



BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH

ThS. NGUYỄN MINH ĐẠO



GIÁO TRÌNH

LẬP TRÌNH WEB VỚI ASP.NET

(Giáo trình dành cho sinh viên ngành Công nghệ Thông tin)



NHÀ XUẤT BẢN
ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT
THÀNH PHỐ HỒ CHÍ MINH

GVC. ThS. NGUYỄN MINH ĐẠO

GIÁO TRÌNH
**LẬP TRÌNH WEB
với ASP.NET**

(Giáo trình dùng cho sinh viên ngành Công nghệ Thông tin)

NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA
THÀNH PHỐ HỒ CHÍ MINH – 2014

GIÁO TRÌNH

LẬP TRÌNH WEB VỚI ASP.NET

Nhà xuất bản ĐHQG-HCM và tác giả/đối tác liên kết giữ bản quyền ©

Copyright © by VNU-HCM Publishing House and author/co-partnership

All rights reserved



TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TPHCM

Xuất bản năm 2014

LỜI NÓI ĐẦU

ASP.NET là một nền tảng ứng dụng web (*web application framework*) được xây dựng, phát triển bởi tập đoàn Microsoft, cho phép các lập trình viên tạo ra những trang web động, những ứng dụng web và những dịch vụ web. Vào tháng 2 năm 2002, phiên bản đầu tiên được đưa ra thị trường cùng với phiên bản 1.0 của .NET framework, ASP.NET là công nghệ nối tiếp của công nghệ Microsoft's Active Server Pages (ASP) trước đó, được biên dịch dưới dạng Common Language Runtime (CLR), cho phép những người lập trình viết mã ASP.NET với bất kỳ ngôn ngữ nào (C#, VB.NET,...) được hỗ trợ bởi .NET framework.

Với xu hướng các ứng dụng web được phát triển mạnh mẽ, ngày càng có nhiều ứng dụng viết bằng ASP.NET, rất nhiều công ty đang chọn ASP.NET làm ngôn ngữ phát triển website cho khách hàng cũng như sản phẩm của riêng họ nên việc nắm vững và sử dụng thành thạo về công nghệ web nói chung và chuyên sâu về ASP.NET nói riêng sẽ là một lợi thế cho các sinh viên và các lập trình viên ứng dụng web.

Giáo trình được biên soạn nhằm giới thiệu công nghệ lập trình ASP.NET 3.5 chạy trên bộ phần mềm ứng dụng Visual Studio.NET 2008 với phiên bản 3.5 của .NET framework. Nội dung giáo trình trình bày các bước từ cơ bản đến chuyên sâu trong việc sử dụng công nghệ ASP.NET để xây dựng ứng dụng web. Sau khi học xong môn Lập trình ứng dụng Web với ASP.NET 3.5, sinh viên sẽ có kiến thức để hiểu và vận dụng được những kỹ năng trong việc tích hợp các điều khiển và các công nghệ liên quan để xây dựng các ứng dụng web trong thực tiễn.

Giáo trình gồm có 17 chương, bao quát hầu hết các vấn đề cốt lõi của công nghệ lập trình web với ASP.NET 3.5. Phần đầu mỗi chương đều có tóm tắt nội dung chương và cuối mỗi chương là phần bài tập để sinh viên có thể tự kiểm tra kiến thức của mình. Tuy nhiên, giáo trình chắc chắn không tránh khỏi những thiếu sót. Tác giả xin chân thành cảm ơn và mong muốn nhận được những góp ý của giảng viên, sinh viên và những ai quan tâm để giáo trình ngày càng hoàn thiện hơn.

Mọi ý kiến đóng góp xin vui lòng gửi theo địa chỉ email:
daonm@fit.hcmute.edu.vn

Tác giả

MỤC LỤC

LỜI NÓI ĐẦU.....	3
MỤC LỤC	5

Chương 1. Lập trình ứng dụng Web với công nghệ ASP.NET	9
1.1. Sự cần thiết phải triển khai ứng dụng Web.....	9
1.2. Giới thiệu về ASP và ASP.NET	14
1.3. Web Server IIS.....	21
1.4. Tạo ứng dụng Web với ASP.NET	23
1.5. Khảo sát giao diện Visual Studio.NET 2008	26
Bài tập chương 1	30
Chương 2. Tìm hiểu và sử dụng các điều khiển Controls.....	33
2.1. Cấu trúc một trang ASP.NET	33
2.2. Biến cố trang ASP.NET	41
2.3. Giới thiệu ASP.NET Server Controls	44
2.4. HTML Server Controls	45
2.5. ASP.NET Server Controls	78
2.6. Sự kiện PageLoad và thuộc tính IsPostBack	115
2.7. Thuộc tính AutoPostBack của một số Web Server Controls....	120
Bài tập chương 2	125
Chương 3. Master Page – Web Navigation	135
3.1. Master Page.....	135
3.2. Web Navigation	148
3.3. Web User Control	160
3.4. Đối tượng Request, Response và Server.....	172
Bài tập chương 3	182

Chương 4. Quản lý trạng thái.....	191
4.1. Vấn đề trạng thái.....	192
4.2. Xem trạng thái (View State)	192
4.3. Chuyển thông tin giữa các trang	195
4.4. Đối tượng Cookies	202
4.5. Đối tượng Session	206
4.6. Đối tượng Application	209
4.7. Tập tin Global.asax	212
4.8. Tập tin Web.config	213
Bài tập chương 4	216
Chương 5. Sử dụng các Validation Controls.....	221
5.1. RequiredFieldValidator.....	223
5.2. Điều khiển RangeValidator.....	226
5.3. Điều khiển CompareValidator	229
5.4. Điều khiển RegularExpressionValidator	234
5.5. Điều khiển Custom Validator	238
5.6. Điều khiển ValidationSummary	241
Bài tập chương 5	248
Chương 6. Các đối tượng dữ liệu (Rich Controls – Login	253
6.1. Điều khiển hiển thị các trang khác nhau MultiView	253
6.2. Điều khiển Wizard	264
6.3. Nhóm Điều khiển Login	274
Bài tập chương 6	282
Chương 7. Giới thiệu ADO.NET	299
7.1. Kiến trúc ADO.NET	299
7.2. Tìm hiểu trình cung cấp dữ liệu của ADO.NET	301
7.3. Các namespace của ADO.NET	303

7.4. Tìm hiểu cơ chế kết nối của ADO.NET qua Connected Layer.....	309
7.5. Disconnected Layer.....	343
7.6. Đối tượng dữ liệu SqlDataSource.....	356
Bài tập chương 7	364
 Chương 8. Tìm hiểu và ứng dụng cơ chế Data Binding	369
8.1. Giới thiệu Data Binding	369
8.2. Các dạng Data Binding.....	370
Bài tập chương 8	377
 Chương 9. Các đối tượng dữ liệu (Data Controls).....	385
9.1. Đối tượng dữ liệu GridView.....	385
9.2. Đối tượng dữ liệu DetailsView	411
9.3. Đối tượng dữ liệu FormView.....	415
9.4. Đối tượng dữ liệu DataList	419
9.5. Đối tượng dữ liệu Repeater.....	423
Bài tập chương 9	426
 Chương 10. Bảo mật các ứng dụng Web	431
10.1. Giới Thiệu về Bảo Mật Trong ASP.NET.....	431
Thí dụ minh họa	433
 Chương 11. Giới thiệu AJAX	439
11.1. Giới thiệu Ajax.....	439
11.2. Ajax làm việc như thế nào?	440
11.3. ASP.NET Ajax & ASP.NET call back.....	449
11.4. ASP.NET Ajax Server Controls.....	450
11.5. Giới thiệu Ajax ControlToolkit	454
11.6. Tìm hiểu một số điều khiển trong Ajax Control Toolkit 3.5....	459
Bài tập chương 11	522

Chương 12. Lập trình LINQ	541
12.1. Giới thiệu LINQ.	541
12.2. Các khái niệm cơ bản	544
12.3. LINQ to Objects	549
12.4. LINQ to DataSet.....	550
12.5. LINQ to SQL.....	553
Chương 13. Lập trình Web Services	561
13.1. Giới thiệu Web services.	561
13.2. Kiến trúc và các thành phần Web services	562
13.3. Xây dựng ứng dụng Web Service.....	568
Chương 14. Xây dựng Website bán hàng.....	575
14.1. Thương mại điện tử.....	575
14.2. Khảo sát hiện trạng	580
14.3. Kết luận	586
Chương 15. Thiết kế Website bán hàng – Mức dữ liệu	587
Chương 16. Thiết kế Website bán hàng – Mức xử lý.....	607
Chương 17. Thiết kế Website bán hàng – Mức trình diễn	633
TÀI LIỆU THAM KHẢO.....	658

Chương I

LẬP TRÌNH ỨNG DỤNG WEB VỚI ASP.NET

Kết thúc chương này các bạn có thể:

- *Nắm vững khái niệm ứng dụng Web*
- *Hiểu rõ về các khái niệm cơ bản về Web, về mô hình kiến trúc Client-Server 2 lớp (two tier) và 3 lớp (three tier)*
- *Nhận biết các ngôn ngữ lập trình ứng dụng Web: HTML, DHTML, VBScript, JavaScript*
- *Cài đặt được Web Server IIS trên máy chạy hệ điều hành Windows*
- *Trình bày được các đặc điểm của ASP và ASP.NET*
- *Sử dụng được Visual Studio.Net 2008 để tạo ứng dụng Web*

1.1. SỰ CẦN THIẾT PHẢI TRIỂN KHAI ỨNG DỤNG WEB

Với sự phát triển nhanh chóng và mạnh mẽ của ngành công nghệ thông tin, đặc biệt là sự phát triển các hệ thống mạng intranet, internet, trong các lĩnh vực như: thương mại, y tế, giáo dục, nhu cầu trao đổi thông tin thực sự là rất cần thiết, giúp cho công việc được triển khai dễ dàng, chính xác, với tốc độ nhanh và tiết kiệm chi phí, thông tin được cập nhật kịp thời. Từ đó vấn đề đặt ra ở đây là chúng ta cần phải có một ứng dụng cho phép trao đổi thông tin mọi lúc, mọi nơi, dễ sử dụng,... thông qua môi trường mạng.

Ứng dụng Web đáp ứng được các yêu cầu đặt ra với các lý do sau:

- ✓ Dễ dàng trao đổi và chia sẻ thông tin qua mạng
- ✓ Sử dụng giao diện đồ họa giúp cho người dùng dễ sử dụng
- ✓ Hỗ trợ về multimedia như: hình ảnh, âm thanh, phim ảnh,...
- ✓ Hỗ trợ nhiều chương trình duyệt web (web-browser) để truy cập Web
- ✓ Hỗ trợ truy cập web trên các thiết bị di động: PocketPC, SmartPhone,...
- ✓ Hỗ trợ nhiều ngôn ngữ để phát triển Web: ASP, ASP.NET, JSP, PHP,...

➤ Web client (Browser)

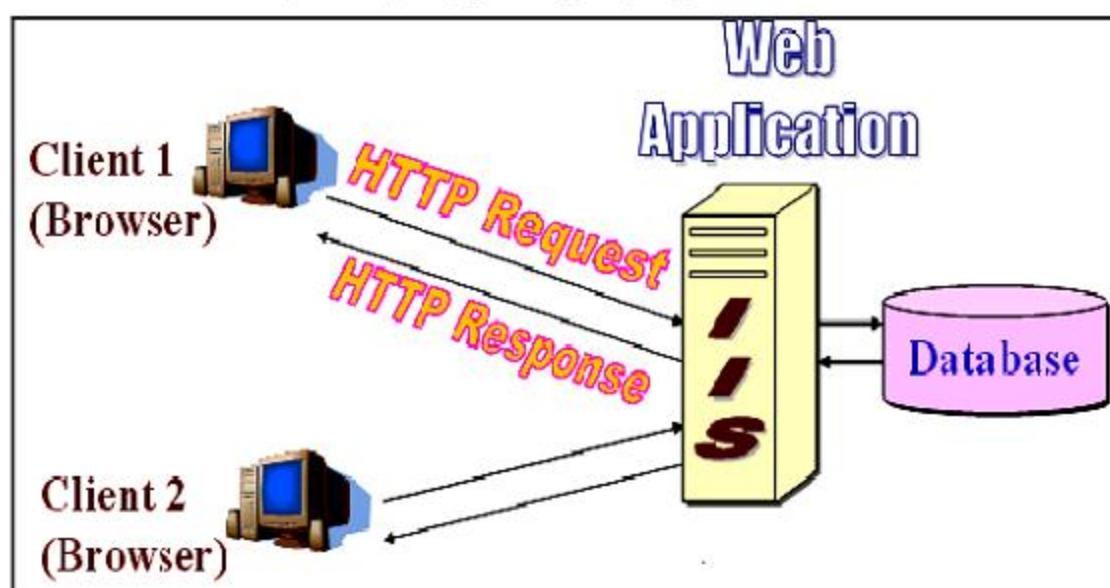
Các máy khách (Client) sẽ sử dụng chương trình để triệu gọi đến các trang web gọi là trình duyệt web hay Web browser. Hiện nay, có rất nhiều trình duyệt web thông dụng như: Internet Explorer, Mozilla FireFox, Google Chrome, Opera,..

➤ Web server

Các máy chủ (Server) chứa các ứng dụng Web, sẵn sàng truy xuất các trang web hay các tài liệu và gửi về cho client khi nhận được yêu cầu từ phía client. Hiện nay có rất nhiều Web server và chạy trên nhiều hệ thống như: Apache, Microsoft, Sun,...

➤ Giao thức HTTP

Quá trình giao tiếp giữa client và server được thực hiện thông qua giao thức chuẩn HTTP (HyperText Transfer Protocol). Hình minh họa sau mô tả việc truy cập ứng dụng Web.



Hình 1.1: Minh họa truy cập ứng dụng Web

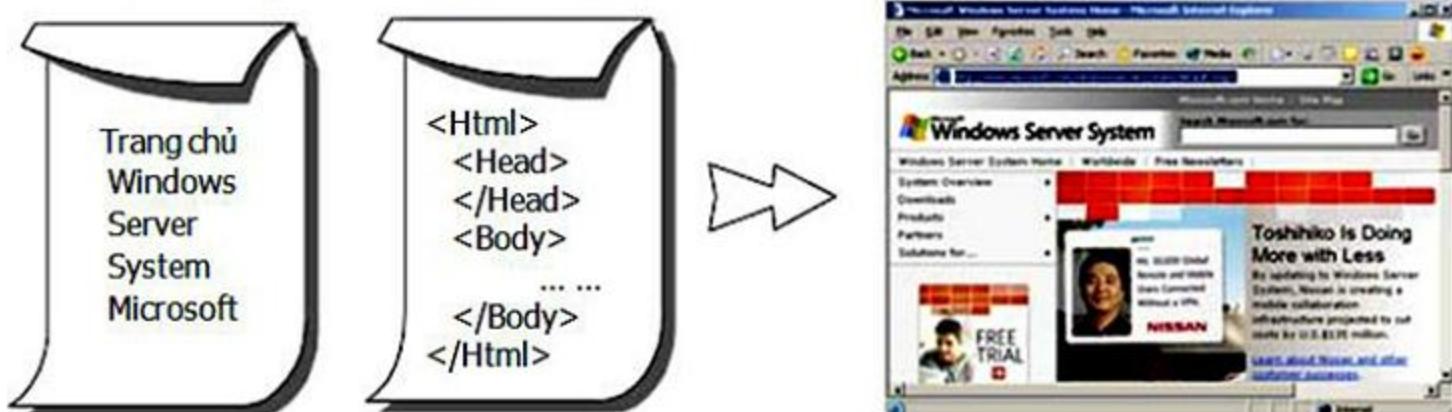
- *Ứng dụng Web được xây dựng theo mô hình Khách-Chủ (Client-Server)*
- *Quá trình giao tiếp giữa client và server được thực hiện thông qua giao thức chuẩn của Web là HTTP (HyperText Transfer Protocol).*
- *Mô hình Client-Server là mô hình Three Tier gồm ba thành phần chính là: máy khách (client tier), máy phục vụ (Web server tier) và máy chủ dữ liệu (data server tier). Máy phục vụ (Web server tier) sẽ chứa các ứng dụng Web và các ứng dụng Web này sẽ được quản lý tập trung bởi trình quản lý gọi là Web Server (IIS, Tomcat, JBoss, WebLogic,...). Các máy khách (client tier) truy cập đến ứng dụng web sử dụng trình duyệt web (Web browser). Hình 1.1 trên minh họa một kiến trúc ứng dụng Web với công nghệ của Microsoft với Webserver là IIS, và tầng máy chủ dữ liệu thường là cơ sở dữ liệu MS Access hoặc MS SQL Server.*

- Client sử dụng giao thức HTTP Request để gửi yêu cầu (trang web) lên Server, Server xử lý và sử dụng giao thức HTTP Response để gửi kết quả về cho Client.

➤ Ngôn ngữ đánh dấu siêu văn bản HTML (HyperText Transfer Markup Language)

- Ngôn ngữ chuẩn được sử dụng để thiết kế một trang Web. HTML cho phép người viết có thể phân chia và trình bày thông tin trên một trang tin. HTML đơn giản, dễ học. HTML đơn giản là tập tin có phần mở rộng .htm (hay .html), sử dụng các thẻ (tag): kiểu văn bản, danh sách, các siêu liên kết (hyperlinks),... Chúng ta có thể dùng các phần mềm thiết kế web như: FrontPage, DreamWeaver,... để thiết kế các trang HTML.
- Ngôn ngữ đánh dấu HTML sử dụng các thẻ ký hiệu quy định sẵn (được gọi là tag) để trình bày nội dung văn bản.

Nội dung + Định dạng = Kết quả hiển thị



Hình 1.2: Trang siêu văn bản HTML

Thí dụ 1.1: Nội dung trang web **Hello.html**

```
<Html>
  <Head>
    <Title>Trang ASP.NET</Title>
  </Head>
  <Body>
    <P align="center">
      <FONT size="12"> Chào các bạn đã đến với ASP.Net 3.5!!! </FONT>
    </P>
  </Body>
</Html>
```

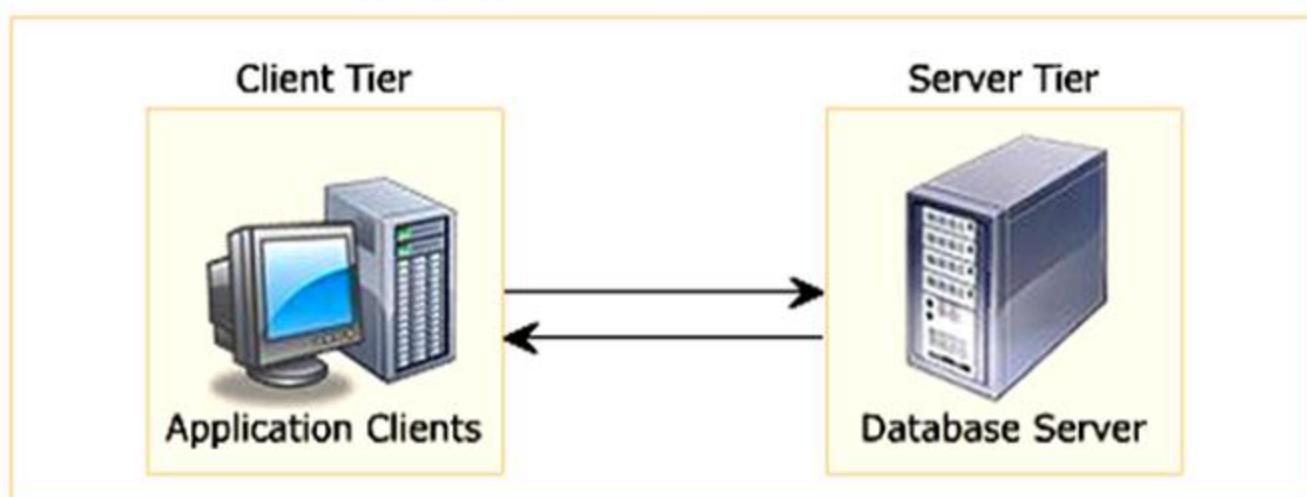
➤ Client Scripting và Server Scripting

Các ngôn ngữ dùng để viết mã lệnh cho trang web. Một trang web được xử lý ở server và trả kết quả về cho client. Do đó, các ngôn ngữ viết mã cho trang web được chia thành hai dạng:

- **ClientScript:** được xử lý tại trình duyệt (Web browser) trên máy client. Các ngôn ngữ thường dùng để viết là: VBScript, JavaScript, DHTML,...
 - JavaScript là ngôn ngữ phổ biến sử dụng nhiều nhất hiện nay. JavaScript được dùng để thực hiện một số tác vụ như kiểm tra thông tin nhập vào, kiểm tra trình duyệt,...
 - DHTML: là sự kết hợp của HTML, Style Sheet (CSS) và JavaScript nhằm làm cho trang web dễ tương tác, điều khiển và giảm bớt việc xử lý phía Server.
 - VBScript là ngôn ngữ script của Microsoft. Chức năng của VBScript cũng giống như JavaScript.
 - Sau này có thêm thư viện JavaScript là JQuery rất mạnh được thiết kế để đơn giản hóa lập trình phía máy client.
- **Server Scripting:** được xử lý tại Web server trên máy Server. Các ngôn ngữ dùng để viết là: ASP, ASP.NET, PHP, JSP,... Trong chương này, chúng ta sẽ tìm hiểu ngôn ngữ ASP và ASP.NET.

➤ Các mô hình ứng dụng

- **Mô hình ứng dụng hai tầng**



Hình 1.3: Mô hình hai tầng (Two Tier)

Đây là một dạng mô hình đơn giản, khá phổ biến của một ứng dụng phân tán. Trong mô hình này, việc xử lý dữ liệu được thực hiện trên máy chủ dữ liệu Database Server, việc nhận và hiển thị dữ liệu được thực hiện ở Client. Mô hình này dùng cho ứng dụng lập trình trên Windows.

➤ Ưu điểm

- ✓ Dữ liệu tập trung → đảm bảo dữ liệu được nhất quán.
- ✓ Dữ liệu được chia sẻ cho nhiều người dùng.

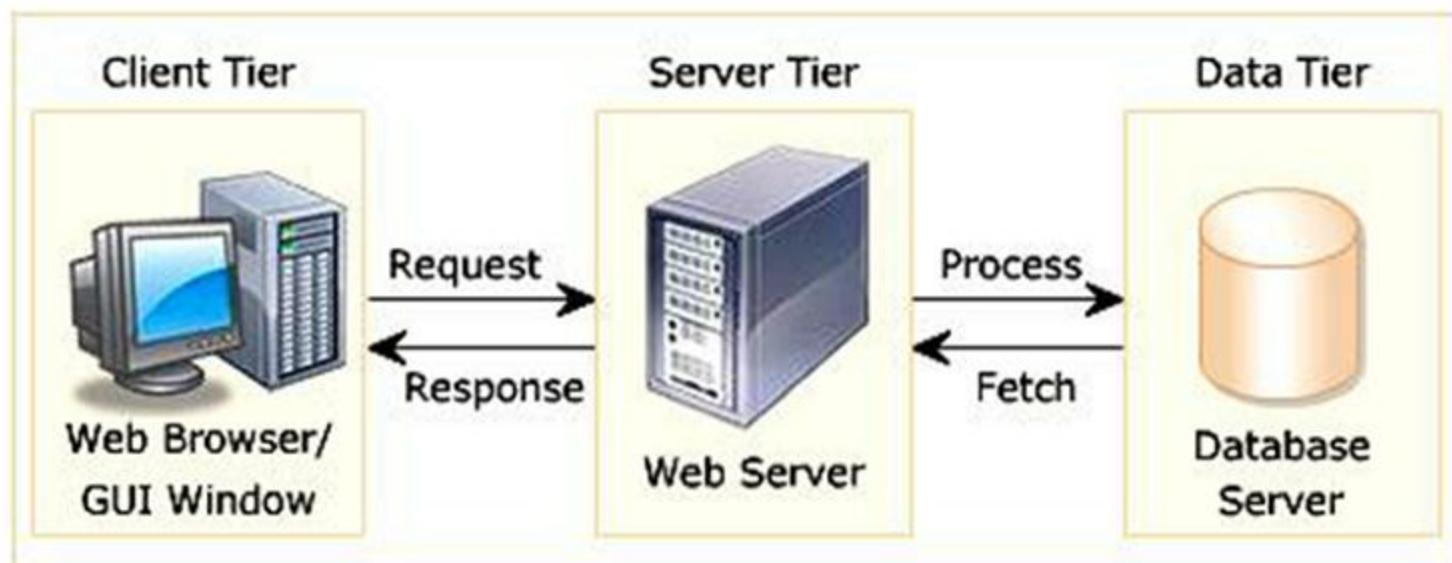
➤ Nhược điểm

- ✓ Các xử lý tra cứu và cập nhật dữ liệu được thực hiện ở Database Server, việc nhận kết quả và hiển thị phải được thực hiện ở Client -> Khó khăn trong vấn đề bảo trì và nâng cấp.
- ✓ Khối lượng dữ liệu truyền trên mạng lớn -> chiếm dụng đường truyền, thêm gánh nặng cho Database Server.

