định nghĩa các thuộc tính của đối tượng ngay từ đầu bằng chỉ số như mycar[5]="25 mpg", bạn có thể lần lượt gọi tới các thuộc tính khác như mycar[5].

Tuy nhiên điều này là không đúng đối với những đối tượng tương ứng của HTML như mảng form. Bạn có thể gọi tới các đối tượng trong mảng bởi số thứ tự hoặc tên của chúng. Ví dụ thẻ <FORM> thứ hai trong một document có thuộc tính NAME là "myform" thì bạn có thể gọi tới form đó bằng document.form[1] hoặc document.form["myForm"] hoặc document.myForm

Định nghĩa thêm các thuộc tính cho một kiểu đối tượng

Bạn có thể thêm thuộc tính cho một kiểu đối tượng đã được định nghĩa trước bằng cách sử dụng thuộc tính property. Thuộc tính được định nghĩa này không chỉ có tác dụng đối với một đối tượng mà có tác dụng đối với tất cả các đối tượng khác cùng kiểu.Ví dụ sau thực hiện thêm thuộc tính **color** cho tất cả các đối tượng kiểu **car**, sau đó gắn một giá trị màu cho thuộc tính **color** của đối tượng **car1:**

car.prototype.color=null

car1.color=”red”

Định nghĩa các cách thức

Một cách thức là một hàm được liên kết với một đối tượng. Bạn định nghĩa một cách thức cũng có nghĩa là bạn định nghĩa một hàm chuẩn. Bạn có thể sử dụng cú pháp sau để gắn một hàm cho một đối tượng đang tồn tại:

object.methodname = function_name

Trong đó object là đối tượng đang tồn tại, methodname là tên cách thức và function_name là tên hàm

Bạn có thể gọi cách thức này từ đối tượng như sau:

object.methodname(<tham số>)

Bạn có thể định nghĩa cách thức cho một kiểu đối tượng bằng cách đưa cách thức đó vào trong hàm xây dựng đối tượng. Ví dụ bạn có thể định nghĩa một hàm có thể định dạng và hiển thị các thuộc tính của các đối tượng kiểu **car** đã xây dựng ở phần trước:

```
function displayCar ()

{

var result = “Abeautiful”+this.year+ “ ”+ this.make + “ ”+ this.model

document.write(result)

}
```

Bạn có thể thêm cách thức này vào cho đối tượng car bằng cách thêm dòng lệnh sau vào hàm định nghĩa đối tượng

this.displayCar= displayCar;

Như vậy có thể định nghĩa lại đối tượng **car** như sau:

```
function car(make, model, year,owner )

{
```

```
this.make = make

this.model = model

this.year = year

this.owner = owner

this.displayCar= displayCar

}
```

Sau đó, bạn có thể gọi cách thức displayCar đối với mỗi đối tượng:

```
car1.displayCar()

car2.displayCar()
```

Sử dụng cho các tham chiếu đối tượng (Object References)

JavaScript có một từ khoá đặc biệt là **this** mà bạn có thể sử dụng nó cùng với một cách thức để gọi tới đối tượng hiện thời. Ví dụ, giả sử bạn có một hàm validate dùng để xác nhận giá trị thuộc tính của một đối tượng nằm trong một khoảng nào đó:

```
function validate(obj, lowval, hival)

{

if ( (obj.value<lowdate)||(obj.value>hival) )

alert("Invalid value!")

}
```

Sau đó bạn có thể gọi hàm **validate** từ mỗi thẻ sự kiện **onChange**:

```
<INPUT TYPE="TEXT" NAME="AGE" SIZE=3 onChange="validate(this,18,99)" >
```

Khi liên kết với một thuộc tính form, từ khoá **this** có thể gọi tới form cha của đối tượng hiện thời. Trong ví dụ sau, **myForm** có chứa đối tượng **Text** và một nút bấm. Khi người sử dụng kích vào nút bấm, trường text sẽ hiển thị tên form. Thẻ sự kiện onClick của nút bấm sử dụng **this.form** để gọi tới form cha là **myForm**.

```
<FORM NAME="myForm">
```

Form name:<INPUT TYPE=”text” NAME=”text1” VALUE=”Beluga”>

<P>

<INPUT TYPE=”button” NAME=”button1”

value=”Show Form Name” onClick=”this.form.text1.value=this.form.name”>

</FORM>

Xoá đối tượng

Trong JavaScript cho Navigator 2.0, bạn không thể xoá các đối tượng-chúng vẫn tồn tại trong khi bạn đã rời khỏi trang đó. Trong khi JavaScript cho Navigator 3.0 cho phép bạn có thể xoá một đối tượng bằng cách đặt cho nó trỏ tới giá trị Null (nếu như đó là lần cuối cùng gọi tới đối tượng). JavaScript sẽ đóng đối tượng đó ngay lập tức thông qua biểu thức gán.

Truy cập thành phần dữ liệu

Khái niệm DOM

DOM là chữ viết tắt từ tiếng Anh*Document Object Model* ("Mô hình Đối tượng Tài liệu"), là một giao diện lập trình ứng dụng (API). Thường thường DOM, có dạng một cây cấu trúc dữ liệu, **được dùng để truy xuất các tài liệu dạng** HTML **và** XML. Mô hình DOM độc lập với hệ điều hành và dựa theo kỹ thuật lập trình hướng đối tượng để mô tả tài liệu.

Ban đầu, chưa có chuẩn thống nhất nên các thành phần trong một tài liệu HTML mô tả bằng các phiên bản khác nhau của DOM được hiển thị bởi các chương trình duyệt web thông qua JavaScript. Điều này buộc World Wide Web Consortium (W3C) phải đưa ra một loạt các mô tả kĩ thuật về tiêu chuẩn cho DOM để thống nhất mô hình này.

Mặc dù một tài liệu hay văn bản có cấu trúc chặt chẽ (*well-structured document*) luôn luôn có thể được mô hình hóa bằng một cấu trúc dạng cây, DOM không có giới hạn về cấu trúc dữ liệu của một tài liệu.

Hầu hết các bộ phân tích XML (*XML parsers*) (ví dụ: Xerces) và bộ xử lí XSL (ví dụ: Xalan) đã được phát triển để sử dụng cấu trúc cây này. Những hiện thực như vậy đòi hỏi toàn bộ nội dụng của một văn bản phải được phân tích và lưu trong bộ nhớ. Vì thế, DOM được sử dụng tốt nhất trong các ứng dụng mà trong đó các thành phần của tài liệu có thể được truy xuất và thao tác một cách ngẫu nhiên. Với các ứng dụng dựa trên XML, bao gồm yêu cầu đọc/ghi có chọn lọc cho mỗi lần phân tích (*one-time selective read/ write per parse*), DOM cho thấy được sự tối ưu về mặt bộ nhớ. Trong các trường hợp đó thì giao diện lập trình ứng dụng SAX trở nên rất tiện lợi về cả mặt tốc độ và bộ nhớ.

Xem sét ví dụ sau:

```
<html>

<head>

<title>DOM Example</title>

</head>

<body>

<p>Hello World!</p>
```

Isn’t this exciting?

You’re learning to use the DOM!

</body>

</html>

Lấy các đối tượng trong document

HTML Code:

var oHtml = document.documentElement; // Lấy tài liệu html

var oHead = oHtml.firstChild; // Lấy head

var oHead = oHtml.childNodes[0];

var oHead = oHtml.childNodes.item(0);

var oBody = oHtml.lastChild; // Lấy body

alert(oHead.parentNode == oHtml); //Lấy parent, trả về true

alert(oBody.previousSibling == oHead); // kết quả “true”

alert(oHead.nextSibling == oBody); //kết quả “true”

alert(oHead.ownerDocument == document); //kết quả “true”

Lấy loại node

Sử dụng thuộc tính **nodeType**

alert(document.nodeType); //outputs “9”

alert(document.documentElement.nodeType); //outputs “1”

alert(document.nodeType == Node.DOCUMENT_NODE); //outputs “true”

alert(document.documentElement.nodeType == Node.ELEMENT_NODE); //outputs “true”

Dưới đây là danh sách các loại node:

ELEMENT_NODE: 1,

ATTRIBUTE_NODE: 2,

TEXT_NODE: 3,

CDATA_SECTION_NODE: 4,

ENTITY_REFERENCE_NODE: 5,

ENTITY_NODE: 6,

PROCESSING_INSTRUCTION_NODE: 7,

COMMENT_NODE: 8,

DOCUMENT_NODE: 9,

DOCUMENT_TYPE_NODE: 10,

DOCUMENT_FRAGMENT_NODE: 11,

NOTATION_NODE: 12

Tương tác với attribute (thuộc tính)

getAttribute(name) — Lấy thuộc tính với tên được chỉ định

setAttribute(name, newvalue) — Gán thuộc tính với tên và value

removeAttribute(name) — Xóa 1 thuộc tính

Truy vấn các node trong DOM

getElementsByTagName() : lấy các node với tag name

getElementsByName() : lấy các node với thuộc tính name

getElementById() : lấy các node với id

Ví dụ:

var oImgs = document.getElementsByTagName(“img”);

var oRadios = document.getElementsByName(“radColor”);

var oDiv1 = document.getElementById(“div1”);

Dùng DOM để truy xuất đến các phần tử trong tài liệu HTML

Các hiện thực khác nhau của DOM trong các trình duyệt web khác nhau đã dẫn đến các vấn đề không tương hợp trong quá khứ. Vì thế, các lập trình viên web tốt hơn hết là nên kiểm tra xem một thuộc tính có tồn tại hay không trước khi sử dụng nó. Đoạn mã sau cho thấy cách kiểm tra phương thức quan trọng của W3CDOM có tồn tại hay không trước khi muốn thực thi đoạn mã phụ thuộc vào sự hỗ trợ của W3CDOM.

```
 if (document.getElementById && document.getElementsByTagName) {

    // vì các phương thức quan trọng getElementById và getElementsByTagName

    // có mặt nên đoạn mã sau có thể an toàn mà sử dụng chúng.

     obj = document.getElementById("navigation")

    // đoạn mã khác có sử dụng W3CDOM.

    // .....

 }
```

Chương trình duyệt Internet Explorer của Microsoft – phiên bản 5 (1999), ... phiên bản 6 (2001) – là một trong các trình duyệt phổ biến nhất trong năm 2005. Internet Explorer và các trình duyệt khác dựa trên động cơ hiển thị Gecko như là Mozilla và Firefox hỗ trợ nhiều thuộc tính của W3C DOM. Vì thế các vấn đề trong việc sử dụng W3CDOM không còn tồn tại nhiều nữa như từng có trước đây với Internet Explorer 4.x và Netscape 4.x vào năm 2000. Bảng trong bài so sánh các động cơ hiển thị (DOM) cho thấy các phương thức và thuộc tính nào có thể được sử dụng một cách an toàn cho mỗi chương trình duyệt.

Giới thiệu về DHTML

Khái niệm

HTML động hay **DHTML** (viết tắt tiếng Anh: Dyna*mic HTML*) là một thể hiện của việc tạo ra một trang web bằng cách kết hợp các thành phần: ngôn ngữ đánh dấuHTML tĩnh, ngôn ngữ kịch bản máy khách (như là Javascript), và ngôn ngữ định dạng trình diễn Cascading Style Sheets và Document Object Model (DOM).

DHTML có thể được sử dụng để tạo ra 1 ứng dụng trên trình duyệt web: ví dụ như dễ dàng điều hướng, tạo một đơn web tương tác với người dùng hoặc tạo ra một bài tập sử dụng cho e-learning. Bởi vì== Cấu trúc của một trang web == Đặc trưng của một trang web sử dụng **DHTML** được cấu thành như sau:

```
</script> </head> <body>

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

Thường thì mã JavaScript được lưu giữ trong một tập tin riêng, được nạp vào trang web bằng liên kết đến tập tin chứa mã JavaScript:

```
<script type="text/javascript" src="script.js"></script>
```

Cấu trúc trang DHTML

Đoạn mã dưới đây minh họa một chức năng thường được sử dụng. Phần thêm vào của trang web sẽ chỉ được hiển thị nếu người dùng yêu cầu nó. Như trong e-learning chức năng này sử dụng để hiển thị gợi ý được thêm vào.

```
<html>

<head><title>Ví dụ</title>

 <style type="text/css">

<!--

   h2 {background-color: lightblue; width: 100%}

   a {font-size: larger; background-color: goldenrod}

   a:hover {background-color: gold}
```

```
    #example1 {display: none; margin: 3%; padding: 4%; background-color:limegreen}
-->
</style>


<script type="text/javascript">
<!--
function changeDisplayState (id) {
    e = document.getElementById(id);
     if (e.style.display == 'none' || e.style.display == '')
{
    e.style.display = 'block';
    showhide.innerHTML = 'Dấu ví dụ'};
    else
{
        e.style.display = 'none';
        showhide.innerHTML = 'Hiển thị ví dụ';
    }
}
//-->
</script>
</head>
<body>
```