▪ Mô hình ứng dụng ba tầng

Mô hình hai lớp phần nào đáp ứng được các yêu cầu khắt khe của một ứng dụng phân tán. Tuy nhiên, khi khối lượng dữ liệu lớn, ứng dụng đòi hỏi nhiều xử lý phức tạp, số người dùng tăng, mô hình hai tầng không thể đáp ứng được.

Mô hình ba lớp sử dụng thêm WebServer hay còn gọi là Application Server giữ nhiệm vụ tương tác giữa Client và Database server, giảm bớt các xử lý trên Database server, tập trung các xử lý nhận và hiển thị dữ liệu tại Application server.



Hình 1.4: Mô hình ba tầng (Three Tier)

➤ Ưu điểm

- ✓ Hỗ trợ nhiều người dùng
- ✓ Giảm bớt xử lý cho Client → Không yêu cầu máy tính ở Client có cấu hình mạnh.
- ✓ Xử lý nhận và hiển thị dữ liệu tập trung tại Application Server → dễ quản lý, bảo trì và nâng cấp.
- ✓ Xử lý truy cập dữ liệu tập trung tại Database Server.

➤ Nhược điểm

- ✓ Phải sử dụng thêm một Application Server → tăng chi phí.

1.2. GIỚI THIỆU VỀ ASP & ASP.NET

1.2.1. Giới Thiệu về ASP

Công nghệ Active Server Page (ASP) do Microsoft phát triển là môi trường lập trình về phía server (server side programming), hỗ trợ mạnh trong việc xây dựng các ứng dụng thương mại điện tử (các trang Web Ecommerce). Các ứng dụng ASP rất dễ viết và dễ sửa đổi, đồng thời tích hợp các công nghệ sẵn có của Microsoft như: COM, DCOM,...

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.

Một ứng dụng ASP được triển khai trên Web Server là IIS (Internet Information Service) có sẵn trong môi trường Windows. Để có thể triển khai ứng dụng ASP trên các môi trường khác ta phải cài đặt các thư viện hỗ trợ ASP.

➤ Đặc điểm của trang ASP

- ASP là một tập tin văn bản (text file) có phần mở rộng .asp. Phần mở rộng này sẽ giúp Web server yêu cầu trình xử lý trang ASP (ASP engine) trước khi trả về cho trình duyệt.
- Ngôn ngữ script thông dụng nhất để viết mã của ASP là VBScript. Ngoài ra, ta cũng có thể viết mã bằng các ngôn ngữ khác như: JavaScript, Perl, Python,... nếu trên Web server có cài đặt các bộ xử lý ngôn ngữ này.
- Các đoạn mã viết trong trang ASP sẽ được các bộ xử lý ngôn ngữ trên Web server xử lý tuần tự từ trên xuống dưới. Kết quả của việc xử lý này là trả về trang mã HTML cho web server và web server sẽ gửi trang HTML này về cho trình duyệt (Browser). Do đó, tại trình duyệt không thể thấy được, các đoạn mã chương trình đã viết trong trang ASP.

- Một trang ASP gồm bốn phần:
 - ✓ *Dữ liệu văn bản (text)*
 - ✓ *Các thẻ (tag) HTML*
 - ✓ *Các đoạn mã chương trình phía client đặt trong cặp thẻ <SCRIPT> và </SCRIPT>*
 - ✓ *Mã chương trình ASP được đặt trong cặp thẻ <% và %>*

Ba thành phần đầu tiên là cấu trúc của một trang HTML thông thường, do đó, có thể xem một trang ASP là một trang HTML được nhúng thêm phần xử lý viết bằng mã ASP (VBScript, JavaScript,...).

Thí dụ 1.2: Minh họa trang Sample.asp

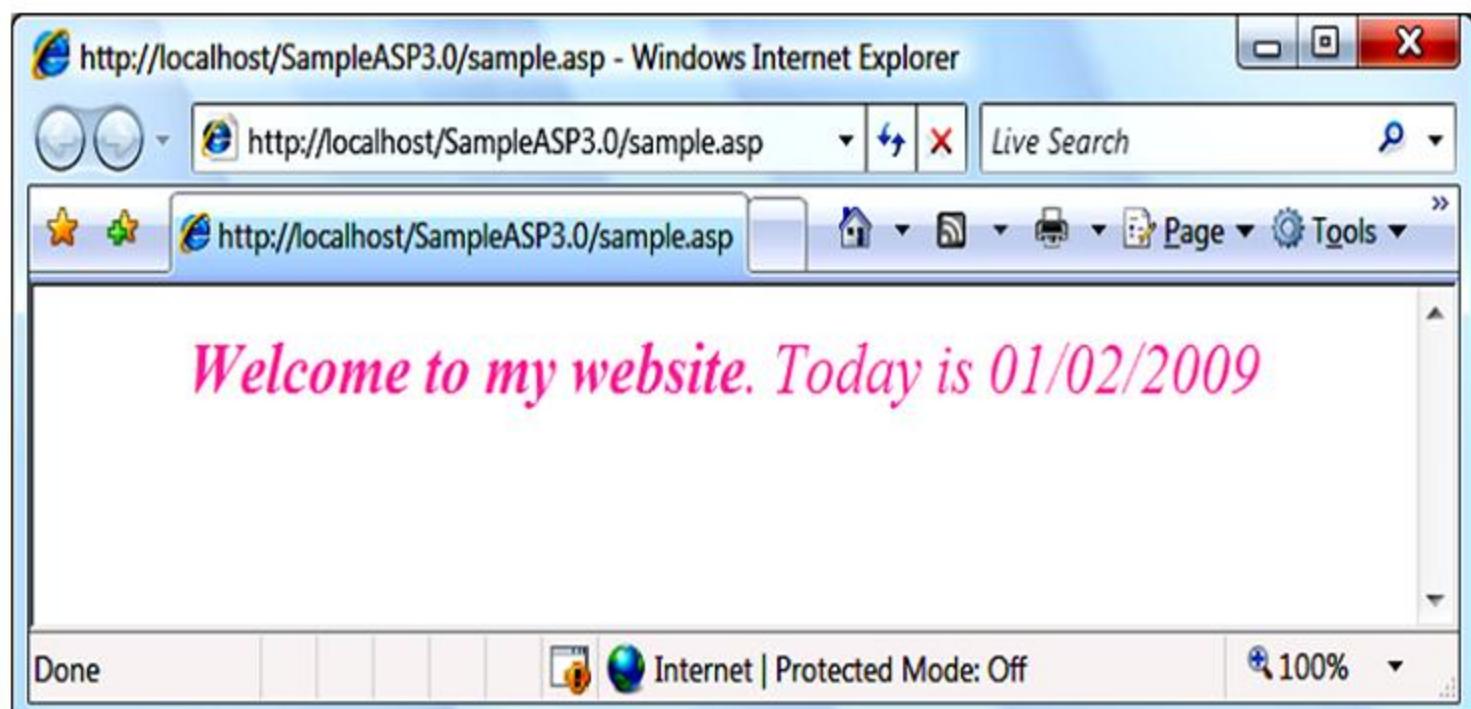
```
<HTML><BODY>

<CENTER><I><FONT COLOR="HOTPINK" size = 5>

<% @Language=VBScript %>
<P> <B> to my website</B>. Today is
<%
    Response.Write Date()
%>

</FONT></I></CENTER>
</BODY>
</HTML>
```

Kết quả thực thi:



Hình 1.5: Minh họa kết quả trang Sample.asp

➤ Ưu điểm

- ✓ Trang ASP được diễn dịch một cách tự động
- ✓ Xây dựng ứng dụng Web động
- ✓ Xử lý dữ liệu động hiệu quả

➤ Nhược điểm

- ✓ Chỉ sử dụng hai ngôn ngữ kịch bản phi định kiểu (non-type): VBScript và JavaScript,
- ✓ Sử dụng trình thông dịch cho các trang ASP
- ✓ Các đoạn mã lệnh và giao diện (HTML) trộn lẫn với nhau
- ✓ Không sử dụng lại được (reuse) các đoạn mã
- ✓ Không hỗ trợ cơ chế bẫy lỗi (Debug)

1.2.2 Giới thiệu về ASP.NET

Như chúng ta đã biết, ASP vẫn còn tồn đọng một số khó khăn như mã lệnh ASP và HTML lẫn lộn, điều này làm cho quá trình viết mã khó khăn, thể hiện và trình bày mã không trong sáng, hạn chế khả năng sử dụng lại mã. Bên cạnh đó, khi triển khai cài đặt, do không được biên dịch trước nên dễ bị mất mã nguồn (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 programming) 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 gửi về cho Client. Client nhận kết quả trả về từ Server và hiển thị lên màn hình.

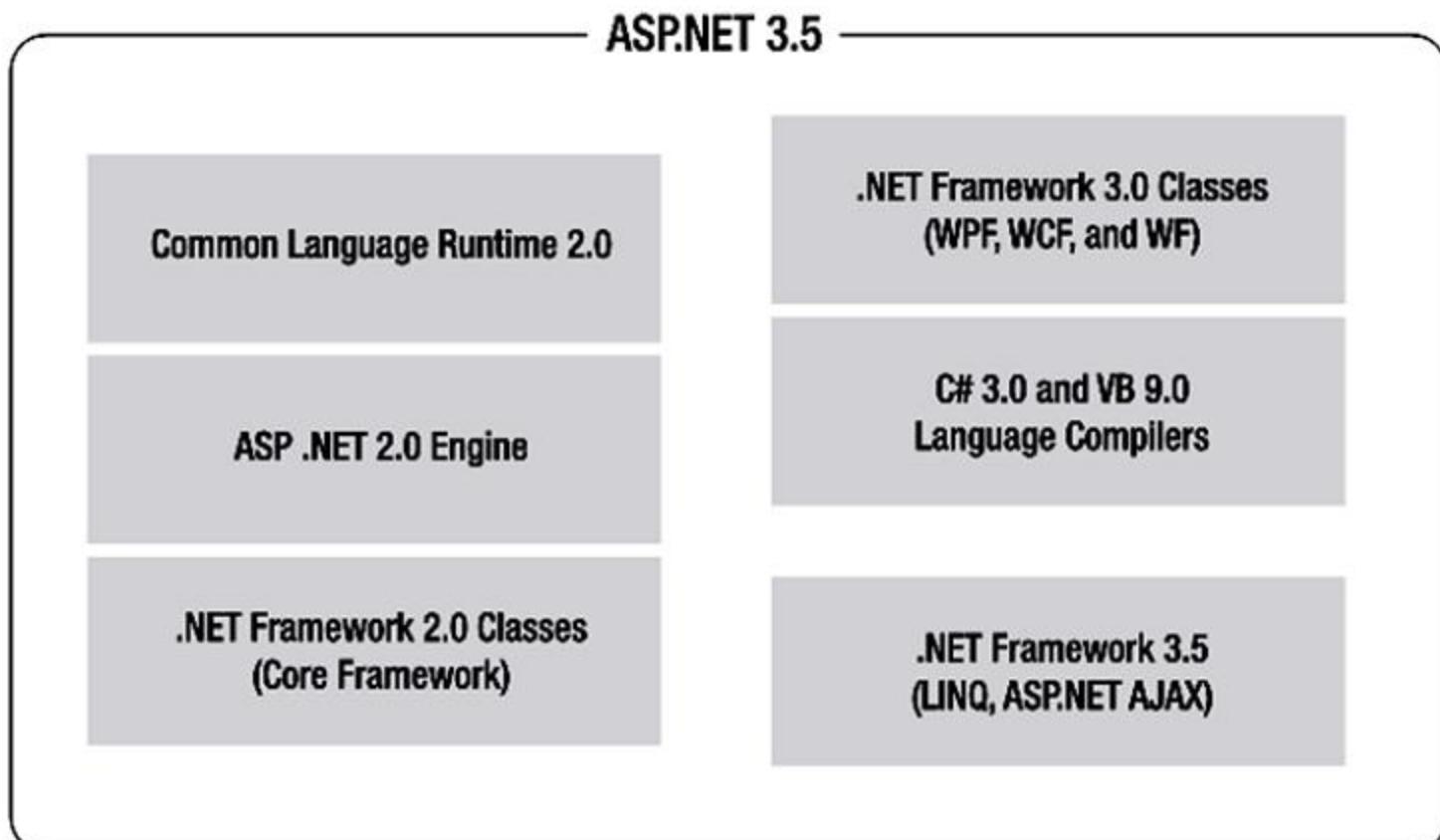
ASP.NET sử dụng kỹ thuật lập trình ở phía server thì hoàn toàn khác, mã lệnh ở phía server (ví dụ: mã lệnh trong trang ASP) sẽ được

bên 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.

ASP.NET được Microsoft phát triển qua nhiều phiên bản từ ASP.NET 1.0, 1.1, 2.0 và gần đây nhất là phiên bản ASP.NET 3.5 chạy trên .NET Framework 3.5 sử dụng môi trường phát triển tích hợp (IDE) Visual Studio.Net 2008. Trong giáo trình này, chúng ta sử dụng ASP.NET 3.5

Tại sao phải sử dụng ASP.NET ?

Yêu cầu về xây dựng các ứng dụng thương mại điện tử ngày càng được phát triển và nâng cao. Khi đó, ASP không còn đáp ứng được yêu cầu đặt ra. ASP được thiết kế riêng biệt và nằm ở tầng phía trên hệ điều hành Windows và Internet Information Service, do đó các công dụng của nó hết sức rắc rối và giới hạn. ASP.NET đưa ra một phương pháp phát triển hoàn toàn mới khác hẳn so với ASP trước kia và đáp ứng được các yêu cầu đặt ra. Hình 1.6 minh họa các thành phần bên trong ASP.NET 3.5

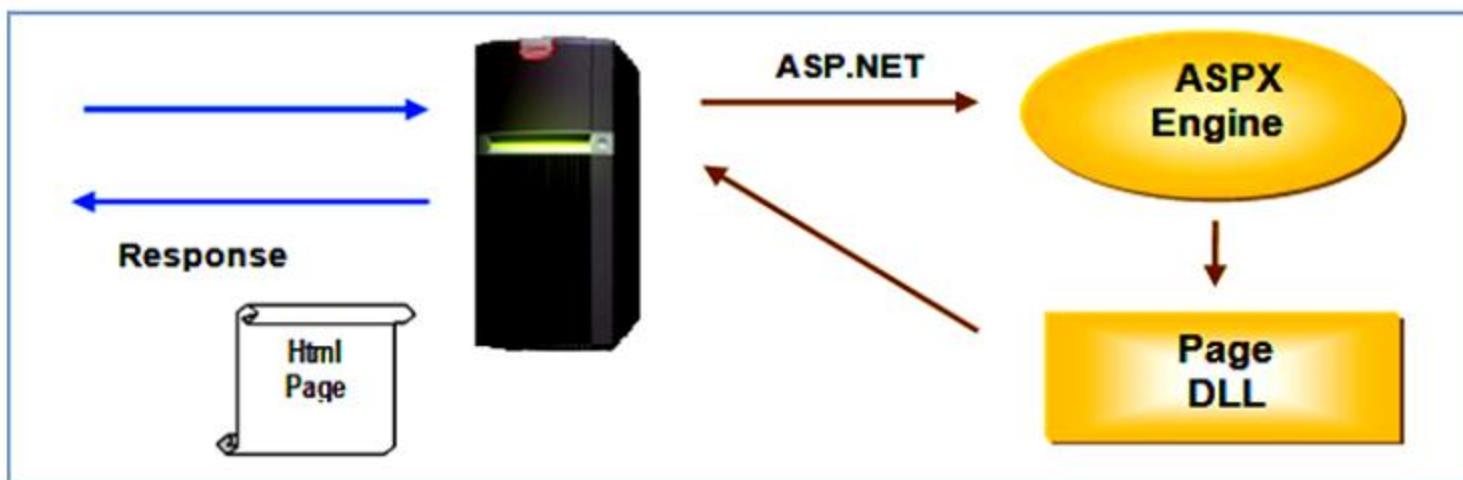


Hình 1.6: Các thành phần của ASP.NET 3.5

➤ **Các ưu điểm của ASP.NET**

- ASP chỉ sử dụng VBScript và JavaScript mà không sử dụng được các ngôn ngữ mạnh khác: Visual Basic, C++... Trong khi đó, ASP.NET cho phép viết nhiều ngôn ngữ: VBScript, JavaScript, C#, Visual Basic.NET,...

- ASP.NET sử dụng phong cách lập trình mới: Code behide song song với dạng Code inline. Tách mã riêng, giao diện riêng. Dễ đọc, dễ quản lý và bảo trì.
- Trong các trang ASP, chúng ta phải viết mã để kiểm tra dữ liệu nhập từ người dùng, ASP.NET hỗ trợ các validation controls để kiểm tra chúng ta không cần viết mã,...
- Hỗ trợ phát triển Web được truy cập trên các thiết bị di động: PocketPC, Smartphone,...
- Hỗ trợ nhiều điều khiển phía máy chủ (web server controls).
- Hỗ trợ thiết kế và xây dựng MasterPage lồng nhau.
- Hỗ trợ bẫy lỗi (debug) với JavaScript.
- Cho phép người dùng thiết lập giao diện trang Web theo sở thích cá nhân sử dụng Theme, Profile, WebPart,..
- Tăng cường các tính năng bảo mật (security).
- Hỗ trợ kỹ thuật truy cập dữ liệu mới LINQ.
- Hỗ trợ kỹ thuật xây dựng các ứng dụng đa phương tiện SilverLight.
- Hỗ trợ kỹ thuật bất đồng bộ ASP.NET Ajax.
- 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 một ứng dụng.
- 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.
- 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 làm gia tăng tốc độ thực thi so với kỹ thuật thông dịch của ASP.



Hình 1.7: Minh họa quá trình biên dịch trang ASP.NET

Thí dụ 1.3: Minh họa trang ASP.NET hiển thị ngày hiện hành

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="Sample._Default" %>

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

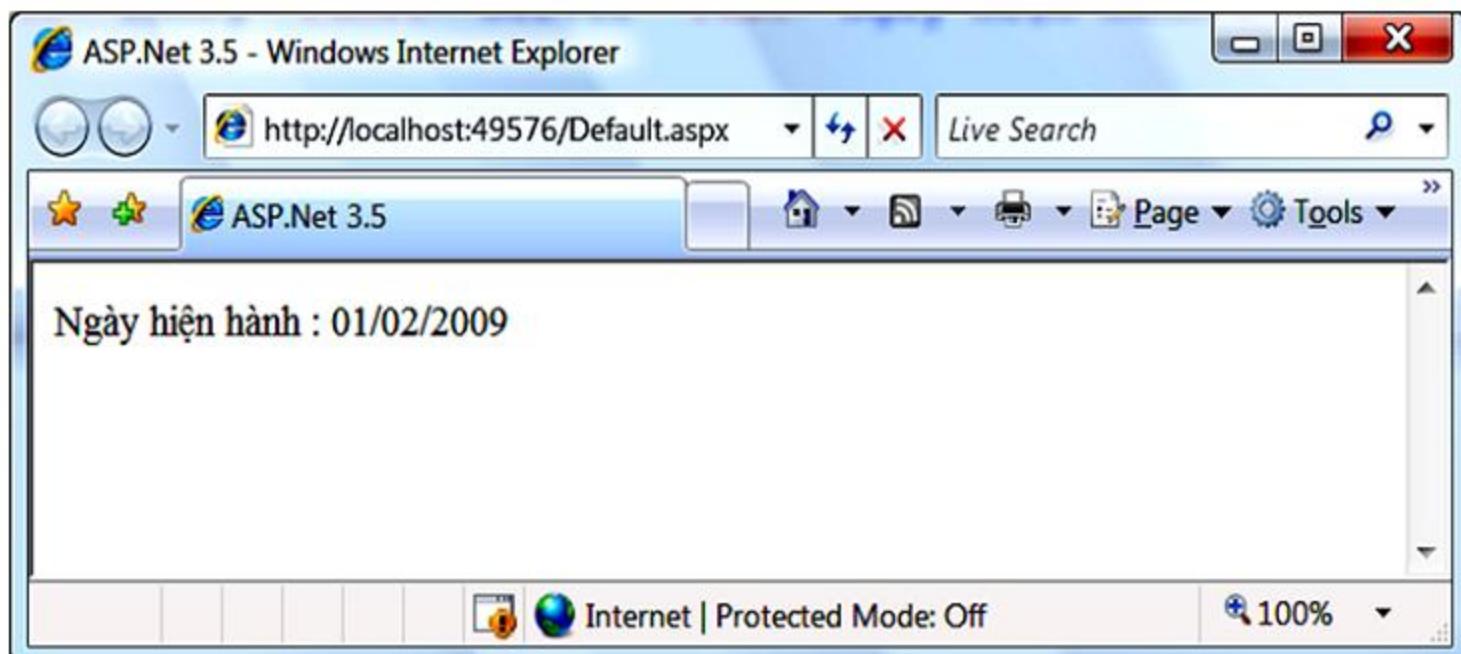
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>ASP.Net 3.5</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="lbMsg" runat="server"
                Text="Ngày hiện hành:"></asp:Label>
        </div>
    </form>
</body>
</html>
```

Hình 1.8: Nội dung trang Default.aspx