```
<a id="showhide" href="#"
    onclick="javascript:changeDisplayState('example1')">Hiển thị ví dụ</a>
<div id="example1">Đây là ví dụ: văn bản chỉ hiển thị khi bấm chuột vào
    liên kết.</div>
<p>Thêm văn bản thường...<p>
</body>
</html>
```

Bài 5 : Xây dựng trang Web động

Cơ bản về trang web động

Khái niệm về trang web động

Website động là trang web mà toàn bộ thông tin được tổ chức bằng một hệ cơ sở dữ liệu (database). Gồm những chức năng giúp cho việc quản lý thông tin cũng như khai thác thông tin rất linh hoạt và dễ dàng. Có thể hiểu website động là một phần mềm chạy trên internet, phần mềm này sẽ trả lại thông tin xuống máy của khách lướt web tương ứng với mỗi.

Web "ĐỘNG" là thuật ngữ được dùng để chỉ những website được hỗ trợ bởi một phần mềm cơ sở web, nói đúng hơn là một chương trình chạy được với giao thức HTTP. Thực chất, website động có nghĩa là một website tĩnh được "ghép" với một phần mềm web (các modules ứng dụng cho Web). Với chương trình phần mềm này, người chủ website thực sự có quyền điều hành nó, chỉnh sửa và cập nhật thông tin trên website của mình mà không cần phải nhờ đến những người chuyên nghiệp.

Bạn hãy tưởng tượng website như một công cụ quảng cáo luôn có thể tiếp cận với khách hàng tiềm năng, cũng như khách hàng hiện tại của bạn bất cứ lúc nào, bất cứ ở đâu, không hạn chế về mặt thời gian và không gian. Giả sử cửa hàng của Bạn là một phòng trưng bày về mẫu mốt thời trang với nhiều cô ma-nơ-canh đứng trưng bày các mẫu mốt mới.

Nếu Bạn làm web tĩnh, cũng giống như các cô ma-nơ-canh này đã được chế tạo rất hoàn thiện nhưng sẽ không bao giờ thay đổi tư thế, về cả những bộ quần áo mà các cô mặc. Nếu muốn làm lại kiểu dáng mới, Bạn phải HOÀN TOÀN PHỤ THUỘC NHÀ CHẾ TẠO, hoặc Bạn phải mất chi phí mua mới. Còn nếu Bạn làm web động, thì cũng giống như các cô ma-nơ-canh này chỉ được dựng lên như một bộ khung mà tự Bạn luôn có thể thay đổi từ dáng đứng, cách ăn mặc, dù là thời trang mùa xuân, mùa hè, mùa thu hay mùa đông, các mẫu mốt luôn hợp thời đại, mà KHÔNG MẤT THÊM MỘT KHOẢN CHI PHÍ NHỎ NÀO cho người tạo ra chúng. Hiểu cách khác, những bộ mốt mới trưng bày chính là những thông tin, thông báo về tình hình phát triển các sản phẩm - dịch vụ mà Bạn luôn muốn cập nhật để khách hàng được rõ.

Hãy tưởng tượng tiếp, các modules của một website động cũng giống như những thành phần của một bộ khung ma-nơ-canh. Bạn có thể chỉnh sửa cẳng tay của những bộ khung này, nâng chúng lên hoặc hạ chúng xuôi xuống, điều chỉnh thành chân bước hay chân đứng thẳng, thành tư thế ngồi hoặc đứng, đó là khả năng tuỳ biến của một chương trình

phần mềm điển hình. Hoặc Bạn có thể tháo rời hay lắp lại đôi tay, đôi chân của ma-nơ-canh, đó là khả năng tương thích của từng module với tổng thể một chương trình.

So sánh giữa trang web động và trang web tĩnh

WEBSITE ĐỘNG	WEBSITE TĨNH
Ưu điểm	
Người quản trị dễ dàng thay đổi cập nhật thông tin bất cứ lúc nào một cách đơn giản gần như tất cả những người dùng internet đều có thể làm được.Có thực hiện những vấn đề phức tạp có thể là tính hóa đơn, quản lý đơn hàng, thanh toán online, so sánh, tìm kiếm sản phẩm theo yêu cầu cụ thể ...Số lượng các trang phụ thuộc vào số lượng thông tin mà khách hàng cập nhật các trang này sẽ tự động phát sinh theo các mục tương ứng và có liên kết với nhau.	Tốc độ truy cập nhanh.Các máy chủ tìm kiếm dễ nhận diện website.
Nhược điểm	
Tốc độ truy cập chậm hơn website tĩnh lý do là mã lệnh của website động cần webserver biên dịch mã lệnh lập trình thành các thẻ html (hyper text make up language_ngôn ngữ đánh dấu siêu văn bản) rồi mới chuyển đến máy của người lướt web.	Thay đổi thông tin khó khăn (mất nhiều thời gian và đòi hỏi có một số kỹ năng sử dụng html, phần mềm ftp). Do không có mã lệnh lập trình vì vậy việc cập nhật, thay đổi nội dung thông tin của website mang nặng tính thủ công nên cần nhiều thời gian.Số lượng các trang thông tin theo lý thuyết là không giới hạn nhưng với số trang càng lớn càng tốn nhiều thời gian chẳng hạn cần thêm một trang thông tin thì phải sửa tất cả những trang còn lại.
Cách thức cập nhật thông tin	
Thông qua tài khoản quản trị admin, khi đăng nhập sẽ xuất hiện chức năng công cụ quản trị tương ứng với quyền hạn của mỗi người quản trị. Điều này làm cho việc kiểm soát thông tin cũng như cập nhật, thay đổi rất đơn giản.	Xử lý trực tiếp vào các file html thông qua tài khoản ftp đưa lên internet.

Cấu trúc một trang web

Nếu ta tạm coi Website như 1 doanh nghiệp trong đời thường, thì để thiết lập và đưa vào hoạt động 1 Website cũng phải đáp ứng được tối thiểu 3 yếu tố cơ bản như doanh nghiệp là:

Tên Website (hay còn gọi là Tên miền ảo hoặc Domain name) Thường tương ứng với tên của tổ chức hay doanh nghiệp trong thực tế. Tên website rất quan trọng vì nó thể hiện phần nào nội dung của website và cũng là thể hiện mục đích tồn tại của trang web.

Web Hosting (hay còn gọi là nơi lưu giữ trên máy chủ Internet) Là nơi đặt toàn bộ nội dung, thành phần của website. Việc lựa chọn được web hosting lý tưởng cũng giúp website được đạt được hiệu quả thực tế cao hơn. Thông thường các website cần một web hosting có độ bảo mật cao, băng thông lớn và ổn định.

Bản thân trang web bao gồm có nội dung website, những tiện ích đi kèm, người quản lý…Đây có thể nói là yếu tố quan trọng nhất của một website, nó thể hiện được cái mà người người tạo ra website mong muốn và là công cụ để thực hiện ý tưởng của tổ chức, doanh nghiệp sở hữu website đó.

Ngôn ngữ cơ bản thường dùng

Servlet

Cùng với Java, Sun đồng thời đưa ra một công nghệ mới gọi là servlet. Các đoạn mã Java sẽ không chạy phía client như với applet; chúng sẽ được chạy trên một ứng dụng phía server. Servlet cũng đồng thời phục vụ các CGI script. Servlet là một bước tiến lớn, nó đưa ra một thư viện hàm API trên Java và một thư viện hoàn chỉnh để thao tác trên giao thức HTTP.

Servlet

JavaServer Page (JSP) là một công nghệ lập trình Web của Sun, JSP là công nghệ đòi hỏi một trình chủ hiểu được Java. Trong nhiều năm trở lại đây JSP luôn là một trong những công cụ xây dựng web được yêu thích số một thế giới. JSP đưa ra nền tảng đầu tiên về xây dựng web ứng dụng code-behide. Là bộ công cụ rất hữu ích cho người lập trình. Ngoài ra nó còn hỗ trợ rất tốt vấn đề bảo mật và tốt độ truy cập rất cao.

ASP

Microsoft đã nghiên cứu các nhược điểm của servlet và tạo ra ASP dễ dàng hơn để thiết kế các trang web động. Microsoft thêm các bộ công cụ rất mạnh và sự tích hợp rất hoàn hảo với các Web server. JSP và ASP có những nét tương đương vì chúng đều được thiết kế để phân tách qua trình xử lí khỏi quá trình biểu diễn. Có sự khác biệt về kỹ thuật, song cả hai đều cho phép các nhà thiết kế Web tập trung vào cách bố trí (layout) trong khi các nhà phát triển phần mềm thì tập trung vào các kỹ thuật lập trình logic.

PHP

PHP (Hypertext Preprocessor) là một trong các công nghệ cung cấp các bộ công cụ rất mạnh cho các nhà phát triển web. Tới nay PHP vẫn dành được sự quan tâm đặc biệt của những nhà phát triển web đặc biệt là trong lĩnh vực mã nguồn mở. PHP tỏ ra làm một công nghệ tương đối dễ sử dụng và rất thích hợp trong môi trường web với khả năng bảo mật rất cao.

ASP.NET

ASP.Net là kỹ thuật lập trình và phát triển ứng dụng web ở phía Server (Server-side) dựa trên nền tảng của Microsoft .Net Framework. Hiện nay ASP.NET là công cụ phát triển web nhanh và dễ ràng nhất hiện nay. ASP.NET hỗ trợ rất nhiều ngôn ngữ lập trình

riêng biệt tích hợp trên nền tảng Microsoft .Net framework. Trong phạm vi của môn học này chúng ta sẽ cùng tìm hiểu công nghệ ASP.NET.

Giới thiệu cơ bản về ngôn ngữ ASP.NET

Khái niệm và nguồn gốc xuất xứ

Từ khoảng cuối thập niên 90, ASP (Active Server Page) đã được nhiều lập trình viên lựa chọn để xây dựng và phát triển ứng dụng web động trên máy chủ sử dụng hệ điều hành Windows. ASP đã thể hiện được những ưu điểm của mình với mô hình lập trình thủ tục đơn giản, sử dụng hiệu quả các đối tượng COM: ADO (ActiveX Data Object) - xử lý dữ liệu, FSO (File System Object) - làm việc với hệ thống tập tin…, đồng thời, ASP cũng hỗ trợ nhiều ngôn ngữ: VBScript, JavaScript. Chính những ưu điểm đó, ASP đã được yêu thích trong một thời gian dài.

Tuy nhiên, ASP vẫn còn tồn đọng một số khó khăn như Code ASP và HTML lẫn lộn, điều này làm cho quá trình viết code khó khăn, thể hiện và trình bày code không trong sáng, hạn chế khả năng sử dụng lại code. Bên cạnh đó, khi triển khai cài đặt, do không được biên dịch trước nên dễ bị mất source code. Thêm vào đó, ASP không có hỗ trợ cache, không được biên dịch trước nên phần nào hạn chế về mặt tốc độ thực hiện. Quá trình xử lý Postback khó khăn, …

Đầu năm 2002, Microsoft giới thiệu một kỹ thuật lập trình Web khá mới mẻ với tên gọi ban đầu là ASP+, tên chính thức sau này là ASP.Net. Với ASP.Net, không những không cần đòi hỏi bạn phải biết các tag HTML, thiết kế web, mà nó còn hỗ trợ mạnh lập trình hướng đối tượng trong quá trình xây dựng và phát triển ứng dụng Web.

ASP.Net là kỹ thuật lập trình và phát triển ứng dụng web ở phía Server (Server-side) dựa trên nền tảng của Microsoft .Net Framework.

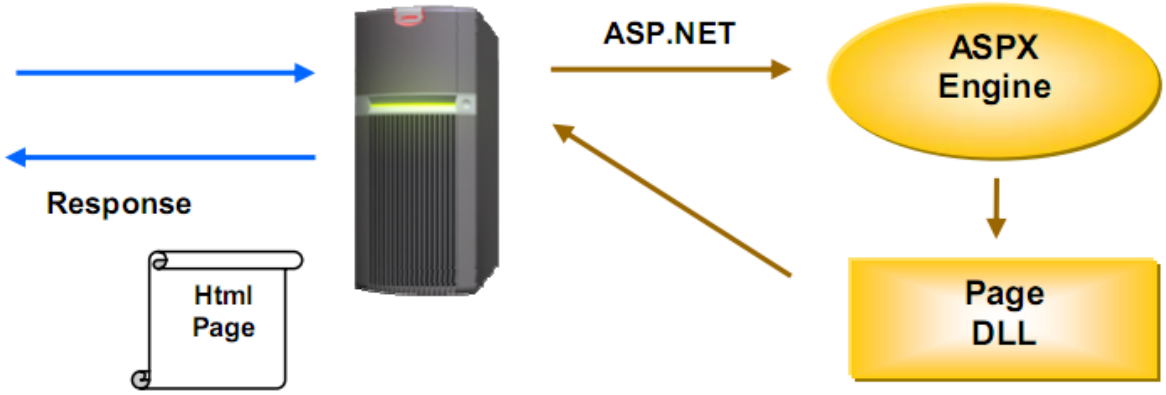
Hầu hết, những người mới đến với lập trình web đều bắt đầu tìm hiểu những kỹ thuật ở phía Client (Client-side) như: HTML, Java Script, CSS (Cascading Style Sheets). Khi Web browser yêu cầu một trang web (trang web sử dụng kỹ thuật client-side), Web server tìm trang web mà Client yêu cầu, sau đó gởi về cho Client. Client nhận kết quả trả về từ Server và hiển thị lên màn hình.

ASP.Net sử dụng kỹ thuật lập trình ở phía server thì hoàn toàn khác, mã lệnh ở phía server (ví dụ: mã lệnh trong trang ASP) sẽ được biên dịch và thi hành tại Web Server. Sau khi được Server đọc, biên dịch và thi hành, kết quả tự động được chuyển sang HTML/JavaScript/CSS và trả về cho Client. Tất cả các xử lý lệnh ASP.Net đều được thực hiện tại Server và do đó, gọi là kỹ thuật lập trình ở phía server.

Những ưu điểm nổi bật của ASP.NET

ASP.Net cho phép bạn lựa chọn một trong các ngôn ngữ lập trình mà bạn yêu thích: Visual Basic.Net, J#, C#,…

Trang ASP.Net được biên dịch trước. Thay vì phải đọc và thông dịch mỗi khi trang web được yêu cầu, ASP.Net biên dịch những trang web động thành những tập tin DLL mà Server có thể thi hành nhanh chóng và hiệu quả. Yếu tố này là một bước nhảy vọt đáng kể so với kỹ thuật thông dịch của ASP.



ASP.Net hỗ trợ mạnh mẽ bộ thư viện phong phú và đa dạng của .Net Framework, làm việc với XML, Web Service, truy cập cơ sở dữ liệu qua ADO.Net, …

ASPX và ASP có thể cùng hoạt động trong 1 ứng dụng.

ASP.Net sử dụng phong cách lập trình mới: Code behide. Tách code riêng, giao diện riêng do vậy dễ đọc, dễ quản lý và bảo trì.

Kiến trúc lập trình giống ứng dụng trên Windows.

Hỗ trợ quản lý trạng thái của các control

Tự động phát sinh mã HTML cho các Server control tương ứng với từng loại Browser

Hỗ trợ nhiều cơ chế cache.

Triển khai cài đặt

Không cần lock, không cần đăng ký DLL

Cho phép nhiều hình thức cấu hình ứng dụng

Hỗ trợ quản lý ứng dụng ở mức toàn cục

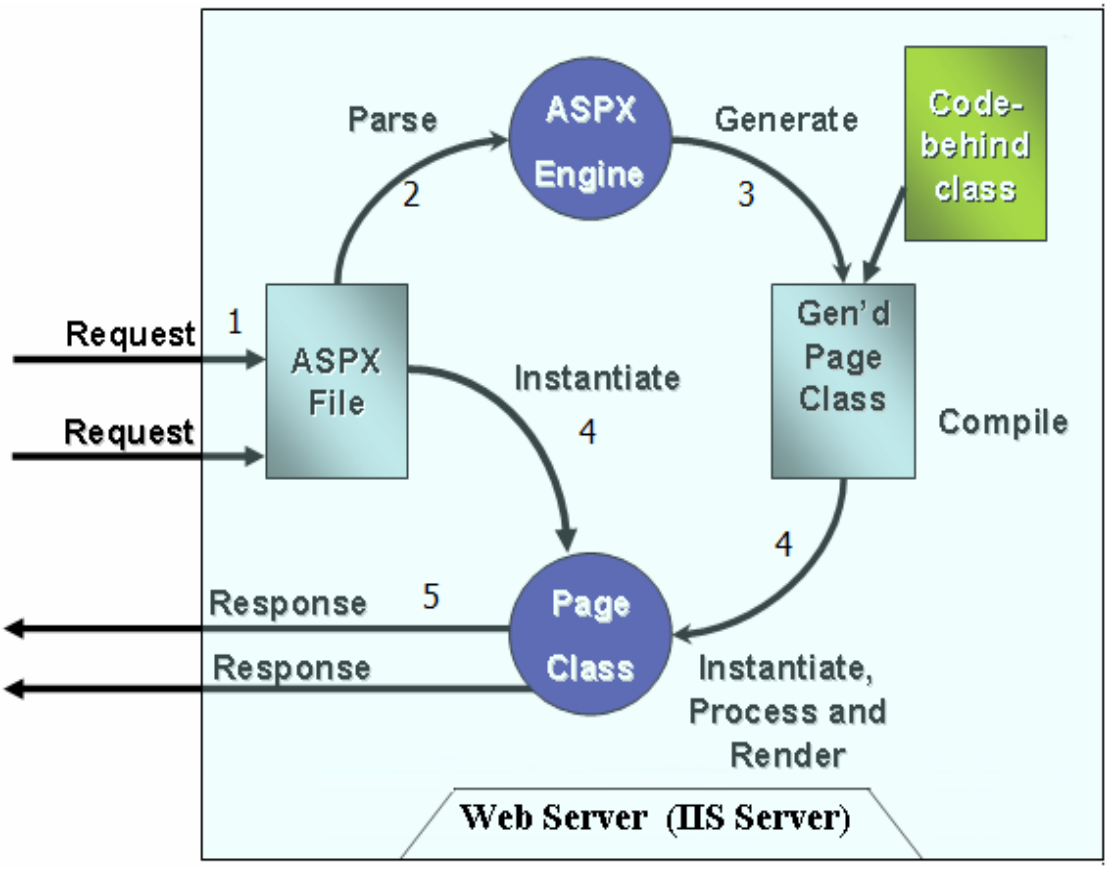
Global.aspx có nhiều sự kiện hơn

Quản lý session trên nhiều Server, không cần Cookies

Cách thức thực thi một trang web động

Thực thi bên phía máy chủ với ASP.NET

Đối với các trang ASP.NET, thì cơ chế xử lý giống như đã mô tả ở trên, tức là theo mô hình xử lý bên phía server. Nhưng có bổ sung thêm tính năng Compile and Cache:



Giải thích cơ chế xử lý ở trên:

- Bước 1: Người lập trình phải tạo các trang ASPX (giả sử tên trang đó là **abc.aspx)** và đặt nó vào trong thư mục web của web server (có tên là www.server.com). Trên thanh địa chỉ của trình duyệt, người dùng nhập trang www.server.com/Default.aspx.

- Bước 2: Trình duyệt gửi yêu cầu tới server với nội dung: *"Làmơngửichotôitrang abc.aspxthìtốt!"*.

- Bước 3: web server sẽ biên dịch code của trang aspx (bao gồm cả các mã code vb.net/ c# - gọi là code behind hay code file) thành class.

- Bước 4: Lớp sau khi được biên dịch sẽ thực thi.

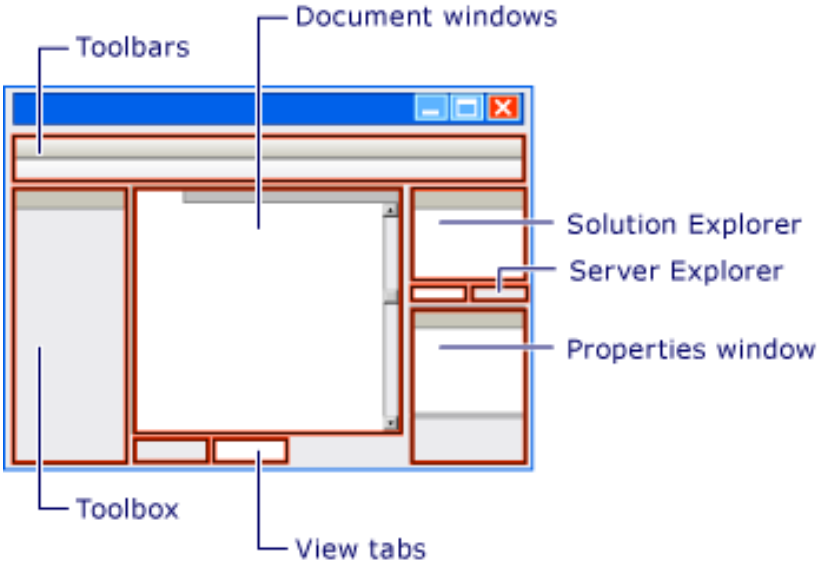
- Bước 5: trả kết quả về cho trình duyệt.

Riêng với ASP.NET thì việc biên dịch sẽ được thực hiện như sau:

Môi trường xây dựng

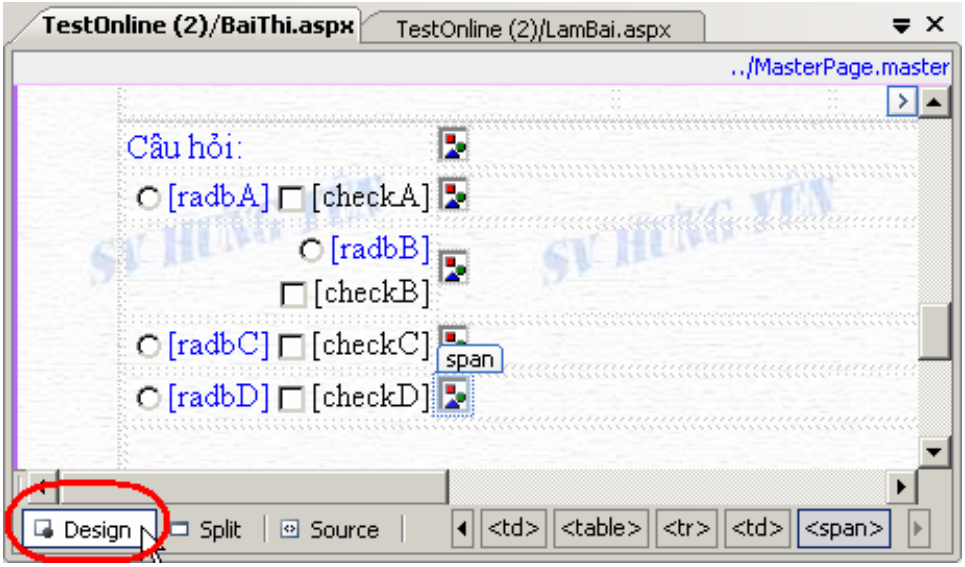
Một điều thật tuyệt vời là Visual Studio sử dụng một trình IDE chung cho toàn bộ ngôn ngữ lập trình (ASP.NET, VB.NET, C#,…). Điều này đảm bảo tính nhất quán cho các ngôn ngữ trên nền .NET, giúp bạn chỉ cần "Học một lần nhưng áp dụng mọi nơi".

Tr o n g đó:



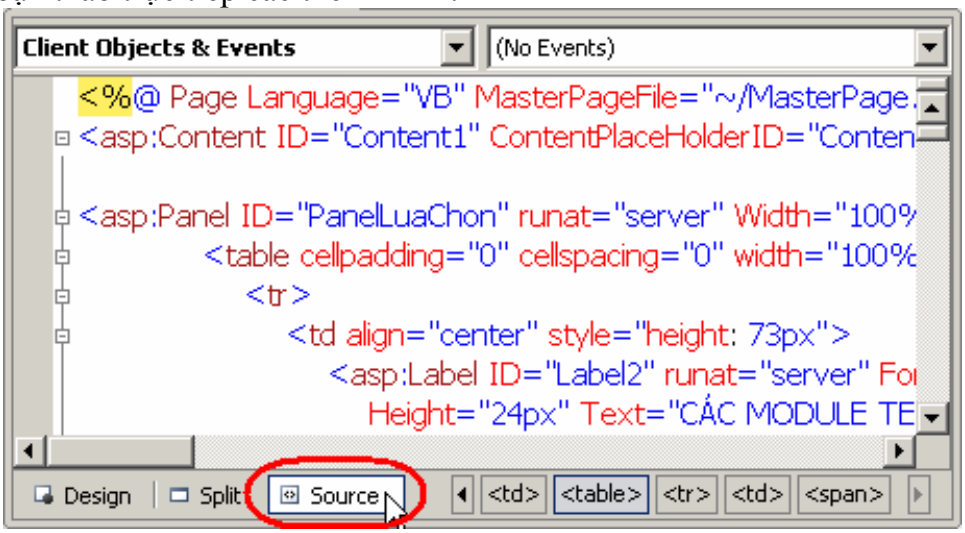
C ửa s ổ giao d i ện chính c ủ a mô i t r ư ờng phá t tr i ể n tích h ợ p.

Tab Design: để hiển thị trang web ở chế độ Design, tức là cho phép sửa chữa nội dung trang web trực quan.



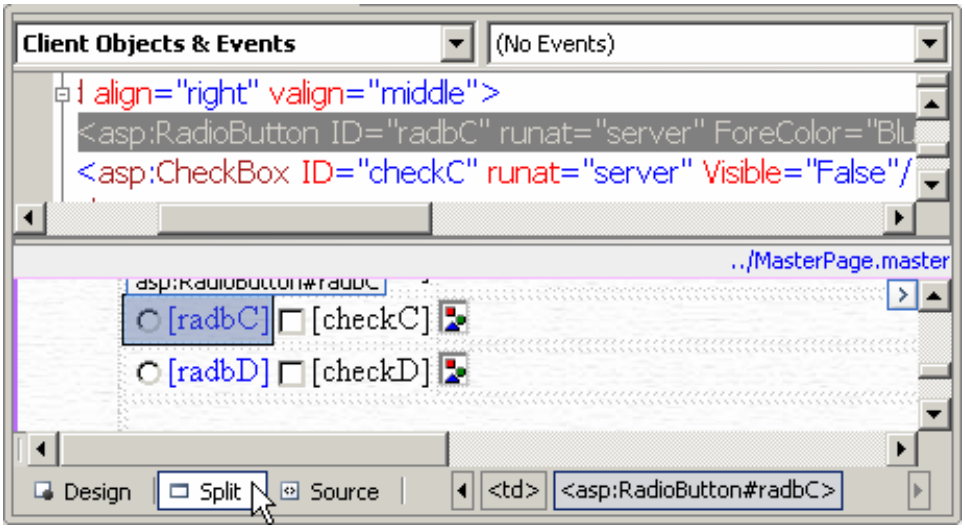
M ở trang ở c hế độ Design

- Tab Source: Mở trang ở chế độ mã nguồn HTML. Tại đây người dùng có thể soạn thảo trực tiếp các thẻ HTML.

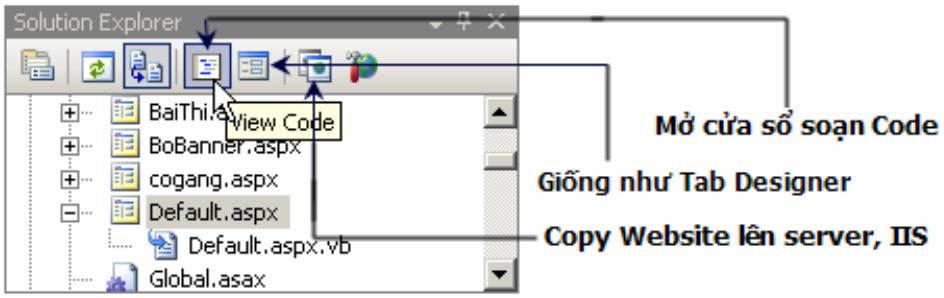


M ở trang ở c hế độ Source

Tab Split: Cho phép xem trang web đồng thời ở cả hai chế độ.

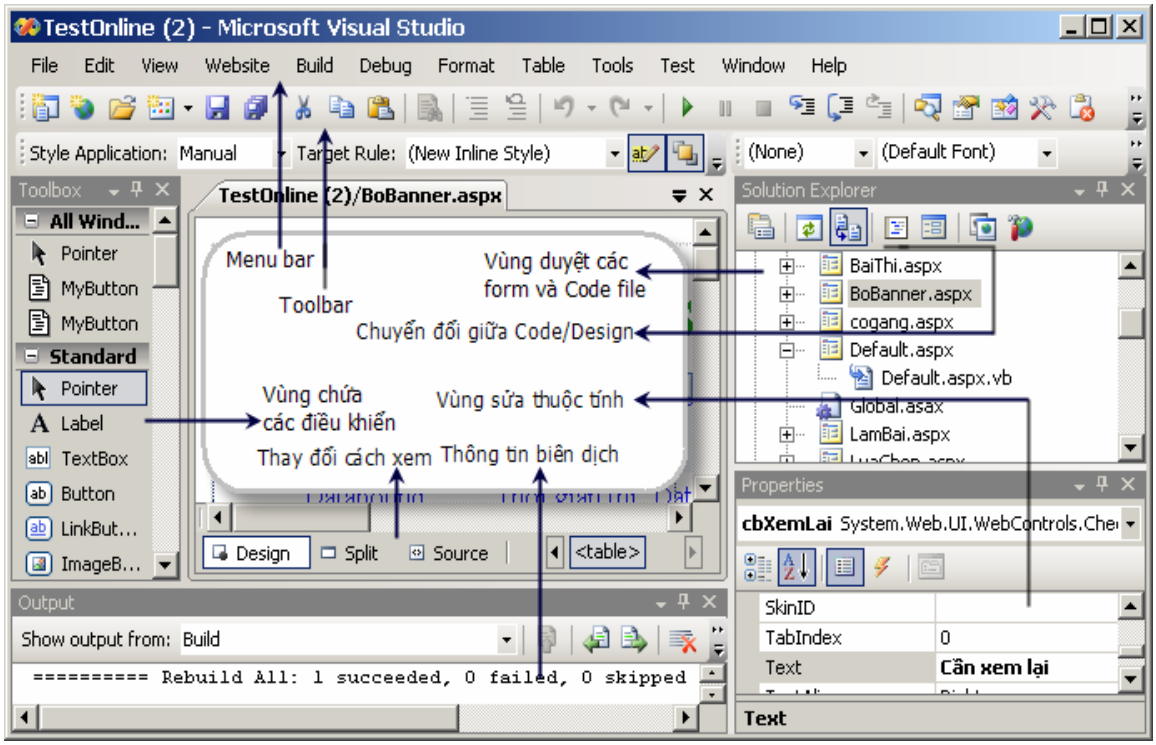


M ở trang ở c hế độ kết h ợp, v ừa xem code HTML v ừ a xem Design.



M ở c ửa s ổ s oạn Code (C#, VB.NET

Ngoài thao tác trực tiếp thông qua hệ thống menu, nút lệnh, người dùng còn có thể sử dụng tổ hợp các phím tắt. (Mở menu bar và xem tổ hợp phím tắt bên cạnh). Ví dụ: Shift+F7 để xem ở chế độ Design, F7 xem ở chế độ Code, F4 Focus tới Properties….



Giao d iệ n c ủa h ệ th ố ng IDE.

Cách thức xây dựng một trang web động đơn giản

Để tạo một trang web động đơn giản chúng ta cần tối thiểu một cơ sở dữ liệu có thể là Access có thể là sql server hoặc file database… Sau đó là việc ta kết nối website với Database trên để có thể thực hiện chức năng cập nhật sửa xóa thông tin.

Chúng ta có thể bắt đầu xây dựng một trang web động đơn giản với ASP.NET đầu tiên.

Ví dụ:

Ta có cơ sở dữ liệu là Nhanxet là tập hợp tất cả những lời nhận xét của mỗi chúng ta trong quá trình học môn công nghệ web đã thực hiện.

Chương trình sẽ hiển thị tất cả những lời nhận xét và một hộp thoại cho phép người dùng thêm một nhận xét.

Cơ sở dữ liệu được xây dựng ngay trên nên visual studio 2008 gồm database có tên là nhanxet.mdf gồm một table duy nhất NX(manhanxet (numberic),noidungnhanxet (nvachar(500))).

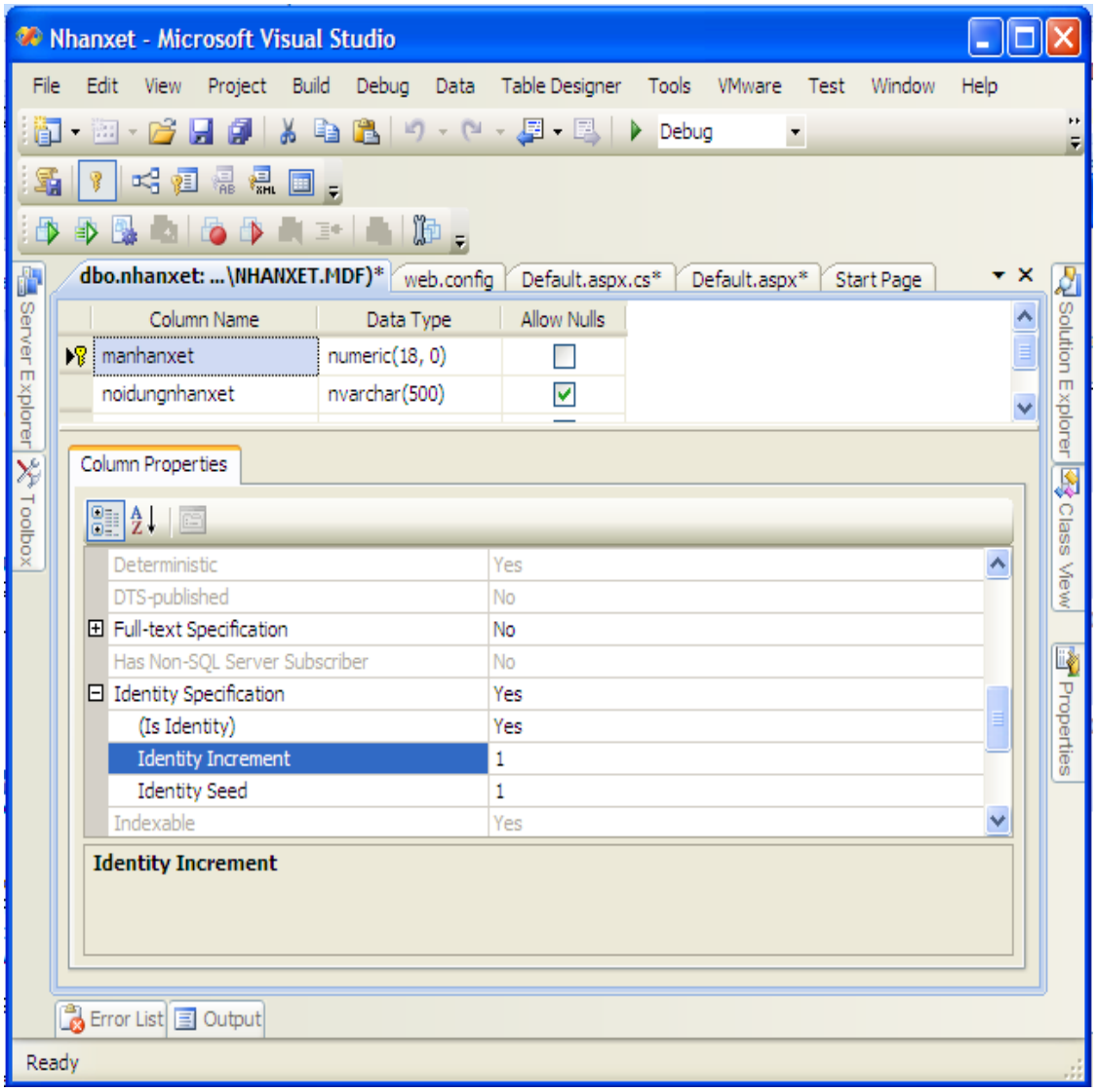
Các bước thực hiện:

Bước 1: Tạo ứng dụng web

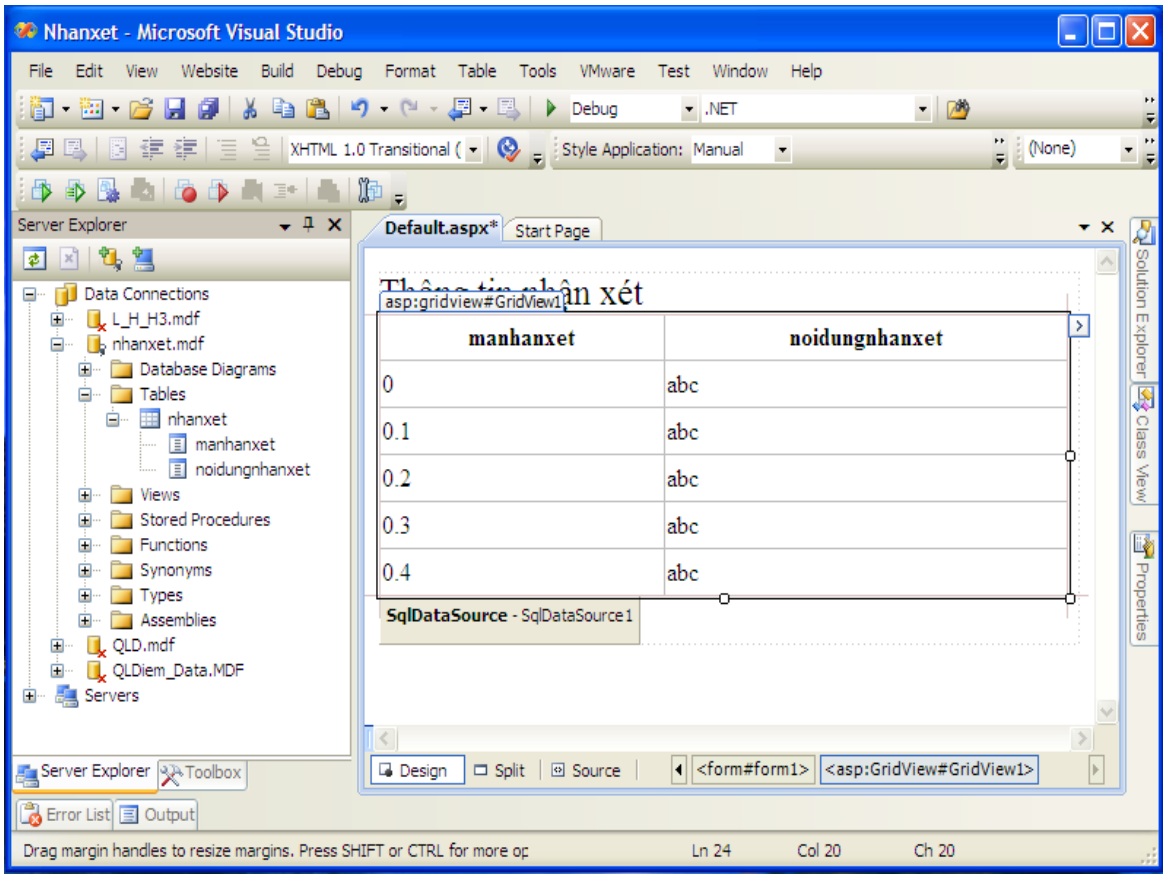
Mở Visual studio 2008 chọn File/New/Web site… Sau đó bạn chọn Language là C#, chọn ứng dụng là ASP.NET web site và đặt tên ứng dụng là nhanxet.

Bước 2: Tạo cơ sở dữ liệu

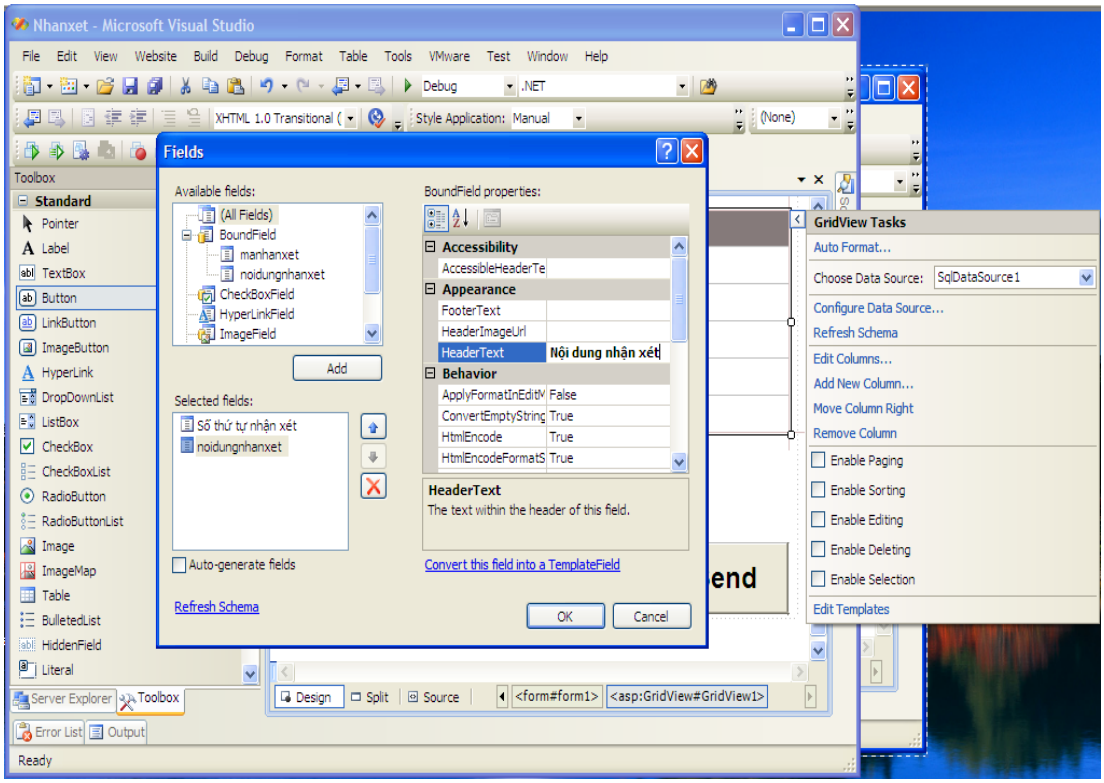
- Chọn Add new item
- Chọn SQL server database và đặt tên cho cơ sở dữ liệu là nhanxet chọn Add sau đó chọn yes.
- Click phải vào table trong data connection của cửa sổ server explorer chọn add new table
- Tạo bảng nhanxet cho trường manhanxet tự động tăng 1 đơn vị khi ta thêm dữ liệu vào bảng:



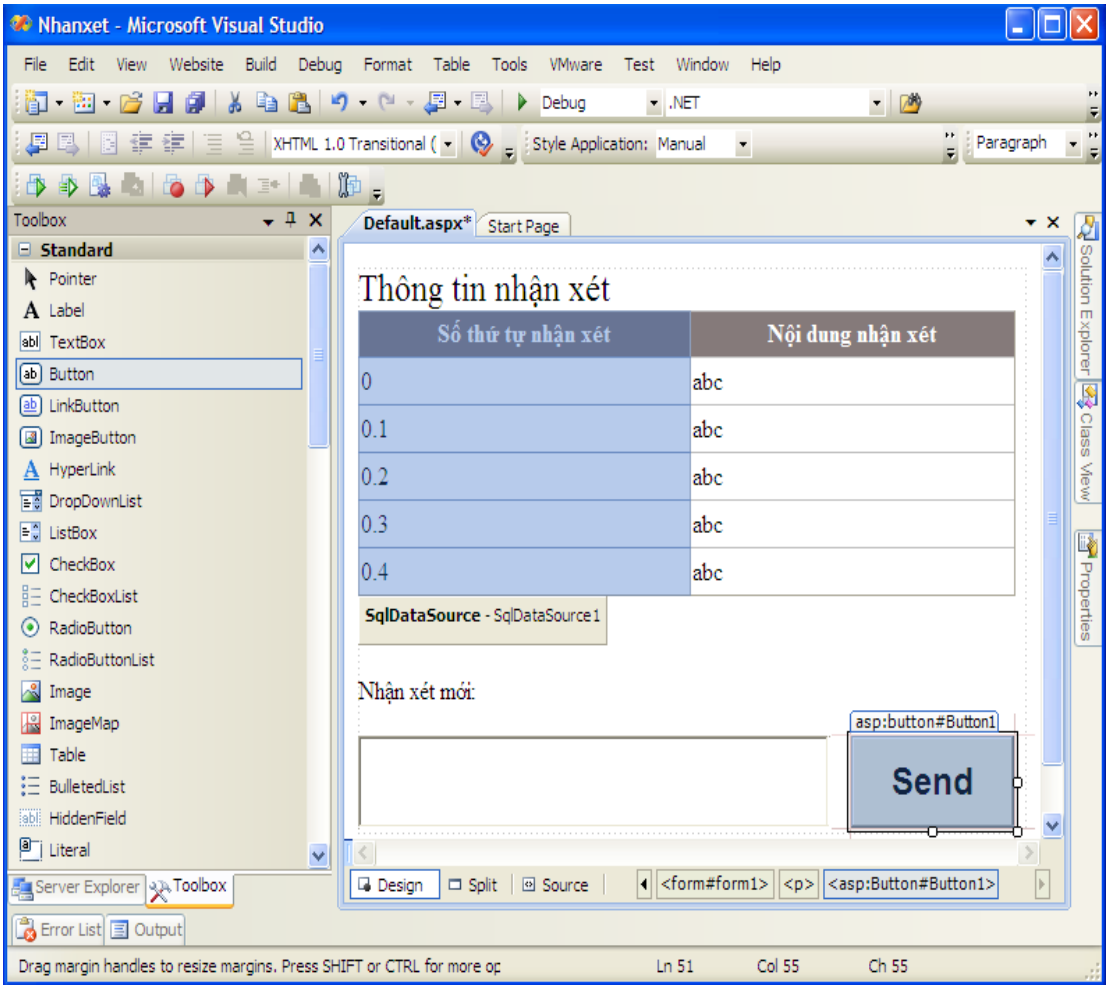
- Để hiển thị nội dung nhận xét chúng ta kéo thẳng bảng nhận xét vào phần Design của trang Default.aspx sau đó tiến hành tô điểm cho Gridview.



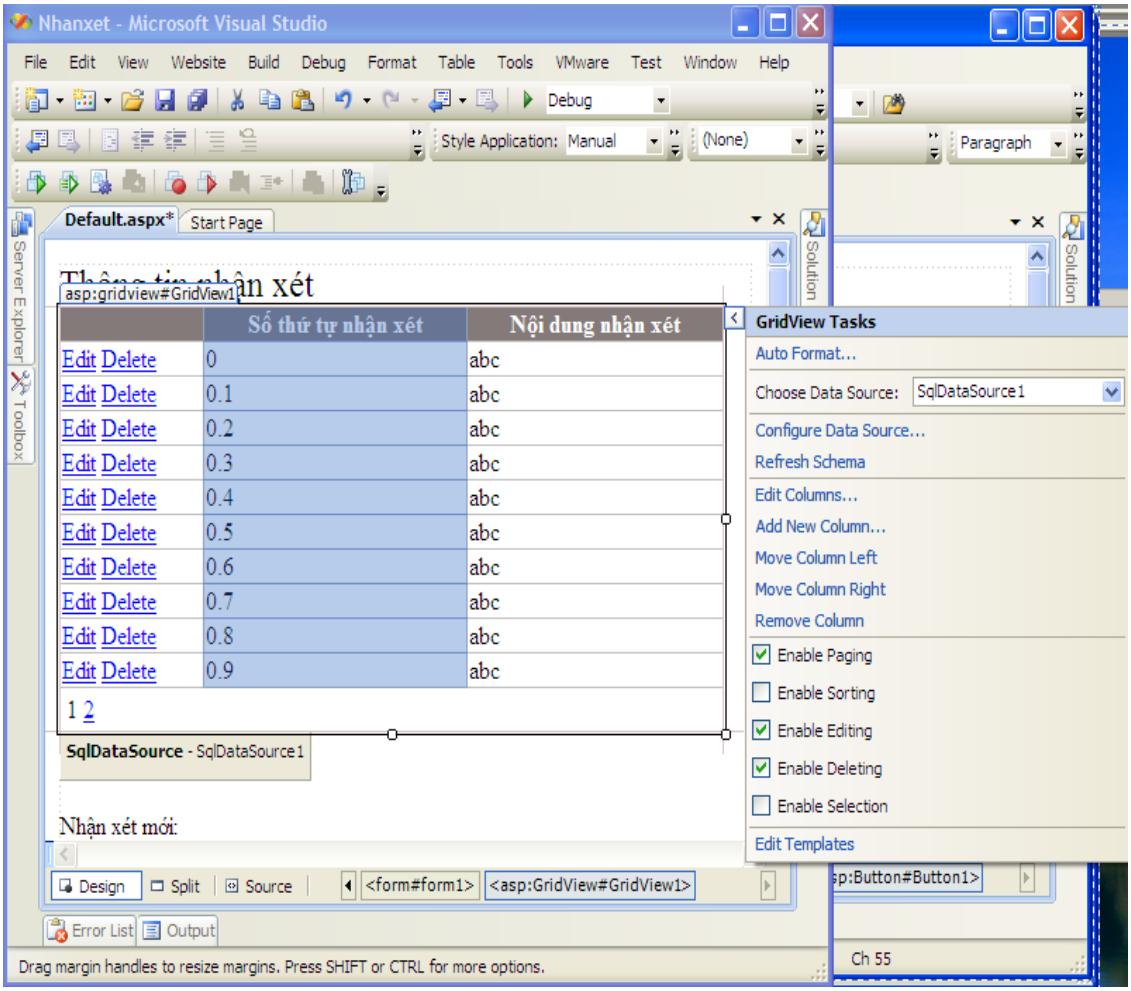
- Tiến hành chỉnh sửa Gridview phần Header:



- Sau khi tiến hành chỉnh sửa ta thêm vào điều khiển TextBox và nút Button như hình vẽ sau:



- Toàn bộ việc thiết kế đã hoàn tất chúng ta có thể thêm điều khiển vào Gridview như chỉnh sửa thông tin và xóa thông tin:



- Chúng ta tiến hành viết code đầu tiên ta click kép vào màn hình và viết sự kiện Page load (khi vào trang web) nội dung như sau:

```
using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.HtmlControls;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Xml.Linq;

public partial class _Default : System.Web.UI.Page
```

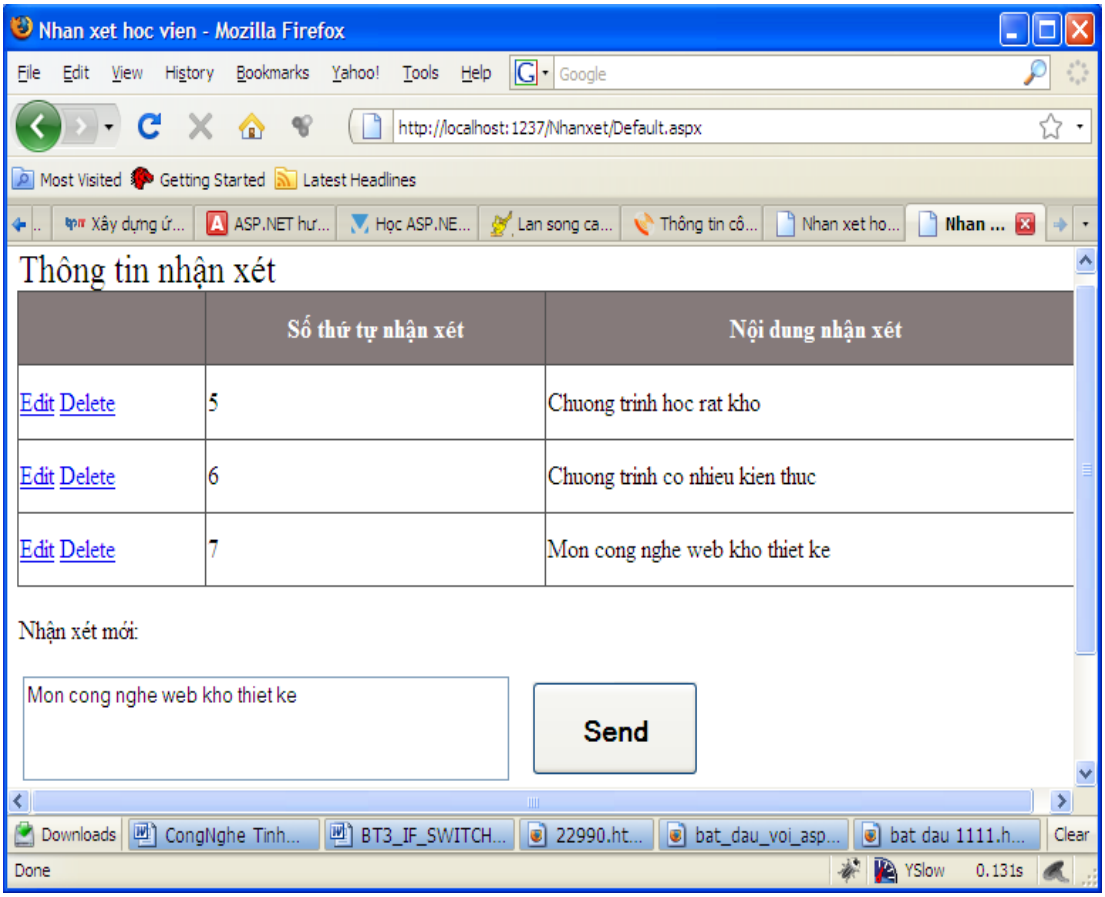
```
{
protected void Page_Load(object sender, EventArgs e)
{
GridView1.DataBind();
}
}
```

- Tiếp theo chúng ta viết thêm sự kiện thêm thông tin:

```
using System;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Data.Sql;
using System.Data.SqlClient;
public partial class _Default : System.Web.UI.Page
{
```

```
protected void Page_Load(object sender, EventArgs e)
{
GridView1.DataBind();
}
protected void Button1_Click(object sender, EventArgs e)
{
string con = ConfigurationManager.ConnectionStrings[
"nhanxetConnectionString1"].ConnectionString;
SqlConnection cn = new SqlConnection(con);
cn.Open();
SqlCommand cmd = new SqlCommand(
"insert into nhanxet(noidungnhanxet) values('" + TextBox1.Text + "')", cn);
cmd.ExecuteNonQuery();
cmd.Dispose();
cn.Close();
GridView1.DataBind();
}
}
```

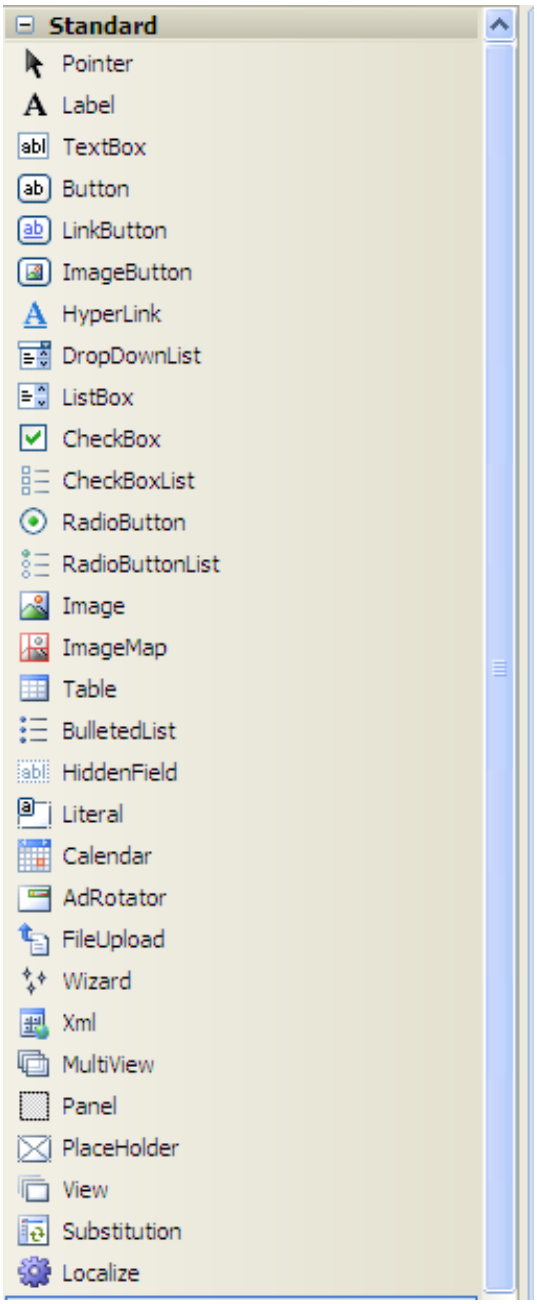
- Chúng ta đã hoàn thành xong trang web tập hợp nhận xét của học viên tham gia môn công nghệ web chúng ta có thể khởi động website bằng phím F5 hoặc nhấn vào nút start.


Nhan xet hoc vien - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://localhost:1237/Nhanxet/Default.aspx
Most Visited Getting Started Latest Headlines
Thông tin nhận xét
Số thứ tự nhận xét
Nội dung nhận xét
Edit Delete
5
Chuong trinh hoc rat kho
Edit Delete
6
Chuong trinh co nhieu kien thuc
Edit Delete
7
Mon cong nghe web kho thiet ke
Nhận xét mới:
Mon cong nghe web kho thiet ke
Send
Done

Các đối tượng điều khiển trong ASP.NET

Điều khiển cơ bản (Standard control)

Bao gồm các điều khiển đưa ra các thành phần chuẩn của form như: Label, Button, TextBox…



Label

Label thường được sử dụng để hiển thị và trình bày nội dung trên trang web. Nội dung được hiển thị trong label được xác định thông qua thuộc tính Text. Thuộc tính Text có thể nhận và hiển thị nội dung với các tag HTML.

Ví dụ:

lblA.Text = "Đây là chuỗi văn bản thường"

lblB.Text = "<B>Còn đây là chuỗi văn bản được in đậm</B>"

Điều khiển Literal

Giống với điều khiển Label bạn có thể dùng Literal để trình bày Text hoặc nội dung Html. Literal hỗ trợ một thuộc tính mà Label không hỗ trợ đó là: thuộc tính Mode gồm 3 giá trị là: Pass through, Endcode, transform.

Ví dụ

```
<asp:Literal ID="lit1" runat="server" Text="<hr/>" Mode="PassThrough" />

<asp:Literal ID="lit2" runat="server" Text="<hr/>" Mode="Encode" />

<asp:Literal ID="lit3" runat="server" Text="<hr/>" Mode="Transform" />
```

Hiên thị:

HyperLink

Điều khiển này được sử dụng để tạo ra các liên kết siêu văn bản.

Các thuộc tính

ImageURL: Qui định hình hiển thị trên điều khiển.

Text: Chuỗi văn bản sẽ được hiển thị trên điều khiển. Trong trường hợp cả 2 thuộc tính

ImageURL và Text được thiết lập, thuộc tính ImageURL được ưu tiên, thuộc tính Text sẽ được hiển thị như Tooltip.

NavigateUrl: Đường dẫn cần liên kết đến.

Target: Xác định cửa sổ sẽ hiển thị cho mối liên kết

_blank: Hiển thị trang liên kết ở một cửa sổ mới.

_self: Hiển thị trang liên kết tại chính cửa sổ chứa liên kết đó.

_parent: Hiển thị trang liên kết ở frame cha.

Ví dụ:

hplASP_net.Text = "Trang chủ ASP.Net"

hplASP_net.ImageUrl = "Hinh\Asp_net.jpg"

hplASP_net.NavigateUrl = "http://www.asp.net"

hplASP_net.Target = "_blank"

Kết quả hiển trị trên trang Web

TextBox

TextBox là điều khiển được dùng để nhập và hiển thị dữ liệu. TextBox thường được sử dụng nhiều với các ứng dụng trên windows form.

Các thuộc tính

Text: Nội dung chứa trong Textbox

TextMode: Qui định chức năng của Textbox, có các giá trị sau:

SingleLine: Hiển thị và nhập liệu 1 dòng văn bản

MultiLine: Hiển thị và nhập liệu nhiều dòng văn bản

Password: Hiển thị dấu * thay cho các ký tự có trong Textbox.

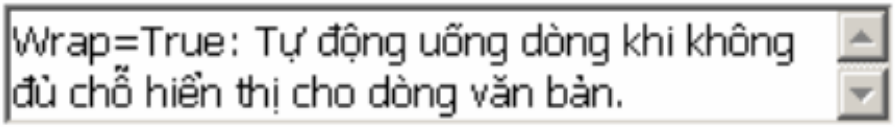
Rows: Trong trường hợp thuộc tính TextMode = MultiLine, thuộc tính Rows sẽ qui định số dòng

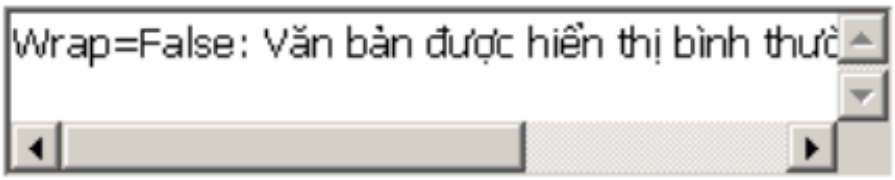
Văn bản được hiển thị.

Maxlength: Qui định số ký tự tối đa được nhập vào cho TextBox

Wrap: Thuộc tính này qui định việc hiển thị của văn bản có được phép tự động xuống dòng khi kích thước ngang của của điều khiển không đủ để hiển thị dòng nội dung văn bản. Giá trị mặc định của thuộc tính này là True - tự động xuống dòng.

Ví dụ:





AutoPostBack: Thuộc tính này qui định điều khiển có được phép tự động PostBack về Server khi nội dung trong Textbox bị thay đổi hay không. Giá trị mặc định của thuộc tính này là False - không tự động Postback.

Image

Điều khiển này được dùng để hiển thị hình ảnh lên trang Web.

Thuộc tính

ImageURL: Đường dẫn đến tập tin hình ảnh cần hiển thị.

AlternateText: Chuỗi văn bản sẽ hiển thị khi tập tin được thiết lập trong thuộc tính ImageURL không tồn tại.

ImageAlign: Vị trí hiển thị giữa hình và nội dung văn bản.

Điều khiển hợp lệ dữ liệu – Validation

Ở Chương trước chúng ta đã học về những điều khiển chuẩn của NetFrameWork3.5, chúng ta có thể dùng những điều khiển đó để thao tác với Webserver phía Server-Side, có thể để cập nhật dữ liệu. Nhưng có một tình huống đặt ra là các điều khiển đó có đảm bảo cho chũng ta cập nhật dữ liệu đúng và không xảy ra lôi?. Ví dụ trên Form của ta có trường nhập số điện thoại nhưng người sử dụng lại nhập vào là 1 dòng text như vậy sẽ gây ra lỗi nhập liệu. Với phiên bản trước của ASP.Net là asp thì để khắc phục lỗi đó chúng ta phải thực hiện viết mã JavaScript để bắt lỗi việc đó, còn với ASPNET nó đã cung cấp cho ta những điều khiển kiểm tra tính hợp lệ của các điều khiển nhập liệu trên Form. Trong chương này các bạn sẽ học về những điều khiển đó và tiếp theo là sẽ học cách mở rộng những điều khiển đó theo ý muốn của chúng ta ví dụ bạn sẽ tạo một AjaxValidator để kiểm tra nhập liệu phía Client.

Điều khiển Validation trong netframework3.5:

RequiredFieldValidator: Yêu cầu người sử dụng nhập giá trị vào trường chỉ định trên Form.

RangeValidator: Kiểm tra giá trị nhập vào có nằm trong một khoảng nhỏ nhất và lớn nhất định trước hay không.

CompareValidator: So sánh giá trị nhập có bằng một giá trị của trường khác trên Form hay không.

RegularExpressionValidator: So sánh giá trị nhập với 1 biểu thức quy tắc nào đấy có thể hòm thư, điện thoại…

CustomValidator: Bạn có thể tuỳ chỉnh đối tượng Validator theo ý của mính.

ValidationSummary: cho phép hiển thị tổng hợp tất cả các lỗi trên 1 trang.

RequiredFieldValidator

Ý nghĩa: với điều khiển này bạn có thể yêu cầu người dùng phải nhập giá trị vào 1 trường chỉ định trên Form.

Cách sử dụng:

Đưa điều khiển RequiredFieldValidator từ ToolBox(trong phần Validation) vào trong Form và thêm vào cho nó 2 thuộc tính

ControlToValidate: chỉ đến điều khiển sẽ được kiểm tra

Text(hoặc ErrorMessage): Thông báo lỗi khi kiểm tra

Ví dụ

Code 1: Trang RequiredValidator.aspx

```
<%@ Page Language="C#" %>

<script runat="server">

    void btnAccept_Click(object sender, EventArgs e)

    {

        if (Page.IsValid)

        {

            this.lblResult.Text = txtHoten.Text;

            this.txtHoten.Text = "";

        }

    }

</script>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title>Validator</title>

</head>

<body>

    <form id="form1" runat="server">

    <div>
```

```
    <asp:Label ID="lblHoten" runat="server" Text="Nhập vào họ tên" />

    <asp:TextBox ID="txtHoten" runat="server"></asp:TextBox>

            <asp:RequiredFieldValidator ID="RequiredFieldtxtHoten"
ControlToValidate="txtHoten" runat="server" Text="* Bạn phải nhập họ
tên"></asp:RequiredFieldValidator><br />

        <asp:Button ID="btnAccept" OnClick="btnAccept_Click" runat="server"
Text="Accept" /><br />

    <asp:Label ID="lblResult" runat="server" Text="" />

  </div>

  </form>

</body>

</html>
```

Điều khiển CompareValidator

Ý nghĩa

Bạn có thể sử dụng CompareValidator để Kiểm tra giá trị nhập vào có nằm trong một khoảng nhỏ nhất và lớn nhất định trước hay không.

Cách sử dụng

Bạn đưa điều khiển CompareValidator từ hộp ToolBox vào Form và thiết lập cho nó một số thuộc tính sau:

ControlToValidate: chỉ đến điều khiển cần kiểm tra

Text(ErrorMessage): Nội dung thông báo lỗi

MinimumValue: Giá trị nhỏ nhất thiết lập cho đối tượng

MaximumValue: Giá trị lớn nhất thiết lập cho đối tượng

Type: Kiểu so sánh, Có thể là các giá trị Interger,String, Double, Date và Currency.

Ví dụ

Code 2: Trang CompareValidator.aspx

```
<%@ Page Language="C#" %>


<script runat="server">

    void btnAccept_Click(object sender,EventArgs e)

    {

        if (Page.IsValid)

        {

            this.lblThongbao.Text = txtDiem.Text;

        }

    }

</script>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

   <title>RangeValidator</title>

</head>

<body>

   <form id="form1" runat="server">

   <div>

    <table>
```

```
<tr>
 <td>Vào điểm</td>
 <td><asp:TextBox ID="txtDiem" runat="server"></asp:TextBox></td>
 <td>
        <asp:RequiredFieldValidator ID="RequiredFieldDiem" ControlToValidate="txtDiem" runat="server" ErrorMessage="Bạn phải nhập điểm"></asp:RequiredFieldValidator>
        <asp:RangeValidator ID="RangeDiem" runat="server" ControlToValidate="txtDiem" Type="Integer" MinimumValue="0" MaximumValue="10" ErrorMessage="Điểm phải nằm trong khoảng từ 0 đến 10"></asp:RangeValidator>
 </td>
</tr>
<tr>
 <td colspan="3">
    <asp:Button ID="btnAccept" OnClick="btnAccept_Click" runat="server" Text="Thực hiện" />
 </td>
</tr>
<tr>
 <td colspan="3">
  <hr />
  <asp:Label ID="lblThongbao" runat="server" />
 </td>
</tr>
```

```
    </table>

   </div>

   </form>

</body>

</html>
```

Trong ví dụ trên ta dùng hai đối tượng Validator cùng kiểm tra giá trị nhập trên điều khiển txtDiem, điều khiển RequiredFieldDiem dùng để kiểm tra và yêu cầu nhập giá trị cho txtDiem còn điều khiển RangeDiem yêu cầu nhập giá trị trong txtDiem phải nằm trong khoảng từ 0 đến 10.

Điều khiển RegularExpressionValidator

ý nghĩa

Điều khiển RegularExpressionValidator cho phép bạn so sánh giá trị nhập tại 1 trường nào đó trên Form với một quy tắc định trước. bạn có thể sử dụng các biểu thức quy tắc để đưa ra các chuỗi mẫu như là email addresses, Social Security numbers, phone numbers, dates, currency, amounts, and product codes.

Cách sử dụng

Bạn đưa điều khiển RegularExpressValidator vào Form của mình và thiết lập cho nó một số thuộc tính sau:

- ID: tên của điều khiển
- ControlToValidate: trỏ đến điều khiển cần kiểm tra
- Text(ErrorMessage): nội dung thông báo khi có lỗi
- ValidatorExpression: quy định mẫu nhập liệu như là hòm thư, số điện thoại…

Ví dụ

Sau đây sẽ là một ví dụ về việc yêu cầu người sử dụng phải cập nhật đúng địa chỉ của hòm thư.

Code 3: trang RegularExpressionValidator.aspx

```
<%@ Page Language="C#"%>


<script runat="server">

    void btnAccept_Click(object sender, EventArgs e)

    {

        if (Page.IsValid)

        {

            lblThongbao.Text = txtEmail.Text;

        }

    }

</script>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

 <html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title>RegularExpressionValidator</title>

</head>

<body>

    <form id="form1" runat="server">

    <div>

        Email:<asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>

                <asp:RequiredFieldValidator ControlToValidate="txtEmail" ID="RequiredFieldValidator1" runat="server" ErrorMessage="Bạn phải nhập địa chỉ hòm thư"></asp:RequiredFieldValidator>
```

```
            <asp:RegularExpressionValidator          ID="RegularExpressionValidator1"
runat="server"

          ErrorMessage="Bạn nhập không đúng định dạng hòm thư"

          ControlToValidate="txtEmail"

                              ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-
.]\w+)*"></asp:RegularExpressionValidator>

      <br />

          <asp:Button    ID="btnAccept"    OnClick="btnAccept_Click"    runat="server"
Text="Accept" />

      <asp:Label ID="lblThongbao" runat="server"></asp:Label>

   </div>

   </form>

</body>

</html>
```

Điều khiển CompareValidator

ý nghĩa

Điều khiển CompareValidator có 3 kiểu khác nhau để kiểm tra giá trị nhập:

Sử dụng để kiểm tra kiểu dữ liệu

Sử dụng để so sánh giá trị nhập với một giá trị cố định

Sử dụng để so sánh gia trị nhập với giá trị của một điều khiển khác trên Form

Cách sử dụng

Bạn đưa điều khiển CompareValidator vào Form và thiết lập cho nó một số thuộc tính sau:

ControlToValidate: điều khiển của Form sẽ được kiểm tra

ControlToCompare: Điều khiển dùng để so sánh giá trị

Text(ErrorMessage): hiển thị nội dung thông báo lỗi kho có lỗi

Type: Kiểu của giá trị sẽ được so sánh

Operator: Toán tử so sánh. Có thể là các giá trị: DataTypeCheck, Equal, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual, NotEqual.

Ví dụ

Ví dụ 4 sau sẽ yêu cầu nhập vào ngày sinh, nếu người sử dụng nhập vào không đúng dữ liệu dạng ngày thì sẽ có lỗi thông báo.

Code 4: Trang CompareValidator.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="CompareValidator.aspx.cs" Inherits="CompareValidator" %>


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title>CompareValidator</title>

</head>

<body>

    <form id="form1" runat="server">

    <div>
```

```
        <asp:Label ID="Label1" runat="server" Text="Ngày sinh"></asp:Label>

        <asp:TextBox ID="txtNgaysinh" runat="server" Width="154px"></asp:TextBox>

        <asp:CompareValidator ID="CompareValidator1" runat="server"

            ControlToValidate="txtNgaysinh"

                ErrorMessage="Sai dữ liệu phải là kiểu ngày tháng"
Operator="DataTypeCheck"

            Type="Date"></asp:CompareValidator>

        <br />

        <asp:Button ID="Button1" runat="server" Text="Accept" />



    </div>

    </form>

</body>

</html>
```

Ví dụ sau đây sẽ hướng đưa ra trường hợp với Form tạo tài khoản trên một trên Web yêu cầu người đăng ký phải nhập mật khẩu 2 lần.

Code 5 trang

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="CompareValidator.aspx.cs" Inherits="CompareValidator" %>



<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">



<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head runat="server">
   <title>CompareValidator</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>


        <asp:Label ID="Label2" runat="server" Text="Nhập mật khẩu "></asp:Label>
                <asp:TextBox ID="txtPass" runat="server" TextMode="Password"></asp:TextBox>
      <br />
      <asp:Label ID="Label3" runat="server" Text="Nhập lại mật khẩu"></asp:Label>
                <asp:TextBox ID="txtrePass" runat="server" TextMode="Password"></asp:TextBox>
      <asp:CompareValidator ID="CompareValidator2" runat="server"
         ControlToCompare="txtPass" ControlToValidate="txtrePass"
            ErrorMessage="Nhập mật khẩu 2 lần phải giống nhau"></asp:CompareValidator>
      <br />
      <asp:Button ID="Button2" runat="server" Text="Accept" />


   </div>
   </form>
```

```
</body>

</html>
```

Điều khiển CustomValidator

ý nghĩa

Nếu những điều khiển Validator trên chưa đủ với bạn hoặc bạn muốn tạo một Validator riêng theo ý mình, bạn có thể dụng điều khiển CustomValidator, bạn có thể kết hợp CustomValidator với một hàm.

Cách sử dụng và Ví dụ

CustomValidator có 3 thuộc tính hay sử dụng là:

ControlToValidator: điểu khiển của Form sẽ được kiểm tra

Text(ErrorMessage): hiển thị nội dung thông báo lỗi kho có lỗi

ClientValidationFunction: tên của một hàm client-side để thực hiện kiểm tra trên client-side

CustomValidator hỗ trợ 1 sự kiện

ServerValidate: Sự kiện được đưa ra khi CustomValidator thực hiện kiểm chứng.

Ví dụ sau sẽ sử dụng sự kiện ServerValidate để kiểm tra độ dài của chuỗi được nhập trong điều khiển TextBox, nếu người nhập, nhập vào chuỗi có độ dài lớn hơn 20 ký tự thì điều khiển CustomValidator sẽ đưa ra thông báo lỗi.

Ví dụ:

Code 6 trang CustomValidator.aspx

```
<%@ Page Language="C#" %>


<script runat="server">
```

```
    void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs e)
    {
        if (e.Value.Length > 20)
            e.IsValid = false;
        else
            e.IsValid = true;
    }
</script>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>CustomValidator</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>


        <asp:Label ID="Label1" runat="server" Text="Ghi chú"></asp:Label>
        <br />
            <asp:TextBox ID="TextBox1" runat="server" Height="95px" TextMode="MultiLine"
```

```
            Width="218px"></asp:TextBox>

        <asp:CustomValidator ID="CustomValidator1" runat="server"

            ErrorMessage="Độ dài ghi chú phải nhỏ hơn 20 ký tự"

                                                        ControlToValidate="TextBox1"
OnServerValidate="CustomValidator1_ServerValidate"></asp:CustomValidator>

        <br />

        <asp:Button ID="Button1" runat="server" Text="Accept" />



    </div>

    </form>

</body>

</html>
```

Ở ví dụ trên trong hàm "CustomValidator1_ServerValidate" Tham số thứ 2 được truyền tới sự kiện ServerValidator để xử lý. Trong thực thể của lớp ServerValidateEventArgs có hai thuộc tính

Value: Giá trị của trường trên Form sẽ được kiểm chứng.

IsValid: Diễn tả việc kiểm chứng cho kết quả thành công hoặc sai.

Trong ví dụ tiếp theo tôi sẽ đưa ra cách sử dụng hàm kiểm chứng Client-side kết hợp với CustomValidator như thế nào, Trang này chỉ kiểm tra độ dài của chuỗi nhập vào bên trong TextBox, nhưng nó sẽ kiểm tra trên cả Server và Client.

Code 7.

```
<%@ Page Language="C#" %>



<script runat="server">
```

```
    void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs e)

    {

        if (e.Value.Length > 20)

            e.IsValid = false;

        else

            e.IsValid = true;

    }

</script>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">



<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title>CustomValidator</title>

    <script language="javascript" type="text/javascript">

     function valComments_ClientValidate(source, args)

     {

        if (args.Value.length > 20)

            args.IsValid = false;

        else

            args.IsValid = true;

     }

    </script>
```

```
</head>

<body>

    <form id="form1" runat="server">

    <div>



        <asp:Label ID="Label1" runat="server" Text="Ghi chú"></asp:Label>

        <br />

            <asp:TextBox ID="TextBox1" runat="server" Height="95px" TextMode="MultiLine"

            Width="218px"></asp:TextBox>

        <asp:CustomValidator ID="CustomValidator1" runat="server"

            ErrorMessage="Độ dài ghi chú phải nhỏ hơn 20 ký tự" ClientValidationFunction="valComments_ClientValidate"

                                                ControlToValidate="TextBox1" OnServerValidate="CustomValidator1_ServerValidate"></asp:CustomValidator>

        <br />

        <asp:Button ID="Button1" runat="server" Text="Accept" />



    </div>

    </form>

</body>

</html>
```

Điều khiển ValidationSummary

ý nghĩa

ValidationSummary cho phép bạn liệt kê tất cả các các lỗi kiểm tra trên trang từ những điều khiển validator vào một vị trí. Điều khiển này đặc biệt tiện ích với Form có độ rộng lớn.

cách sử dụng

Bạn đưa điều khiển ValidationSummary vào Form và thiết lập cho nó một số thuộc tính sau:

DisplayMode: Cho phép bạn chỉ rõ định dạng hiển thị lỗi, nó có thể là các giá trị như BulletList, List, và SingleParagraph.

HeaderText: Cho phép bạn hiển thị tiêu đề tóm tắt cho các lỗi.

ShowMessageBox: Cho hiện thị một popup thông báo

ShowSummary: Cho phép bạn ẩn ValidationSummary trên trang.

ví dụ

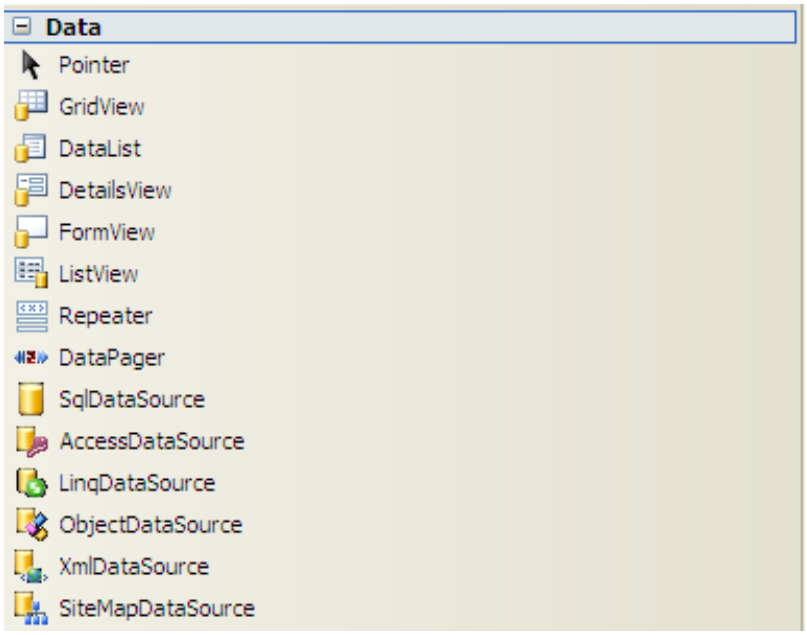
Code 8 Trang ValidationSummary.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" %>


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

   <title>ValidationSummary</title>


</head>
```

```
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ValidationSummary ID="ValSummary" runat="server" />
        <table>
            <tr>
                <td>
                    <asp:Label ID="Label1" runat="server" Text="Họ tên"></asp:Label>
                </td>
                <td>
                    <asp:TextBox ID="txtHoten" runat="server"></asp:TextBox>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server"
                        ErrorMessage="Bạn phải nhập họ tên"
ControlToValidate="txtHoten">*(Yêu cầu)</asp:RequiredFieldValidator>
                </td>
            </tr>
            <tr>
                <td>
                    <asp:Label ID="Label2" runat="server" Text="Email"></asp:Label>
                </td>
                <td>
                    <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>
```

```
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator2"
runat="server"

                     ErrorMessage="Bạn phải nhập hòm thư"
ControlToValidate="txtEmail">*(Yêu cầu)</asp:RequiredFieldValidator>

                <asp:RegularExpressionValidator ID="RegularExpressionValidator1"
runat="server"

                ErrorMessage="Hòm thư bạn nhập không đúng định dạng"

                ControlToValidate="txtEmail"></asp:RegularExpressionValidator>

            </td>

        </tr>

        <tr>

            <td colspan="2">

                <asp:Button ID="Button1" runat="server" Text="Accept" />

            </td>

        </tr>

    </table>

    </div>

  </form>

</body>

</html>
```

Điều khiển làm việc với CSDL (Data Control)



DetailView

Hiển thị dữ liệu với DetailView

DetailView được đưa ra hiển thị như một bảng(<Table>) trong HTML để hiển thị dữ liệu một bản ghi.

Ví dụ: Trang DetailView.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="DetailView.aspx.cs"
Inherits="_DetailView" %>


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

    <title>Detail View</title>
```

```
</head>

<body>

   <form id="form1" runat="server">

   <div id="navcontain">

      <asp:DetailsView ID="DetailsView1"

            DataSourceID="SqlDataSource1"        runat="server"        Height="50px"
Width="125px">

      </asp:DetailsView>

      <asp:SqlDataSource ID="SqlDataSource1"

       ConnectionString="<%$ConnectionStrings:hcubiuData %>"

                SelectCommand="select        *        from        tblIntrodure"
runat="server"></asp:SqlDataSource>

   </div>

   </form>

</body>

</html>
```

Vẫn với cơ sở dữ liệu từ chương trước bạn đưa dữ liệu của bảng tblIntrodure vào SqlDataSource và điền nó vào DetailView1 với thuộc tính DataSourceID của nó

Bạn cũng có thể đưa dữ liệu vào DetailView từ một mảng hay danh sách dữ liệu

Bạn tạo một lớp Employee.cs

```
using System;



public class Employee

{
```

```
private int _PersonID;
public int PersonID
{
    get { return _PersonID; }
    set { _PersonID = value; }
}

private string _Hoten;
public string Hoten
{
    get { return _Hoten; }
    set { _Hoten = value; }
}

private int _Tuoi;
public int Tuoi
{
    get { return _Tuoi; }
    set { _Tuoi = value; }
}
  public Employee()
  {
```

```
    }

    public Employee(int _PersonID, string _Hoten, int _Tuoi)
    {
        this._PersonID = _PersonID;
        this._Hoten = _Hoten;
        this._Tuoi = _Tuoi;
    }
}
```

DetailViewPerson.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="DetailViewPerson.aspx.cs" Inherits="DetailViewPerson" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Detail View</title>
</head>
<body>
    <form id="form1" runat="server">
    <div id="navcontain">
```

```
        <asp:DetailsView ID="DetailsView1" runat="server" Height="50px" Width="125px">

        </asp:DetailsView>

    </div>

    </form>

</body>

</html>
```

DetailViewPerson.aspx.cs

```
using System;

using System.Collections;

using System.Collections.Generic;

using System.Data;


public partial class DetailViewPerson : System.Web.UI.Page

{

    protected void Page_Load(object sender, EventArgs e)

    {

        if (!IsPostBack)

        {

            Employee newEmploy=new Employee(1,"HCUBIU",25);

            List<Employee> listEmploy=new List<Employee>();

            listEmploy.Add(newEmploy);

            DetailsView1.DataSource = listEmploy;
```

```
            DetailsView1.DataBind();
        }
    }
}
```

Trong ví dụ này chúng ta tạo ra một lớp Employee và chúng ta đưa dữ liệu vào DetailView1 với thuộc tính DataSource và phương thức DataBind điền dữ liệu vào.

Sử dụng Fields với điều khiển DetailView

DetailView hỗ trợ tất cả các Field như GridView

BoundField: cho phép bạn hiển thị giá trị của dữ liệu như Text

CheckBoxField: hiển thị dữ liệu dưới dạng một CheckBox

CommandField: hiển thị liên kết cho phép chỉnh sửa, thêm mới, xoá dữ liệu của dòng.

ButtonField: hiển thị dữ liệu như một button(ImageButton,)

HyperLinkField: hiển thị môt liên kết

ImageField: hiển thị ảnh

TemplateFile: cho phép hiển thị các điều khiển tuỳ biến.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="DetailViewfield.aspx.cs" Inherits="DetailViewfield" %>


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

    <title>Fields</title>
```

```
</head>

<body>

   <form id="form1" runat="server">

   <div id="navcontain">

      <asp:DetailsView ID="DetailsView1" AutoGenerateRows="false"

            DataSourceID="SqlDataSource1"        runat="server"        Height="50px"
Width="125px">

       <Fields>

         <asp:BoundField DataField="pkIntrodureID" HeaderText="ID" />

         <asp:BoundField DataField="sTitle" HeaderText="Tiêu đề" />

         <asp:BoundField DataField="iPosition" HeaderText="Vị trí" />

       </Fields>

      </asp:DetailsView>

      <asp:SqlDataSource ID="SqlDataSource1"

       ConnectionString="<%$ConnectionStrings:hcubiuData %>"

                SelectCommand="select          *          from          tblIntrodure"
runat="server"></asp:SqlDataSource>

   </div>

   </form>

</body>

</html>
```

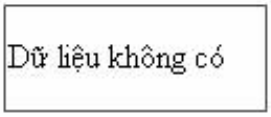
Trong ví dụ trên bạn đưa vào 3 BoundField và điền vào dữ liệu với thuộc tính DataField và thiết đặt cho nó tiêu dề với HeaderText, để đưa ra dữ liệu như thế này bạn cần thiết lập thuộc tính AutoGenerateRows=”false”.

Hiển thị DetailView với dữ liệu rỗng

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="DetailViewDatanull.aspx.cs" Inherits="DetailViewDatanull" %>


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

   <title>Null Data</title>

</head>

<body>

   <form id="form1" runat="server">

   <div id="navcontain">

      <asp:DetailsView ID="DetailsView1"

         DataSourceID="SqlDataSource1" EmptyDataText="Dữ liệu không có" runat="server" Height="50px" Width="125px">

      </asp:DetailsView>

      <asp:SqlDataSource ID="SqlDataSource1"

       ConnectionString="<%$ConnectionStrings:hcubiuData %>"

                SelectCommand="select * from tblProduct" runat="server"></asp:SqlDataSource>

   </div>

   </form>
```

```
</body>

</html>
```

Kết xuất của chương trình



Trong ví dụ trên ta đưa dữ liệu vào DetailView1 với dữ liệu từ bảng tblProduct(chưa được nạp dữ liệu), trong DetailView1 ta thêm vào thuộc tính EmptyDataText="Dữ liệu không có" để khi trong bảng không có dữ liệu chuỗi Text nằm trong thuộc tính EmptyDataText sẽ được đưa ra.

Bạn cũng có thể Customize chuỗi text hiển thị ra khi chưa có nội dung bằng EmptyDataTemple như ví dụ sau:

Ví dụ: DetailViewDatanull.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="DetailViewDatanull.aspx.cs" Inherits="DetailViewDatanull" %>


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

   <title>Null Data</title>

   <style type="text/css">

      .noMatch{background-color:#ffff66;padding:10px;font-family:Arial,Sans-Serif;}

      .noMatch h1{color:red;font-size:16px;font-weight:bold;}

   </style>
```

```
</head>

<body>

    <form id="form1" runat="server">

    <div id="navcontain">

        <asp:DetailsView ID="DetailsView1"

                DataSourceID="SqlDataSource1"        runat="server"        Height="50px"
Width="125px">

         <EmptyDataTemplate>

            <div class="noMatch">

                <h1>No Matching Results!</h1>

                Please select a different record.

            </div>

         </EmptyDataTemplate>

        </asp:DetailsView>

        <asp:SqlDataSource ID="SqlDataSource1"

         ConnectionString="<%$ConnectionStrings:hcubiuData %>"

                    SelectCommand="select        *        from        tblProduct"
runat="server"></asp:SqlDataSource>

    </div>

    </form>

</body>

</html>
```

Sử dụng điều khiển GridView

GridView trình bày dữ liệu như thẻ Table của HTML mà mỗi mục dữ liệu như vói thẻ TR.

Chúng ta cùng đi vào xây dựng một lớp gridViewHelper giúp việc điền dữ liệu vào gridView trong các ví dụ của chúng ta.

Trong chương này ngoài điều khiển ngoài điều khiển GridView các bạn sẽ được giới thiệu thêm về điều khiển sqlDatasource.

Ta đi vào một ví dụ đơn giản: Bạn hiển thị dữ liệu từ bảng Giới thiệu ra 1 GridView

Trong file web.config: bạn thêm vào

```
  <connectionStrings>

      <add name="Gridview" connectionString="Data Source=.\SQLEXPRESS;
AttachDbFilename=|DataDirectory|Database.mdf;Integrated Security=True;user
Instance=True" />

  </connectionStrings>
```

Bạn tạo một trang SimpleGridview.aspx và đưa vào một điều khiển SqlDataSource và điền vào nó các thuộc tính như sau:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="SimpleGridview.aspx.cs" Inherits="_Default" %>


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

   <title>GridView</title>

</head>

<body>
```

```
    <form id="form1" runat="server">

    <div id="navcontain">

        <asp:GridView AllowSorting="true" DataSourceID="SqlDataSource1"

         ID="GridView1" runat="server">

        </asp:GridView>

        <asp:SqlDataSource ConnectionString="<%$ ConnectionStrings:Gridview %>"

            SelectCommand="select * from tblIntrodure" ID="SqlDataSource1"
runat="server"></asp:SqlDataSource>

    </div>

    </form>

</body>

</html>
```

Như bạn thấy trong ví dụ trên đối tượng SqlDatasource chứa chỗi kết nối String được lấy ra từ file web.config và thuộc tính selectCommand sẽ đưa vào một chuỗi sql dạng select để lấy tất cả dữ liệu trong bảng tblIntrodure

Và điều khiển GridView của ta sẽ điền vào thuộc tính DataSourceID=”_tên_sqlDatasource”.

Sorting Data

Bạn có thể trình bày sắp xếp dữ liệu trong GridView với thuộc tính AllowSorting

Ví dụ: cũng với ví dụ 1 bạn thêm vào thuộc tính AllowSorting="true" khi này bạn sẽ thấy trên dòng Header của Gridview sẽ xuất hiện như LinkButton và khi bạn nhấn vào nó, nó cho phép bạn sắp xếp thông tin theo thứ tự giảm dần và tăng dần của dữ liệu

Paging Data

Khi số trường dữ liệu lớn bạn có thể thực hiện phân trang cho dữ liệu với việc thiết đặt thuộc tính AllowPaging="true" cũng với ví dụ trên bạn thêm vào thuộc tính AllowPaging, cho nó giá trị bằng true và thiết lập thuộc tính PageSize(số dòng trên một trang) bằng 3 bạn sẽ thấy sự thay đổi

Bạn có thể chỉnh sửa trình bày xuất hiện phân trang theo ý mình thay vì mặc định nó sẽ trình bày bởi những con số ở cuối của GridView với thuộc tính PagerSetting

Ví dụ bạn thêm vào 1 số thuộc tính cho GridView của chúng ta như sau

```
<asp:GridView AllowSorting="true" PageSize="3"

                PagerSettings-Mode="NextPreviousFirstLast"          PagerSettings-
Position="TopAndBottom" PagerStyle-HorizontalAlign="Center"

      AllowPaging="true" DataSourceID="SqlDataSource1"

       ID="GridView1" runat="server">

      </asp:GridView>
```

Lớp PagingSetting hỗ trợ các thuộc tính sau:

FirtPageImageURL: cho phép hiển thị ảnh của liên kết tới trang đầu tiên

FirstPageText: Cho phép hiển thị Text của liên kết đến trang đầu tiên

LastPageImageUrl: cho phép hiển thị ảnh của liên kết tới trang cuối cùng.

LastPageTex: Cho phép hiển thị Text của liên kết đến trang cuối cùng.

Mode: cho phép bạn lựa chọn hiển thị kiểu cho giao diện phân trang, nó có thể có các giá trị sau:

NextPrevious, NextPreviousFirstLast, Numeric, and NumericFirstLast.

NextPageImageUrl: Cho phép hiển thị ảnh liên kết tới trang tiếp theo.

NextPageText: Text hiển thị cho liên kết đến trang tiếp theo .

PageButtonCount: hiển thị tổng số trang.

Position: chỉ định vị trí hiển thị phân trang. Giá trị của nó có thể là: Bottom, Top, and TopAndBottom.

PreviousPageImageUrl: ảnh hiển thị cho liên kết tới trang trước đó.

PreviousPageText: Text hiển thị cho liên kết tới trang trước đó.

Visible: Cho phép hiển thị hay ẩn giao diện phân trang.

Ví dụ tiếp theo chúng ta cùng customize phân trang 1 GridView với PagerTemplate GridView như sau:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="GridViewpage.aspx.cs" Inherits="GridViewpage" %>



<script runat="server">

    protected void GridView1_DataBound(object sender, EventArgs e)

    {

        Menu menuPager = (Menu)this.GridView1.BottomPagerRow.FindControl("menuPager");

        for (int i = 0; i < GridView1.PageCount; i++)

        {

            MenuItem item = new MenuItem();

            item.Text = Convert.ToString(i+1);

            item.Value = i.ToString();

            if (GridView1.PageIndex == i)

                item.Selected = true;

            menuPager.Items.Add(item);

            menuPager.DataBind();

        }

    }



    protected void menuPager_MenuItemClick(object sender, MenuEventArgs e)
```

```
    {
        GridView1.PageIndex = Int32.Parse(e.Item.Value);
    }
</script>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Gridview</title>
    <style type="text/css">
        .menu td{padding:5px 0px;}
        .selectedPage a{font-weight:bold;color:red;}
    </style>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:GridView ID="GridView1" AllowPaging="true" PageSize="3"
            DataSourceID="SqlDataSource1"     OnDataBound="GridView1_DataBound" runat="server">
         <PagerTemplate>
            <table>
```

```
            <tr>

                <td>

                        <asp:LinkButton id="lnkPrevious" Text="&lt; Prev"
CommandName="Page" CommandArgument="Prev" ToolTip="Previous Page"
Runat="server" />

                </td>

                <td>

                        <asp:Menu id="menuPager" Orientation="Horizontal"
OnMenuItemClick="menuPager_MenuItemClick" StaticSelectedStyle-
CssClass="selectedPage" CssClass="menu" Runat="server" />

                </td>

                <td>

                        <asp:LinkButton id="lnkNext" Text="Next &gt;"
CommandName="Page" CommandArgument="Next" ToolTip="Next Page"
Runat="server" />

                </td>

            </tr>

          </table>

        </PagerTemplate>

       </asp:GridView>

                        <asp:SqlDataSource ID="SqlDataSource1"
ConnectionString="<%$ConnectionStrings:Gridview %>"

                SelectCommand="select * from tblIntrodure"
runat="server"></asp:SqlDataSource>

    </div>

    </form>
```

```
</body>

</html>
```

Ở đây trong PagerTemple bạn thêm vào 2 điều khiển Linkbutton và 1 điều khiển Menu để thực hiện phân trang. 2 điều khiển Linkbutton với các thuộc tính Command và CommandArgument được GridView hỗ trợ lên ta không phải viết các phương thức để thực thi còn với điều menu trong sự kiện DataBound của GridView bạn cung cấp một phương thức GridView1_DataBound để điền dữ liệu cho Menu.

Thay đổi dữ liệu trong GridView

Điều khiển GridView chỉ cho phép bạn thay đổi hoặc xoá dữ liệu trong Database được điền vào nó.

Ví dụ sau sẽ hướng dẫn bạn cách chỉnh sửa dữ liệu và xoá dữ liệu trong điều khiển GridView.

Ví dụ trang GridviewEdit.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="GridviewEdit.aspx.cs" Inherits="_Default" %>



<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">



<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

   <title>GridView</title>

</head>

<body>

   <form id="form1" runat="server">

   <div id="navcontain">
```

```
<asp:GridView AllowSorting="true" PageSize="10"

PagerSettings-Mode="NextPreviousFirstLast" PagerSettings-Position="TopAndBottom" PagerStyle-HorizontalAlign="Center"

AutoGenerateDeleteButton="true"

AutoGenerateEditButton="true"

DataKeyNames="pkIntrodureID"

AllowPaging="true" DataSourceID="SqlDataSource1"

ID="GridView1" runat="server">

</asp:GridView>

<asp:SqlDataSource ConnectionString="<%$ ConnectionStrings:Gridview %>"

SelectCommand="select pkIntrodureID,sTitle,sSummary,sContent,iPosition from tblIntrodure"

UpdateCommand="Update tblIntrodure set sTitle=@sTitle, sSummary=@sSummary, sContent=@sContent,iPosition=@iPosition where pkIntrodureID=@pkIntrodureID"

DeleteCommand="Delete from tblIntrodure where pkIntrodureID=@pkIntrodureID"

ID="SqlDataSource1" runat="server"></asp:SqlDataSource>

</div>

</form>

</body>

</html>
```

Khi nhấn vào nút Edit bạn sẽ thấy các TextBox sẽ hiện ra tương ứng với dòng được nhấn và chúng ta có thể thay đổi dữ liệu trong đó để xác nhận thay đổi dữ liệu bạn nhấn Update.

Chú ý rằng GridView sẽ tự động đưa ra CheckBox nếu có trường trong bảng dữ liệu là Boolean. để thay đổi hay xoá dữ liệu bạn phải thiết lập thuộc tính DataKeyNames với giá trị là khoá chính trong bảng cơ sở dữ liệu của bạn.

Hiển thị dữ liệu trống:

GridView bao gồm hai thuộc tính cho phép bạn hiển thị nội dung cho GridView khi không có dữ liệu, bạn có thể sử dụng EmptyDataText hoặc thuộc tính EmptyDataTemplate để hiển thị nội dung khi dữ liệu rỗng.