```
using System;
using System.Collections;
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;

namespace Sample
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            lbMsg.Text = lbMsg.Text + DateTime.Now.ToString("dd/MM/yyyy");
        }
    }
}
```

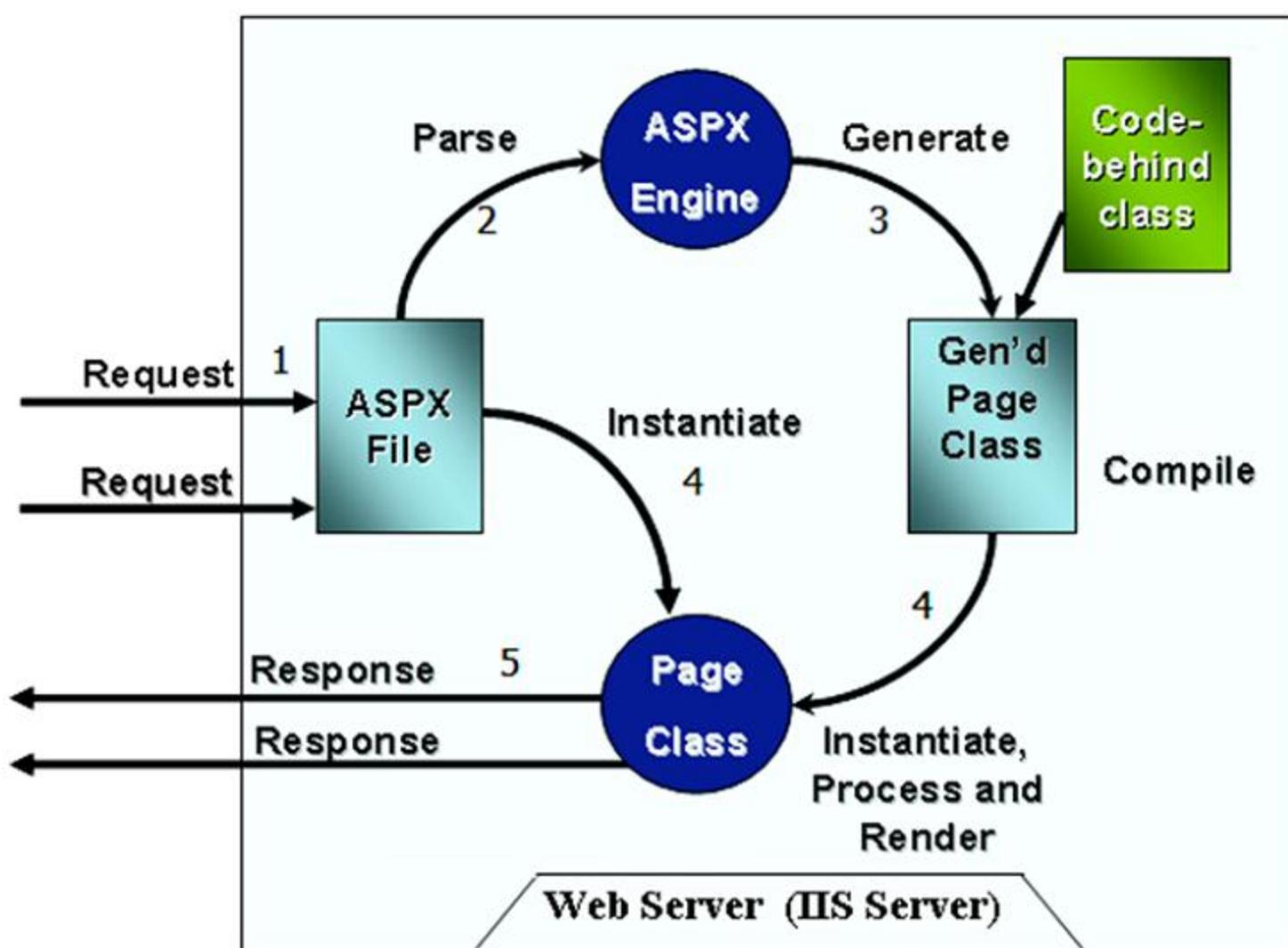
Hình 1.9: Nội dung trang Default.aspx.cs



Hình 1.10: Kết quả trang Default.aspx

➤ Quá trình xử lý tập tin .ASPX

Khi Web Server nhận được yêu cầu từ phía client, nó sẽ tìm kiếm tập tin được yêu cầu thông qua chuỗi URL được gửi về, sau đó, tiến hành xử lý theo sơ đồ sau:



Hình 1.11: Quá trình xử lý tập tin .aspx

1.3. WEB SERVER IIS

Trong phần này, chúng ta khảo sát về IIS (phần mềm Web Server của Microsoft dành cho Windows), đồng thời hướng dẫn bạn cài đặt, cấu hình và kiểm tra Web Server trên các hệ thống sử dụng Windows 2000, Windows XP, Windows Server 2003, Windows Vista,...

➤ Internet Information Services

IIS có thể được sử dụng như một Web server, kết hợp với ASP để xây dựng các ứng dụng Web tận dụng các điểm mạnh của Server-side Script, COM component, ... theo mô hình Client/Server.

IIS có rất nhiều phiên bản, đầu tiên được phát hành rời trong bản Service pack của WinNT.

- Các phiên bản Windows 2000 đã có tích hợp IIS 5.0
- Windows XP tích hợp IIS 5.5
- Windows Vista tích hợp IIS 6

➤ Cài đặt Web Server

Các bước cài đặt Web Server trên Windows XP Professional

Windows XP tích hợp sẵn IIS nhưng không tự động cài đặt. Do đó, bạn phải tự cài IIS theo các bước sau:

Bước 1. Chọn Control Panel | Add/Remove programs.

Bước 2. Add/Remove Windows Components.



Hình 1.12: Cài đặt IIS từ đĩa Windows XP

Bước 3. Đánh dấu vào mục Internet Information Services (IIS).

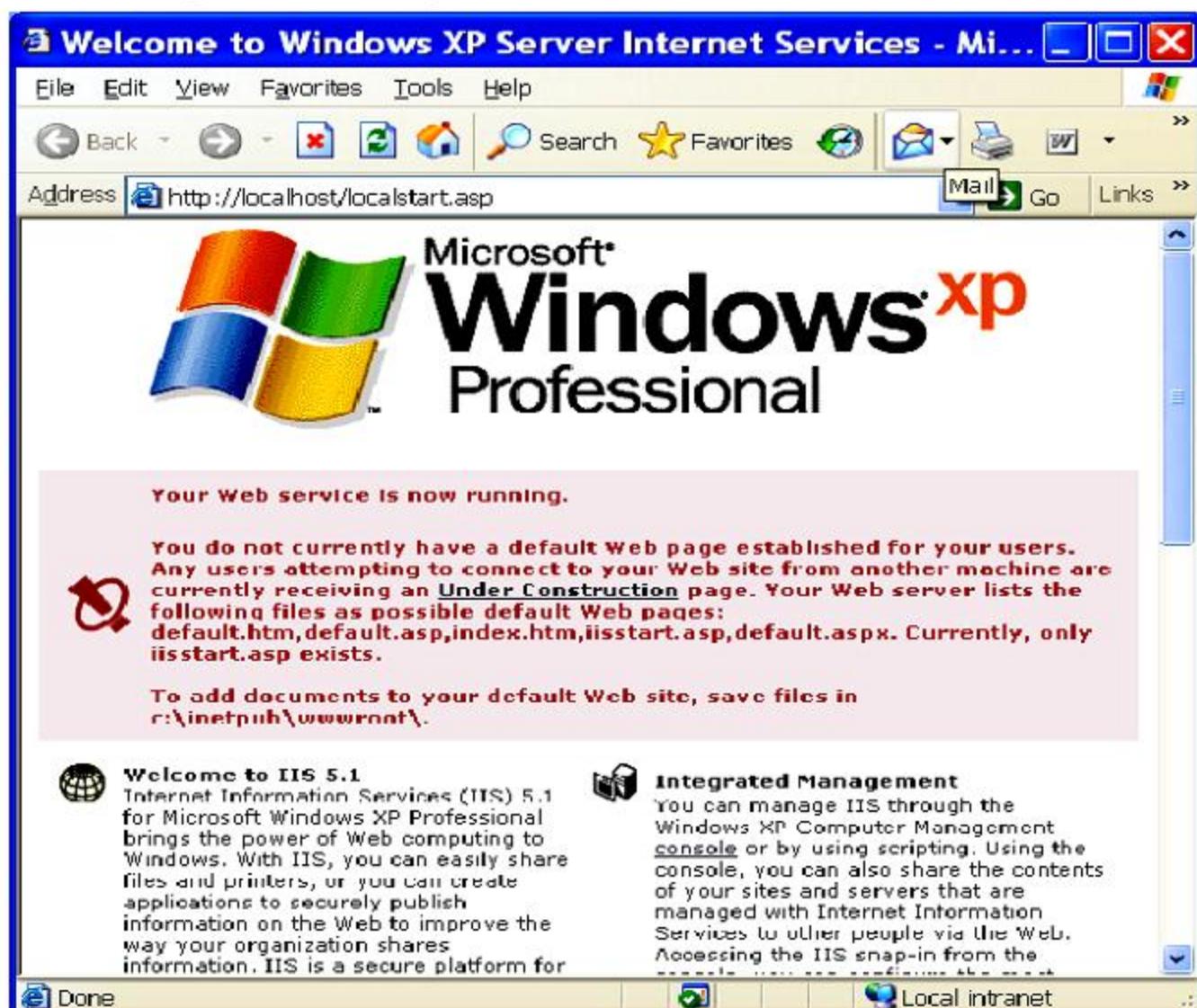
Bước 4. Chọn nút Details để chọn các mục chi tiết.

Bước 5. Chọn các mục cần cài đặt. Trong đó, bạn nhớ chọn: *FrontPage 2000 Server Extensions* và *Internet Information Services Snap-In*

Bước 6. Nhấp nút Next (có thẻ Windows yêu cầu đĩa CD Windows XP) để cài đặt hoàn tất.

Bước 7. Để xác định việc cài thành công Web Server, ta có thể kiểm tra như sau:

Mở trình duyệt (Browser): Microsoft Internet Explorer và gõ <http://localhost/localstart.asp> vào hộp địa chỉ và sau đó nhấn Enter thì trang localstart.asp mặc định sẽ xuất hiện như sau.



Hình 1.13: Minh họa cài đặt IIS thành công

- ❖ Localhost là địa chỉ của máy cục bộ mà bạn đang làm việc. Nếu máy của bạn đang kết nối vào mạng LAN và có một địa chỉ IP, bạn có thể dùng địa chỉ này thay cho localhost.

Để xác định địa chỉ IP của máy mình:

- Vào menu Start|Run và gõ lệnh: *command* hoặc *cmd*
- Trên màn hình DOS, gõ lệnh: *ipconfig* và xem phần IP Address

- Khi gõ //localhost, bạn sẽ thấy trong thanh địa chỉ tự động đổi thành: http://localhost. HTTP là giao thức mặc định được dùng trên Internet. Vì HTTP là một giao thức thuộc bộ TCP/IP, bạn cần có địa chỉ IP để các máy tính khác trong mạng có thể truy cập được đến trang web của bạn.
- Sau khi cài đặt Web Server, mặc định trên ổ đĩa C:\ sẽ có sẵn thư mục **C:\inetpub\wwwroot**. Đây là thư mục mà Web Server mặc định ánh xạ vào //localhost, do đó, các trang web đặt trong **wwwroot** có thể được truy cập bởi các máy tính khác.

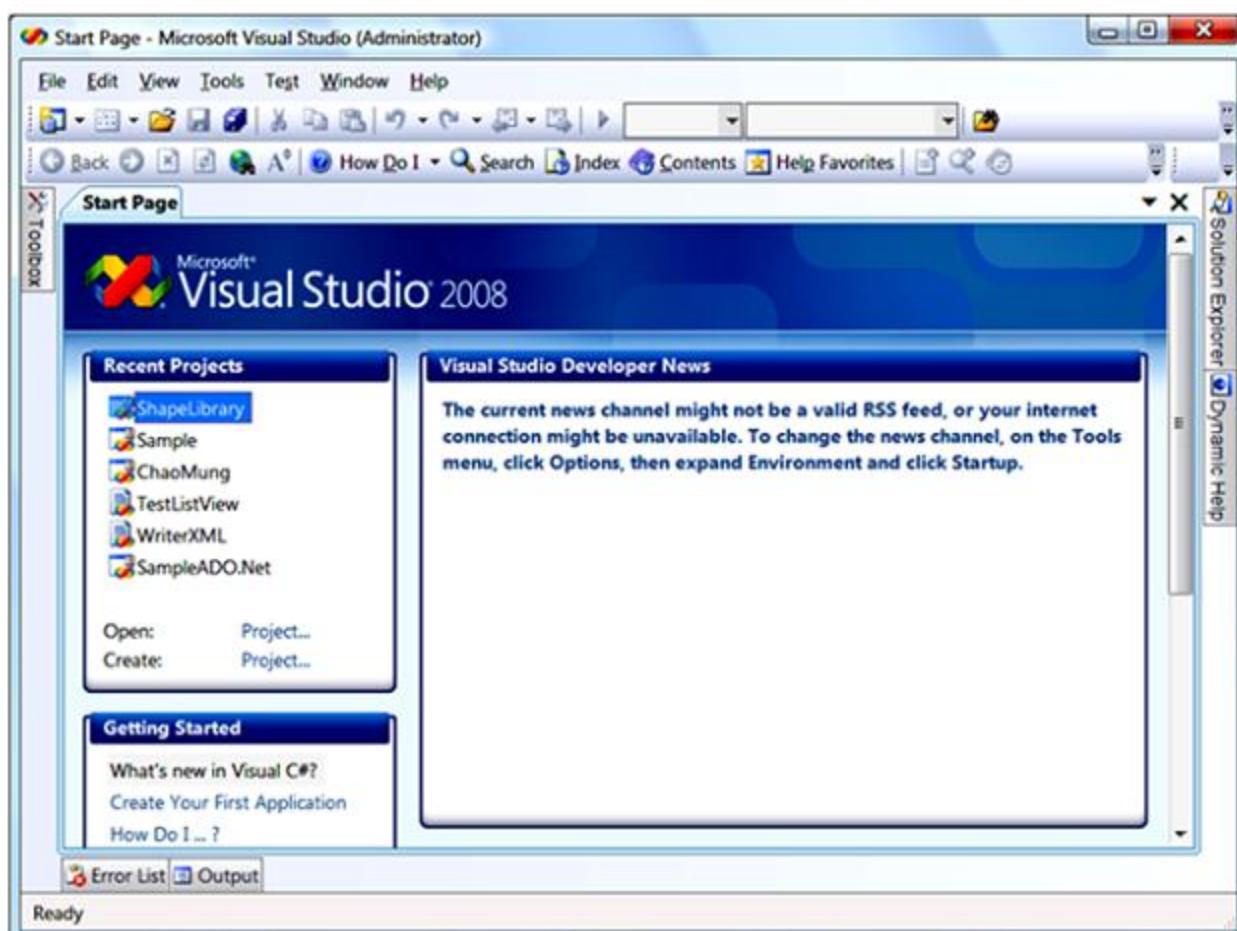
1.4. TẠO MỚI ỨNG DỤNG WEB VỚI ASP.NET

Chúng ta sẽ bắt đầu bằng việc làm quen với môi trường phát triển ứng dụng (IDE) của Visual Studio.NET. VS.NET 2008 có nhiều thay đổi so với các phiên bản trước.

Hình dưới là màn hình khởi đầu của VS.NET 2008. Vùng làm việc chính giữa đang hiển thị trang Start Page, Recent Projects, Visual Studio Developer News.

Visual Studio Developer News cần một kết nối với Internet để download các thông tin từ website của Microsoft về máy tính của chúng ta.

Recent Projects liệt kê các project mà chúng ta đã làm việc trong thời gian gần đây. Trên mục này, chúng ta cũng có thể tạo mới một project bằng cách nhấn vào nút New Project.



Hình 1.14: Cửa sổ giao diện MS Visual Studio.NET 2008

1.4.1. Tạo ứng dụng ASP.NET đầu tiên

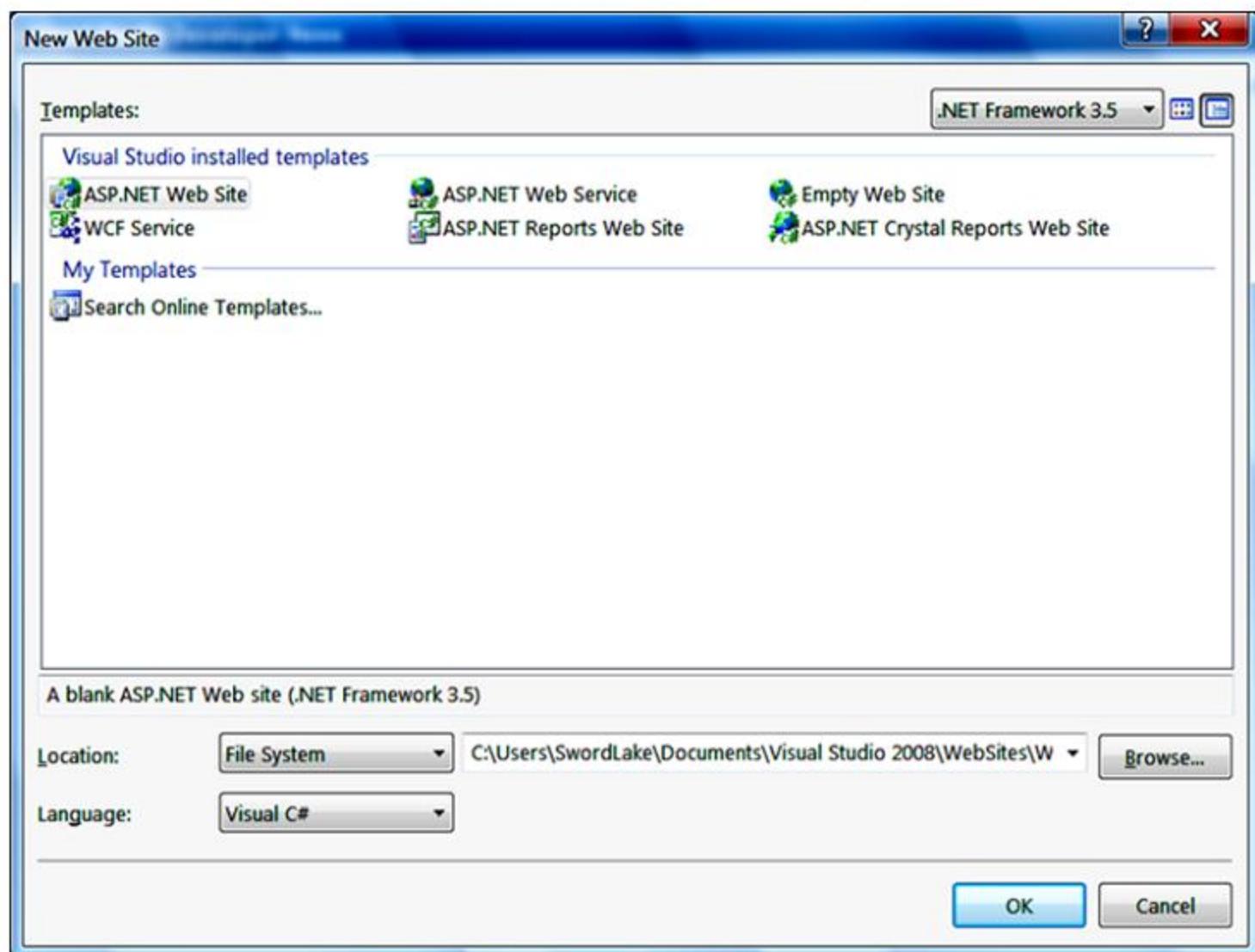
Chúng ta có thể tạo ứng dụng ASP.NET sử dụng Visual C# Project theo các bước sau:

Bước 1. Chọn từ thực đơn File | New | WebSite. Xuất hiện hộp thoại tạo mới Project (hình 1.15).

- Chọn loại *Language* là **Visual C#**
- Chọn ASP.NET Web Site từ vùng Templates
- Ứng dụng mới được tạo mặc định có tên là **WebSiteXX** (XX là số thứ tự tự động). Chúng ta có thể thay đổi tên của Project tại mục Location. Trong ví dụ này, chúng ta thay đổi tên Project **WebSite1** thành **MinhHoa**.

❖ Tại mục **Location**:

Nếu ta chọn giá trị là **File System** thì ứng dụng sẽ được tạo ra trong thư mục theo đường dẫn mà ta chỉ định ví dụ: **D:\DotNet2008\WebSite1**, khi ta chạy ứng dụng thì VS.Net sẽ tạo ra một Web Server ảo và sử dụng Web server này để thực thi ứng dụng.



Hình 1.15: Màn hình tạo mới WebSite

Nếu như ta chọn giá trị là **HTTP**, ta gõ vào đường dẫn: <http://localhost/WebSite1> thì ứng dụng sẽ được tạo ra trong thư mục mặc định là **C:\Inetpub\wwwroot** với tên là **WebSite1**, khi ta chạy ứng dụng thì VS.NET sẽ sử dụng Web server là IIS mà ta đã cài đặt trên máy.

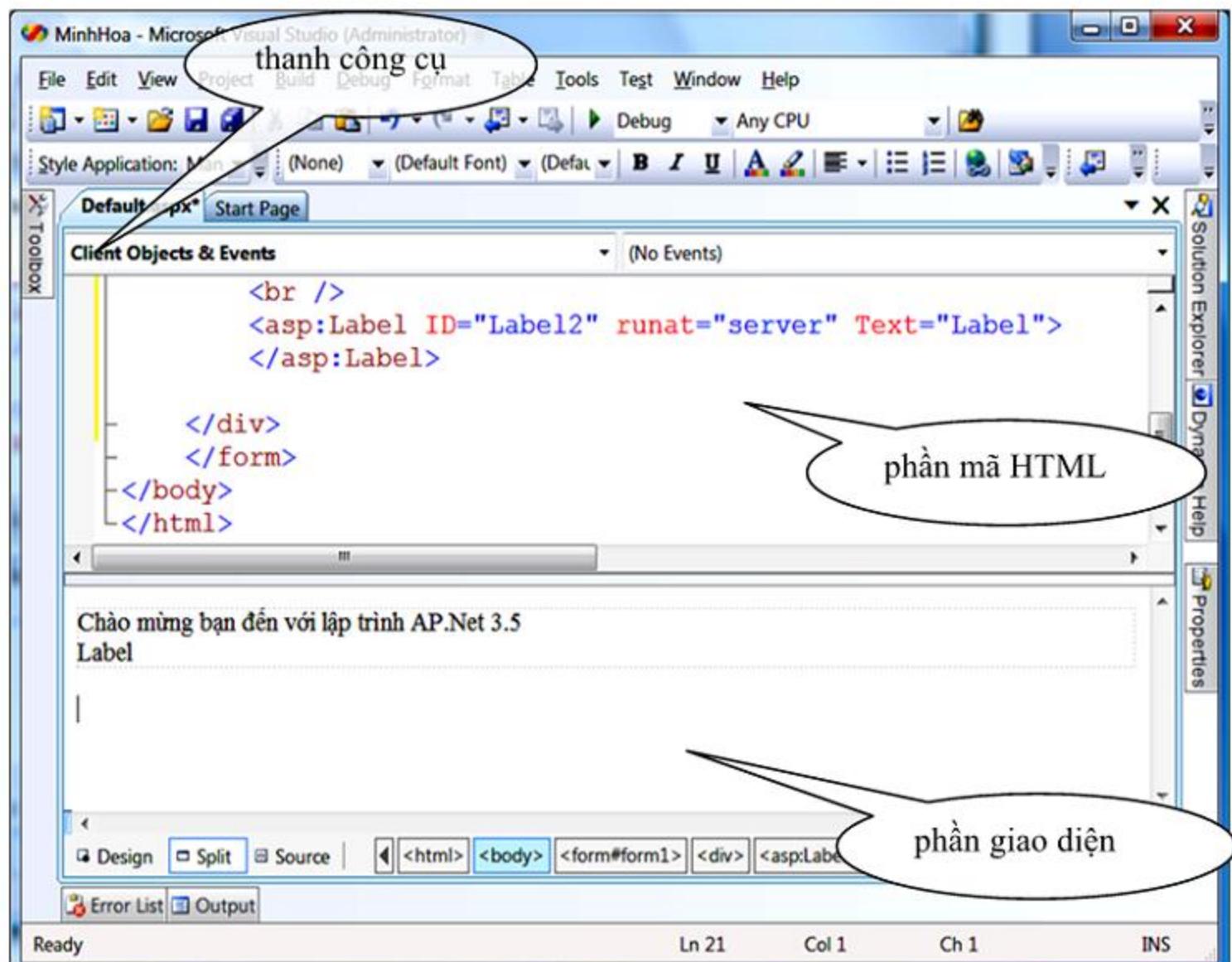
1.4.2. Thiết kế giao diện thực thi và ứng dụng

Trên hộp công cụ (Toolbox), nếu chưa có hộp công cụ chọn View/ToolBox, mở thẻ Standard (chứa các Web server control) click vào lần lượt hai điều khiển dạng nhãn (Label control) và dán vào trang Default.aspx.

Nhập nội dung thuộc tính Text cho hai điều khiển dạng nhãn theo bảng 1.1

Bảng 1.1

Tên điều khiển	Thuộc tính Text
lblChao	Chào bạn đến với lập trình ASP.NET 3.5
lblThoiGian	[Chuỗi rỗng]

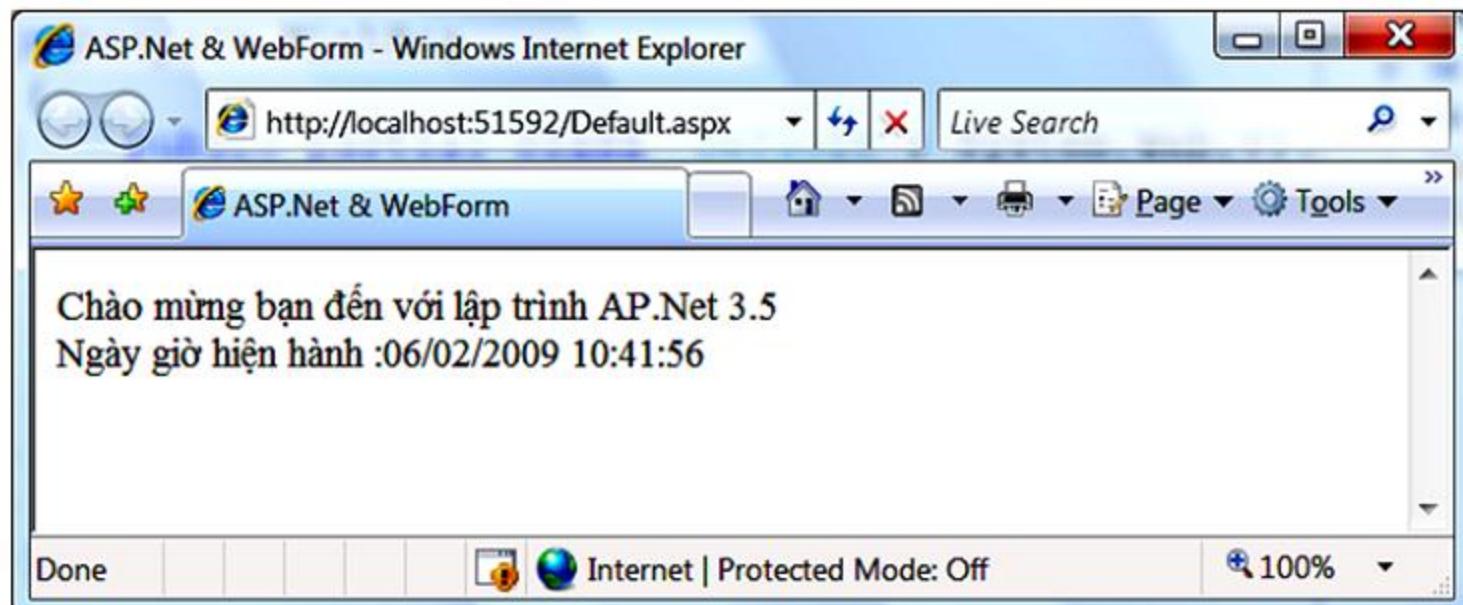


Hình 1.16: Màn hình thiết kế trang Default.aspx

Để viết lệnh cho trang Default.aspx các bạn vào menu View | Code hay nhấn phím F7, màn hình viết lệnh xuất hiện như hình 4.10 và viết lệnh cho sự kiện Page_Load:

```
namespace MinhHoa
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            lblThoiGian.Text = "Ngày giờ hiện hành" +
                DateTime.Now.ToString("dd/MM/yyyy hh:mm:ss");
        }
    }
}
```

Thực hiện việc gán nội dung cho thuộc tính Text của điều khiển nhãn lblThoiGian như trong hình. Nhấn F5 hoặc Ctrl + F5 để thi hành ứng dụng. Ứng dụng sẽ được biên dịch (compiler) để kiểm tra lỗi và hiện nội dung ra trình duyệt mặc định trên máy. Kết quả như hình 1.17.



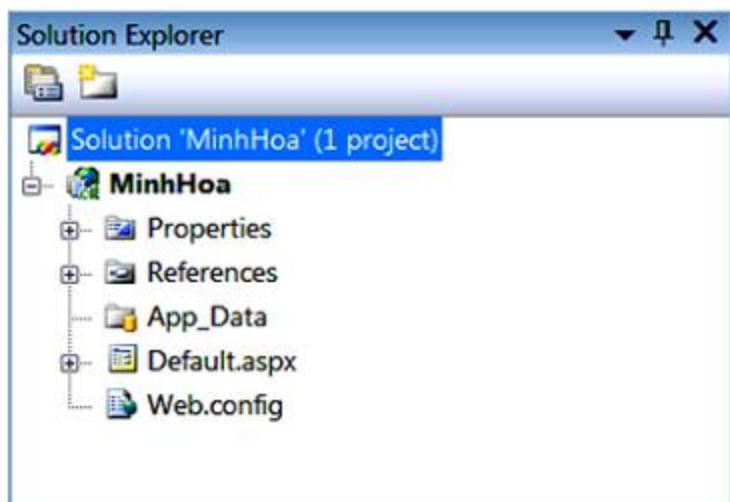
Hình 1.17: Màn hình kết quả hiển thị trang Default.aspx

1.5. KHẢO SÁT GIAO DIỆN VISUAL STUDIO.NET 2008

➤ Solution Explorer

Hiển thị cửa sổ Solution Explorer:

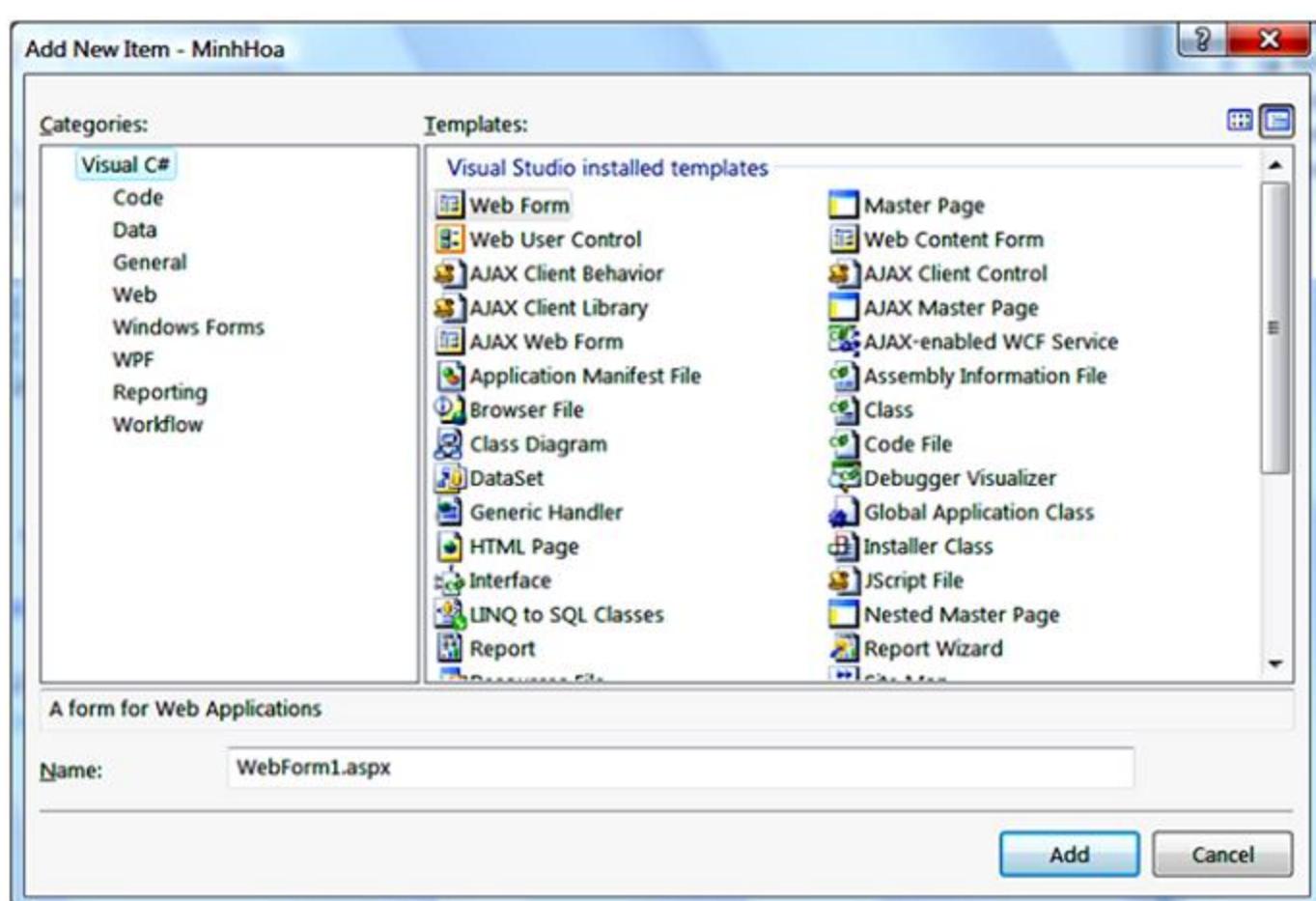
Menu **View | Solution Explorer**



Hình 1.18: Cửa sổ Solution Explorer

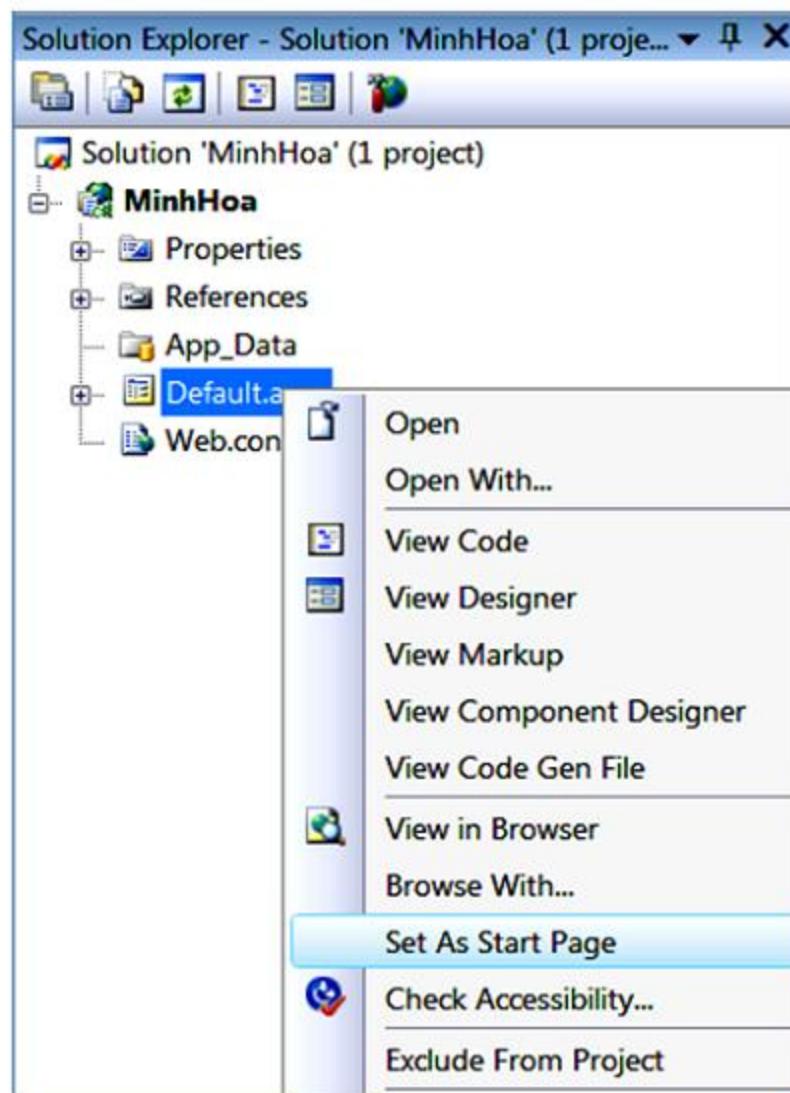
Đây là cửa sổ quản lý các "tài nguyên" có trong ứng dụng. Thông qua cửa sổ này, chúng ta có thể:

- Thực hiện các chức năng: sao chép, cắt, dán trên tập tin, thư mục như Windows Explorer.
- Tổ chức thư mục quản lý ứng dụng: Sử dụng chức năng Add | New Folder từ thực đơn ngữ cảnh.
- Thêm thành phần mới cho ứng dụng: Sử dụng chức năng Add | Add New Item,...từ thực đơn ngữ cảnh. Xuất hiện hộp thoại **Add New Item**, hình 1.19.
 - Web Form: Thêm trang Web
 - Class: Thêm lớp đối tượng
 - Web User Control: Thêm điều khiển người dùng,...



Hình 1.19: Màn hình thêm mới Item

- Xác định trang web khởi động cho ứng dụng trong trường hợp chúng ta có nhiều trang web
 - Chọn trang cần khởi động -> Nhấp chuột phải (xuất hiện thực đơn ngữ cảnh) -> Chọn **Set As Start Page**.
 - Xác định Project khởi động (trong trường hợp Solution có nhiều Project): Chọn Solution và từ (thực đơn) menu ngữ cảnh chọn **Set as StartUp Project** từ thực đơn ngữ cảnh.



Hình 1.20: Chọn trang thực thi đầu tiên trong ứng dụng

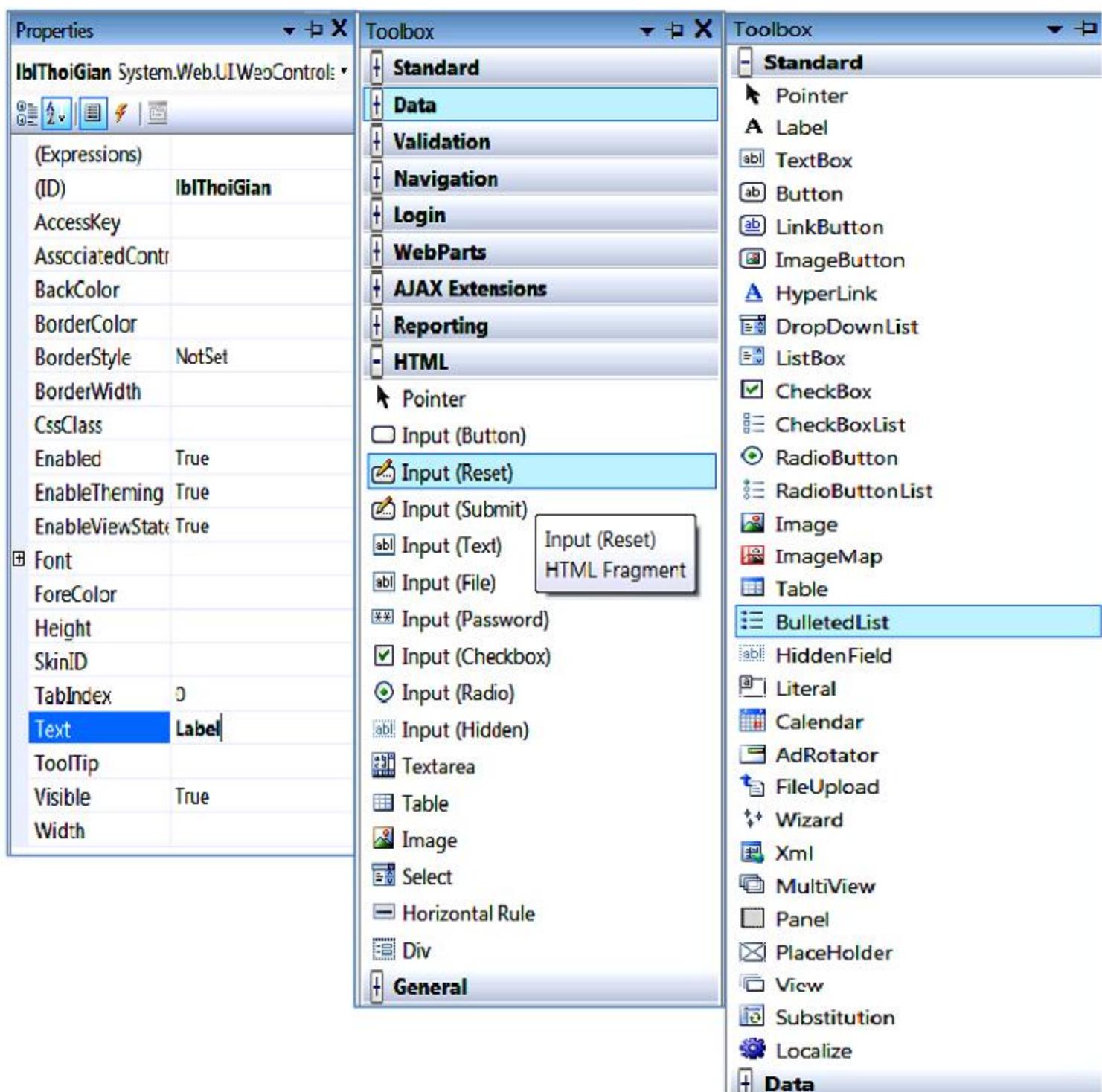
➤ Property Window

- Hiển thị cửa sổ Properties Window: Thực đơn **View | Properties Window**.
- Thông qua cửa sổ thuộc tính, chúng ta có thể thiết lập thuộc tính cho trang web và các đối tượng có trong trang web.

Hình 1.20

➤ Toolbox

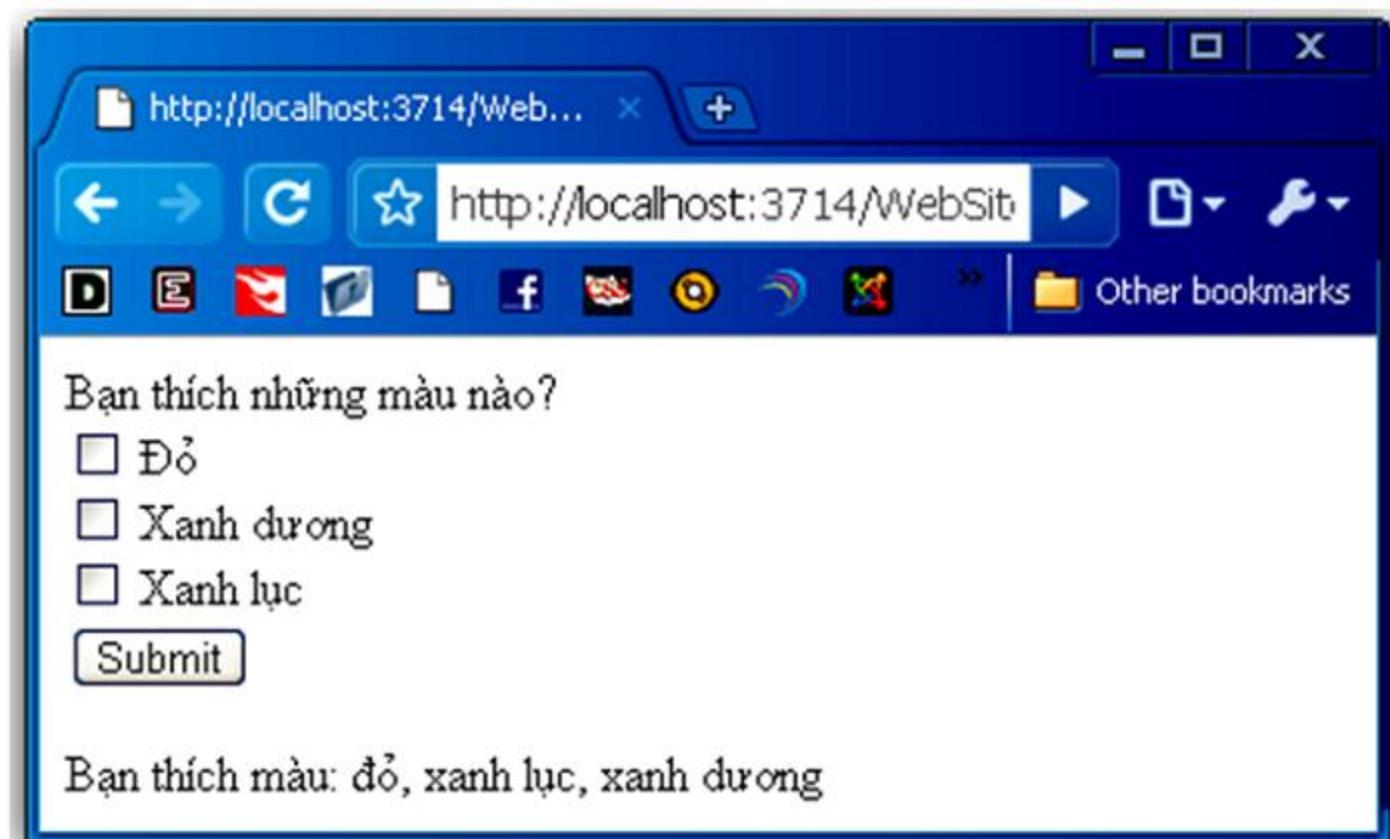
- Hiển thị Toolbox: Thực đơn **View | Toolbox**



Hình 1.21: Web Server Control

BÀI TẬP CHƯƠNG 1

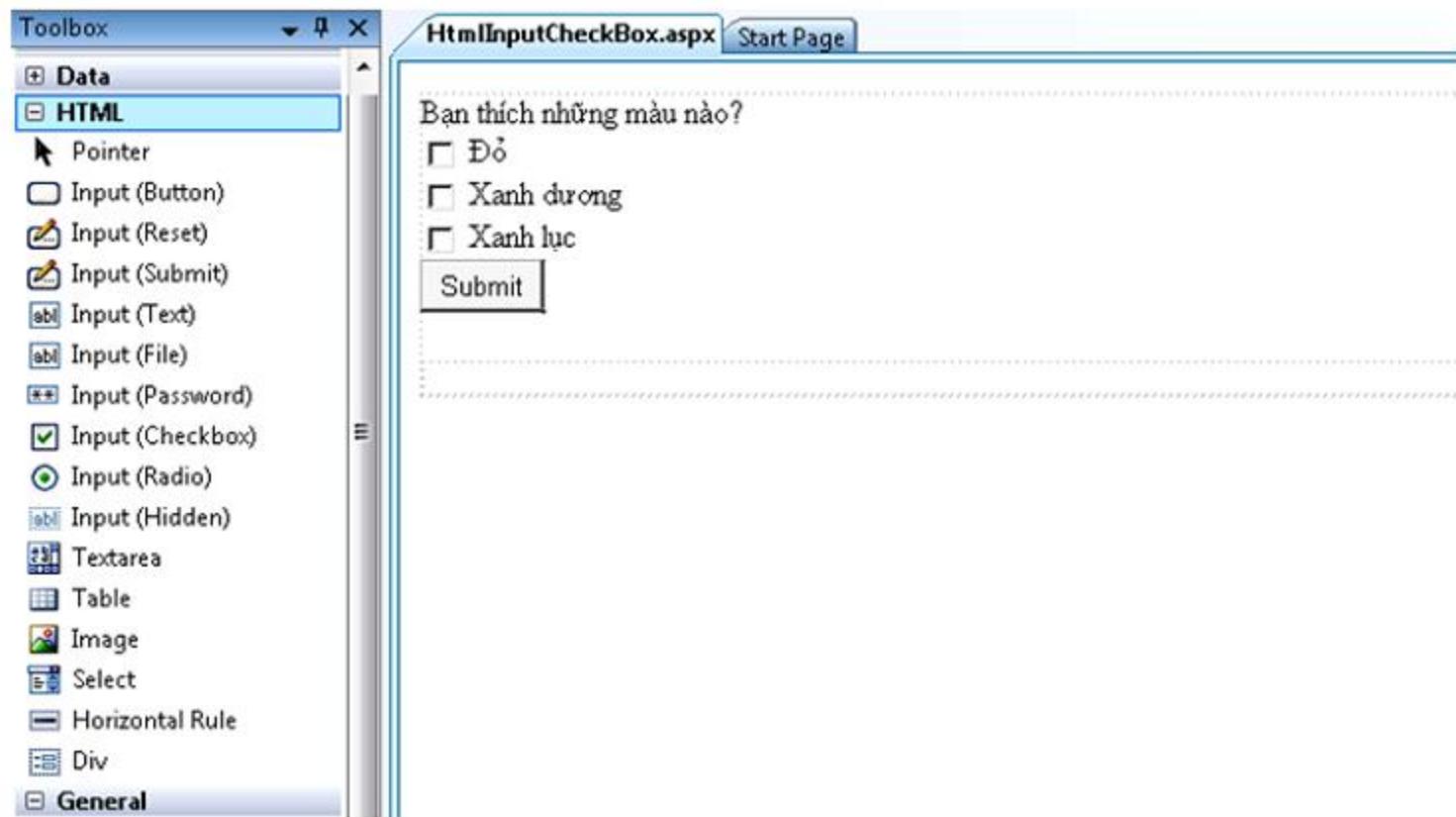
Bài 1: Tạo ứng dụng cho phép người dùng **lựa chọn màu ưa thích** trên trang ASP.NET như sau:



Giao diện của chúng ta có ba checkbox và một button. Khi button được click thì hàm xử lý sự kiện button click được thực hiện và kiểm tra checkbox nào được chọn và hiển thị nội dung tương ứng lên.

Hướng dẫn:

Chúng ta có file aspx như sau: phần nội dung trong thẻ `<form>` được nhập và kéo thả các điều khiển **CheckBox** và **Button** trong thẻ **HTML** của hộp công cụ.



Nội dung trang Default.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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></title>
</head>
<body>
    <form id="form1" runat="server">
        Bạn thích những màu nào?
        <br />
        <input id="red" type="checkbox" runat="server" /> Đỏ    <br />
        <input id="blue" type="checkbox" runat="server" /> Xanh dương    <br />
        <input id="green" type="checkbox" runat="server" /> Xanh lục    <br />
        <input id="Button1" type="button" value="Submit" OnServerClick="submit"
runat="server"/>
        <p id="p1" runat="server" />
    </form>
</body>
</html>
```

Code xử lý phía server như sau:

```
public partial class _Default : System.Web.UI.Page
{
    protected void submit(object sender, EventArgs e)
    {
        string sResult = "";
        if (red.Checked)          sResult = "đỏ, ";
        if (green.Checked)        sResult = sResult + "xanh lục, ";
        if (blue.Checked)
            sResult = sResult + "xanh dương";
        p1.InnerHtml = "Bạn thích màu: " + sResult;
        red.Checked = false;
        green.Checked = false;
        blue.Checked = false;
    }
}
```


Chương 2

TÌM HIỂU VÀ SỬ DỤNG CÁC SERVER CONTROLS

Trong bài này, chúng ta tập trung tìm hiểu các loại Server controls.