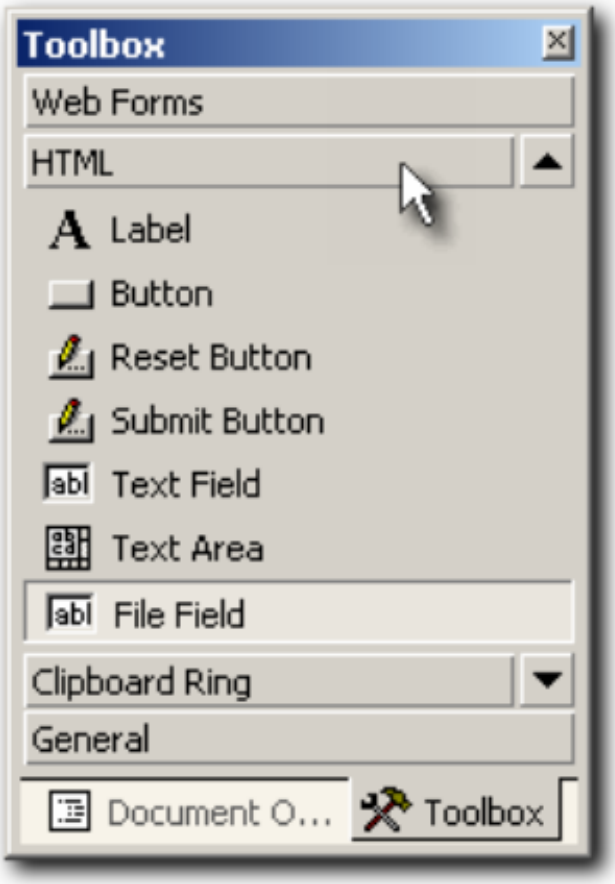
Ví dụ trang GridviewdataNull.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="GridviewdataNull.aspx.cs" Inherits="_Default" %>


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

   <title>GridView</title>

</head>

<body>

   <form id="form1" runat="server">

   <div id="navcontain">

      <asp:GridView AllowSorting="true" PageSize="10"

                  PagerSettings-Mode="NextPreviousFirstLast" PagerSettings-Position="TopAndBottom" PagerStyle-HorizontalAlign="Center"

      EmptyDataText="trong bảng không có dữ liệu"

      DataKeyNames="pkIntrodureID"
```

```
        AllowPaging="true" DataSourceID="SqlDataSource1"
         ID="GridView1" runat="server">
        </asp:GridView>
        <asp:SqlDataSource ConnectionString="<%$ ConnectionStrings:Gridview %>"
         SelectCommand="select * from tblHello"
         ID="SqlDataSource1" runat="server"></asp:SqlDataSource>
    </div>
    </form>
</body>
</html>
```

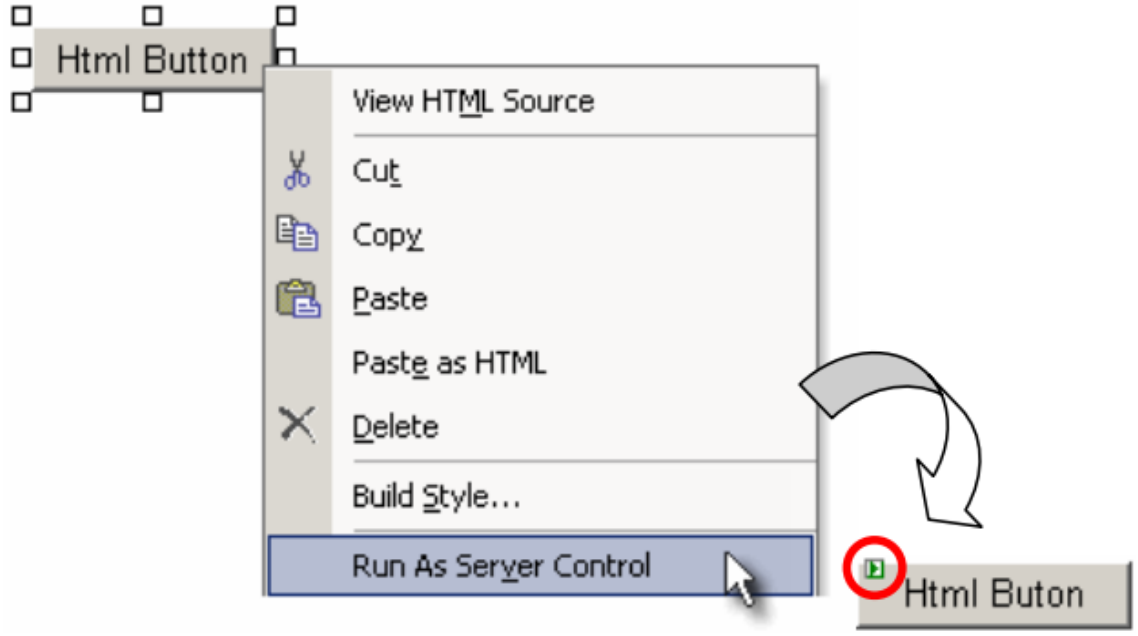
Điều khiển HTML

Điều khiển HTML (tag HTML) trong trang ASP.Net có thể xem như những chuỗi văn bản bình thường. Để có thể được sử dụng lập trình ở phía Server, ta gán thuộc tính runat="Server" cho các điều khiển HTML đó. Những điều khiển HTML (tag HTML) có thuộc tính runat="Server" được gọi là HTML Server Control.



Các điều khiển HTML trên thanh công cụ

Để chuyển các điều khiển HTML thành điều khiển HTML Server, ta chọn Run As Server Control từ thực đơn ngữ cảnh.



Chuyển điều khiển HTML thành điều khiển HTML Server

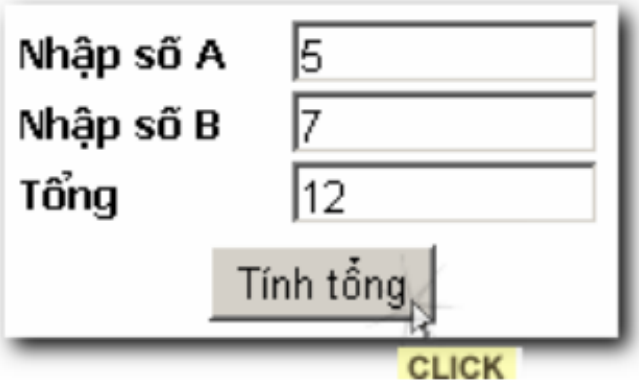
Ví dụ: Các điều khiển HTML: Label, Textbox, Button

Xử lý sự kiện:

Private Sub butTong_ServerClick(…) …

txtTong.Value = Val(txtA.Value) + Val(txtB.Value)

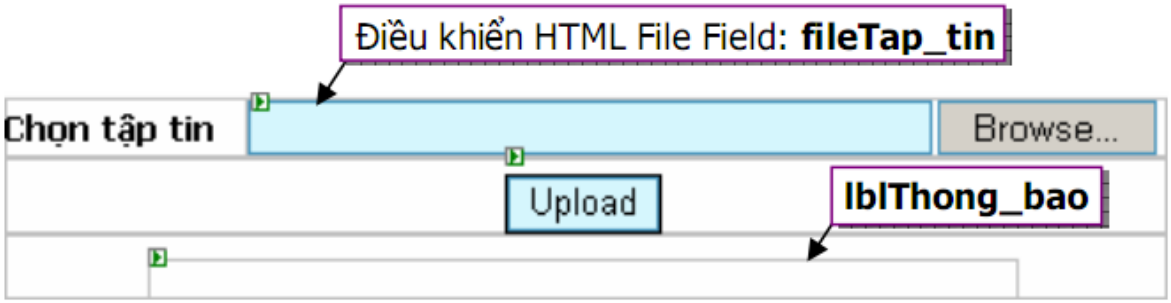
End Sub



Khi thi hành ứng dụng

Ví dụ: Upload file với điều khiển HTML File Field Trong ví dụ sau, chúng ta sẽ thực hiện Upload tập tin lên server, cụ thể hơn, tập tin vừa Upload sẽ được lưu trong thư mục Upload.

Chú ý: Để chép được tập tin lên thư mục Upload, bạn cần phải cấp quyền cho phép ghi trên thư mục Upload



Màn hình ở chế độ thiết kế

Xử lý sự kiện:

Private Sub butUpload_ServerClick(…, e …) …

Dim sTap_tin As String

Dim sTen_file As String

sTap_tin = fileTap_tin.PostedFile.FileName

'Phân tích đường dẫn tập tin để lấy tên tập tin

```
sTen_file = sTap_tin.Substring(sTap_tin.LastIndexOf("\") + 1,
sTap_tin.Length - sTap_tin.LastIndexOf("\") + 1))
'Thực hiện chép tập tin lên thư mục Upload
fileTap_tin.PostedFile.SaveAs(Server.MapPath("Upload\") & sTen_file)
lblThong_bao.InnerHtml = "<B>Thông báo: Bạn đã upload file thành
công</B>"
End Sub
```

Các điều khiển khác

Sử dụng ListControl

Trong chương này các bạn sẽ được học các điều khiển trình bày danh sách như DropDownList, RadioButtonList… và kết thúc chương các bạn sẽ được học 1 cách chi tiết để sử dụng các List Control này tạo một Module bình chọn cho trang web của bạn

Điểm chung cho tất cả các điều khiển danh sách là nó gồm 3 thuộc tính chính

Bạn có thể đưa dữ liệu vào DropDownList từ một mảng danh sách hoặc dữ liệu từ một cơ sở dữ liệu:

Thuộc tính quan trọng

DataSource: chỉ đến nguồn dữ liệu

DataTextField: trường dữ liệu được hiển thị

DataValueField: trường dữ liệu thiết lập giá trị với tương ứng với Text hiển thị

Phương thức OnSelectedIndexChanged

Xảy ra khi người dùng thay đổi lựa chọn phần tử trên DropDownList

Điều khiển DropdownList

Cho phép hiển thị một danh sách các lựa chọn, người sử dụng chỉ chọn một lựa chọn 1 lần

Ví dụ:

Bạn tạo một lớp phục vụ đưa dữ liệu vào DropDownList như sau:

để sử dụng lớp này bạn tạo 1 trang aspx và trong phần code behind bạn nhập khẩu gói Website.Library, trong trong sự kiện Load của trang bạn gọi như sau

```
protected void Page_Load(object sender, EventArgs e)

{

    DropdownListHelper.Fillcombobox(DropDownList1, "tblIntrodure", "sTitle", "pkIntrodureID");
```

```
    }
```

Để sử dụng sự kiện OnSelectedIndexchanged bạn cần thêm vào cho DropDownList thuộc tính AutoPostBack và thiết lập cho nó giá trị là true

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="dropdownlist.aspx.cs" Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

    <title>DropDownList</title>

</head>

<body>

    <form id="form1" runat="server">

    <div>

                <asp:DropDownList AutoPostBack="true" OnSelectedIndexChanged="DropDownList1_Changed" ID="DropDownList1" runat="server">

        </asp:DropDownList><hr />

        <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>

    </div>

    </form>

</body>

</html>
```

Trang dropdownlist.aspx.cs

Code 9.3

```
using System;

using Website.Library;



public partial class _Default : System.Web.UI.Page

{

   protected void Page_Load(object sender, EventArgs e)

   {

      if (!IsPostBack)

      {

         DropdownListHelper.Fillcombobox(DropDownList1,  "tblIntrodure",  "sTitle",
"pkIntrodureID");

      }

   }



   protected void DropDownList1_Changed(object sender, EventArgs e)

   {

      Label1.Text = "Text:" + DropDownList1.SelectedItem.Text + "<br>giá tri:" +
DropDownList1.SelectedValue.ToString();

   }

}
```

Sử dụng điều khiển RadiobuttonList

Điều khiển RadioButtonList cho phép hiển thị một danh sách các RadioButton mà có thể sắp xếp theo hướng ngang hay dọc, để ngừơi sử dụng có thể chọn một trong các Radiobutton đó.

Ví dụ: khi chúng ta cần thăm dò ý kiến khách hàng về một vấn đề gì đó chúng ta cần tạo một module bình chọn cho website của chúng ta.

Chúng ta sẽ thiết lập 1 bảng sau

bảng tblSurveyAnswer

pkAnswerID (int)

sContent (nvarchar(100))

iVote (int)

iPosition (int)

Chúng ta sẽ tạo một trang radiobuttonlist.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="radiobuttonlist.aspx.cs" Inherits="radiobuttonlist" %>


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

   <title>RadioButtonList</title>

   <style type="Text/css">

      body{background-color:#e5e5e5}

            #navcontain{width:399px;Height:299px;Background-color:white;margin:0px auto; padding:15px 15px 15px 15px;}

      A:link{COLOR: #31659C; TEXT-DECORATION: none;}

      A:visited{COLOR: #31659C; TEXT-DECORATION: none;}
```

```
        A:active{COLOR: #FC8800; TEXT-DECORATION: none;}
        A:hover{COLOR: #FC8800; TEXT-DECORATION: none;}
    </style>
</head>
<body>
    <form id="form1" runat="server">
    <div id="navcontain">
      <b>Bạn biết đến Điễn đàn Sinh viên Hoa Sen qua:</b><br />
       <asp:RadioButtonList ID="RadioButtonList1" runat="server">
       </asp:RadioButtonList><hr />
         <asp:LinkButton ID="lbnVote" OnClick="lbnVote_Click" Text="Vote"
runat="server" />
      <hr />
      Bạn chọn: <asp:Label runat="server" ID="lblResult" />
    </div>
    </form>
</body>
</html>
```

Trang radiobuttonlist.aspx.cs

```
using System;
using Website.Library;


public partial class radiobuttonlist : System.Web.UI.Page
```

```
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            ListControlHelper.fillRadioButtonList(RadioButtonList1,  "tblSurveyAnswer",
"sContent", "pkAnswerID");
        }
    }

    protected void lbnVote_Click(object sender, EventArgs e)
    {
        this.lblResult.Text = RadioButtonList1.SelectedItem.Text + "<br> và giá trị của nó
là:" + RadioButtonList1.SelectedItem.Value;
    }
}
```

Bạn thấy ở Code 9.6 lớp radiobuttonlist.aspx.cs cớ nhập khẩu gói Website.Library có sử dụng phương thức fillRadioButtonList từ lớp ListControlHelper với 4 đối số tương ứng như code ở cuối chương trình

Sử dụng điều khiển ListBox

Nó là một điều khiển giống với DropDownList nhưng nó sẽ hiển thị một danh sách trên trang và chúng ta có thể lựa chọn nhiều phần tử một lúc với thuộc tính selectionMode với hai giá trị là Singer và Multiple.

Ví dụ sau mình sẽ đưa ra với một ListBox nhiều lựa chọn

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="ListBox.aspx.cs"
Inherits="ListBox" %>
```

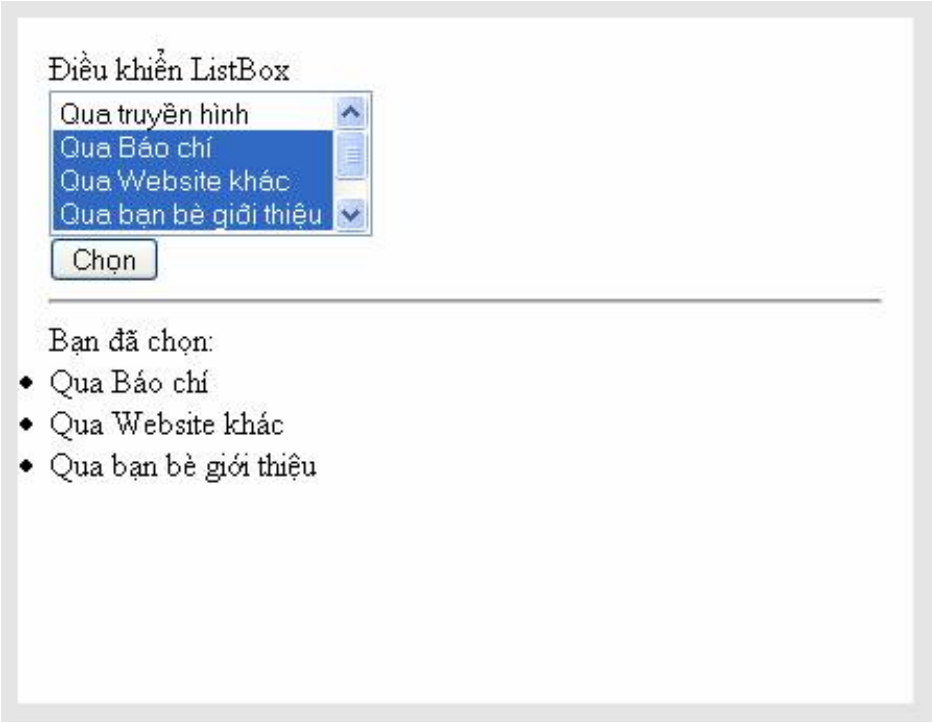
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

   <title>Untitled Page</title>

</head>

<body>

   <form id="form1" runat="server">

   <div id="navcontain">

      Điều khiển ListBox<br />

      <asp:ListBox SelectionMode="Multiple" ID="listbox1" runat="server" /><br />

            <asp:Button ID="btnChon" runat="server" Text="Chọn" OnClick="btnChon_Click" /><hr />

      Bạn đã chọn: <asp:Label ID="lblresult" runat="server" />

   </div>

   </form>

</body>

</html>
```

ListBox.aspx.cs

```
using System;

using System.Collections;
```

```
using System.Web;

using System.Web.UI.WebControls;

using Website.Library;


public partial class ListBox : System.Web.UI.Page

{

    protected void Page_Load(object sender, EventArgs e)

    {

        if(!IsPostBack)

                ListControlHelper.fillListBox(listbox1,     "tblSurveyAnswer",     "sContent",
"pkAnswerID");

    }


    protected void btnChon_Click(object sender, EventArgs e)

    {

        lblresult.Text = "";

        foreach (ListItem item in listbox1.Items)

        {

            if (item.Selected)

                lblresult.Text += "<li>" + item.Text;

        }

    }

}
```

Kết xuất của chương trình



Các đối tượng và biến trong ASP.NET

Respone-Requesy-Server-Cookie

Trong bất kỳ ứng dụng nào, dù là winform based hay webform based thì việc giao tiếp (tương tác) với người dùng và giữa các webform với nhau là điều bắt buộc. Ví dụ ta cần phải lấy thông tin đặt hàng do người dùng nhập vào và hiển thị trở lại người dùng một số thông tin hữu ích khác, như kết quả thanh toán…hay một trang chuyển tiếp kết quả cho một trang khác để xử lý v.v…

Ở các bài trước, để làm điều này chúng ta thực hiện dễ dàng thông qua các server controls như textbox, listbox, dropdownlist, label,… Tuy nhiên những điều khiển này chỉ có tác dụng trong một Page còn các trang khác thì hoàn toàn không thể đọc/ghi giá trị nằm trong các điều khiển này.

Để thực hiện việc giao tiếp (truyền dữ liệu) giữa các webform ASP.NET cung cấp một tập các điều khiển giúp ta làm việc đó một cách dễ dàng, đó là: Đối tượng Request và đối tượng Response.

Trong bài học này, chúng ta cũng tìm hiểu thêm một số đối tượng khác cũng rất hay dùng khi xây dựng ứng dụng là đối tượng Server, Application và Session.

Respone

Đối tượng Response dùng để làm gì ?

Đối tượng này được dùng để gửi nội dung (một xâu) bất kỳ về cho trình duyệt.

Các thành phần (thuộc tính và phương thức) chính

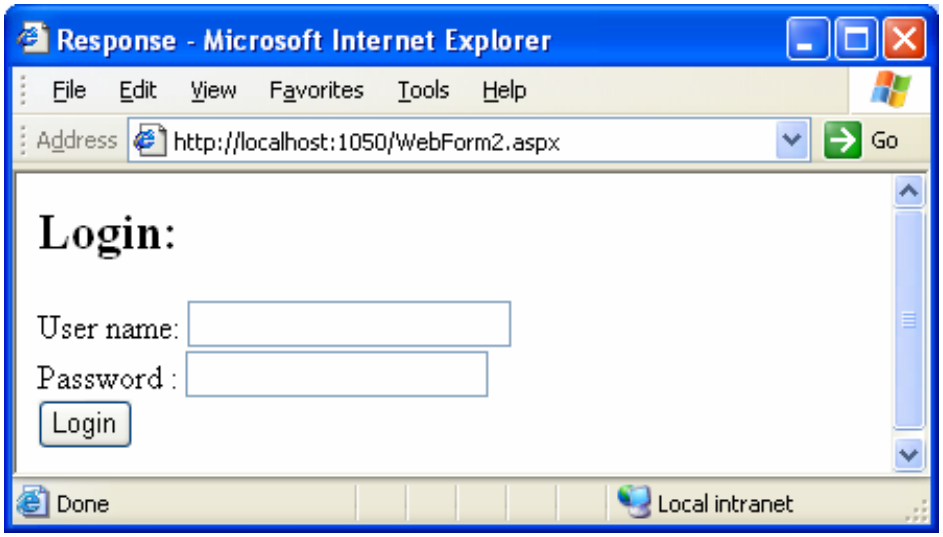
Phương thức: Response.write(<Biểu thức>) dùng để gửi giá trị biểu thức truyền vào cho phía trình duyệt.

Phương thức: Flush dùng để đưa dữ liệu còn trong bộ đệm phía server về cho phía trình duyệt.

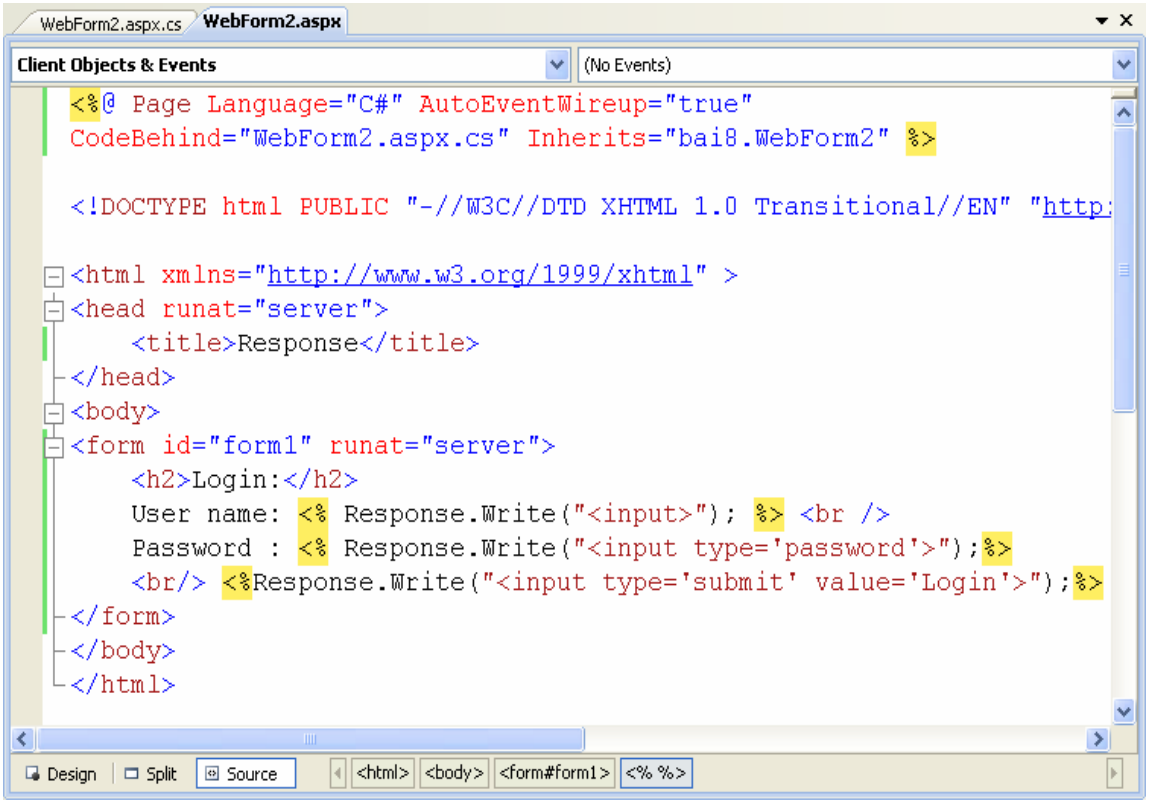
Phương thức Response.Redirect("địa chỉ URL"): Chuyển tới một trang khác.

Ví dụ sử dụng

Tạo một trang Login hoàn toàn bằng phương thức Response.write như sau:



Trang code sẽ như sau:



Request

Đối tượng Request dùng để làm gì ?

Request là một đối tượng của ASP.NET, nó cho phép đọc các thông tin do các trang khác gửi (Submit) đến.

Các thành phần (thuộc tính và phương thức) chính

Phương thức:

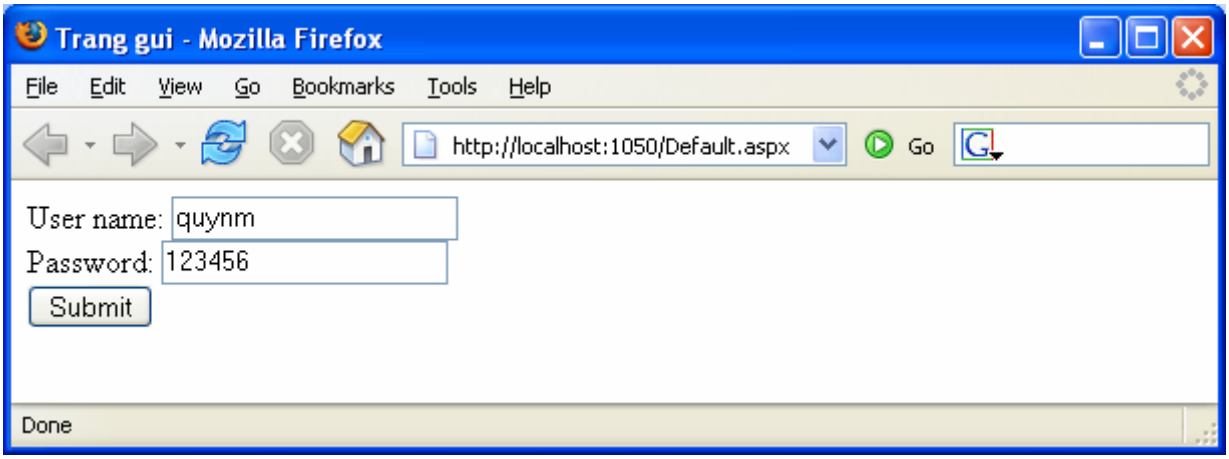
Request.QueryString.Get("Tên_Phần tử cần đọc"): Để đọc giá trị của một phần tử được gửi theo phương thức Get (Method = "Get")

Phương thức Request.Form.Get("Tên_Phần tử cần đọc"): Để đọc giá trị của một phần tử được gửi theo phương thức Post (Method = "Post").

Chúý: Có thể dùng Request.Form.GetValues và Request.Form.GetValues để đọc.

Ví dụ sử dụng

Xây dựng 2 trang web : trang Default.aspx, trong đó có 2 textbox chứa tên và mật khẩu. Khi người dùng click vào nút submit thì gửi tên và mật khẩu sang trang Webform1.aspx để hiển thị.



Trang nguồn (gửi): Default.aspx

Code của 2 trang sẽ như sau:

Kết quả nhận về.

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="Default.aspx.cs" Inherits="bai8._Default" %>
<head runat="server"><title>Trang gửi</title></head>
<body>
    <form id="form1" runat="server" method="Post">
        User name:
        <asp:TextBox runat="server" ID="txtUserName">
        </asp:TextBox><br />

        Password:
        <asp:TextBox runat="server" ID="txtPassword" >
        </asp:TextBox><br />

        <asp:Button runat="server" ID="cmdSubmit"
        Text="Submit" PostBackUrl="~/WebForm1.aspx" />
    </form>
</body>
</html>
```

Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="bai8.WebForm1" %>
<head runat="server">
    <title>Trang nhận</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:Label runat="server" ID="lblKetQua">
        </asp:Label>
    </form>
</body>
</html>
```

Webform1.aspx

```
using System;
namespace bai8
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArg
        {
            lblKetQua.Text = "Username: " +
                 Request.Form.Get("txtUserName") +
                                "<br>Password: " +
                 Request.Form.Get("txtPassword");
        }
    }
}
```

Code xử lý của trang webform1.aspx.cs

Server

Đối tượng Server dùng để làm gì ?

- Dùng để tạo các đối tượng COM
- Lấy thông tin về tên máy
- Ánh xạ đường dẫn ảo thành đường dẫn vật lý.

Các thành phần (thuộc tính và phương thức) chính

CreateObject("COM Specification") ∈ Ít dùng trong ứng dụng .NET

MachineName: String; Trả về tên của máy tính server đang chạy.

Mappath("Virtual path"): Trả về đường dẫn vật lý của đường dẫn ảo tương ứng.

Ví dụ sử dụng

In ra tên của máy chủ hiện hành: Response.Write(Server.MachineName);

Cho biết đường dẫn thực sự trên ổ cứng (thư mục vật lý) của trang hiện hành (trang default.aspx) : **Server.Mappath("default.aspx");**

Cho biết đường dẫn vật lý ứng với tệp QLCB.Mdb, biết rằng tệp này nằm trong một thư mục con là "App_Data": **Server.Mappath("App_Data/QLDB.MDB");**

Application

Đối tượng Application dùng để làm gì ?

Dùng để quản lý các biến có phạm vi toàn ứng dụng. Có tác dụng đến mọi người dùng.

Khái niệm biến toàn ứng dụng

Biến toàn ứng dụng là biến có tác dụng đối với mọi người dùng truy cập vào website. Mọi trang aspx.cs đều có thể truy cập đến biến này và dù ở bất kỳ thời điểm nào.

Đối tượng Application

Dùng để quản lý (Tạo, đọc, ghi) các biến có phạm vi toàn ứng dụng.

Cú pháp tạo biến Application:

Application.Add("Tên_Biến", <Giá trị khởi tạo>);

Ví dụ: Tạo biến So_Nguoi_Truy_Cap

Application.Add("So_Nguoi_Truy_Cap", 0)

Truy xuất đến biến Application:

Application.Contents["Tên_Biến"] hoặc chỉ số: Application.Contents[i]

Ví dụ : Đọc và ghi biến Application.

Application.Contents["SoNguoiTC"] = Application.Contents[" SoNguoiTC "] + 1

Response.write("Bạn là vị khách thứ: " & Application.Contents["SoNguoiTC"])

Ngoài ra, đối tượng Application còn có 2 phương thức thường dùng là **Application**.**Lock()**: Để khóa không cho người khác sửa đổi các biến toàn cục và **Application.UnLock**() để mở khóa .

Đối tượng Application cũng có 2 sự kiện đó là Application_OnStart và Application_On**END**. Sự kiện OnStart chỉ được kích hoạt duy nhất một lần khi yêu cầu đầu tiên phát sinh. Sự kiện On**END** được kích hoạt khi dịch vụ web dừng (unload).

Đối tượng Application có 2 phương thức là Lock và Unlock. Khi gọi phương thức Lock (khóa) thì tất cả các ứng dụng không được phép thay đổi các giá trị Application. Để các ứng dụng khác được phép thay đổi các biến Application thì gọi phương thức Unlock.

Mã lệnh viết cho 2 sự kiện này cũng được đặt trong file Global.asa.

Cookie

Cookie làm việc như thế nào?

Khi trình duyệt web tạo một Cookie, một nội dung sẽ được lưu vào header của trang web với nội dung giống như sau:

Set-Cookie: Message=Hello

Phần tiêu đề Set-Cookie này gây ra cho trình duyệt web tạo một Cookie có tên là Message và giá trị của nó là Hello.

Sau khi một Cookie được tạo trên trình duyệt, Mỗi khi trình duyệt yêu cầu một trang web trong ứng dụng, trình duyệt sẽ gửi một header có dạng giống như sau:

Cookie: Message=Hello

Tiêu đề Cookie chứa đựng tất cả các Cookie mà được tạo trên Web Server. Cookie được gửi trở lại mỗi khi một yêu cầu được đưa ra trên trình duyệt web.

Bạn có thể tạo hai kiểu của Cookie, Session Cookies và Persistent Cookies. Session cookies chỉ tồn tại trong bộ nhớ khi trình duyệt web bị đóng lại thì nó cũng bị xóa đi.

Còn Persistent Cookies có thể tồn tại hàng tháng hoặc hàng năm. Khi bạn tạo một Persistent Cookies, nó sẽ được lưu trữ trên web browse trên máy tính của bạn. với IE ví dụ nó sẽ được lưu trữ trong một file Text theo thư mục

\Documents and Settings\[user]\Cookies

Còn với FireFox nó lưu trữ trong thư mục theo đường dẫn sau:

\Documents and Settings\ [user]\ Application Data\ Mozilla\ Firefox\ Profiles\ [randomfolder name]\ Cookies.txt

bởi vì sự lưu trữ cookies trên các trình duyệt khác nhau để ở các thư mục khác nhau lên khi bạn tạo Cookies trên IE thì nó sẽ không tồn tại trên FireFox và ngược lại.

Tạo Cookies

Bạn có thể tạo cookies với câu lệnh Response.Cookies, tất cả các Cookies sẽ được gửi từ Web Server đến Web Browser.

Ví dụ sau đây sẽ tạo ra một Cookies Message với giá trị được lấy từ hộp TextBox trên Form

Ví dụ 1: Trang setCookies.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>

<script runat="server">

   protected void Add_Click(object sender, EventArgs e)

   {

      Response.Cookies["Message"].Value = txtCookies.Text;

   }

</script>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

   <title>Create Cookies</title>

</head>

<body>

   <form id="form1" runat="server">

   <div>

      <asp:Label ID="Label1" runat="server" Text="Cookie Value"></asp:Label>

      <asp:TextBox ID="txtCookies" runat="server"></asp:TextBox>

      <asp:Button ID="Add" runat="server" OnClick"Add_Click" Text="Button" />

   </div>

   </form>

</body>

</html>
```

Trong ví dụ một là chúng ta tạo ra một Session Cookies, còn nếu bạn muốn tạo một Persistent Cookies bạn cần chỉ định thời hạn kết thúc cho Cookies .

Ví dụ 2 trang setPersistentCookies.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="setPersistentCookies.aspx.cs" Inherits="setPersistentCookies" %>

<script runat="server">
```

```
    protected void Page_Load(object sender, EventArgs e)
    {
        int counter=0;
        if (Request.Cookies["counter"] != null)
            counter = Int32.Parse(Request.Cookies["counter"].Value);
        counter++;
        Response.Cookies["counter"].Value = counter.ToString();
        Response.Cookies["counter"].Expires = DateTime.Now.AddYears(2);
        this.Label1.Text = Response.Cookies["counter"].Value;
    }
</script>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Set Persitent Cookies</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
    </div>
```

```
    </form>

</body>

</html>
```

Trong ví dụ trên khi chạy chương trình mỗi lần bạn Refresh lại trang thì giá trị của Label1 sẽ tăng lên một. Và với câu lệnh

Response.Cookies[“counter”].Expires=Datetime.Now.AddYears(2), có nghĩa là thời gian tồn tại của Cookie này sẽ là 2 năm.

Đọc dữ liệu từ Cookies

Bạn sử dụng lện Request.Cookies để lấy dữ liệu từ Cookies, bạn xem lại ví dụ 2 trang setPersistentCookies.aspx.

Khi bạn có một tập hợp các Cookies bạn có thể lấy tât cả giá trị của các Cookies trên website của mình, ví dụ sau đây sẽ hướng dẫn bạn làm việc đó.

Ví dụ 3 trang GetallCookies

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="GetAllCookies.aspx.cs" Inherits="GetAllCookies" %>

<script runat="server">

    void Page_Load()

    {

        ArrayList colCookies = new ArrayList();

        for (int i = 0; i < Request.Cookies.Count; i++)

            colCookies.Add(Request.Cookies[i]);

        grdCookies.DataSource = colCookies;

        grdCookies.DataBind();

    }

</script>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

 <html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

   <title>Get All Cookies</title>

</head>

<body>

   <form id="form1" runat="server">

   <div>

      <asp:GridView ID="grdCookies" runat="server">

      </asp:GridView>

   </div>

   </form>

</body>

</html>
```

Thiết lập thuộc tính cho Cookies

Cookies được đưa ra với lớp HttpCookie, khi bạn tạo hoặc lấy giá trị từ một Cookie có thể bạn sẽ sử dụng một vài thuộc tính của lớp này:

Domain: cho phép bạn chỉ định Domain kết hợp với Cookie. Giá trị mặc định là Domain hiện tại.

Expires: Cho phép tạo Persistent Cookies với thời hạn được chỉ định

HasKeys: Cho phép bạn định rõ Cookies có nhiều giá trị hay không.

HttpOnly: Cho phép đưa ra một Cookies từ JavaScript.

Name: chỉ định tên cho Cookies.

Path: Cho phép chỉ định đường dẫn kết hợp với Cookies. Giá trị mặc định là /.

Secure: Cho phép một Cookie được chuyển tác ngang tới kết nối Sercure Sockets

Layer (SSL).

Value: Cho phép lấy hoặc thiết lập giá trị cho Cooki

Values: Cho phép bạn lấy hoặc thiết lập giá trị riêng khi làm việc với Cookies có nhiều giá trị.

Xóa Cookies

Để xóa một Cookie bạn thiết lập ngày hết hạn cho Cookies là -1

Ví dụ như câu lệnh dưới đây:

Response.Cookies[“Message”].Expires = DateTime.Now.AddDays(-1);

Trên ví dụ trên chúng ta sẽ xóa Cookie vơi tên là Message.

Làm việc với Cookies nhiều giá trị:

Trong trình duyệt không lên lưu trưc hơn 20 Cookies từ một Domain, thay vào đó bạn có thể làm việc với một Cookie nhiều giá trị.

Một Cookies nhiều giá trị là một Cookies đơn chứa đựng nhiều khóa con, bạn có thể tạo nhiều khóa con như bạn muốn.

Như ví dụ dưới đây bạn tạo ra một Cookies Person nhiều giá trị, Cookie Person lưu trữ các giá trị Họ tên, Ngày sinh và màu sắc yêu thích.

Ví dụ 4 trang SetCookieValues.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="SetCookieValues.aspx.cs" Inherits="SetCookieValues" %>

<script runat="server">

   protected void btnsubmit_Click(object sender, EventArgs e)

   {

      Response.Cookies["Person"]["Hoten"] = txtHoten.Text;
```

```
        Response.Cookies["Person"]["Ngaysinh"] = txtNgaysinh.Text;

        Response.Cookies["Person"]["Color"] = txtColor.Text;

        Response.Cookies["Person"].Expires = DateTime.MaxValue;

    }

</script>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">



<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

    <title>Set Cookie MutilValues</title>

</head>

<body>

    <form id="form1" runat="server">

    <div>

        <table>

            <tr>

                <td>Họ tên</td>

                <td><asp:TextBox ID="txtHoten" runat="server" /></td>

            </tr>

            <tr>

                <td>Ngày sinh</td>

                <td><asp:TextBox ID="txtNgaysinh" runat="server" /></td>
```

```
            </tr>
            <tr>
                <td>Màu yêu thích</td>
                <td><asp:TextBox ID="txtColor" runat="server" /></td>
            </tr>
            <tr>
                <td colspan="2">
                    <asp:Button ID="btnsubmit" runat="server" Text="Submit" OnClick="btnsubmit_Click" />
                </td>
            </tr>
        </table>
    </div>
    </form>
</body>
</html>
```

Việc lấy giá trị của Cookie nhiều giá trị tương tự như các phần trên, học viên về nhà hoàn thiện nốt.

Session-Sử dụng Profiles

Session

Bạn có thể chưa thực sự dùng Cookies để lưu trữ Shoping Cart. Một Cookie vừa quá nhỏ và quá đơn giản. Để làm việc ngoài giới hạn của Cookie ASP.NET Framework hỗ trợ một chức năng mới được gọi là Session State

Giống với Cookie Session lưu trữ dữ liệu trong phạm vi riêng với từng người sử dụng. Nhưng không giống với Cookie Session không giới hạn dung lượng, nếu bạn cần bạn có thể lưu trữ hàng Gigabyte dữ liệu, hơn thế nữa Session có thể đưa ra điều đối tượng phức tạp hơn là chuỗi Text. Bạn có thể lưu trữ một vài đối tượng trong Session. Ví dụ bạn có thể lưu trữ một Dataset hay một Shoping cart trong Session.

thêm dữ liệu vào Session

Bạn thêm dữ liệu vào trạng thái Session bằng việc sử dụng đối tượng Session, Ví dụ sau đây sẽ thêm một dữ liệu vào Session có tên là Message và giá trị của nó là “Hello World”

Ví dụ 1: Trang Sessionset.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Sessionset.aspx.cs" Inherits="_Default" %>

<script runat="server">

   protected void Page_Load(object sender, EventArgs e)

   {

      Session["Message"] = "Hello World";

   }

</script>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">



<html xmlns="http://www.w3.org/1999/xhtml" >
```

```
<head runat="server">
    <title>Session set</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
    <h1>Session state item added!</h1>
    </div>
    </form>
</body>
</html>
```

Lấy dữ liệu từ một Session

Ví dụ: Trang Sessionset.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Sessionget.aspx.cs"
Inherits="Sessionget" %>


<script runat="server">
    protected void Page_Load(object sender, EventArgs e)
    {
        lblsession.Text = Session["Message"].ToString();
    }
</script>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

   <title>Session get</title>

</head>

<body>

   <form id="form1" runat="server">

   <div>

      <asp:Label ID="lblsession" runat="server" />

   </div>

   </form>

</body>

</html>
```

Bạn lưu ý rằng cũng như Cookie khi một Session được tạo ra, một trạng thái session có tên ASP.NET_SessionID được tự động thêm vào trình duyệt của bạn và Session này được lưu trữ trên web server và không lưu trữ trên webClient. Và khi bạn tắt trình duyệt đi thì Session này của bạn vẫn tồn tại trong khoảng thời gian quy định, mà ASP.NET Framework quy định thời gian mặc định của Session là 20 phút. bạn có thể thiết lập thời gian nhiều hơn.

Lưu trữ cơ sở dữ liệu trong Session

Bạn có thể tạo Session state để tạo một vùng nhớ cho người sử dụng. ví dụ bạn có thể tải dữ liệu cho một người sử dụng và cho phép người sử dụng đó sắp xếp hay lọc dữ liệu.

Sử dụng đối tượng Session

Chương trình ứng dụng chính giao tiếp và làm việc với Session State là lớp HttpSessionState. Đối tượng này được thể hiện bới các thuộc tính Page.Session, Context.Session, UserControl.Session, Webservice.Session và Application.Session. có nghĩa là bạn có thể làm việc với Session bất kỳ đâu trong ứng dụng web.

Lớp HttpSessionState hỗ trợ các thuộc tính sau:

? CookieMode: có cho phép Cookie Session hay không?

Count: cho phép lấy số dữ liệu trong Session State

IsCookieless: Cho phép chỉ rõ có cho phép Cookieless hay không?

IsNewSession—Enables you to determine whether a new user session was created with the current request.

IsReadOnly—Enables you to determine whether the Session state is read-only.

Keys—Enables you to retrieve a list of item names stored in Session state.

Mode—Enables you to determine the current Session state store provider. Possible

values are Custom, InProc, Off, SqlServer, and StateServer.

SessionID—Enables you to retrieve the unique session identifier.

Timeout—Enables you to specify the amount of time in minutes before the web server assumes that the user has left and discards the session. The maximum value is 525,600 (1 year).

Đối tượng HttpSessionState hỗ trợ các phương thức sau:

Abandon: Cho phép kết thúc Session của một người sử dụng.

Clear: Cho phép xoá toàn bộ dữ liệu trong Session State.

Remove: cho phép bạn xoá từng phần tử trong Session State

Điều khiển sự kiện Session

Có hai sự kiện có liên quan với Session State mà bạn có thể điều khiển nó trong file Global.asax là sự kiện Session_Start và Session_End.

Session_Start xảy ra khi một Sesion mới của người sử dụng được tạo ra. Bạn có thể sử dụng sự kiện này để load thông tin của người sử dụng ra từ cơ sở dữ liệu. Ví dụ bạn có thể tải dữ liệu về Shoping cart của người sử dụng trong sự kiện này .

Session_End xảy ra khi kết thúc Session, một Session kết thúc khi thời hạn của Session hết hoặc bởi viêcj chỉ định của phương thức Session.Abadon. Ví dụ Khi bạn muốn tự động ghi giỏ hàng của người sử dụng vào bảng dữ liệu trong cơ sở dữ liệu khi Session_End xảy ra.

Điều khiển khi Session quá hạn

Mặc định ASP.NET Framework quy định thời gian quá hạn của Session là 20 phút, trong nhiều trường hợp bạn thấy như vậy là quá ít, và bạn nghĩ rằng bạn cần thay đổi thời gian này.

Ví dụ trong trường hợp bạn tạo một trang quản trị của website, khi cập nhật dữ liệu bạn có một bài viết dài, và thời hạn 20 phút không thể đủ thời gian cho bạn cập nhật tin đó, và để hoàn thành bài đó có thể bạn phải mất 1 giờ.

Sự bất lợi là nếu tăng thời gian quá hạn của Session lên thì bộ nhớ của ứng dụng càng phải sử dụng nhiều, vì vậy khi bạn tăng thời hạn của Session thì bộ nhớ của Server sẽ phải dùng càng nhiều.

Tuy nhiên nếu cần bạn vẫn có thể tăng thời hạn của Session bằng cách bạn có thể chỉ định thời gian quá hạn của Session trong file web.config

Ví dụ:

```
   <configuration>

      <system.web>

         <sessionState timeout="60" />

      </system.web>

</configuration>
```

sử dụng Cookieless Session State

Mặc định Session State dựa trên Cookie. ASPNET Framework sử dụng ASP.NET_SessionId Cookie để định danh người sử dụng trên website mà dữ liệu có thể

được kết hợp với người sử dụng, nếu người sử dụng vô hiệu hóa Cookie trên trình duyệt thì Session State sẽ không làm việc.

Để Session có thể làm việc khi trình duyệt vô hiệu hóa Cookie bạn cần thêm vào Cookieless Session. Khi Cookieless Session được cho phép, thì Session ID của người sử dụng sẽ được thêm vào trang URL.

Đây là một ví dụ của trang URL nhìn giống với khi Cookieless Session được cho phép.

http://localhost:4945/Original/(S(5pnh11553sszre45oevthxnn))/SomePage.aspx

Bạn cho phép Cookieless Session bằng việc chỉnh sửa các thành phần SessionState trong file web.config. Thành phần SessionState bao gồm các một đặc tính cookieless mà nó chấp nhận các giá trị sau:

AutoDetect: SessionID được lưu trữ trong một cookie khi trình duyệt có cho phép Cookie. Ngược lại thì nó lưu trữ vào địa chỉ URL.

UseCookies: Session ID luôn luôn lưu trữ trong cookie

UseDeviceProfile: Session ID lưu trữ trong cookie khi trình duyệt hỗ trợ Cookie, trường hợp ngược lại nó lưu trữ trong địa chỉ URL.

UseUri: Session ID luôn luôn được thêm vào URL.

Trong ví dụ sau đây chúng ta thiết lập cookieless là AutoDetect như vậy ASP.NET Framework sẽ kiểm tra sự tồn tại cảu HTTP Cookie header, nếu Cookie header được tìm thấy thì Framework sẽ lưu trữ Session trong một cookie và ngược lại thì nó sẽ thêm vào URL.

Ví dụ:

```
<configuration>

   <system.web>

     <sessionState cookieless="AutoDetect" regenerateExpiredSessionId="true"/>

   </system.web>

</configuration>
```

Sử dụng Profiles

Profiles dùng để làm gì?

ASP.NET Framework cung cấp cho bạn một thay thế cho Session hay cookie để lưu trữ thông tin của người sử dụng đó là đối tượng Profile

Bạn tạo một Profile bằng cách định nghĩa một danh sách các thuộc tính Profile trong ứng dụng ở file web.config trong thư mục root. ASP.NET Framework tự động biên dịch một lớp chứa đựng các thuộc tính này.

Ví dụ sau đây sẽ đưa ra một Profile với ba thuộc tính: firstName, lastName và NumberOfVisits:

Trang web.config

```
<?xml version="1.0"?>



<configuration>

   <system.web>

     <profile>

       <properties>

         <add name="firstName" />

         <add name="lastName"/>

         <add name="NumberOfVisits" type="Int32" defaultValue="0"/>

       </properties>

     </profile>

   </system.web>

</configuration>
```

Khi làm việc với Profile bạn chú ý một số thuộc tính sau:

Name: chỉ định tên của thuộc tính.

Type: cho phép chỉ định kiểu dữ liệu của thuộc tính

Defaultvalue: cho phép chỉ định giá trị mặc định của thuộc tính

ReadOnly: cho phép tạo thuộc tính chỉ đọc

serializeAs: Enables you to specify how a property is persisted into a static repre-

sentation. Possible values are Binary, ProviderSpecific, String, and Xml.

allowAnonnyMous: cho phép người sử dụng nặc danh đọc và thiết lập thuộc tính

Provider: Cho phép bạn kết hợp thuộc tính với một Profile Provider riêng biêt.

customeProviderData:Enables you to pass custom data to a Profile provider.

Sau khi định nghĩa một Profile trong web.config, bạn có thể sử dụng đối tượng Provider để chỉnh sửa các thuộc tính Profile. Như ví dụ sau đây bạn sẽ chỉnh sửa hai thuộc tính firstName và lastName trên Form, hơn thế nữa chúng ta sẽ thấy mỗi lần trang web được load lại thì giá trị của NumberOfVisit sẽ tăng lên một.

Ví dụ: trang showProfile.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="showProfile.aspx.cs"
Inherits="showProfile" %>



<script runat="server">



    void Page_PreRender()

    {

        firstName.Text = Profile.firstName;

        lastName.Text = Profile.lastName;
```

```
        Profile.NumberOfVisits++;

        numbervisit.Text = Profile.NumberOfVisits.ToString();

    }


    protected void btnUpdate_Click(object sender, EventArgs e)

    {

        Profile.firstName = txtFirstName.Text;

        Profile.lastName = txtLastName.Text;

    }

</script>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

    <title>show Profile</title>

</head>

<body>

    <form id="form1" runat="server">

    <div>

     FirstName: <asp:Label ID="firstName" runat="server" /><br />

     LastName: <asp:Label ID="lastName" runat="server" /><br />

     Number of Visit: <asp:Label ID="numbervisit" runat="server" /><hr />
```

```
    <asp:Label ID="lblfistname" runat="server" Text="FirstName:" />

    <asp:TextBox ID="txtFirstName" runat="server" /><br />

    <asp:Label ID="lbllastName" runat="server" Text="LastName:" />

    <asp:TextBox ID="txtLastName" runat="server" /><br />

        <asp:Button ID="btnUpdate" Text="Update" runat="server"
OnClick="btnUpdate_Click" />

   </div>

   </form>

</body>

</html>
```

Và kết xuất của chương trình như sau:

Chú ý rằng thuộc tính của Profile được trình bày với kiểu dữ liệu đã quy định trong thuộc tính type mà ta định nghĩa trong Profile.

Một thuộc tính quan trọng của Profile nó là nó có khả năng giữ lại giá trị của thuộc tính khi người sử dụng rời khỏi trang web, ví dụ nếu bạn gán thuộc tính Profile cho một người sử dụng, người sử dụng đó không quay lại website trong 500 năm thì giá trị đó vẫn được giữ lại cho người sử dụng.

Đối tượng Profile sử dụng models Provider. mặc định Profile Provider là SqlProfileProvider, Mặc định Provider lưu trữ dữ liệu Profile trong cơ sở dữ liệu MS SQL Server 2005 Express được đặt tên là ASPNETDB.mdf, được định vị trong thư mục App_Data. Nếu cơ sở dữ liệu không tồn tại thì nó sẽ được tạo tự động khi chạy chương trình có sử dụng Profile.

MẶc định bạn không thể lưu trữ thông tin Profile cho một người sử dụng nặc danh. ASP.NET Framework tính đồng nhất authenticate của bạn kết hợp với thông tin Profile, bạn có thể sử dụng đối tượng Profile với các kiểu chuẩn mà authentication hỗ trợ bởi ASP.NET Framework, bao gồm cả hai kiểm chứng Forms và Windows

Creating Profile Groups

Nếu bạn cần định nghĩa nhiều thuộc tính của Profile, bạn có thể tạo các thuộc tính bằng quản lý bởi việc tổ chức các thuộc tính trong Groups. Ví dụ trong file web.config sau định nghĩa hai nhóm thuộc tính Preferences và ContactInfo.

Ví dụ Trang web.config

```
<?xml version="1.0"?>



<configuration>

   <system.web>

     <profile>

       <properties>

         <group name="Preferences">

           <add name="BackColor" defaultValue="lightblue"/>

           <add name="font" defaultValue="Arial"/>

         </group>

         <group name="ContactInfo">

           <add name="Email" defaultValue="hiepgia@hiepgia.com"/>

           <add name="phone" defaultValue="0933030411"/>

         </group>

       </properties>

     </profile>
```

```
    </system.web>

</configuration>
```

Ví dụ sau đây sẽ hướng dẫn bạn cách tạo

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="showProfilegoups.aspx.cs" Inherits="showProfilegoups" %>

<%@ Import Namespace="System.Drawing" %>

<script runat="server">

    protected void Page_Load()

    {

        lblEmail.Text = Profile.ContactInfo.Email;

        lblPhone.Text = Profile.ContactInfo.phone;



        Style pagestyle = new Style();

        pagestyle.BackColor = ColorTranslator.FromHtml(Profile.Preferences.BackColor);

        pagestyle.Font.Name = Profile.Preferences.font;

        Header.StyleSheet.CreateStyleRule(pagestyle, null,"html");

    }

</script>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">



<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">
```

```
    <title>show profile group</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        Email: <asp:Label ID="lblEmail" runat="server" /><br />
        Phone: <asp:Label ID="lblPhone" runat="server" />
    </div>
    </form>
</body>
</html>
```

Hỗ trợ người sử dụng nặc danh

Mặc định người sử dụng nặc danh không thể chỉnh sửa các thuộc tính của Profile, vấn đề là ASPNET Framework không có phương thức kết hợp dữ liệu Profile với người sử dụng riêng biêt trừ khi người sử dụng được kiểm chứng.

Nếu bạn muốn cho phép người sử dụng nặc danh chỉnh sửa các thuộc tính Profile, bạn có phải cho phép đặc tính của ASP.NET Framework được gọi là định danh nặc danh. Khi định danh nặc danh được cho phép, khi định danh duy nhất được gán đến người sử dụng nặc danh và được lưu trữ trong trình duyệt cookie ổn định.

Hơn thế nữa, bạn phải đánh dấu tất cả các thuộc tính Profile mà bạn muốn cho phép người sử dụng nặc danh với các đặc tính cho phép nặc danh. Ví dụ trang web.config sau cho phép định danh nặc danh và định nghĩa một thuộc tính Profile mà có thể chỉnh sửa được bởi người sử dụng nặc danh.

```
<?xml version="1.0"?>
<configuration>
    <system.web>
```

```
    <authentication mode="Windows" />

  <anonymousIdentification enabled="true"/>

  <profile>

    <properties>

        <add name="NumberOfVisits" type="Int32" defaultValue="0"
allowAnonymous="true"/>

    </properties>

  </profile>

 </system.web>

</configuration>
```

Thuộc tính NumberOfVisits bao gồm thuộc tính allowAnonymous. Chú ý rằng file web.config và chỉ cho phép Form Authencation. Khi Form Authentication được cho phép và bạn không login, và khi đó bạn là người sử dụng nặc danh.

Trong ví dụ sau sẽ hướng dẫn cách bạn sửa thuộc tính định danh khi định danh nặc danh được cho phép.

Trang ShowAnonymousIdentification.aspx;

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="ShowAnonymousIdentification.aspx.cs"
Inherits="ShowAnonymousIdentification" %>



<script runat="server">

 void Page_PreRender()

 {

    lblName.Text = Profile.UserName;

    lblanonymous.Text = Profile.IsAnonymous.ToString();
```

```
        Profile.NumberOfVisits++;

        lblnumbetofanonymous.Text = Profile.NumberOfVisits.ToString();

    }


    protected void btnLogin_Click(object sender, EventArgs e)

    {

        FormsAuthentication.SetAuthCookie("Bob", false);

        Response.Redirect(Request.Path);

    }


    protected void btnLogout_Click(object sender, EventArgs e)

    {

        FormsAuthentication.SignOut();

        Response.Redirect(Request.Path);

    }

</script>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

    <title>Show Anonymous Identification</title>

</head>
```

```
<body>

   <form id="form1" runat="server">

   <div>

       UseName: <asp:Label ID="lblName" runat="server" /><br />

       Is Anonymous: <asp:Label ID="lblanonymous" runat="server" />

       Number of Visits: <asp:Label ID="lblnumbetofanonymous" runat="server" /><br
/><hr />

       <asp:Button ID="btnReload" Text="Reload" runat="server" />

          <asp:Button ID="btnLogin" Text="Login" OnClick="btnLogin_Click"
runat="server" />

         <asp:Button ID="btnLogout" Text="Logout" OnClick="btnLogout_Click"
runat="server" />

   </div>

   </form>

</body>

</html>
```

Kết nối cơ sở dữ liệu trong trang web sử dụng ASP.NET

Tầm quan trọng của việc sử dụng cơ sở dữ liệu

Hầu hết ứng dụng hay các website đều cần phải có cơ sở dữ liệu, để lưu trữ dữ liệu, xử lý thông tin và đưa ra các báo cáo, hỗ trợ tìm kiếm…

Khi dữ liệu trở thành trung tâm của ứng dụng thì cung cấp các chức năng tới người dùng phụ thuộc vào khả năng thao tác dữ liệu, vấn đề mà người thiết kế và người xây dựng ứng dụng quan tâm khi sử dụng dữ liệu là:

Lưu dữ liệu tập trung

Đảm bảo toàn vẹn dữ liệu

Đảm bảo khả năng truy xuất đồng thời của nhiều người dùng trên dữ liệu

Đảm bảo thời gian hồi đáp ngắn cho mỗi người dùng

Bảo mật dữ liệu

Trao đổi dữ liệu giữa các hệ thống khác nhau

Những vấn đề này được giải quyết dựa vào khả năng của các hệ quản trị cơ sở dữ liệu(HQT CSDL) và các phần mềm xử lý dữ liệu do HQL CSDL cung cấp. Net truy xuất dữ liệu qua ADO.NET, đặc điểm chính của ADO.NET là khả năng làm việc vơi dữ liệu không kết nối, dữ liệu được lưu trữ trong bộ nhớ như một csdl thu nhỏ gọi là dataset, nhằm tăng tốc độ tính toán, xử lý tính toán và hạn chế sử dụng tài nguyên trên Database Server. ĐẶc điểm quan trọng thứ 2 là khả năng xử lý dữ liệu chuẩn XML, dữ liệu ở dạng XMl có thể trao đổi giữa bất kỳ hệ thống nào nên ứng dụng của bạn sẽ có nhiều khả năng làm việc với nhiều ứng dụng khác.