Các vấn đề chính sẽ được đề cập:

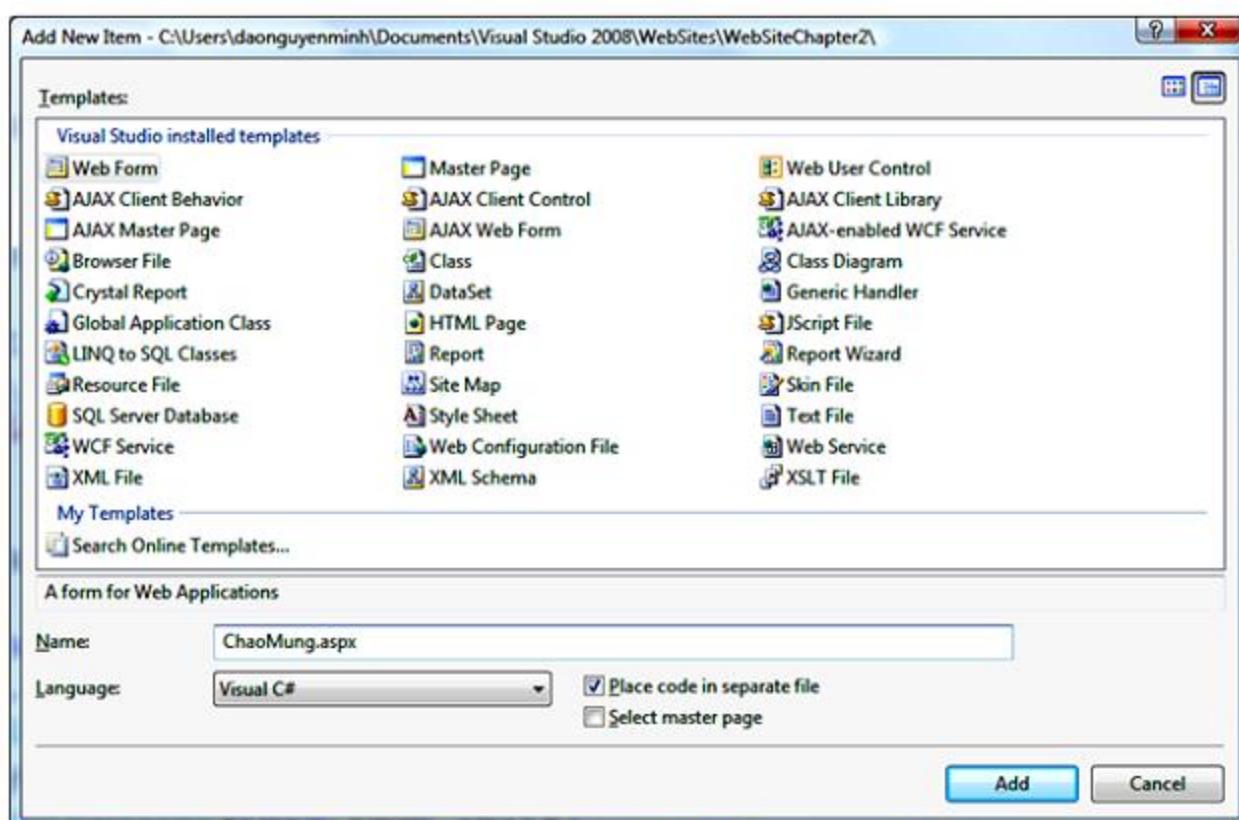
- ✓ *Cấu trúc một trang ASP.NET*
- ✓ *Biến cố của trang ASP.NET*
- ✓ *Giới thiệu ASP.NET Server Controls*
- ✓ *Các loại HTML Server Controls và Web server Controls*
- ✓ *Thuộc tính IsPostBack của trang ASP.NET và AutoPostBack của các Web Server Controls*

Kết thúc bài này các bạn có thể:

- *Sử dụng được các Web Server Controls để xây dựng các trang ASP.NET.*

2.1. CẤU TRÚC TRANG ASP.NET

Chúng ta tạo một trang ASP.NET tên **ChaoMung.aspx**, nhấp chọn **File|New|File...** (hoặc **Ctrl+N**), hoặc trong cửa sổ Solution Explorer nhấp **R-Click|Add New Item...** xuất hiện hộp thoại sau:



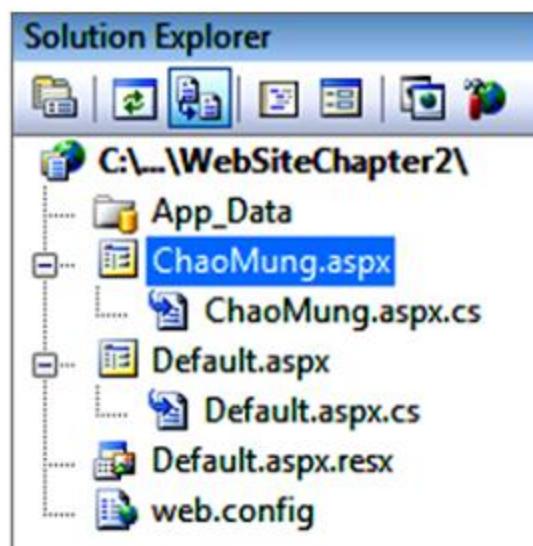
Hình 2.1: Hộp thoại thêm thành phần mới vào ứng dụng đang mở

Chọn đề mục **Web Form**, nhập **Name: ChaoMung.aspx**, nhấp nút **Add**, một trang mới được thêm vào ứng dụng.

2.1.1. Các phương pháp viết mã trong ASP.NET

ASP.NET cho phép viết mã lệnh theo hai mô hình sau:

- **Mô hình Code Inline (Code Inline Model)**
- **Mô hình Code Behind (Code Behind Model)**



Hình 2.2: Trang ChaoMung.aspx mới được thêm vào

➤ **Code Inline Model**

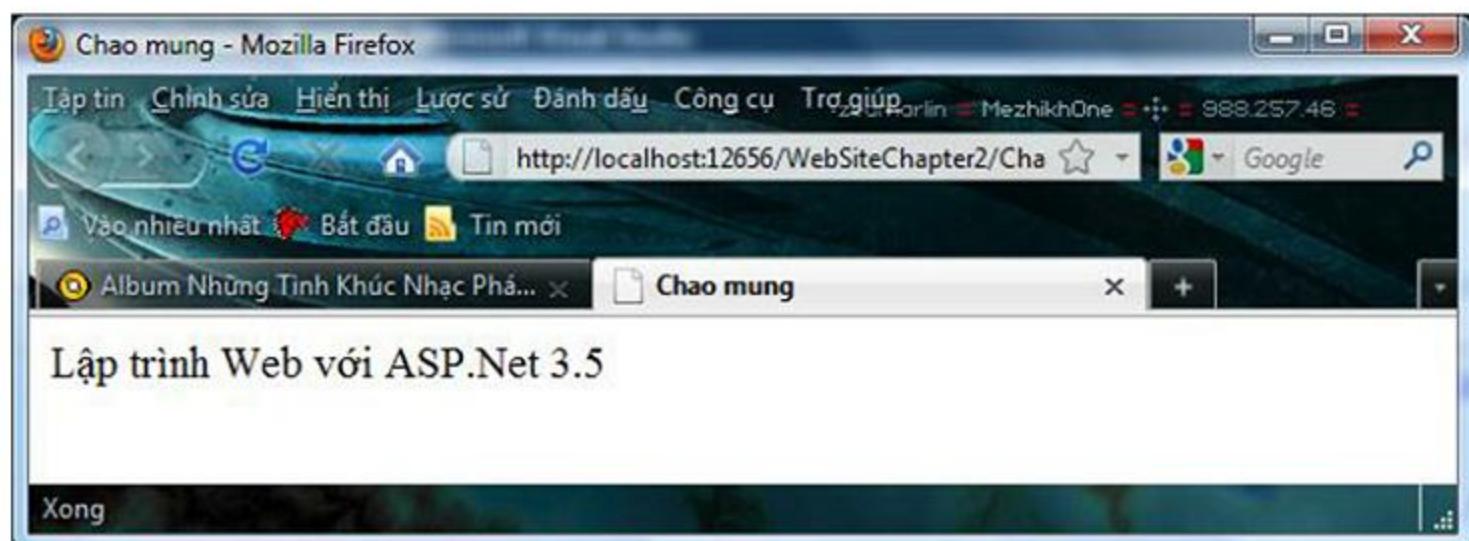
Trong mô hình này, phần mã ASP.NET và mã HTML được viết trong cùng một trang, mã ASP.NET được viết ở phần **<script runat="server">....</script>** nằm trong trang ASP.NET nhưng không trộn lẫn với mã HTML dành cho phần nội dung (content section). Chẳng hạn, chúng ta có thể đặt phần mã trong trang **ChaoMung.aspx** lên trên cùng tách khỏi phần mã HTML.

```

<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="ChaoMung.aspx.cs" Inherits="ChaoMung" %>
<!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">
<%-- Phần mã ASP.NET(code section) --%>
<script runat="server" language="c#">
void Page_Load(object sender, EventArgs e)
{
    lblChao.Text = "Lập trình Web với ASP.Net 3.5";
}
</script>
<%-- Phần mã HTML(content section) --%>
<head runat="server">
    <title>Chao mung</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="lblChao" runat="server" Width="302px" Height="38px">
            </asp:Label>
        </div>
    </form>
</body>
</html>

```

Kết quả thực thi trang trên:



Hình 2.3: Minh họa phần HTML code của trang ChaoMung.aspx

➤ Code Behind Model

Trong mô hình này, phần mã ASP.NET được được sắp xếp trong một tập tin khác riêng biệt với phần mã HTML. Ta có thể viết mã theo xử lý biến cố cho trang **ChaoMung.aspx** theo các bước sau:

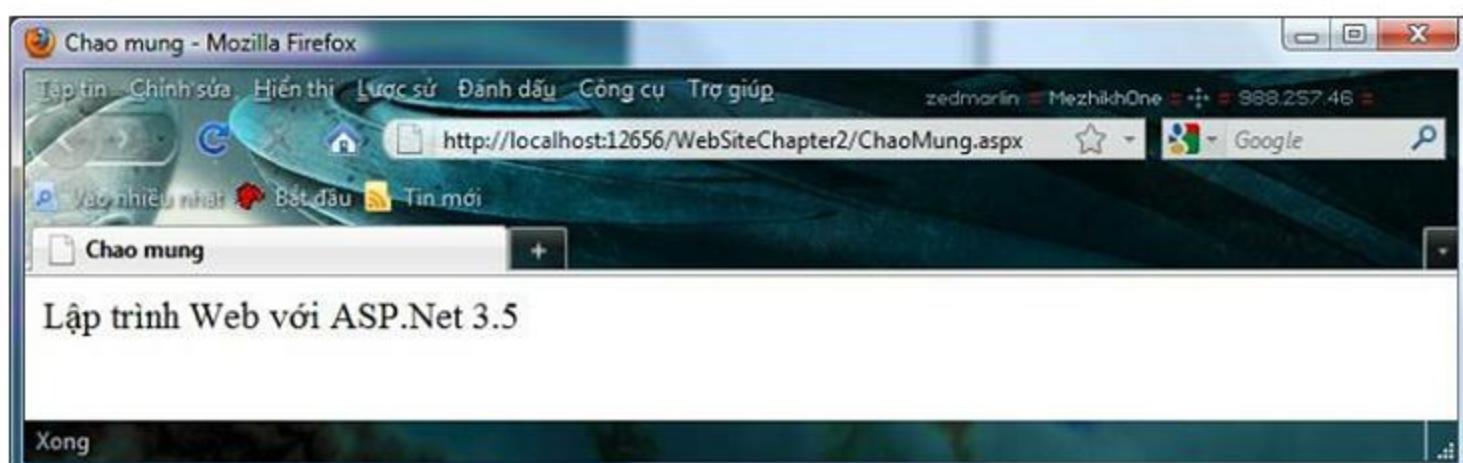
Bước 1: Trong cửa sổ Solution Explorer, chọn trang **ChaoMung.aspx**, nhấn phải chuột và chọn **ViewCode**.

Bước 2: Sau khi VS.Net tạo một tập tin tên là **ChaoMung.aspx.cs** ta viết lệnh:

lblChao.Text = "Lập trình Web với ASP.NET 3.5" vào trong sự kiện **Page_Load**.

```
public partial class ChaoMung : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        lblChao.Text = "Lập trình Web với ASP.NET 3.5";
    }
}
```

Kết quả như trên:



Hình 2.4: Minh họa phần Code behind của trang ChaoMung.aspx

So với CodeInline thì Code Behind viết mã lệnh (*code*) dễ hơn do tách được phần giao diện và phần mã HTML riêng biệt đồng thời có thể sử dụng lại các đoạn mã đã viết (reuseable codes) trong tập tin **.aspx.cs**.

2.1.2. Cấu trúc của trang ASP.NET

Một trang ASP.NET thông thường gồm ba phần:

Phần 1: Được gọi là **Page Directives**, phần này cung cấp cho ASP.NET những thông tin đặc biệt để trình biên dịch biết cách thực thi trang ASP.NET, cũng như những thông tin dùng trong tiến trình biên dịch (during the compiling process), gồm các thông tin sau:

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

- **<%@ Page %>**: Khai báo các biên dịch trang.

- **Language:** khai báo ngôn ngữ được sử dụng để viết mã cho trang (C#, VB.NET,...)
- **AutoEventWireup:** nếu giá trị là true thì các sự kiện của trang được tự động gọi đúng tên như **Page_Load** mà không cần khởi tạo sự kiện chỉ đến phương thức **Page_Load**.
- **CodeFile:** chỉ rõ tên tập tin code behind có phần mở rộng **.aspx.cs** (chứa các đoạn mã thực thi các biến cố) được liên kết với trang ASP.NET có phần mở rộng **.aspx**.
- **Inherits:** cho biết là trang giao diện thừa kế từ lớp nào là tên của lớp (class) của tập tin code behind, theo thí dụ là trang **ChaoMung.aspx.cs**.

Phần 2: `<script runat="server">... </script>`, phần này còn gọi là **Code Declaration Block**, giống như mã ở phía client (*Client Side*) nhưng có kèm theo thuộc tính **runat="server"** cho biết đoạn mã này được thực thi ở phía server (*Server Side*). Ta có thể đặt để phần này ở bất cứ nơi nào trong trang web, nhưng để phân biệt mã của ASP.NET với mã của HTML ta nên sắp xếp ở phần đầu tiên của trang.

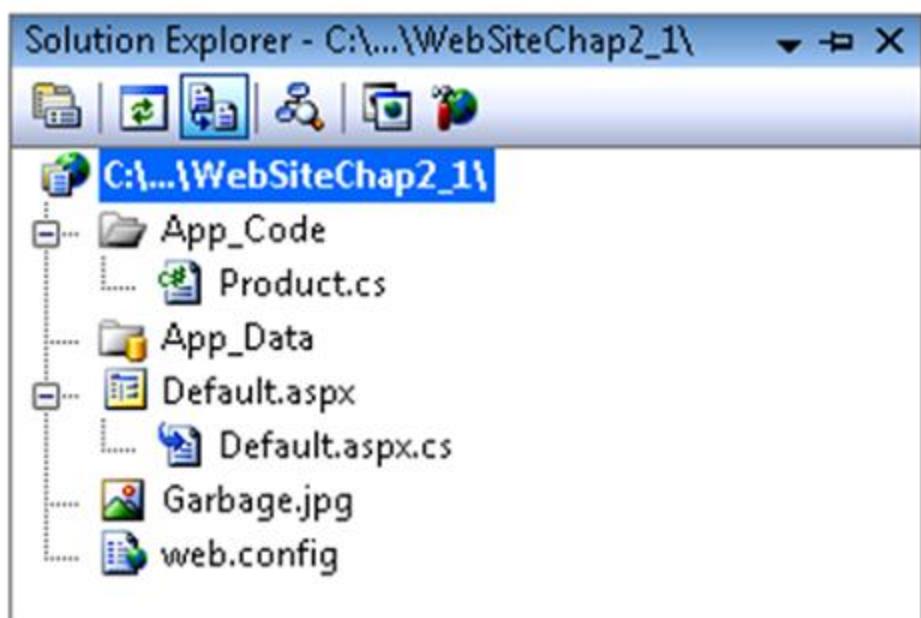
```
<script runat="server" language="c#>
void Page_Load(object sender, EventArgs e)
{
    lblChao.Text = "Lập trình Web với ASP.Net 3.5";
}
</script>
```

Phần mã này tạo ra một phương thức (hàm) có tên là **Page_Load** mặc định (*default*) cho các trang ASP.NET, phương thức này sẽ thực hiện gán chuỗi "Lập trình Web với ASP.NET 3.5" vào trong thuộc tính **Text** của **Label Control** mỗi khi trang được thực thi.

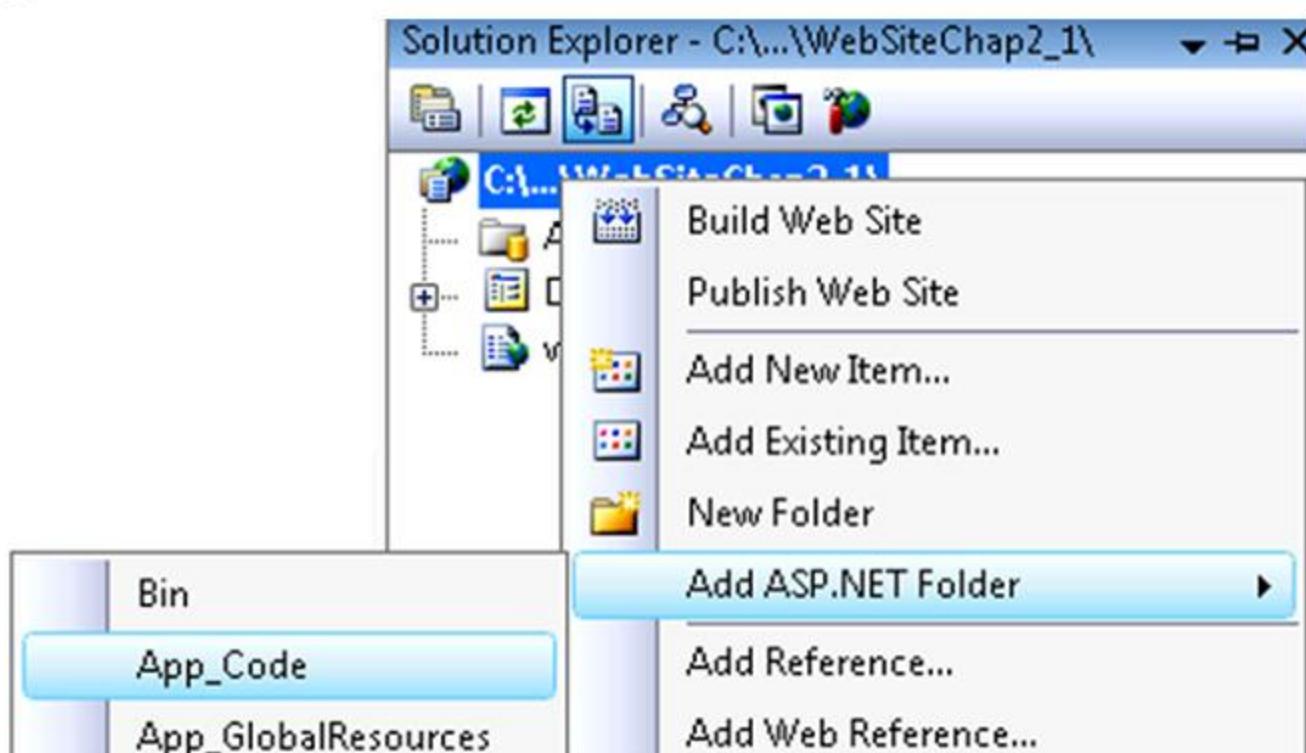
Phần 3: `<html>... </html>`, đây là nơi ta bắt đầu phần mã của HTML. Phần này chính là hình thức trình bày nội dung của trang được soạn bởi mã ASP.NET trước khi gửi về và hiển thị trong trình duyệt (*browser*) của Client. Ngoài ra, ASP.NET cũng cho phép ta kèm theo những chỉ thị (*instructions*) trong Code Render Block bắt đầu với `<%` và kết thúc với `%>`

Bài thực hành 2.1

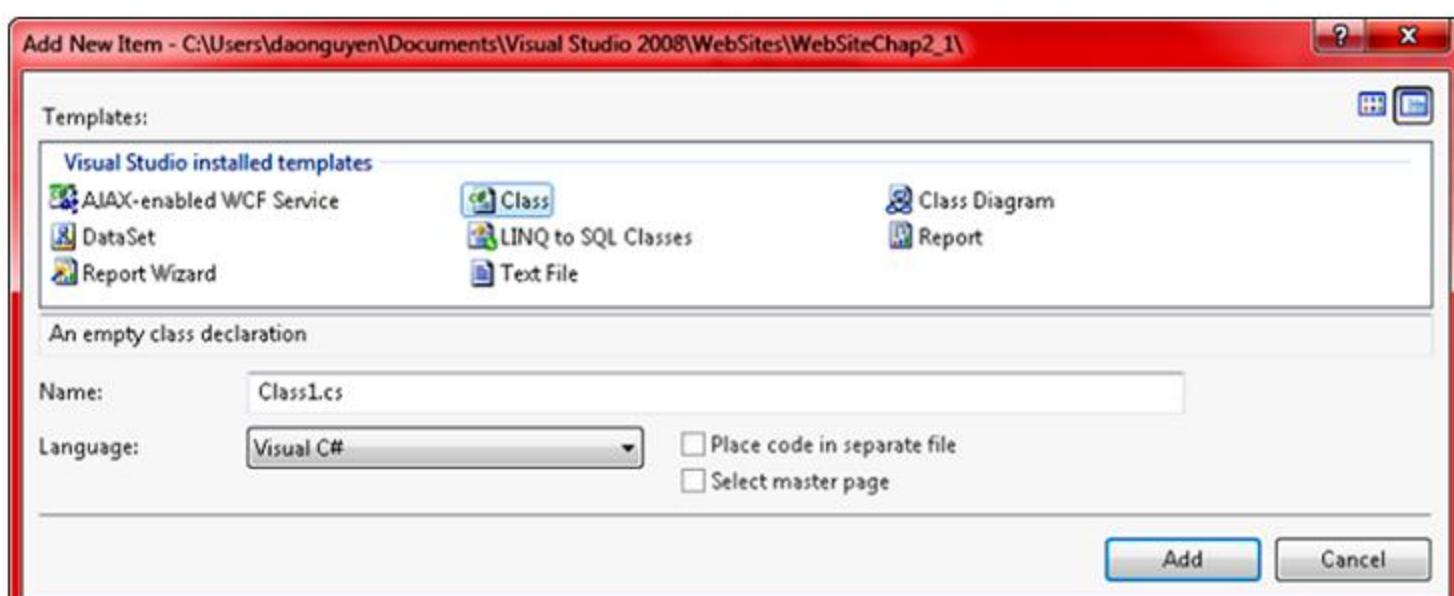
Xây dựng một ứng dụng Web với tên WebSiteChap2_1 có các thành phần sau:



Để thêm thư mục App_Code, R-Click trên tên ứng dụng trong cửa sổ Solution Explorer, chọn Add ASP.NET folder và chọn tiếp App_Code:



Sau đó tạo một lớp **Product.cs** trong thư mục **App_Code**, R-Click trên thư mục **App_Code**, chọn **Add New Item**:



Gõ tên trong khung Name là **Product.cs** và nhập nội dung sau:

```
// Define the delegate that represents the event.  
public delegate void PriceChangedEventHandler();  
  
public class Product  
{  
    private string name;  
    private decimal price;  
    private string imageUrl;  
    public string Name  
    {  
        get { return name; }  
        set { name = value; }  
    }  
  
    // Define the event.  
    public event PriceChangedEventHandler PriceChanged;  
    public decimal Price  
    {  
        get { return price; }  
        set {  
            price = value;  
            // Fire the event, provided there is at least one listener.  
            if (PriceChanged != null) {  
                PriceChanged();  
            }  
        }  
    }  
  
    public string ImageUrl  
    {  
        get { return imageUrl; }  
        set { imageUrl = value; }  
    }  
    public string GetHtml()  
    {  
        string htmlString;  
        htmlString = "<h1>" + name + "</h1><br>";  
    }  
}
```

```

    htmlString += "<h3>Costs: " + price.ToString() + "</h3><br>";
    htmlString += "<img src=\"" + imageUrl + "\" />";
    return htmlString;
}

public Product(string name, decimal price)
{
    Name = name;
    Price = price;
}
}