Kết nối CSDL sử dụng ADO.NET

Kiến trúc của ADO.NET và Các đối tượng

Khi phát triển các ứng dụng trên nền web thì công việc chủ yếu phải giải quyết là xử lý các nghiệp vụ, trong đó phần lớn là xử lý Cơ sở dữ liệu. Trong môi trường phát triển Microsoft .NET tất cả các ứng dụng webform hay winform đều thống nhất sử dụng chung một bộ thư viện để truy xuất và thao tác Cơ sở dữ liệu gọi là ADO.NET (Active Data Object).

ADO.NET là một tập các lớp nằm trong bộ thư viện lớp cơ sở của .NET Framework, cho phép các ứng dụng windows (như C#, VB.NET) hay ứng dụng web (như ASP.NET) thao tác dễ dàng với các nguồn dữ liệu.

Mụ c ti êu c h ính c ủ a A D O. N ET là:

Cung cấp các lớp để thao tác CSDL trong cả hai môi trường là phi kết nối (Disconected data) và kết nối (Connected data).

Tích hợp chặt chẽ với XML (Extensible Markup Language)

Tương tác với nhiều nguồn dữ liệu thông qua mô tả dữ liệu chung.

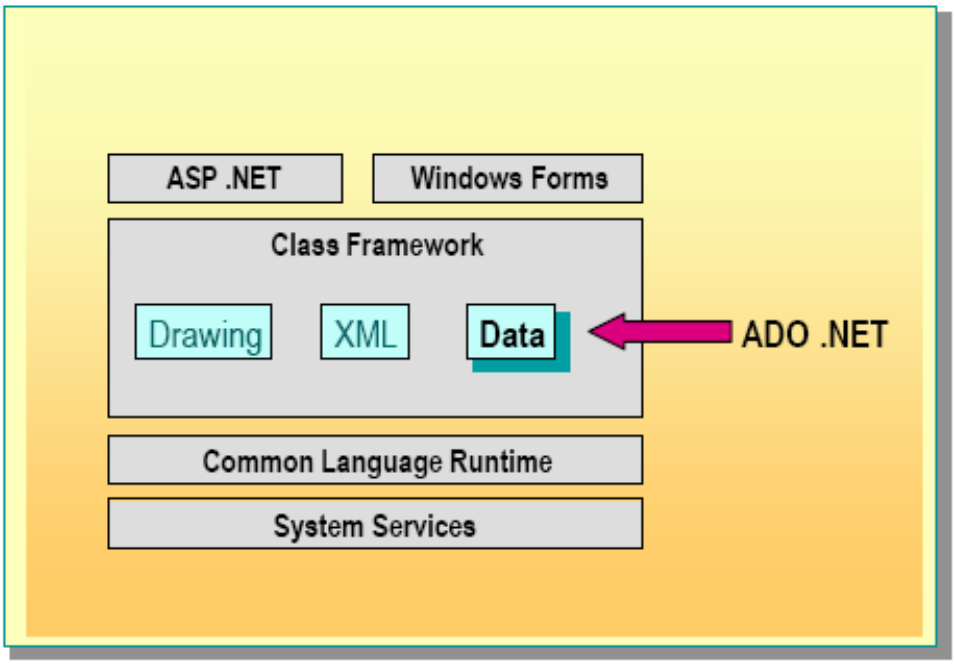
Tối ưu truy cập nguồn dữ liệu (OLE DB & SQL server).

Làm việc trên môi trường Internet.

Các lớp của ADO.NET được đặt trong Namespace là System.Data/ System.Data.oledb

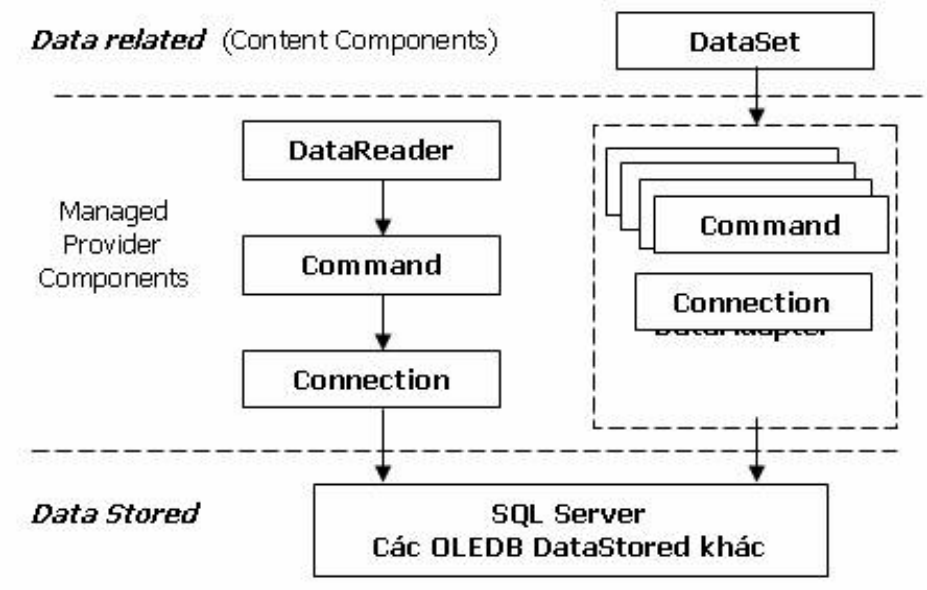
ADO.NET bao gồm 2 Provider (2 bộ thư viện thường dùng) để thao tác với các CSDL là: OLE DB Provider (nằm trong System.Data.OLEDB) dùng để truy xuất đến **bấtkỳ CSDLnàocóhỗtrợOLEDB**; SQL Provider (nằm trong System.Data.SQLClient) chuyên dùng để truy xuất đến CSDL SQL Server (Không qua OLE DB nên nhanh hơn). Hiện nay, các hãng thứ ba còn cung cấp các Provider khác như : MySQL, Oracle… provider để cho phép ứng dụng .NET truy xuất đến các cơ sở dữ liệu không phải của Microsoft khác.

Vị trí của ADO.NET trong kiến trúc của .NET Framework



V ị trí c ủ a A D O . N E T t r ong kiến trúc c ủ a .net Framework

Từ kiến trúc ta thấy rằng: ADO.NET là một thành phần nội tại (Instrict) của .NET framework, do vậy nó có thể được sử dụng trong tất cả các ngôn ngữ hỗ trợ .NET như C#, VB.NET… mà không có sự khác biệt nào (Tức là các chức năng cũng như cách sử dụng hoàn toàn giống nhau).



Hình biểu diễn kiến trúc ADO.NET

Kiến trúc ADO.NET có thể chia làm 2 phần chính:

Managed Provider Component: bao gồm các đối tượng như DataAdapter, DataReader,… giữ nhiệm vụ làm việc trực tiếp với dữ liệu như database, file,…

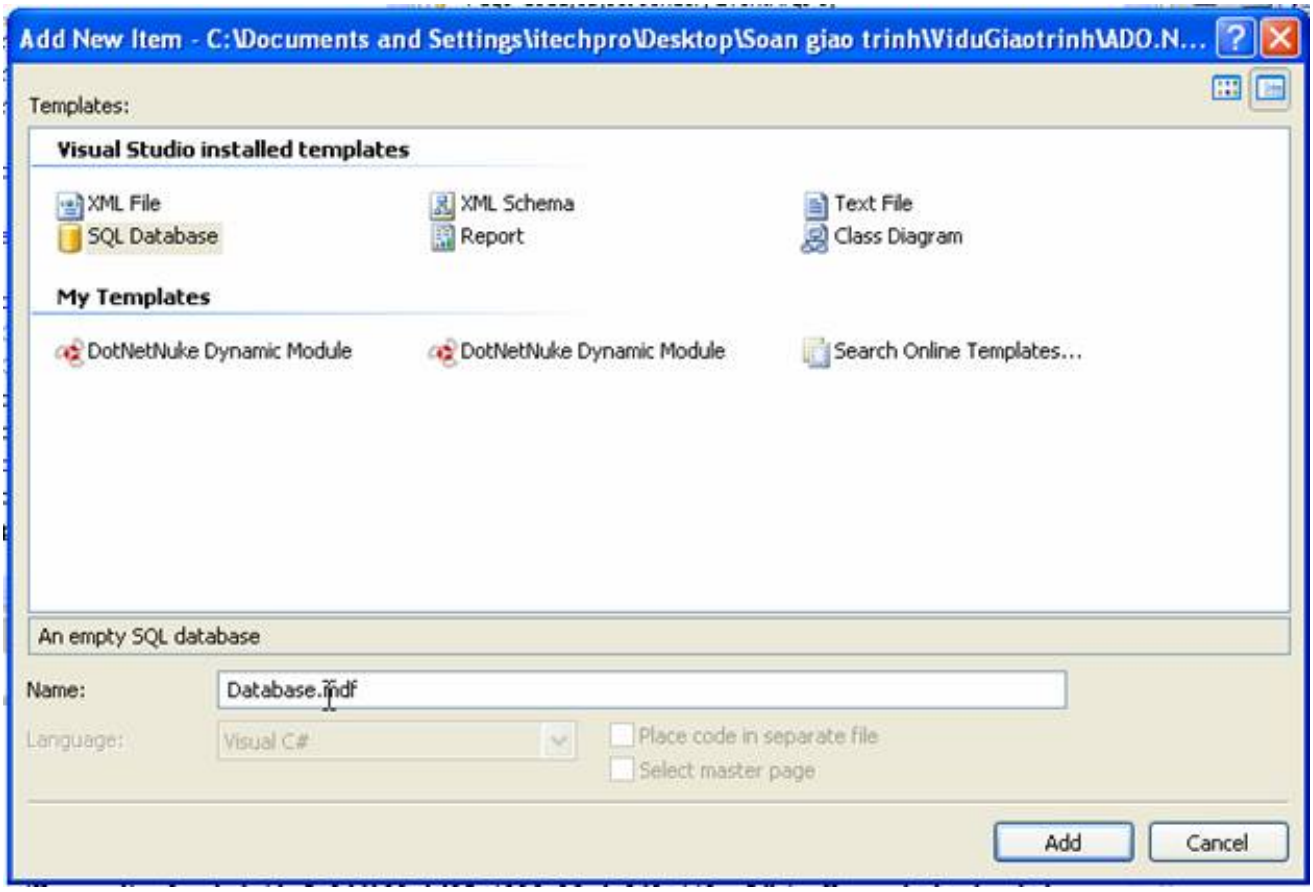
Content Component: bao gồm các đối tượng như DataSet, DataTable,… đại diện cho dữ liệu thực sự cần làm việc. DataReader là đối tượng mới, giúp truy cập dữ liệu nhanh chóng nhưng forward-only và read-only giống như ADO RecordSet sử dụng Server cursor, OpenFowardOnly và LockReadOnly.

DataSet cũng là một đối tượng mới, không chỉ là dữ liệu, DataSet có thể coi là một bản sao gọn nhẹ của CSDL trong bộ nhớ với nhiều bảng và các mối quan hệ. DataAdapter là đối tượng kết nối giữa DataSet và CSDL, nó bao gồm 2 đối tượng Connection và Command để cung cấp dữ liệu cho DataSet cũng như cập nhật dữ liệu từ DataSet xuống CSDL.

Trước khi đi vào học cụ thể các đối tượng của ADO.NET chúng ta cùng xem qua một ví dụ HelloWorld với ADO.NET qua đó bạn sẽ thấy được công việc cần thực hiện khi thao tác với database(ở ví dụ này mình dùng với SQLExpress).

Để làm ví dụ này bạn thực hiện theo các bước sau:

bước 1. Nhấn chuột phải vào thư mục App_Data chọn new Item, Cửa sổ Add New Item hiện ra bạn chọn SqlDatabase như hình 1 sau

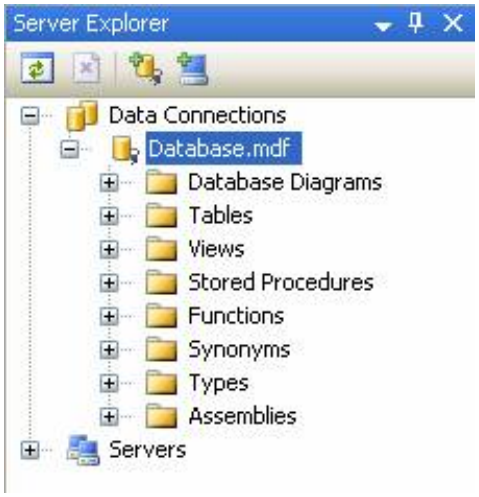


bước 2. Nhập tên Database vào hộp Name sau đó nhấn Add.

Trong Solution Explorer sẽ thêm vào Database trong thư mục App_Data.

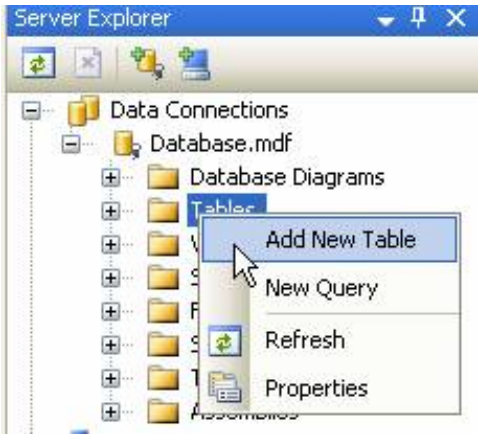
bước 3. tạo bảng dữ liệu cho Database.mdf

Bạn click đúp chuột vào Datatabase.mdf -> Server Explorer hiện ra như sau:



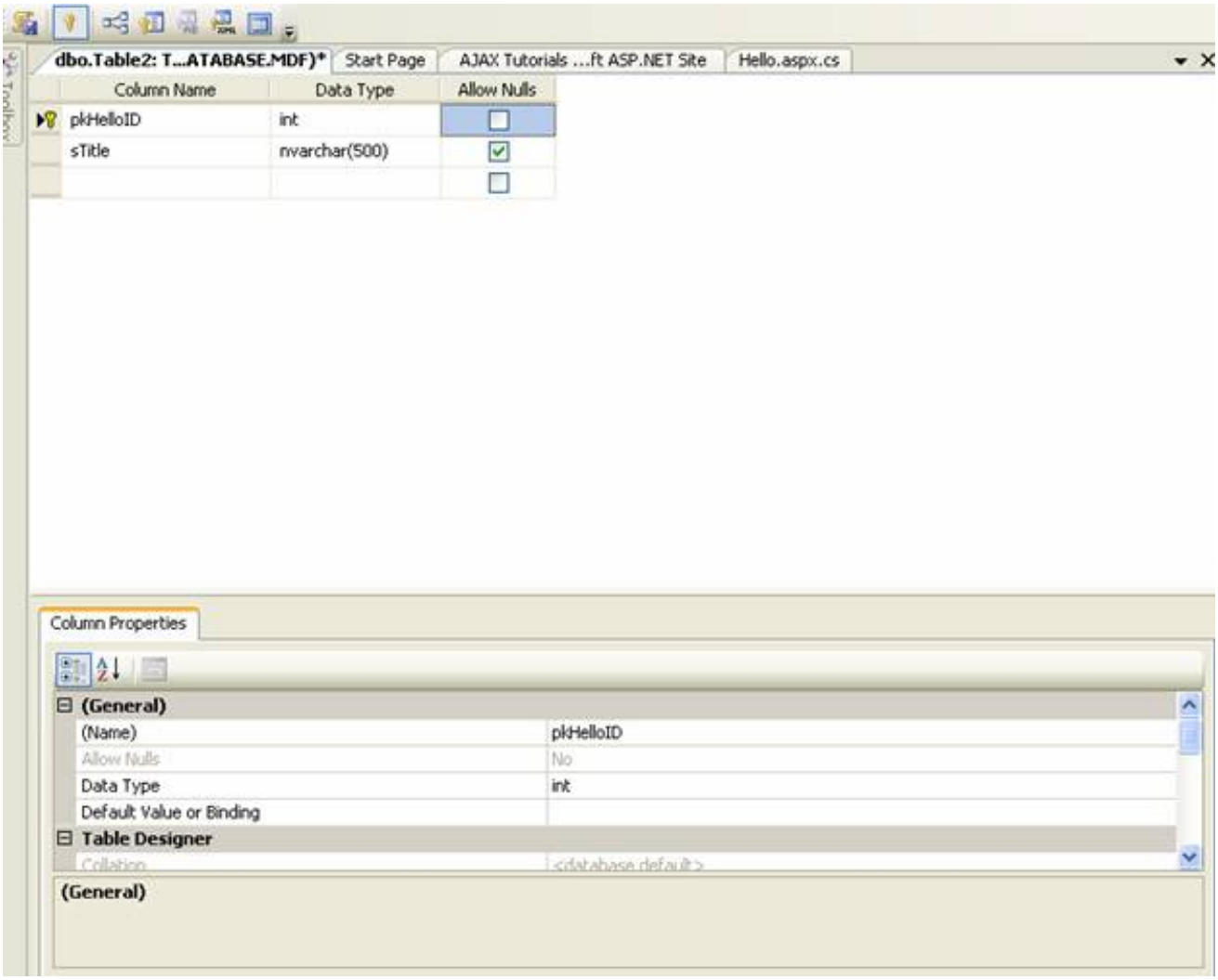
Hình 4

Bạn nhấn chuột phải vào Tables và chọn Add New Table



Hình 5

Trong màn hình của VS sẽ hiện ra như hình 6 và bạn thao tác tạo các trường dữ liệu như thao tác với Access hay MSSQL 2000/2005



Hình 6

Viết code cho Hello.aspx.cs

```
using System;

using System.Data;

using System.Data.SqlClient;


public partial class _Default : System.Web.UI.Page

{
```

```
    protected void Page_Load(object sender, EventArgs e)
    {
        //chuỗi kết nối đến nguồn dữ liệu
        string driver = "Data Source=(local)\\SQLEXPRESS;AttachDbFilename=|DataDirectory|Database.mdf;Integrated Security=True;User Instance=True";
        //đối tượng kết nối tới cơ sở dữ liệu
        SqlConnection sqlconn = new SqlConnection(driver);
        //Command điều khiển truy vấn sql
        SqlCommand sqlcom = sqlconn.CreateCommand();
        sqlcom.CommandText = "select sTitle from tblHello where pkHelloID=1";
        //mở kết nối dữ liệu
        sqlconn.Open();
        //lấy về chuỗi giá trị trong cơ sở dữ liệu
        string result = (string)sqlcom.ExecuteScalar();
        //đóng kết nối
        sqlconn.Close();
        //in giá trị ra màn hình
        Response.Write(result);
    }
}
```

Cơ bản các bước thực hiện với database

bước 1: Tại kết nối

bước 2: Tạo lệnh điều khiển truy vấn SQL

bước 3:Mở kết nối dữ liệu

bước 4: thực thi lệnh

bước 5: đóng kết nối

bước 6: in kết quả

Đối tượng Connection

Kết nối cơ sở dữ liệu SQLServer

Bạn cần nhập khẩu lớp SqlClient

```
using System.Data.SqlClient;
```

Khai báo và khởi tạo:

```
SqlConnection sqlcon;
string driver="server=localhost; UID=sa; PWD=; database=name_database";
    sqlcon=new Sqlconnection();
sqlcon.ConnectionString=driver;
```

Driver là chuỗi kết nối đến cơ sở dữ liệu trong trường hợp này mình kết nối với sqlserver 2000/2005

Kết nối với cơ sở dữ liệu Access

Bạn cận nhập khẩu lớp OleDb

```
        using System.Data.OleDb;
    OleDbConnection oleconn;
        string driver = "Provider=Microsoft.jet.OLEDB.4.0; Data Source=duongdan_tendata";
    oleconn = new OleDbConnection();
```

```
oleconn.ConnectionString = driver;
```

Thuộc tính:

ConnectString: chứa đựng chuỗi kết nối tới cơ sở dữ liệu

Database: Chứa đựng tên cơ sở dữ liệu trong chuỗi kết nối ConnectString ở trên và bạn có thể thay đổi cơ sở dữ liệu trong lúc thực thi bằng phương thức ChangeDataBase:

```
Sqlconn.ChangeDatabase("name_database_thaydoi");
```

Server: tên máy chủ bạn trỏ tới

Connect Timeout: số thời gian(tính bằng giây) chờ kết nối dữ liệu mặc đình là 15giây, nếu trong khoảng thời gian này mà vẫn chưa kết nối xong một lỗi Connect Timeout được đưa ra.

State: trả về trạng thái của đối tượng SqlConnection: bạn có thể kiểm tra trạng thái của State như sau

```
Response.Write(sqlconn.State)
```

2. Phương thức của đối tượng Connection

Các phương thức của đối tượng Connection

Open: cho phép mở dữ liệu với các thuộc tính đã khai báo trong ConnectString

Close: Đóng cơ sở dữ liệu đang mở

CreateCommand: phương thức cho phép gán hay trả về một đối tượng Command ứng với đối tượng Connection, như ví dụ HelloWorld

```
SqlConnection sqlconn = new SqlConnection(driver);

SqlCommand sqlcom = sqlconn.CreateCommand();

sqlcom.CommandText = "select sTitle from tblHello where pkHelloID=1";
```

BeginTransaction: Phương thức này khai báo bắt đầu một chuyển tác, để kết thúc chuyển tác bạn dùng Table Commit

Rollback: trong trương hợp có lỗi trong quá trình thực thi bạn có thể sử dụng phương thức Rollback để huỷ bỏ các chuyển tác đã thực hiện.

Dispose: dùng để huỷ bỏ hay giải phóng đối tượng Connection đang sử dụng

Đối tượng SqlCommand

Khai báo và khởi tạo đối tượng

Cách 1:

```
SqlCommand sqlcom;

    sqlcom=new SqlCommand(ssql,sqlconn)
```

cách 2:

```
        SqlCommand sqlcom = new SqlCommand();

    sqlcom.Connection = sqlconn;

    sqlcom.CommandType = CommandType.Text;

    sqlcom.CommandText = "select sTitle from tblHello where pkHelloID=1";
```

Phương thức

ExcuteReader: dùng để thực thi đọc cơ sở dữ liệu từ bảng cơ sở dữ liệu

ExcuteNoneQuery: Dùng để thực thi các phát biểu T-Sql như: Insert, Update, Delete, Create,…

ExcuteScalar: trả về từ phát biết SQL dạng Select chỉ có một cột một hàng.

Đối tượng SqlDataReader

Đối tượng này được net cung cấp để đọc dữ liệu từ bảng cơ sở dữ liệu, nó là đối tượng chỉ phục vụ thao tác đọc dữ liệu(Read only). Trong khi truy xuất dữ liệu nó sẽ giữ kết nối liên tục với database(hướng kết nối)

Khai báo và khởi tạo đối tượng

```
SqlDataReader sqlreader;

    sqlreader = sqlcom.ExecuteReader();
```

Đối tượng DataAdapter

OleDataAdapter được xem như bộ đọc dữ liệu từ cơ sở dliệu nguồn và điền chúng vào đối tượng DataSet hay DataTable

Khai báo, khởi tạo và giải phóng đối tượng.

```
string ssql;
```

Khai báo đối tượng

```
Dim sqlcom As SqlCommand

Dim sqlconn As SqlConnection

Dim sqladapter As SqlDataAdapter

sqlconn.Open();
```

cách 1.

```
sqladapter = New OleDbDataAdapter(ssql, sqlconn)

sqlcom = New SqlCommand(ssql, sqlconn)
```

cách 2.

```
sqladapter = new SqlDataAdapter(sqlcom);
```

Giải phóng đối tượng

```
sqladapter.Dispose();
```

Thuộc tính:

Các thuộc tính bao gồm SelectCommand, InsertCommand, UpdateCommand, DeleteCommand: thực hiện các thao tác select, insert, update, delete dữ liệu

Phương thức:

Fill: Phương thức thực thi câu lệnh select trong sql rồi điền kết quả cho DataSet hoặc Datatable.