```

Viết mã lệnh cho trang Default.aspx:

```

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

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

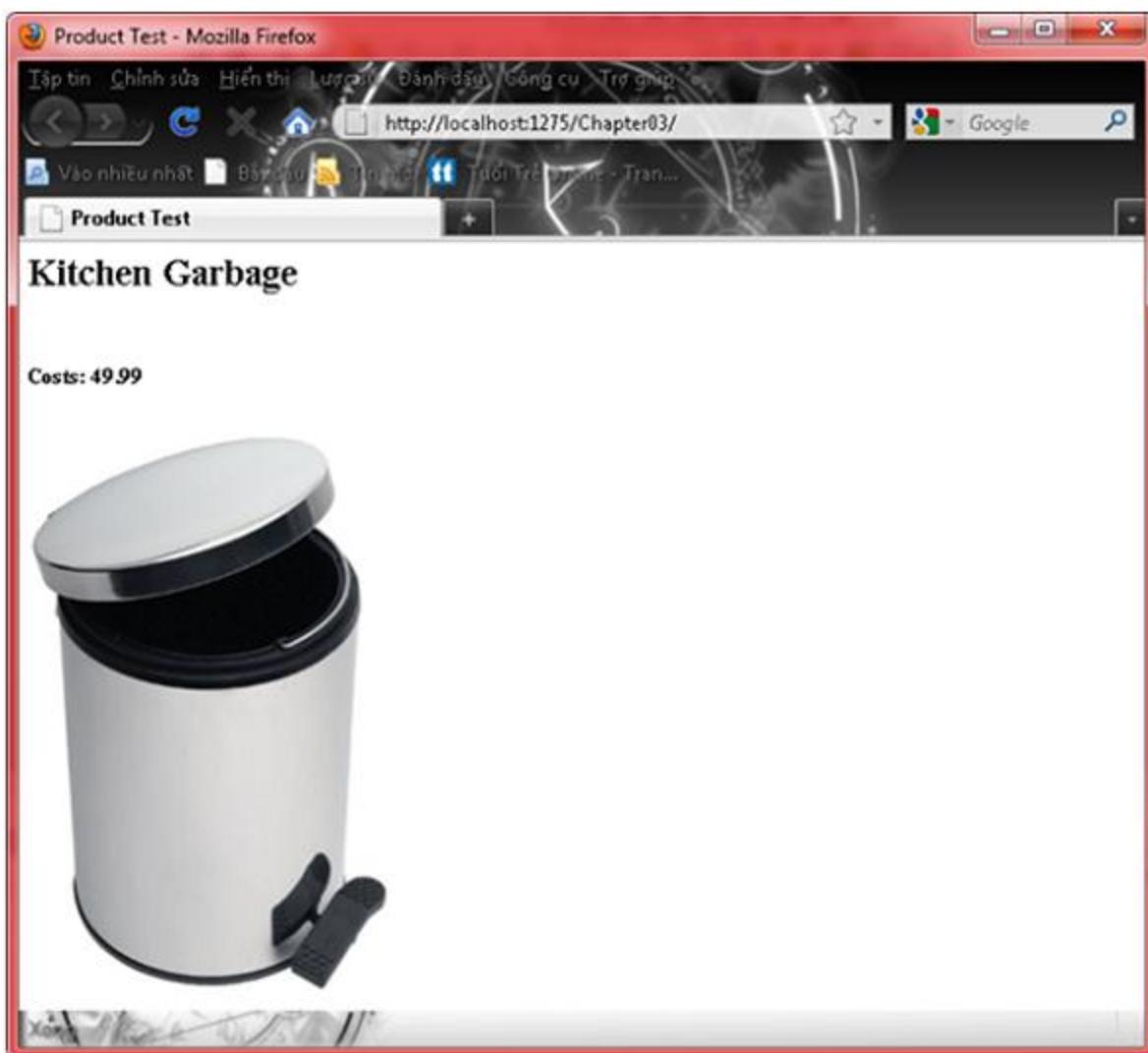
<script runat="server">

    private void Page_Load(object sender, EventArgs e)
    {
        Product saleProduct = new Product("Kitchen Garbage", 49.99M);
        saleProduct.ImageUrl = "garbage.jpg";
        Response.Write(saleProduct.GetHtml());
    }
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Product Test</title>
</head>
<body>
</body>
</html>

```

Kết quả thực thi:

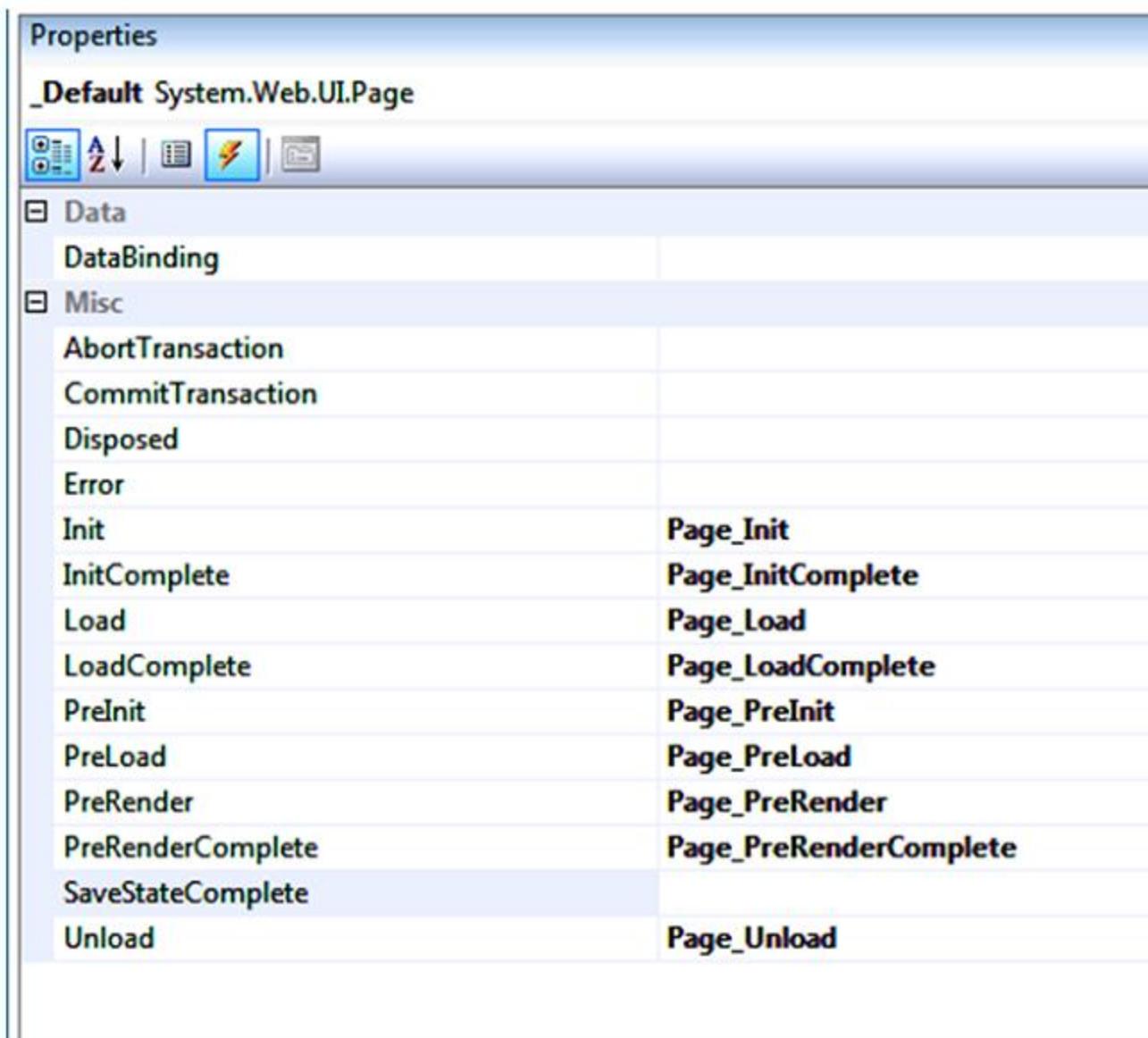


2.2. BIẾN CỐ CỦA TRANG ASP.NET

Khi làm việc với trang ASP.NET, bạn có thể bắt gặp một số biến cố của trang theo thứ tự như sau: **PreInit, Init, InitComplete, PreLoad, Load, LoadComplete, PreRender, PreRenderComplete, UnLoad**.

Để khai báo các biến cố trang ASP.NET, bạn vào thực đơn **View|Component Design** hay **R-Click| View Component Design** trong cửa sổ **Solution Explorer**.

Sau đó, nhấp chọn biểu tượng event (⚡) trong cửa sổ **Properties**, danh sách các biến cố của trang sẽ được hiển ra như hình 2.5



Hình 2.5: Danh sách các biến cố của trang ASP.NET

- **Init**

Sự kiện Page_Init xảy ra đầu tiên khi trang web được yêu cầu.

- **Load**

Sự kiện này là nơi mà bạn sẽ đặt phần lớn các xử lý, giá trị khởi web. Sự kiện này luôn xảy ra mỗi khi trang web được yêu cầu.

- **PreRender**

Sự kiện này xảy ra khi trang Web chuẩn bị được trả về cho Client.

- **Unload**

Sự kiện này đối lập với sự kiện Page_Init. Nếu như sự kiện Page_Init xảy ra đầu tiên khi trang Web được yêu cầu, thì đây, Page_Unload là sự kiện sau cùng, xảy ra sau tất cả những sự kiện khác.

Thí dụ 2-1: Khai báo các biến cõ trang ASP.NET (Default.aspx.cs)

```
public partial class _Default: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    { Page.Response.Write("Page_Load <br/>"); }

    private void InitializeComponent()
    {
        this.PreRenderComplete += new
System.EventHandler(this.Page_PreRenderComplete);

        this.PreLoad += new System.EventHandler(this.Page_PreLoad);
        this.Unload += new System.EventHandler(this.Page_Unload);
        this.InitComplete += new System.EventHandler(this.Page_InitComplete);
        this.Init += new System.EventHandler(this.Page_Init);
        this.PreRender += new System.EventHandler(this.Page_PreRender);
        this.Load += new System.EventHandler(this.Page_Load);
        this.PreInit += new System.EventHandler(this.Page_PreInit);
        this.LoadComplete += new System.EventHandler(this.Page_LoadComplete);
    }

    protected void Page_Init(object sender, EventArgs e)
    { Page.Response.Write("Page_Init <br/>"); }

    protected void Page_InitComplete(object sender, EventArgs e)
    { Page.Response.Write("Page_InitComplete <br/>"); }

    protected void Page_LoadComplete(object sender, EventArgs e)
    { Page.Response.Write("Page_LoadComplete <br/>"); }

    protected void Page_PreInit(object sender, EventArgs e)
    { Page.Response.Write("Page_PreInit <br/>"); }

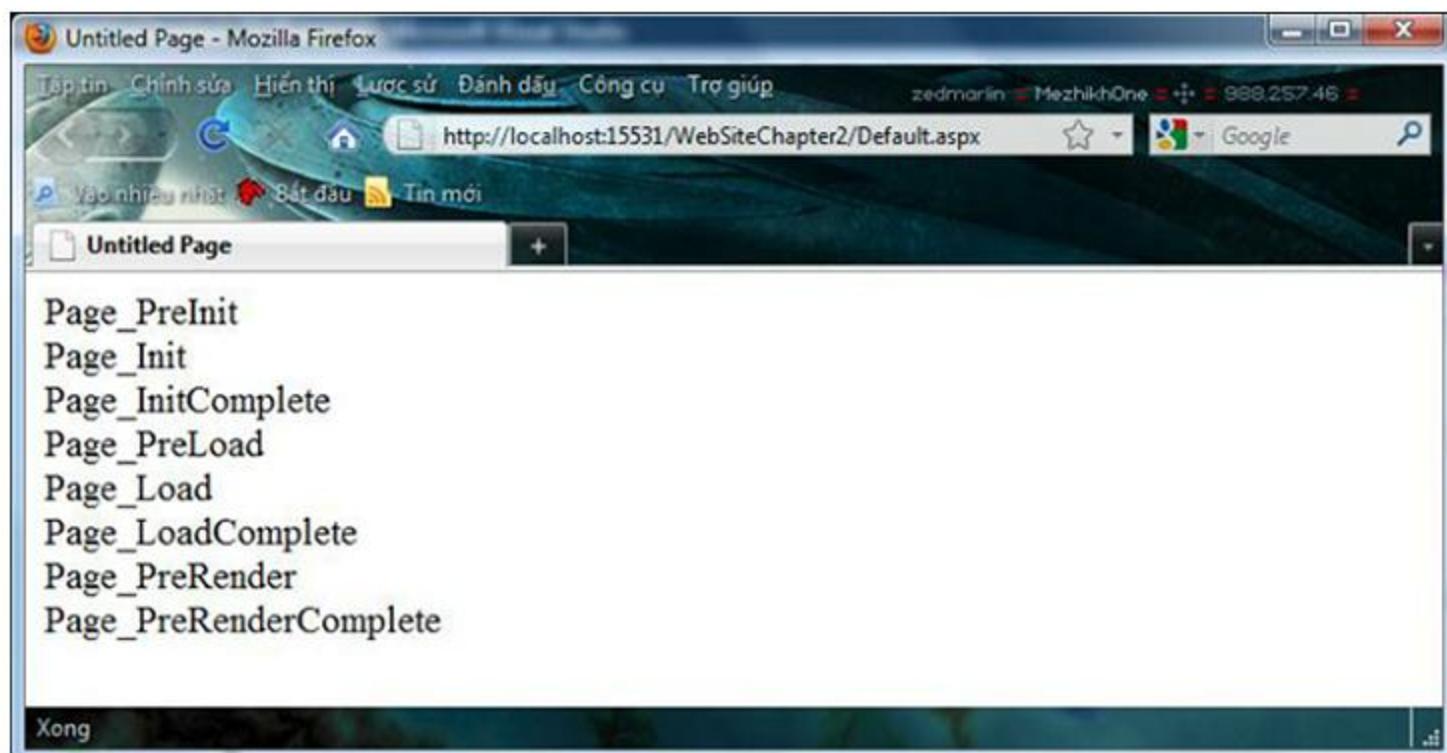
    protected void Page_PreLoad(object sender, EventArgs e)
    { Page.Response.Write("Page_PreLoad <br/>"); }

    protected void Page_PreRender(object sender, EventArgs e)
    { Page.Response.Write("Page_PreRender <br/>"); }

    protected void Page_PreRenderComplete(object sender, EventArgs e)
    { Page.Response.Write("Page_PreRenderComplete <br/>"); }

    protected void Page_Unload(object sender, EventArgs e) { }
}
```

Kết quả sau khi thực thi trang trên:



Hình 2.6: Danh sách các biến cố của trang ASP.NET

2.3. GIỚI THIỆU ASP.NET SERVER CONTROLS

Để giúp cho việc phát triển các ứng dụng web nhanh chóng và thuận tiện, ASP.NET cung cấp cho chúng ta một tập hợp các điều khiển sẵn có để thực hiện hầu hết các công việc phổ biến hàng ngày. Các điều khiển này chia làm hai loại: **HTML Server Control** và **ASP.NET Server Control**.

- **HTML Server Control:** tiền thân là thẻ HTML mà ta vẫn tạo trong trang HTML, chỉ khác một điều là có thêm **runat = "server"**; trong khai báo thẻ và được thực thi tại Web Server. Các đối tượng thẻ HTML server controls khai báo trong namespace **System.Web.UI.HtmlControls** được lấy từ lớp cơ sở **HtmlControl**.

Thí dụ: `<input type="submit" value="OK" ID="Convert" runat="server" OnServerClick="Convert_ServerClick" />`

- Web server controls: nằm trong namespace **System.Web.UI.WebControls**. Các control này cũng gọi là **Web Controls**.

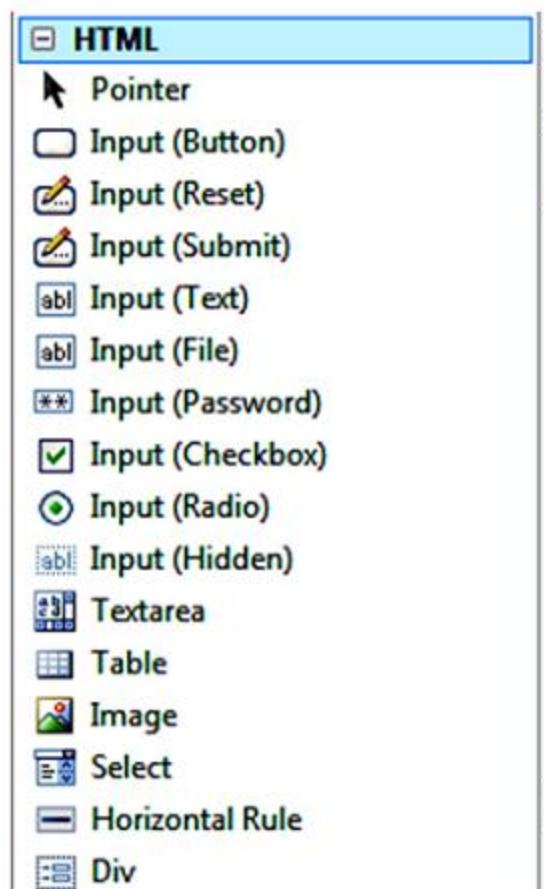
Thí dụ: `<asp:dropdownlist ID="lstBackColor" runat="server" Height="22px" Width="194px"></asp:dropdownlist>`

Điểm khác biệt giữa HTML Server control và ASP.NET Server control ở chỗ:

- **Ánh xạ tới thẻ HTML(Mapping to HTML tags):** HTML server controls ánh xạ trực tiếp tới thẻ HTML, nó được chuyển đổi thành server control bằng việc dùng thuộc tính **runat = "server"**. Web control không ánh xạ trực tiếp tới thẻ HTML. Do đó, bạn phải sử dụng thêm các control của ASP.NET.
- **Mô hình hướng đối tượng(Object Model):** HTML server control thiết lập các thuộc tính dùng cặp chuỗi tên/giá trị không định kiểu mạnh. Web control thiết lập theo chuẩn thuộc tính (property).
- **Trình duyệt đích (Target browser):** HTML server control không thay đổi phụ thuộc vào trình duyệt đích → cần đảm bảo control trả về đúng với trình duyệt. Web control trả về đầu ra tự động điều chỉnh phụ thuộc vào trình duyệt đích → chắc chắn control trả về đúng với trình duyệt.

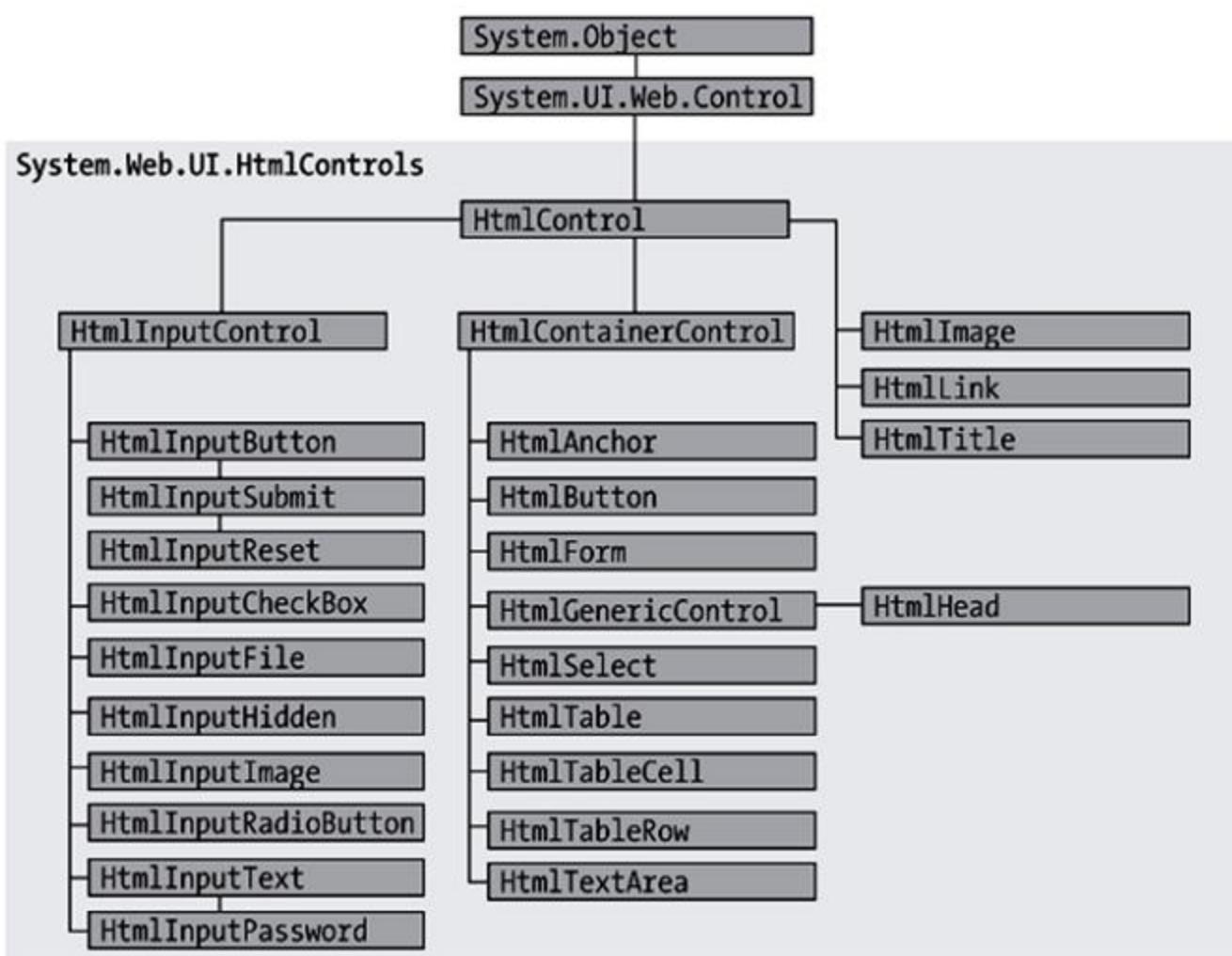
2.4. HTML SERVER CONTROLS

Như trình bày ở trên, HTML Server Control bao gồm các thẻ HTML được khai báo thêm thuộc tính **runat="server"**



Hình 2.7: Các HTML Server Controls

HTML Server Controls bao gồm hai nhóm chính thuộc lớp **HtmlControl** là: **HTMLInputControl**, **HTMLContainerControl** và ba điều khiển phụ là: **HTMLImage**, **HTMLLink** và **HTMLTitle** theo mô hình cấu trúc phân cấp sau:



Hình 2.8: Cấu trúc lớp `HTMLControl`

Các thuộc tính chính trong khai báo các thẻ `HTMLControl` trên dựa theo bảng sau:

Control	Most Important Properties
<code>HtmlAnchor</code>	<code>Href</code> , <code>Target</code> , <code>Title</code>
<code>HtmlImage</code> and <code>HtmlInputImage</code>	<code>Src</code> , <code>Alt</code> , <code>Width</code> , and <code>Height</code>
<code>HtmlInputCheckBox</code> and <code>HtmlInputRadioButton</code>	<code>Checked</code>
<code>HtmlInputText</code>	<code>Value</code>
<code>HtmlSelect</code>	<code>Items</code> (collection)
<code>HtmlTextArea</code>	<code>Value</code>
<code>HtmlGenericControl</code>	<code>InnerText</code>

Hình 2.9: Các thuộc tính chính của các `HTMLControls`

Các sự kiện chính của các thẻ `HTMLControl` chia theo bảng sau:

Event	Controls That Provide It
<code>ServerClick</code>	<code>HtmlAnchor</code> , <code>HtmlButton</code> , <code>HtmlInputButton</code> , <code>HtmlInputImage</code> , <code>HtmlInputReset</code>
<code>ServerChange</code>	<code>HtmlInputText</code> , <code>HtmlInputCheckBox</code> , <code>HtmlInputRadioButton</code> , <code>HtmlInputHidden</code> , <code>HtmlSelect</code> , <code>HtmlTextArea</code>

Hình 2.10: Các sự kiện của `HTMLControl`

Khảo sát chi tiết các HTML Server Controls.

1. HtmlAnchor

HtmlAchor control được sử dụng tương tự như một thẻ HTML `<a>`. Trong HTML, thẻ `<a>` được sử dụng để tạo một Hyperlink. Hyperlink này có thể link tới một bookmark hoặc tới một trang web khác.

Các thuộc tính:

Thuộc tính	Mô tả
Attributes	Trả về tất cả tên thuộc tính và giá trị tương ứng của thẻ
Disabled	Giá trị boolean xác định control không/có hiển thị (disabled) trên trang hay không. Mặc định là false
Href	Địa chỉ URL của liên kết (link)
Id	Id duy nhất của control
innerHTML	Điền vào hay trả về nội dung giữa thẻ đóng và thẻ mở, những ký tự đặc biệt thì không tự động chuyển thành các thực thể (entities)
innerText	Điền vào hay trả về nội dung giữa thẻ đóng và thẻ mở. Những ký tự đặc biệt tự động chuyển thành các thực thể (entities)
Name	Tên của thẻ
OnServerClick	Tên hàm được thực thi khi link được click
Runat	Xác định rằng control này là server control. Phải được xác định là “server”
Style	Xác định hay trả về thuộc tính CSS được áp dụng cho control
TagName	Trả về tên của thẻ
Target	Xác định cửa sổ sẽ được mở
Title	Tựa sẽ được hiển thị
Visible	Giá trị boolean xác định control sẽ được hiển thị hay không

Chúng ta có trang aspx như sau:

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

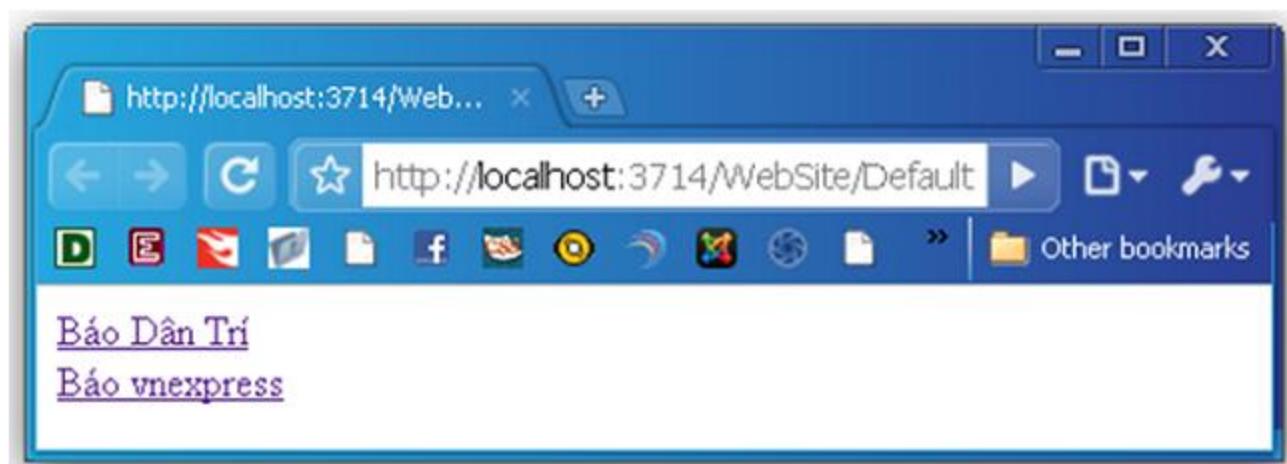
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <a id="link1" href="http://dantri.com">Báo Dân Trí</a>
            <br />
            <a id="link2" runat="server">Báo vnexpress</a>
        </div>
    </form>
</body>
</html>
```

Code xử lý phía server như sau:

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        link2.HRef = "http://vnexpress.net";
        link2.Target = "_blank";
    }
}
```

Chúng ta có **link1** gán thuộc tính **href** được gán trực tiếp trong thẻ **<a>** còn **link2** chúng ta gán **href** khi trang được tải lên ở mã lệnh phía server.



2. HtmlButton

HtmlButton được sử dụng tương ứng với thẻ HTML **<button>**. Trong HTML, thẻ **<button>** được sử dụng để tạo một nút bấm.

Các thuộc tính:

Thuộc tính	Mô tả
Attributes	Trả về tất cả tên thuộc tính và giá trị tương ứng của thuộc thẻ
Disabled	Giá trị boolean xác định control không/có hiển thị (disabled) trên trang hay không. Mặc định là false
Id	Id duy nhất của control
innerHTML	Điền vào hay trả về nội dung giữa thẻ đóng và thẻ mở, những ký tự đặc biệt thì không tự động chuyển thành các entities
innerText	Điền vào hay trả về nội dung giữa thẻ đóng và thẻ mở. Những ký tự đặc biệt tự động chuyển thành các entities
OnServerClick	Tên hàm được thực thi khi link được click.
Runat	Xác định rằng control này là server control. Phải được xác định là “server”
Style	Xác định hay trả về thuộc tính CSS được áp dụng cho control
TagName	Trả về tên của thẻ
Visible	Giá trị boolean xác định control sẽ được hiển thị hay không

Chúng ta có trang aspx như sau:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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></title>

</head>
<body>
    <form id="form1" runat="server">
        <button id="b1" style="background-color:#0000ff;height:25;width:100"
runat="server" onserverclick="b1_Click">
            Blue button!
        </button>
        <button id="b2" style="background-color:#fff0f5;height:25;width:100"
runat="server" onserverclick="b2_Click">
            Pink button!
        </button>
        <p id="p1" runat="server" />
    </form>
</body>
</html>
```

Code xử lý phía server như sau:

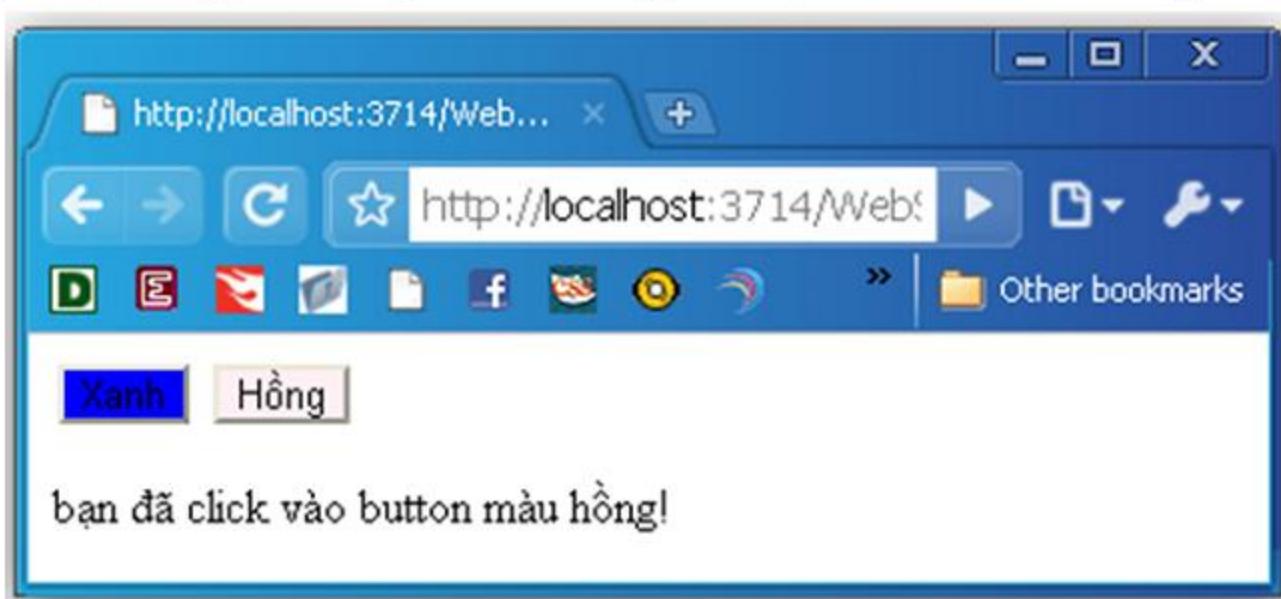
```
public partial class _Default : System.Web.UI.Page
{
    protected void b1_Click(object sender, EventArgs e)
    {
        p1.InnerHtml = "Bạn đã click vào button màu xanh!";
    }
}
```

```

protected void b2_Click(object sender, EventArgs e)
{
    p1.InnerHtml = "Bạn đã click vào button màu hồng!";
}

```

Giao diện chúng ta có hai button một button màu xanh và một button màu hồng. Khi chúng ta click vào button thì hàm xử lý sự kiện tương ứng sẽ gán nội dung vào thẻ **p** và hiển thị giao diện như hình dưới đây:



3. HtmlForm

HtmlForm control được sử dụng tương ứng thẻ HTML **<form>**. Trong HTML, thẻ **<form>** được sử dụng để tạo một form.

Các thuộc tính:

Thuộc tính	Mô tả
Action	URL nơi mà dữ liệu được gửi đến khi form được submit
Attributes	Trả về tất cả tên thuộc tính và giá trị tương ứng của thuộc thẻ
Disabled	Giá trị boolean xác định control có bị disabled hay không. Mặc định là false
Id	Id duy nhất của control
innerHTML	Điền vào hay trả về nội dung giữa thẻ đóng và thẻ mở. những ký tự đặc biệt thì không tự động chuyển thành các entities
innerText	Điền vào hay trả về nội dung giữa thẻ đóng và thẻ mở. Những ký tự đặc biệt tự động chuyển thành các entities
Method	Xác định cách post dữ liệu lên server. Có hai giá trị là “post” và “get”. Mặc định là “post”

Name	Tên của form
Runat	Xác định rằng control này là server control. Phải được xác định là “server”
Style	Xác định hay trả về thuộc tính CSS được áp dụng cho control
TagName	Trả về tên của thẻ
Target	Cửa sổ để load URL
Visible	Giá trị boolean xác định control sẽ được hiển thị hay không

Chúng ta có file aspx như sau:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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></title>
</head>

<body>
    <form id="form1" runat="server">
        Nhập tên của bạn: <input id="name" type="text" size="30" runat="server" />
        <br /><br />
        <input id="Submit1" type="submit" value="Submit" OnServerClick="submit_Click"
runat="server" />
        <p id="p1" runat="server" />
    </form>
</body>
</html>
```

Code xử lý phía server như sau:

```
public partial class _Default : System.Web.UI.Page
{
    protected void submit_Click(object sender, EventArgs e)
    {
```

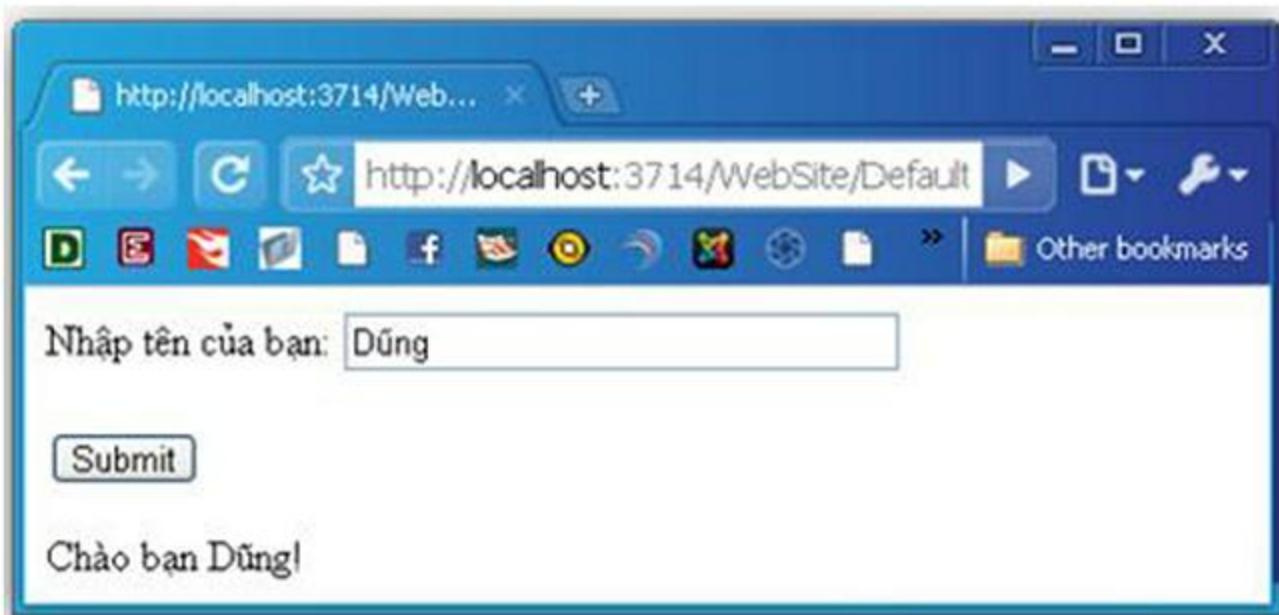
```

    p1.InnerHtml = "Chào bạn " + name.Value + "!";
}

}

```

Giao diện chúng ta có một textbox và một button. Khi chúng ta click vào button submit thì hàm xử lý sự kiện submit_Click được thực hiện và ghi ra trang web: “chào bạn” cùng với tên được nhập vào textbox như hình dưới đây:



4. HtmlGeneric

HtmlGeneric control được dùng để điều khiển những thẻ HTML khác chưa được chỉ rõ bởi một HTML server control đặc biệt, như **<body>**, **<div>**, ****, ****, **<p>**...

Các thuộc tính

Thuộc tính	Mô tả
Attributes	Trả về tất cả tên thuộc tính và giá trị tương ứng của thuộc thẻ
Disabled	Giá trị boolean xác định control có bị disabled hay không. Mặc định là false
Id	Id duy nhất của control
innerHTML	Điền vào hay trả về nội dung giữa thẻ đóng và thẻ mở. Những ký tự đặc biệt thì không tự động chuyển thành các entities
innerText	Điền vào hay trả về nội dung giữa thẻ đóng và thẻ mở. Những ký tự đặc biệt tự động chuyển thành các entities
Method	Xác định cách post dữ liệu lên server. Có hai giá trị là “post” và “get”. Mặc định là “post”
Runat	Xác định rằng control này là server control. Phải được xác

	Xác định là “server”
Style	Xác định hay trả về thuộc tính CSS được áp dụng cho control
TagName	Trả về tên của thẻ
Visible	Giá trị boolean xác định control sẽ được hiển thị hay không

5. HtmlImage

HtmlImage control được sử dụng tương ứng thẻ ****. Trong HTML, thẻ **** được sử dụng để hiển thị hình ảnh.

Các thuộc tính

Thuộc tính	Mô tả
Align	Xác định vị trí của hình: <ul style="list-style-type: none"> • Top • Middle • Bottom • Left • right
Alt	Mô tả ngắn về hình
Attributes	Trả về tất cả tên thuộc tính và giá trị tương ứng của thuộc thẻ
Border	Độ dày của viền xung quanh hình
Disabled	Giá trị boolean xác định control có bị disabled hay không. Mặc định là false
Height	Chiều cao của hình
Id	Id duy nhất của control
Runat	Xác định rằng control này là server control. Phải được xác định là “server”
Src	Địa chỉ URL của hình được hiển thị
Style	Xác định hay trả về thuộc tính CSS được áp dụng cho control
TagName	Trả về tên của thẻ
Visible	Giá trị boolean xác định control sẽ được hiển thị hay không
Width	Chiều rộng của hình

Chúng ta có file aspx như sau:

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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></title>
</head>
<body>
    <form id="form1" runat="server">
        <select id="select1" runat="server">
            <option value="Smile.jpg">Mặt cười</option>
            <option value="evil.jpg">Giận dữ</option>
        </select>
        <input id="Submit1" type="submit" runat="server" value="Hiển thị hình"
OnServerClick="choose_image">
        <br /><br />
        
    </form>
</body>
</html>

```

Code xử lý phía server như sau:

```

public partial class _Default : System.Web.UI.Page
{
    protected void choose_image(object sender, EventArgs e)
    {
        image1.Src = select1.Value;
    }
}

```

Giao diện của chúng ta có một combobox, một button. Khi chúng ta chọn một lựa chọn khác trên combobox và click vào button hiển thị thì hình ảnh hiển thị lên sẽ thay đổi tương ứng như hình dưới đây:



6. HtmlInputButton

HtmlInputButton control được sử dụng để điều khiển các thẻ `<input type="button">`, `<input type="submit">`, và `<input type="reset">`. Trong HTML những thẻ này được sử dụng để tạo một nút lệnh, một submit button và một reset button.

Các thuộc tính:

Thuộc tính	Mô tả
Attributes	Trả về tất cả tên thuộc tính và giá trị tương ứng của thuộc thẻ
Disabled	Giá trị boolean xác định control có bị disabled hay không. Mặc định là false
Name	Tên của thẻ
Runat	Xác định rằng control này là server control. Phải được xác định là “server”
Style	Xác định hay trả về thuộc tính CSS được áp dụng cho control
TagName	Trả về tên của thẻ tag
Type	Loại thẻ
Value	Giá trị của thẻ
Visible	Giá trị boolean xác định control sẽ được hiển thị hay không.

Thí dụ phần này chúng ta có thể xem lại thí dụ ở mục **HtmlButton**.

7. HtmlInputCheckBox

HtmlInputCheckBox control được sử dụng để điều khiển thẻ `<input type=“checkbox” >`.

Trong HTML, thẻ này được sử dụng để tạo một checkbox.

Các thuộc tính và các sự kiện:

Thuộc tính	Mô tả
Attributes	Trả về tất cả tên thuộc tính và giá trị tương ứng của thuộc thẻ
Checked	Giá trị boolean xác định thẻ có được chọn hay không
Disabled	Giá trị boolean xác định control có bị disabled hay không Mặc định là false
Id	Id duy nhất của control
Name	Tên của thẻ
Runat	Xác định rằng control này là server control. Phải được xác định là "server"
Style	Xác định hay trả về thuộc tính CSS được áp dụng cho control
TagName	Trả về tên của thẻ
Type	Loại thẻ
Value	Giá trị của thẻ
Visible	Giá trị boolean xác định control sẽ được hiển thị hay không
Sự kiện	Mô tả
ServerChange	Xảy ra khi trạng thái của control thay đổi

Chúng ta có file aspx như sau:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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></title> </head>
<body>

    <form id="form1" runat="server">
        Bạn thích những màu nào? <br />
        <input id="red" type="checkbox" runat="server" /> Đỏ <br />
        <input id="blue" type="checkbox" runat="server" /> Xanh dương <br />
        <input id="green" type="checkbox" runat="server" /> Xanh lục <br />
    </form>
</body>
</html>
```

```

<input id="Button1" type="button" value="Submit" OnServerClick="submit"
runat="server"/>
<p id="p1" runat="server" />
</form>
</body>
</html>

```

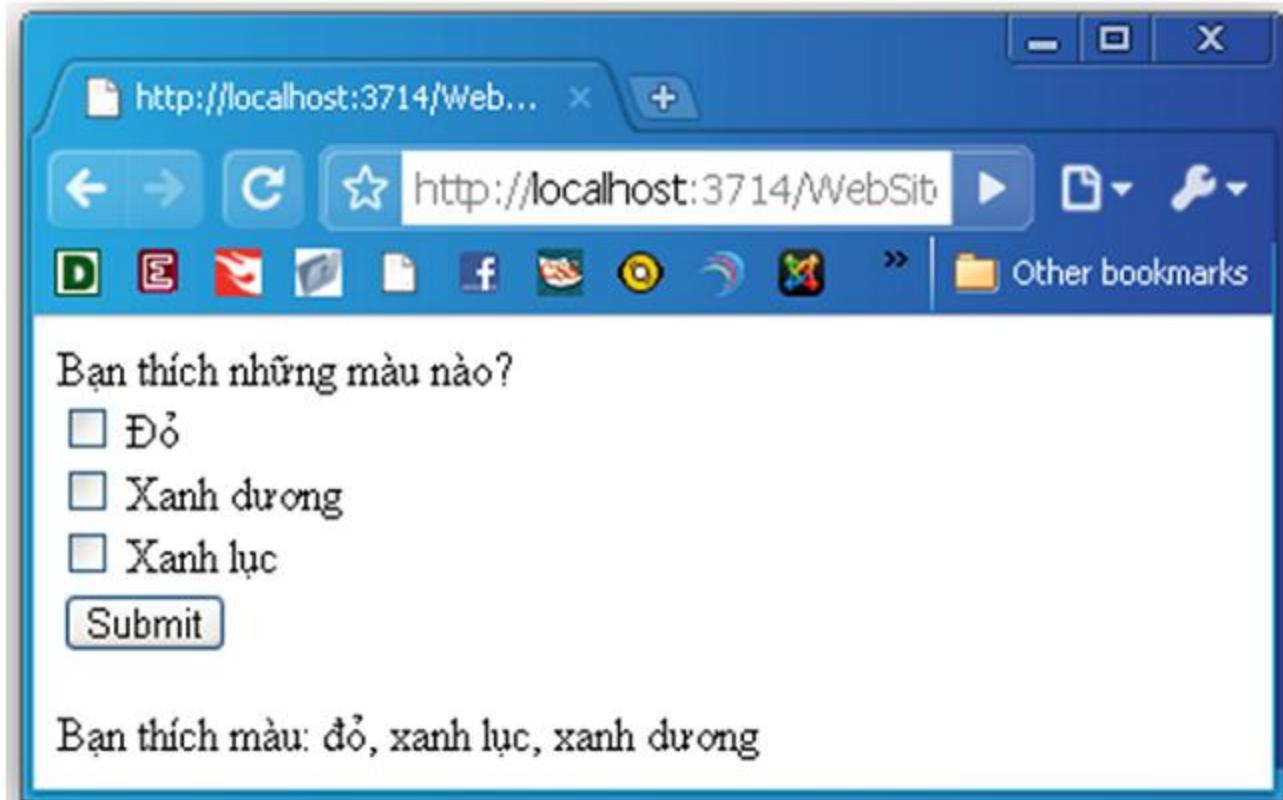
Code xử lý phía server như sau:

```

public partial class _Default : System.Web.UI.Page
{
    protected void submit(object sender, EventArgs e) {
        string sResult = "";
        if (red.Checked)      sResult = "đỏ, ";
        if (green.Checked)    sResult = sResult + "xanh lục, ";
        if (blue.Checked)     sResult = sResult + "xanh dương";
        p1.InnerHtml = "Bạn thích màu: " + sResult;
        red.Checked = false;
        green.Checked = false;
        blue.Checked = false;
    }
}

```

Giao diện của chúng ta có ba checkbox và một button. Khi button được click thì hàm xử lý sự kiện button click được thực hiện và kiểm tra checkbox nào được chọn và hiển thị nội dung tương ứng lên.



8. HtmlInputFile

HtmlInputFile control được sử dụng để điều khiển thẻ **<input type="file">**. Trong HTML, thẻ này được sử dụng để upload một file lên server.

Các thuộc tính

Thuộc tính	Mô tả
Accept	Danh sách những loại MIME được chấp nhận
Attributes	Trả về tất cả tên thuộc tính và giá trị tương ứng của thuộc thẻ
Disabled	Giá trị boolean xác định control có bị disabled hay không. Mặc định là false
Id	Id duy nhất của control
MaxLength	Số ký tự tối đa được cho phép trong thẻ này
Name	Tên của thẻ
PostedFile	Xác định xem có truy xuất được tới file được post lên không
Runat	Xác định rằng control này là server control. Phải được xác định là "server"
Size	Chiều rộng của thẻ
Style	Xác định hay trả về thuộc tính CSS được áp dụng cho control
TagName	Trả về tên của thẻ
Type	Loại thẻ
Value	Giá trị của thẻ
Visible	Giá trị boolean xác định control sẽ được hiển thị hay không.