Update: gọi lệnh cập nhật các thay đổi vào dữ liệu lên các dữ liệu nguồn

Điền dữ liệu từ Adapter vào DataSet

Dataset là một thùng chứa dữ liệu không kết nối

```
  public static DataSet Filldataset(string ssql)
      {
         DataSet dataset = new DataSet();
         opendata();
         try
         {
            sqladapter = new SqlDataAdapter(ssql, sqlconn);
            sqladapter.Fill(dataset);
            sqladapter.Dispose();
         }
         catch (Exception exp)
         {
            closedata();
            System.Web.HttpContext.Current.Response.Write(exp.ToString());
         }
         closedata();
         return dataset;
      }
```

Điền dữ liệu vào DataTable

```
public static DataTable FillDatatable(string ssql)
    {
        opendata();
        DataTable datatable = new DataTable();
        try
        {
            sqladapter = new SqlDataAdapter(ssql, sqlconn);
            sqladapter.Fill(datatable);
            sqladapter.Dispose();
        }
        finally
        {
            closedata();
        }
        closedata();
        return datatable;
    }
```

Đối tượng Dataset và DataTable-1

Là thành phần chính của kiến trúc không kết nối cơ sở dữ liệu, được dùng để nắm giữ dữ liệu của mọi cơ sở dữ liệu và cho phép thay đổi dữ liệu bên trong đối tượng này để sau đó cập nhật trở lại cơ sở dữ liệu nguồn bằng phương thức Update của đối tượng DataAdapter

Khởi tạo

```
DataSet dataset = new DataSet();

        DataSet dataset = new DataSet("Mydataset");
```

Thuộc tính Tables, dataset được dùng để chứa danh sách các đối tượng DataTable

Ví dụ:

```
 private void button1_Click(object sender, EventArgs e)

     {

        string strQuery = "select * from tblEmployees";

        DataSet dataSet = new DataSet("Employees");

        try

        {

           SqlDataAdapter sqlDataAdapter = new

              SqlDataAdapter(strQuery, Connection.sqlConnection);

           sqlDataAdapter.Fill(dataSet);



           sqlDataAdapter.Dispose();

        }

        catch (Exception ex)
```

```
            {
                MessageBox.Show("Error: " + ex.Message);
            }
            this.dataGridView1.DataSource = dataSet.Tables[0];

            //lay ve ten cua doi tuong dataset
            label1.Text ="DataSetName: " + dataSet.DataSetName ;
        }
        private void button2_Click(object sender, EventArgs e)
        {

//khai bao phat bieu sql 1
            string strQuery = "select * from tblEmployees";
            DataSet dataSet = new DataSet("Employees");
            try
            {
                SqlDataAdapter sqlDataAdapter = new
                    SqlDataAdapter(strQuery, Connection.sqlConnection);
                sqlDataAdapter.Fill(dataSet);

//khai bao phat bieu sql 2
                strQuery = "select * from tblContracts";
```

```
        sqlDataAdapter = new
            SqlDataAdapter(strQuery, Connection.sqlConnection);
        DataTable dataTable = new DataTable();
        sqlDataAdapter.Fill(dataTable);
        dataSet.Tables.Add(dataTable);

        sqlDataAdapter.Dispose();
        string dataTableName="";
        foreach(DataTable dt in dataSet.Tables)
        {
            dataTableName += dt.TableName + " ";
        }
        label1.Text = "Number of tables: " + dataTableName ;
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }

    this.dataGridView1.DataSource = dataSet.Tables[1] ;

}
```

Phuong thuc Add, Remove

```
DataSet dataset=new DataSet();
DataTable datatable=new DataTable("datatablename");
dataset.Tables.Add(datatable);
dataset.Tables.Remove(datatable);
```

xoa voi datatable duoc dat tên

```
dataset.Tables.Remove(datatablename);
dataset.Tables.RemoveAt(0);
```

phương thức Clear loại bỏ tất cả các đối tượng trong DataTable

```
dataset.Tables.Clear();
```

De dem so dong du lieu trong bang ta co the thuc hien

```
int sodong=dataset.Tables[0].Rows.Count;
```

2. Đối tượng DataTable

```
  private void button1_Click(object sender, EventArgs e)
        {
            string strQuery = "select top 10 * from tblEmployees";
//khoi tao doi tuong DataTable
            dataTable = new DataTable("Employees");
            try
            {
```

```
            SqlDataAdapter sqlDataAdapter = new
                SqlDataAdapter(strQuery, Connection.sqlConnection);
//dien du lieu vao datatable
            sqlDataAdapter.Fill(dataTable);



            sqlDataAdapter.Dispose();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error: " + ex.Message);
        }
//gan du lieu va dataGrid voi thuoc tinh DataSource
        this.dataGridView1.DataSource = dataTable;
        label1.Text= dataTable.TableName ;
    }



// thuoc tinh DataRow tra ve cac mau tin dang chua trong doi tuong DataTable
    private void button2_Click(object sender, EventArgs e)
    {
        if (dataTable != null)
```

```
        {
            string name = "";
            foreach (DataRow dataRow in dataTable.Rows)
            {
                name += Convert.ToString(dataRow[1]) + "\n";
            }
            label1.Text = name;
        }
    }
```

//thuoc tinh Columns tra ve tap doi tuong DataColumn bao gom danh sach cot du lieu cua bang chua trong doi tuong DataTable

```
    private void button3_Click(object sender, EventArgs e)
    {
        if (dataTable != null)
        {
            string name = "";
            foreach (DataColumn dataColumn in dataTable.Columns)
            {
                name += Convert.ToString(dataColumn.ColumnName) + "\n";
            }
            label1.Text = name;
        }
```

}

Ví dụ chúng ta sẽ ứng dụng 3 dối tượng trên vào việc, cập nhật và hiển thị dữ liệu cho bảng sản phẩm

Bước 1: tạo bảng cơ sở dữ liệu

Ví dụ chúng ta có một bảng dữ liệu tblIntrodure gồm các trường:

pkIntrodureID (int)

sTitle (nvarchar(300)

sSummary (nText)

iContent (nText)

iPosition (int)

Bước 2: tạo thủ tục StoreProcedure

ta tạo ra 3 thủ tục sql cho bảng giới thiệu của ta như sau

spIntrodure_insert - Thủ tục thêm mới dữ liệu

```
Create PROCEDURE spIntrodure_insert
    @sTitle nvarchar(100),
    @sSummary ntext,
    @sContent ntext,
    @iPosition int
AS
    insert into tblIntrodure(sTitle, sSummary, sContent, iPosition)
        values(@sTitle, @sSummary, @sContent, @iPosition)
    GO
```

spIntrodure_edit - Thủ tục sửa dữ liệu

```
Create PROCEDURE spIntrodure_edit
    @pkIntrodureID int,
    @sTitle nvarchar(100),
    @sSummary ntext,
    @sContent ntext,
    @iPosition int
AS
    update tblIntrodure set
        sTitle=@sTitle, sSummary=@sSummary, sContent=@sContent, iPosition=@iPosition
    where pkIntrodureID=@pkIntrodureID
    GO
```

spIntrodure_deletebyID - Thủ tục xoá dữ liệu

```
Create PROCEDURE spIntrodure_deletebyID
    @pkIntrodureID int
AS
    delete from tblIntrodure where pkIntrodureID=@pkIntrodureID
    GO
```

Chú ý: trên là cách tạo 3 thủ tục theo cú pháp của MSSQL nếu bạn tạo thủ tục SQL trong VS thì từ khoá Create sẽ chuyển thành Alter và GO chuyển thành Return

Bước 3: Tạo các lớp(nằm trong thư mục App_Code)

IntrodureInfo.cs

```
using System;


namespace iTechPro.Modules.Introdure
{
    public class IntrodureInfo
    {


        int _pkIntrodureID;
        public int pkIntrodureID
        {
            get { return _pkIntrodureID; }
            set { _pkIntrodureID = value; }
        }


        string _sTitle;
        public string sTitle
        {
            get { return _sTitle; }
            set { _sTitle = value; }
        }


        string _sImage;
```

```
public string sImage
{
    get { return _sImage; }
    set { _sImage = value; }
}

string _sSumary;
public string sSumary
{
    get { return _sSumary; }
    set { _sSumary = value; }
}

string _sComment;
public string sComment
{
    get { return _sComment; }
    set { _sComment = value; }
}

int _iPosition;
public int iPosition
```

```
        {
            get { return _iPosition; }
            set { _iPosition = value; }
        }
    }
}
```

IntrodureDB.cs (chứa tất cả phương thức xử lý và lấy dữ liệu cho bảng tblIntrodure)

```
using System;
using System.Data;
using System.Data.SqlClient;

using Website.Library;

namespace Website.Modules.Introdure
{
    public class IntrodureDB : ExcuteDataHelper
    {

        public IntrodureDB()
        {
            //
```

```
        // TODO: Add constructor logic here
        //
    }

    public static void Delete(string _pkIntrodureID)
    {
        string[] parameters = new string[] { "@pkIntrodureID"};
        string[] values = new string[] { _pkIntrodureID};
        executeData("spIntrodure_deletebyID", parameters, values);

    }

    public static void Insert(IntrodureInfo _introdure)
    {
        string[] parameters = new string[7] { "@sTitle", "@sImage", "@sSumary",
"@sComment", "@sPage", "@sLang", "@iPosition" };
        string[] values = new string[7] { _introdure.sTitle, _introdure.sImage,
_introdure.sSumary, _introdure.sComment, _introdure.sPage, _introdure.sLang,
_introdure.iPosition.ToString() };
        executeData("spIntrodure_insert", parameters, values);
    }

    public static void Update(IntrodureInfo _introdure)
    {
```

```
        string[] parameters = new string[7] { "@pkIntrodureID" ,"@sTitle", "@sImage",
"@sSumary", "@sComment", "@sPage", "@iPosition" };

         string[] values = new string[7] { _introdure.pkIntrodureID.ToString(),
_introdure.sTitle, _introdure.sImage, _introdure.sSumary, _introdure.sComment,
_introdure.sPage, _introdure.iPosition.ToString() };

        executeData("spIntrodure_edit", parameters, values);

    }



    public static void UpdateIndex(string _pkIntrodureID, string _giatri)

    {

         string ssql = "update tblIntrodure set iPosition=" + _giatri + " where
pkIntrodureID=" + _pkIntrodureID;

        executeData(ssql);

    }



    public static IntrodureInfo Getinfo(string _pkIntrodureID)

    {

         DataTable mydata = iTechProData.FillDatatable("spIntrodure_selectbyID",
"@pkIntrodureID", _pkIntrodureID);

        IntrodureInfo _introdure = new IntrodureInfo(); ;

        _introdure.sTitle = mydata.Rows[0]["sTitle"].ToString();

        _introdure.sImage = mydata.Rows[0]["sImage"].ToString();

        _introdure.sSumary = mydata.Rows[0]["sSumary"].ToString();

        _introdure.sComment = mydata.Rows[0]["sComment"].ToString();

        _introdure.sPage = mydata.Rows[0]["sPage"].ToString();
```

```
            _introdure.sLang = mydata.Rows[0]["sLang"].ToString();

            _introdure.iPosition = int.Parse(mydata.Rows[0]["iPosition"].ToString());

            return _introdure;

        }

    }

}
```

Tại lớp IntrodureDB này chúng ta sẽ kế thừa các phương thức thực thi dữ liệu từ lớp ExcuteDataHelper.cs

Lớp ExcuteDataHelper.cs

```
using System;

using System.Data;

using System.Data.SqlClient;




namespace Website.Library

{

    public class ExcuteDataHelper : iTechProData

    {

        //phuong thuc thuc thi du lieu(them moi, chinh sua, xoa) khi dua vao mot tham so sql

        #region executeData(string sql)"Thực thi dữ liệu"

        public static void executeData(string sql)

        {
```

```
        opendata();
        sqlcom = new SqlCommand(sql, sqlconn);
        try
        {
            sqlcom.ExecuteNonQuery();
            closedata();
        }
        catch (Exception exp)
        {
            closedata();
            HttpContext.Current.Response.Write(sql + "<br/>");
            HttpContext.Current.Response.Write("Có lỗi trong quá trình thực thi " +
exp.ToString());
        }
    }
    #endregion


    //phuong thuc thuc thi du lieu voi tham so dua vao
    #region executeData(string store, string[] Parameter, string[] Values)
    public static void executeData(string store, string[] Parameter, string[] Values)
    {
        opendata();
        sqlcom = new SqlCommand();
```

```
            sqlcom.CommandText = store;
            sqlcom.Connection = sqlconn;
            sqlcom.CommandType = CommandType.StoredProcedure;
            for (int i = 0; i < Parameter.Length; i++)
            {
                sqlcom.Parameters.AddWithValue(Parameter[i], Values[i]);
            }
            try
            {
                sqlcom.ExecuteNonQuery();
                closedata();
            }
            catch (DataException exp)
            {
                sqlconn.Close();
                HttpContext.Current.Response.Write(exp.ToString());
            }
        }
        #endregion
    }
}
```

Trong lớp này chúng ta có 2 phương thức thực thi dữ liệu có thể là thêm mới, chỉnh sửa hay xoá dữ liệu void executeData(string sql) cho phép bạn thực thi dữ liệu với một

chuỗi sql đưa vào còn executeData(string store, string[] Parameter, string[] Values) sẽ thực thi dữ liệu với hàm thủ tục từ SQL truyền vào với hai mảng giá trị và tham số và lơp này này thừa kế từ lớp dẫn xuất iTechProData.cs

Lớp iTechProData.cs

```
using System;

using System.Data;

using System.Configuration;

using System.Data.SqlClient;



namespace Website.Library

{

   public class WebsiteData

   {

      #region khai bao bien


      protected string ssql;

      protected static SqlConnection sqlconn;

      protected static SqlCommand sqlcom;

      protected static SqlDataAdapter sqladapter;

      protected static DataSet mydata;

      protected static SqlDataReader sqlreader;
```

```
#endregion


//phuong thuc mo du lieu
#region opendata() "Mở dữ liệu"
public static void opendata()
{
    //đọc chuỗi kết nối từ trong file web.config
        System.Configuration.AppSettingsReader settingsReader = new AppSettingsReader();
    string driver = (string)settingsReader.GetValue("hcubiudata", typeof(String));


    try
    {
        sqlconn = new SqlConnection(driver);
        if (sqlconn.State != ConnectionState.Open)
        {
            sqlconn.Open();
        }
    }
    catch (Exception exp)
    {
        HttpContext.Current.Response.Write("Lỗi mở dữ liệu" + exp.ToString());
    }
```

```
    }
    #endregion

    //phuong thuc dong du lieu
    #region closedata() "Đóng dữ liệu"
    public static void closedata()
    {
        if (sqlconn.State != ConnectionState.Closed)
        {
            sqlconn.Close();
            sqlconn.Dispose();
        }
    }
    #endregion

    // điền dữ liệu vào DataTable từ một thủ tục trong Database
public static DataTable FillDatatable(string store,string _thamso, string _giatri)
    {
        opendata();
        DataTable datatable = new DataTable();
        sqlcom = new SqlCommand();
        sqlcom.CommandText = store;
```

```
            sqlcom.Connection = sqlconn;
            sqlcom.Parameters.AddWithValue(_thamso, _giatri);
            sqlcom.CommandType = CommandType.StoredProcedure;
            try
            {
                sqladapter = new SqlDataAdapter(sqlcom);
                sqladapter.Fill(datatable);
                sqladapter.Dispose();
            }
            finally
            {
                closedata();
            }
            return datatable;
        }
    }
}
```

Trong lớp trên bạn thấy có 2 đối tượng data mới đó là DataAdapter và DataTable chúng ta sẽ học kỹ hơn trong phần sau trong ví dụ này các bạn chỉ cần hiểu qua là DataAdapter là bộ đọc dữ liệu từ nguồn dữ liệu, và DataTable là đối tượng lưu trữ dữ liệu không kết nối, nó như một bảng tạm để chứa dữ liệu và nó ko cần biết dữ liệu đó từ nguồn nào.

Đối tượng Dataset và DataTable-2

Bước 4: Tạo giao diện sử dụng

Code: adminIntrodure.aspx

```
<%@ Page Language="C#" MasterPageFile="~/admin.master" AutoEventWireup="true" CodeFile="adminIntrodure.aspx.cs" Inherits="Desktop_Introdure_adminIntrodure" Title="Admin - Introdure" %>



<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">

<!--Trinh bay du lieu-->

 <table cellpadding="0" cellspacing="0" width="100%" style="padding-right:3px; height:390px">


  <tr>

   <td style="padding:15px 15px 15px 15px" valign="top">


<table width="100%" cellpadding="0" cellspacing="0">

    <tr>

     <td align="left" class="hcubiufontlarger">Giới thiệu</td>

    </tr>

    <tr><td style="height:15px;"></td></tr>
```

```
<!--start them moi-->
<tr>
 <td align="left">
    <asp:Panel ID="panelupdate" Width="100%" runat="Server" Visible="false">
        <table width="100%" style="padding-left:20px;">
         <tr>
          <td colspan="2"><b>Cập nhật thông tin giới thiệu</b></td>
         </tr>
         <tr><td style="width: 78px; height:15px;"></td></tr>
         <tr>
           <td align="left" style="width: 78px">Tiêu đề</td>
             <td align="left"><input type="text" name="txtTitle" id="txtTitle" runat="server" style="width: 329px" /></td>
         </tr>
         <tr>
           <td valign="middle" align="left" style="width: 78px">Tóm tắt</td>
           <td align="left">

                <asp:TextBox ID="txtTomtat" runat="server" TextMode="MultiLine"></asp:TextBox>

           </td>
         </tr>
```

```
<tr>
  <td align="left" style="height: 88px; width: 78px;">Nội dung</td>
  <td align="left" style="height: 88px">

    <asp:TextBox ID="txtNoidung" runat="server" TextMode="MultiLine" />

  </td>
</tr>
<tr>
 <td align="left">Vị trí</td>
 <td align="left">
  <asp:TextBox ID="txtvitri" runat="server" Text="1"></asp:TextBox>
        <asp:RangeValidator ID="RangeValidator1" runat="server" ControlToValidate="txtvitri"
        ErrorMessage="Vị trí phải là kiểu số" MaximumValue="100" MinimumValue="0" Type="Integer"></asp:RangeValidator></td>
</tr>
<tr><td style="width: 78px; height:15px;"></td></tr>
<tr>
 <td colspan="2" align="left">
   <asp:Button ID="btnaccept" runat="server" Text="Ghi" Width="100px" OnClick="btnaccept_Click"/>
  <asp:Button ID="btcancel" runat="server" Text="Bỏ qua" Width="100px" OnClick="btcancel_Click" />
```

```
                    <asp:Label ID="lblidintro" runat="server" Text="" Visible="false"></asp:Label></td>

            </tr>

          </table>

         </asp:Panel>


<!--End them moi-->


    </td>

   </tr>


   <tr>

    <td  style="height:5px;"></td>

   </tr>


   <asp:Panel ID="panelview" runat="server">


   <tr>

     <td  align="left"  style="padding-bottom:3px;"><asp:LinkButton  ID="btnaddnew" CssClass="linkbutton" runat="server" Text="Thêm mới" OnClick="btnaddnew_Click" /></td>

   </tr>


   <tr>
```

```
<td valign="top" align="left">
<asp:DataGrid id="gridintro" runat="server"
    BorderColor="black"
    Width="100%"
    BorderWidth="1"
    CellPadding="3"
    Font-Size="10pt"
    HeaderStyle-BackColor="#aaaadd"
    OnItemCommand="gridintro_OnItemCommand"
    AutoGenerateColumns="false">
    <HeaderStyle BackColor="#AAAADD"></HeaderStyle>
    <Columns>
        <asp:TemplateColumn HeaderStyle-HorizontalAlign="Center" ItemStyle-HorizontalAlign="Center" HeaderStyle-Width="80px" HeaderText="STT">
        <ItemTemplate>
         <%#Container.ItemIndex +1 %>
         </ItemTemplate>
       </asp:TemplateColumn>
        <asp:BoundColumn HeaderStyle-HorizontalAlign="Left" ItemStyle-HorizontalAlign="Left" DataField="sTitle" ReadOnly="true" HeaderText="Tiêu đề"></asp:BoundColumn>

      <asp:TemplateColumn HeaderText="Vị trí" ItemStyle-HorizontalAlign="Center" HeaderStyle-HorizontalAlign="Center" HeaderStyle-Width="100px" ItemStyle-Width="100px" ItemStyle-Height="24px" >
```

```
<ItemTemplate>

<asp:TextBox ID="txtVitri" Width="39px" runat="server" Text='<%#Eval("iPosition") %>' />

</ItemTemplate>

</asp:TemplateColumn>


<asp:TemplateColumn HeaderText="Chỉnh sửa" ItemStyle-HorizontalAlign="Center" HeaderStyle-Width="80px" HeaderStyle-HorizontalAlign="Center" ItemStyle-Width="100px" ItemStyle-Height="24px" >

<ItemTemplate>

<asp:LinkButton ID ="Edit" CommandArgument ='<%#DataBinder.Eval(Container,"DataItem.pkIntrodureID")%>' runat ="server" CommandName="Edit" Text ="Edit"></asp:LinkButton>

</ItemTemplate>

</asp:TemplateColumn>


<asp:TemplateColumn HeaderText="Xóa" HeaderStyle-HorizontalAlign="Center" HeaderStyle-Width="80px" ItemStyle-HorizontalAlign="Center" ItemStyle-Width="100px" ItemStyle-Height="24px" >

<ItemTemplate>

<asp:LinkButton ID ="Delete" CommandArgument ='<%#DataBinder.Eval(Container,"DataItem.pkIntrodureID")%>' runat ="server" CommandName="Delete" Text ="Delete"></asp:LinkButton>

</ItemTemplate>

</asp:TemplateColumn>


</Columns>
```

```
    </asp:DataGrid>
   </td>
  </tr>
  <tr>
   <td align="right" style="padding-top:3px;">
    <asp:Label ID="lblthongbao" runat="server"></asp:Label>
      <asp:LinkButton ID="lbncapnhatvitri" CssClass="linkbutton" runat="server" Text="Cập nhật vị trí" OnClick="lbncapnhatvitri_Click" />
   </td>
  </tr>


  </asp:Panel>
 </table>


   </td>
  </tr>
  <tr><td style="height:30px;"></td></tr>
 </table>


</asp:Content>
```

Code adminIntrodure.aspx.cs

```
using System;
```

```
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Website.Library;
using Website.Modules.Introdure;

public partial class Desktop_Introdure_adminIntrodure : System.Web.UI.Page
{
    string ssql;
    void Loaddatagrid()
    {
        ssql = "select pkIntrodureID,sTitle,iPosition from tblIntrodure";
        DatagridHelper.fill_datagrid(gridintro, ssql, "pkIntrodureID");
        foreach (DataGridItem item in this.gridintro.Items)
        {
            LinkButton lbn = (LinkButton)this.gridintro.Items[item.ItemIndex].FindControl("Delete");
```

```
        lbn.Attributes.Add("onclick", "javascript:return confirm('Bạn có chắc chắn xoá mục giới thiệu này')");
    }
}

protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        Loaddatagrid();
    }
}

private IntrodureInfo Getcontent()
{
    IntrodureInfo intro = new IntrodureInfo();
    try
    {
        intro.pkIntrodureID = int.Parse(lblidintro.Text);
    }
    catch
    {
    }
```

```
        intro.sTitle = txtTitle.Value;
        intro.sSumary = txtTomtat.Text;
        intro.sContent = txtNoidung.Text;
        intro.iPosition = int.Parse(txtvitri.Text);
        return intro;
    }

    protected void btnaddnew_Click(object sender, EventArgs e)
    {
        panelupdate.Visible = true;
        panelview.Visible = false;
        txtNoidung.Text = "";
        txtTitle.Value = "";
        this.txtTomtat.Text = "";
        txtvitri.Text = "1";
        btnaccept.Text = "Ghi";
    }

        protected void gridintro_OnItemCommand(object sender,
DataGridCommandEventArgs e)
    {
        lblidintro.Text = e.CommandArgument.ToString();
        if (e.CommandName == "Edit")
```

```
    {
        IntrodureInfo introdure = IntrodureDB.Getinfo(lblidintro.Text);
        txtTitle.Value = introdure.sTitle;
        txtTomtat.Text = introdure.sSumary;
        txtvitri.Text = introdure.iPosition.ToString();
        txtNoidung.Text = introdure.sContent;
        btnaccept.Text = "Cập nhật";
        panelupdate.Visible = true;
        panelview.Visible = false;
    }
    else
    {
        IntrodureDB.Delete(lblidintro.Text);
        Loaddatagrid();
    }
}

protected void btnaccept_Click(object sender, EventArgs e)
{
    IntrodureInfo introdure = Getcontent();
    if (btnaccept.Text == "Ghi")
    {
```

```
        IntrodureDB.Insert(introdure);
    }
    else
    {
        IntrodureDB.Update(introdure);
    }
    panelupdate.Visible = false;
    panelview.Visible = true;
    Loaddatagrid();
        }

protected void btcancel_Click(object sender, EventArgs e)
{
    panelview.Visible = true;
    panelupdate.Visible = false;
    Loaddatagrid();
}

protected void lbncapnhatvitri_Click(object sender, EventArgs e)
{
    foreach (DataGridItem item in gridintro.Items)
    {
```

```
                    TextBox txt =
(TextBox)this.gridintro.Items[item.ItemIndex].FindControl("txtVitri");

            IntrodureDB.UpdateIndex(gridintro.DataKeys[item.ItemIndex].ToString(),
txt.Text);

        }

    }

}
```

Bài 6 : Thảo luận các bước xây dựng Website

Các bước xây dựng website

Tiếp cận dự án (Initiation)

Tìm hiểu yêu cầu thực tế của khách hàng. Xác nhận các yêu cầu của khách hàng về trang WEB và các thông tin khách hàng cung cấp.

Phân tích yêu cầu (Analisys, planning)

Dựa vào những thông tin và yêu cầu thiết kế của khách hàng, chúng tôi phân tích rõ ràng mục đích, yêu cầu, nguyện vọng ... của khàch hàng. Lập kế hoạch thực hiện dự án, Căn cứ trên kết quả khảo sát, thông tin do khách hàng cung cấp chúng tôi thiết kế lập cấu trúc cho WEB Site và thiết kế chung cho toàn hệ thống.

Đặc tả yêu cầu (Requirement Specs)

Chúng tôi tiến hành các bản đặc tả chi tiết về yêu cầu để chuyển sang giai đoạn thiết kế.

Thiết kế (System Design)

Dựa trên các bản đặ tả yêu cầu, chuyên gia thiết kế của chúng tôi sẽ tiến hành thiết kế bản giao diện và các chức năng của hệ thống

Thiết kế kế hệ thống Phát triển hệ thống (Development)

Tại bước này dựa trên các thiết kế, chuyên gia lập trình của chúng tôi sẽ tiến hành xây dựng và hoàn thiện hệ thống.

Kiểm thử (Testing)

Khi hệ thống được xây dựng xong, chúng tôi bắt đầu quá trình kiểm thử từng module chức năng, cũng như toàn bộ hệ thống. Đảm bảo việc loại bỏ các lỗi còn sót lại.

Chuyển giao (Delivery)

Hệ thống được hoàn tất, chúng tôi chuyển giao lại cho khách hàng, đào tạo sử dụng và hỗ trợ kỹ thuật cần thiết.

Hỗ trợ (Support)

Chuyển giao các tài liệu, hướng dẫn sử dụng. Hỗ trợ các vấn đề kỹ thuật phát sinh khi sử dụng hệ thống. Chúng tôi cũng thường xuyên đánh giá và bảo trì hệ thống.

Tham gia đóng góp

Tài liệu: Công nghệ Web và Ứng dụng

Biên tập bởi: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://voer.edu.vn/c/bac5238b

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Giới thiệu về môn học

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/a99c5320

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Lịch sử Phát triển công nghệ web

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/9aba02d5

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Khái quát về công nghệ web

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/c16da5e4

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Những thành phần cấu tạo nên một website

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/1b74a5e7

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Giới thiệu về HTML

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/847b621b

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Thành phần của HTML

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/ea5d608b

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Các loại thẻ nâng cao trong HTML

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/9765f7fe

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Cơ bản về cascading style sheets

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/45100eca

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Các loại Style trong ứng dụng website

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/591eea66

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Giới thiệu về javascript

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/f76231ff

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Nhúng mã-Cách khai báo biến

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/100f27d8

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Các điều khiển

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/85707e18

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Đối tượng navigator- window- location-frame

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/ad519577

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Đối tượng Document- Anchors- Forms- History

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/83712707

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Xây dựng các hàm và sự kiện trong javascript

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/c692b189

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Xây dựng đối tượng trong javascript

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/6b89364e

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Truy cập thành phần dữ liệu

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/ef00c5aa

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Giới thiệu về DHTML

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/70cb5324

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Cơ bản về trang web động

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/760850f0

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Cấu trúc một trang web

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/4398b11a

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Ngôn ngữ cơ bản thường dùng

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/b29a4d75

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Giới thiệu cơ bản về ngôn ngữ ASP.NET

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/0da1b890

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Cách thức thực thi một trang web động

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/ded656eb

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Điều khiển cơ bản (Standard control)

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/ac5b80cc

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Điều khiển hợp lệ dữ liệu – Validation

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/41bf893d

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Điều khiển làm việc với CSDL (Data Control)

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/a13e1513

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Điều khiển HTML

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/4da098d0

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Các điều khiển khác

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/95b761c8

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Respone-Requesy-Server-Cookie

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/c93615ec

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Session-Sử dụng Profiles

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/3bf549bd

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Tầm quan trọng của việc sử dụng cơ sở dữ liệu

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/37dc4eba

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Kiến trúc của ADO.NET và Các đối tượng

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/1d011e63

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Đối tượng Dataset và DataTable-1

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/1ba9dbc2

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Đối tượng Dataset và DataTable-2

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/51ba5f43

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Các bước xây dựng website

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/22cb14ae

Chương trình Thư viện Học liệu Mở Việt Nam

Chương trình Thư viện Học liệu Mở Việt Nam (Vietnam Open Educational Resources – VOER) được hỗ trợ bởi Quỹ Việt Nam. Mục tiêu của chương trình là xây dựng kho Tài nguyên giáo dục Mở miễn phí của người Việt và cho người Việt, có nội dung phong phú. Các nội dung đều tuân thủ Giấy phép Creative Commons Attribution (CC-by) 4.0 do đó các nội dung đều có thể được sử dụng, tái sử dụng và truy nhập miễn phí trước hết trong trong môi trường giảng dạy, học tập và nghiên cứu sau đó cho toàn xã hội.

Với sự hỗ trợ của Quỹ Việt Nam, Thư viện Học liệu Mở Việt Nam (VOER) đã trở thành một cổng thông tin chính cho các sinh viên và giảng viên trong và ngoài Việt Nam. Mỗi ngày có hàng chục nghìn lượt truy cập VOER (www.voer.edu.vn) để nghiên cứu, học tập và tải tài liệu giảng dạy về. Với hàng chục nghìn module kiến thức từ hàng nghìn tác giả khác nhau đóng góp, Thư Viện Học liệu Mở Việt Nam là một kho tàng tài liệu khổng lồ, nội dung phong phú phục vụ cho tất cả các nhu cầu học tập, nghiên cứu của độc giả.

Nguồn tài liệu mở phong phú có trên VOER có được là do sự chia sẻ tự nguyện của các tác giả trong và ngoài nước. Quá trình chia sẻ tài liệu trên VOER trở lên dễ dàng như đếm 1, 2, 3 nhờ vào sức mạnh của nền tảng Hanoi Spring.

Hanoi Spring là một nền tảng công nghệ tiên tiến được thiết kế cho phép công chúng dễ dàng chia sẻ tài liệu giảng dạy, học tập cũng như chủ động phát triển chương trình giảng dạy dựa trên khái niệm về học liệu mở (OCW) và tài nguyên giáo dục mở (OER) . Khái niệm chia sẻ tri thức có tính cách mạng đã được khởi xướng và phát triển tiên phong bởi Đại học MIT và Đại học Rice Hoa Kỳ trong vòng một thập kỷ qua. Kể từ đó, phong trào Tài nguyên Giáo dục Mở đã phát triển nhanh chóng, được UNESCO hỗ trợ và được chấp nhận như một chương trình chính thức ở nhiều nước trên thế giới.