Chúng ta có file aspx như sau:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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></title>
</head>
```

```

<body>
    <form id="Form1" method="post" enctype="multipart/form-data" runat="server">
        <p>
            Chọn file để upload lên server:
            <input id="MyFile" type="file" size="40" runat="server">
        </p>
        <p>
            <input id="Submit1" type="submit" value="Upload!" OnServerclick="submit"
            runat="server">
        </p>
        <p>
            <div id="Div1" runat="server">
                Tên file: <span id="fname" runat="server"/><br />
                ContentLength: <span id="clength" runat="server"/> bytes
            </div>
        </p>
    </form>
</body>
</html>

```

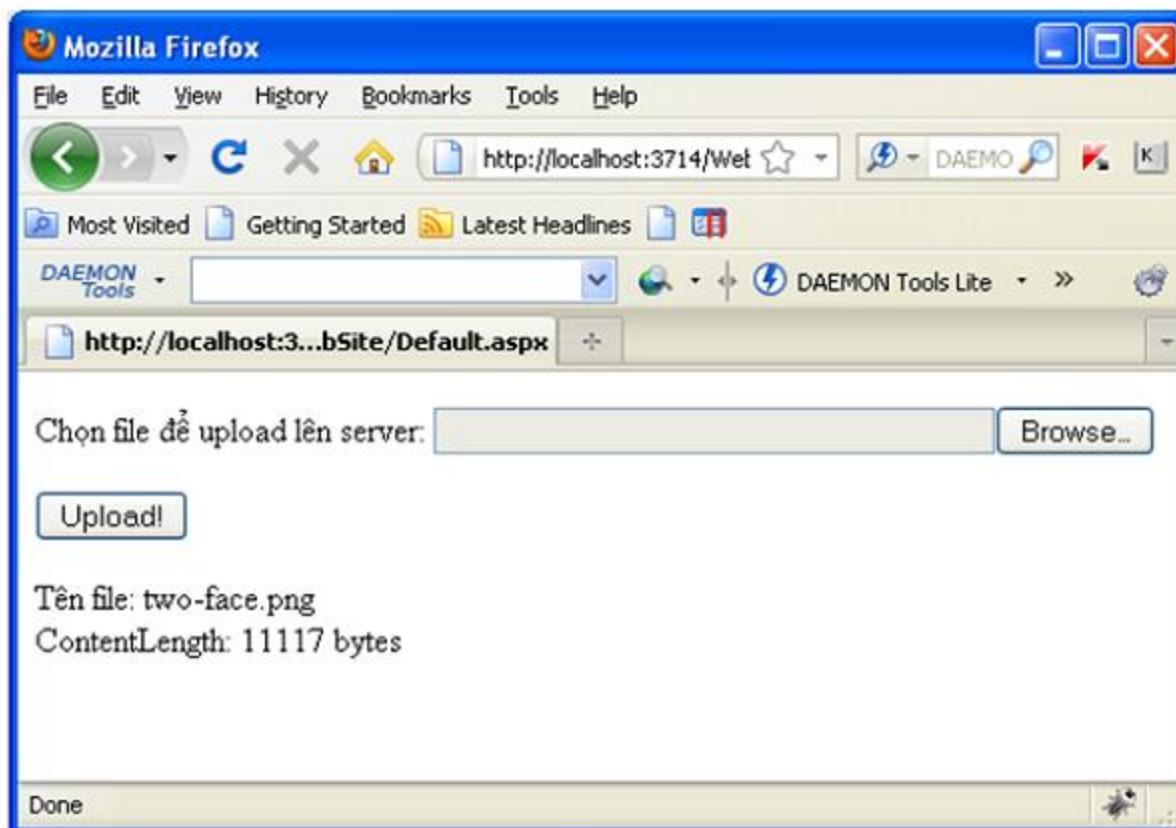
Code xử lý phía server như sau:

```

public partial class _Default: System.Web.UI.Page
{
    protected void submit(object sender, EventArgs e)
    {
        fname.InnerHtml = MyFile.PostedFile.FileName;
        clength.InnerHtml = MyFile.PostedFile.ContentLength.ToString();
        MyFile.PostedFile.SaveAs("c:\\\" + MyFile.PostedFile.FileName);
    }
}

```

Giao diện của chúng ta có một `HtmlInputFile` control, một `HtmlButton` control. Khi chúng ta chọn file để upload lên server và nhấn submit thì hàm xử lý sự kiện click của button sẽ được thực hiện và sẽ lấy tên và độ lớn của file hiển thị lên trang web như hình dưới đây.



9. HtmlInputHidden

HtmlInputHidden control được sử dụng để điều khiển thẻ **<input type="hidden">**. Trong HTML, thẻ này được sử dụng để tạo một hidden input field.

Các thuộc tính và sự kiện:

Thuộc tính	Mô tả
Attributes	Trả về tất cả tên thuộc tính và giá trị tương ứng của thuộc thẻ
Disabled	Giá trị boolean xác định control có bị disabled hay không. Mặc định là false
Id	Id duy nhất của control
Name	Tên của thẻ
PostedFile	Xác định xem có truy xuất được tới file được post lên không
Runat	Xác định rằng control này là server control. Phải được xác định là “server”
Style	Xác định hay trả về thuộc tính CSS được áp dụng cho control
TagName	Trả về tên của thẻ
Type	Loại thẻ
Value	Giá trị của thẻ
Visible	Giá trị boolean xác định control sẽ được hiển thị hay không
Sự kiện	Mô tả
ServerChange	Khi nội dung của thẻ thay đổi

Chúng ta có file aspx như sau:

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

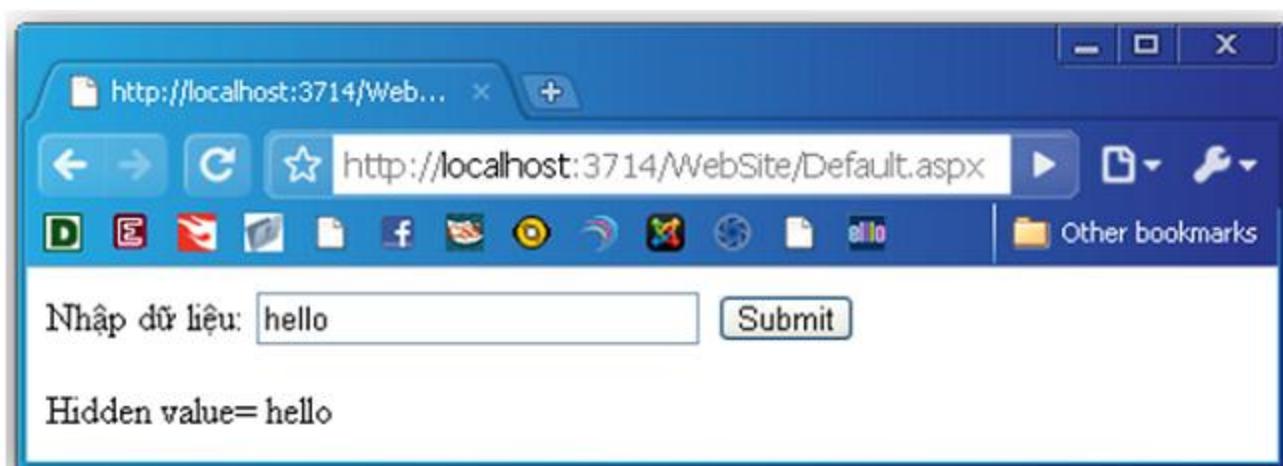
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="Form1" runat="server">
        Nhập dữ liệu: <input id="string1" type="text" size="25" runat="server" />
        <input id="Submit1" type="submit" value="Submit" OnServerClick="submit"
runat="server" />
        <input id="hidden1" type="hidden" runat="server" />
        <p id="p1" runat="server" />
    </form>
</body>
</html>
```

Code phía server như sau:

```
public partial class _Default : System.Web.UI.Page
{
    protected void submit(object sender, EventArgs e)
    {
        hidden1.Value = string1.Value;
        p1.InnerHtml = "Hidden value= " + hidden1.Value;
    }
}
```

Giao diện của chúng ta có một HtmlInputHidden control, một HtmlInputText control, một HtmlInputButton. Khi chúng ta click vào button submit thì hàm xử lý sự kiện click của button được thực thi, khi đó giá trị của hidden field sẽ được gán bằng giá trị của textbox và sau đó hiển thị giá trị của hidden field trong thẻ p như hình sau đây:



10. HtmlInputImage

HtmlInputImage control được sử dụng để điều khiển thẻ **<input type="image">**. Trong HTML, thẻ này được sử dụng để tạo một input button sử dụng hình, thay cho loại button thông thường.

Các thuộc tính:

Thuộc tính	Mô tả
Align	Cách sắp xếp của hình
Alt	Text thay thế để hiển thị cho hình
Attributes	Trả về tất cả tên thuộc tính và giá trị tương ứng của thuộc thẻ
Border	Độ dày của viền xung quanh của thẻ
Disabled	Giá trị boolean xác định control có bị disabled hay không. Mặc định là false
Id	Id duy nhất của control
Name	Tên của thẻ
OnServerClick	Tên của hàm được thực thi khi hình được click
Runat	Xác định rằng control này là server control. Phải được xác định là “server”
Src	Source của image
Style	Xác định hay trả về thuộc tính CSS được áp dụng cho control
TagName	Trả về tên của thẻ
Type	Loại thẻ
Value	Giá trị của thẻ
Visible	Giá trị boolean xác định control sẽ được hiển thị hay không.

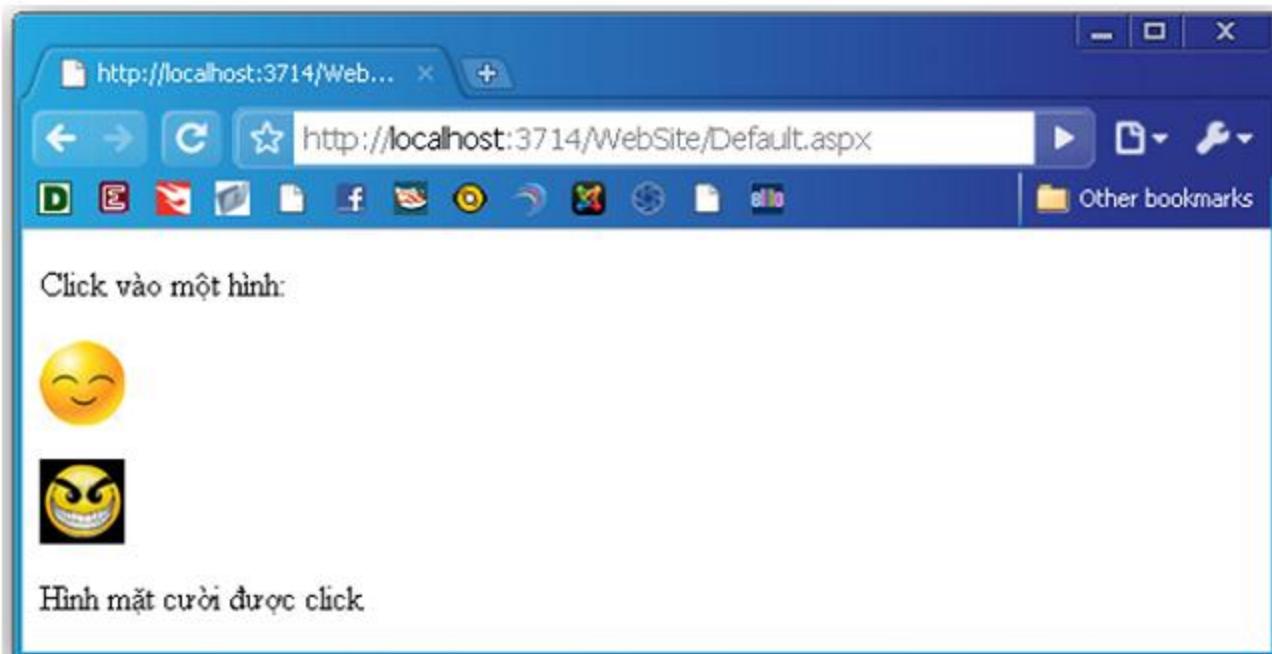
Chúng ta có file aspx như sau:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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></title>
</head>
<body>
    <form id="Form1" runat="server">
        <p>Click vào một hình:</p>
        <p>
            <input id="Image1" type="image" src="Smile.jpg"
OnServerClick="button1" runat="server" width="40" height="40" />
        </p>
        <p>
            <input id="Image2" type="image" src="evil.jpg"
OnServerClick="button2" runat="server" width="40" height="40" />
        </p>
        <p id="p1" runat="server" />
    </form>
</body>
</html>
```

Code phía server như sau:

```
public partial class _Default : System.Web.UI.Page
{
    protected void button1(object sender, EventArgs e)
    {
        p1.InnerHtml = "Hình mặt cười được click";
    }
    protected void button2(object sender, EventArgs e)
    {
        p1.InnerHtml = "Hình mặt giận dữ được click";
    }
}
```

Giao diện của chúng ta có hai **HtmlInputImage** control. Khi chúng ta click vào mỗi hình thì hàm xử lý sự kiện click của hai hình được thực thi và thông điệp được hiện thị trong thẻ **p** như hình dưới đây.



11. HtmlInputRadioButton

HtmlInputRadioButton được sử dụng để điều khiển thẻ **<input type="radio">**. Trong HTML, thẻ này được sử dụng để tạo một radiobutton.

Các thuộc tính:

Thuộc tính	Mô tả
Attributes	Trả về tất cả tên thuộc tính và giá trị tương ứng của thuộc thẻ
Checked	Giá trị boolean xác định thẻ có được chọn hay không
Disabled	Giá trị boolean xác định control có bị disabled hay không. Mặc định là false
Id	Id duy nhất của control
Name	Tên của thẻ
Runat	Xác định rằng control này là server control. Phải được xác định là “server”
Style	Xác định hay trả về thuộc tính CSS được áp dụng cho control
TagName	Trả về tên của thẻ
Type	Loại thẻ
Value	Giá trị của thẻ
Visible	Giá trị boolean xác định control sẽ được hiển thị hay không

Chúng ta có file aspx như sau:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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></title>

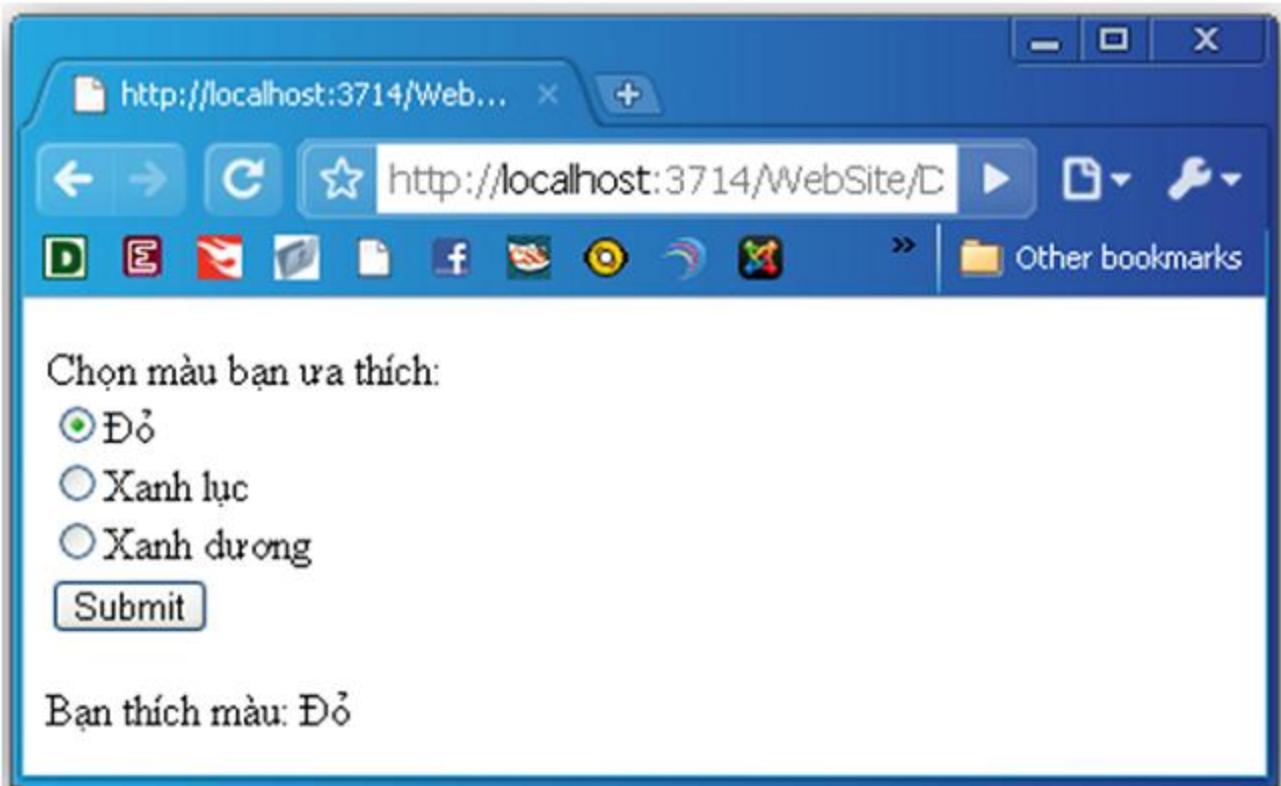
</head>
<body>
    <form id="Form1" runat="server">
        <p>Chọn màu bạn ưa thích:<br />
        <input id="r1" name="col" type="radio" runat="server" >Đỏ</input>
        <br />
        <input id="r2" name="col" type="radio" runat="server">Xanh lục</input>
        <br />
        <input id="r3" name="col" type="radio" runat="server">Xanh dương</input>
        <br />
        <input id="Button1" type="button" value="Submit" OnServerClick="submit"
runat="server"/>
        <p id="p1" runat="server" />
    </form>
</body>
</html>
```

Code phía server như sau:

```
public partial class _Default: System.Web.UI.Page
{
    protected void submit(object sender, EventArgs e)
    {
        string sColor="";
        if(r1.Checked)
            sColor = "Đỏ";
        else if(r2.Checked)
            sColor = "Xanh lục";
        else
            sColor = "Xanh dương";

        p1.InnerHtml = "Bạn thích màu: " + sColor;
    }
}
```

Giao diện của chúng ta có ba `HtmlInputRadioButton`, một `HtmlInputButton`. Khi button submit được click thì hàm xử lý sự kiện click của nó được thực thi, nếu radiobutton nào được chọn thì màu tương ứng đó sẽ được hiển thị lên thông qua thẻ `p` như hình dưới đây:



12. HtmlInputText

HtmlInputText control được sử dụng để điều khiển hai thẻ **<input type="text">** và **<input type="password">**. Trong HTML, những thẻ được sử dụng để tạo một text field và một password field.

Các thuộc tính:

Thuộc tính	Mô tả
Attributes	Trả về tất cả tên thuộc tính và giá trị tương ứng của thuộc thẻ
Disabled	Giá trị boolean xác định control có bị disabled hay không. Mặc định là false
Id	Id duy nhất của control
MaxLength	Số ký tự tối đa được cho phép của thẻ này
Name	Tên của thẻ
Runat	Xác định rằng control này là server control. Phải được xác định là “server”
Size	Chiều rộng của thẻ
Style	Xác định hay trả về thuộc tính CSS được áp dụng cho control
TagName	Trả về tên của thẻ
Type	Loại thẻ
Value	Giá trị của thẻ
Visible	Giá trị boolean xác định control sẽ được hiển thị hay không

Chúng ta có thể xem thí dụ ở mục HtmlButton.

13. HtmlSelect

HtmlSelect control được sử dụng điều khiển thẻ **<select>**. Trong HTML, thẻ này được sử dụng để tạo drop-down list.

Các thuộc tính:

Thuộc tính	Mô tả
Attributes	Trả về tất cả tên thuộc tính và giá trị tương ứng của thuộc thẻ
DataMember	Tên của bảng dữ liệu để sử dụng

DataSource	Data source để sử dụng
DataTextField	Trường trong data source được hiển thị trong drop-down list
DataValueField	Trường trong data source để xác định giá trị của mỗi item trong drop-down list
Disabled	Giá trị boolean xác định control có bị disabled hay không. Mặc định là false
Id	Id duy nhất của control
innerHTML	Điền vào hay trả về nội dung giữa thẻ đóng và thẻ mở. Những ký tự đặc biệt thì không tự động chuyển thành các entities
innerText	Điền vào hay trả về nội dung giữa thẻ đóng và thẻ mở. Những ký tự đặc biệt tự động chuyển thành các entities
Items	Danh sách của những item trong drop-down list.
Multiple	Xác định nhiều item có thể được chọn tại cùng một thời điểm
OnServerChange	Tên của hàm được thực thi khi item được chọn bị thay đổi
Runat	Xác định rằng control này là server control. Phải được xác định là “server”
SelectedIndex	Index của item được chọn hiển tại
Size	Chiều rộng của thẻ
Style	Xác định hay trả về thuộc tính CSS được áp dụng cho control
TagName	Trả về tên của thẻ
Value	Giá trị của thẻ
Visible	Giá trị boolean xác định control sẽ được hiển thị hay không

Chúng ta xem thí dụ ở mục HtmlImage.

14. HtmlTable

HtmlTable control được sử dụng để điều khiển thẻ <table>. Trong HTML, thẻ table được sử dụng để tạo một table.

Các thuộc tính:

Thuộc tính	Mô tả
Align	Xác định cách sắp xếp của table
Attributes	Trả về tất cả tên thuộc tính và giá trị tương ứng thuộc thẻ
BGColor	Xác định màu nền của table
Border	Xác định độ dày của viền
BorderColor	Xác định màu của border
CellPadding	Xác định khoảng cách giữa viền của ô và nội dung bên trong
CellSpacing	Xác định khoảng cách giữa các ô
Disabled	Giá trị boolean xác định control có bị disabled hay không. Mặc định là false
Id	Id duy nhất của control
innerHTML	Điền vào hay trả về nội dung giữa thẻ đóng và thẻ mở. những ký tự đặc biệt thì không tự động chuyển thành các entities
innerText	Điền vào hay trả về nội dung giữa thẻ đóng và thẻ mở. Những ký tự đặc biệt tự động chuyển thành các entities
Rows	Trả về đối tượng HtmlRowCollection thể hiện tất cả các dòng trong table
Style	Xác định hay trả về thuộc tính CSS được áp dụng cho control
TagName	Trả về tên của thẻ
Visible	Giá trị boolean xác định control sẽ được hiển thị hay không
Width	Xác định chiều rộng của table

Chúng ta có trang aspx như sau:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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></title></head>
<body>
```

```

<form id="Form1" runat="server">
    <p>số dòng:<br />
    <select id="rows1" runat="server">
        <option value="1">1</option>
        <option value="2">2</option>
        <option value="3">3</option>
    </select>
    <br />số cột:<br />
    <select id="cells1" runat="server">
        <option value="1">1</option>
        <option value="2">2</option>
        <option value="3">3</option>
    </select>
    <br /><br />
    <input id="Submit1" type="submit" value="Hiển thị Table" runat="server"
OnServerClick="submit">
</p>
<table id="t1" border="1" runat="server" visible="false"/>
</form>
</body>
</html>

```

Code phía server như sau:

```

public partial class _Default: System.Web.UI.Page
{
    protected void submit(object sender, EventArgs e)
    {
        int row, numrows, numcells, j, i;
        row = 0;
        numrows = int.Parse(rows1.Value);
        numcells = int.Parse(cells1.Value);
        for (j = 1; j <= numrows; j++)
        {
            HtmlTableRow r = new HtmlTableRow();

```

```

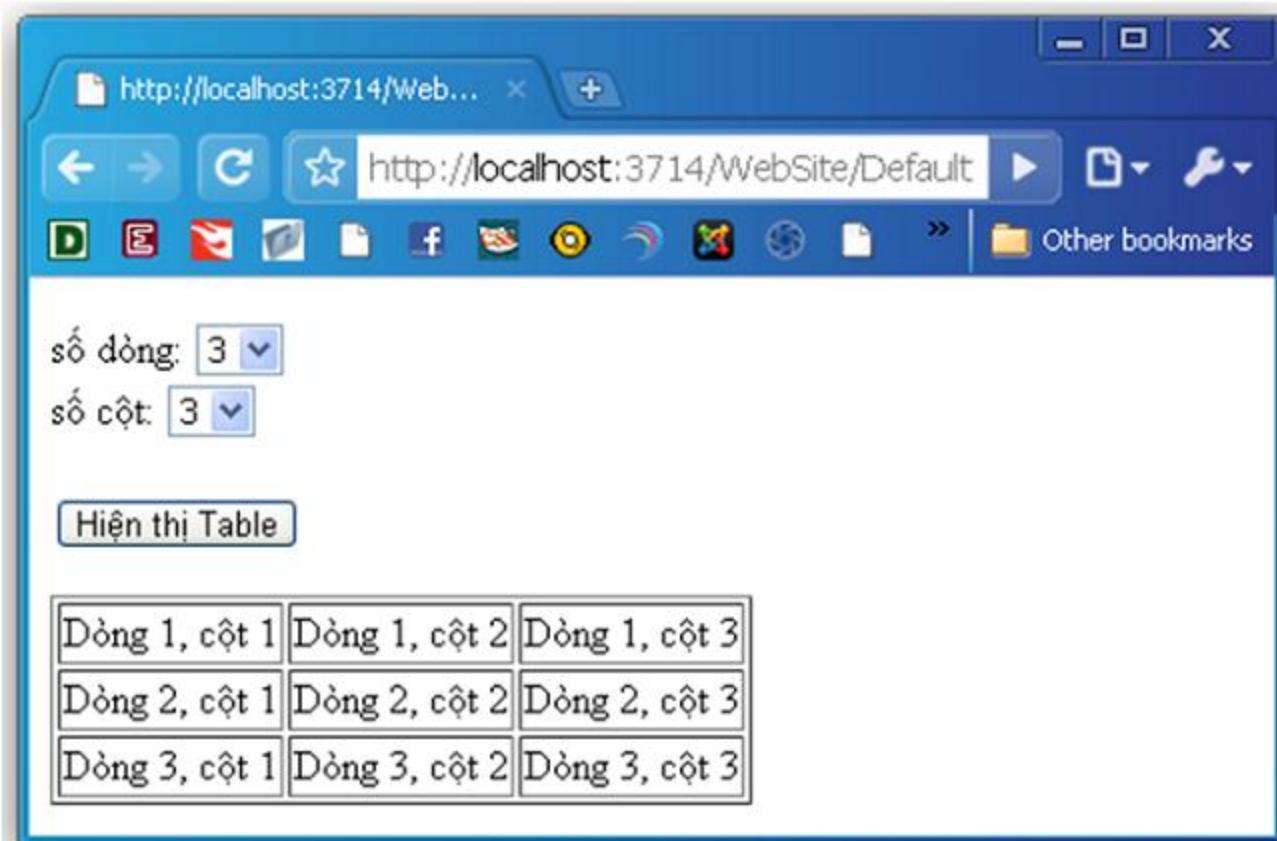
row = row + 1;

for (i = 1; i <= numcells; i++)
{
    HtmlTableCell c = new HtmlTableCell();
    c.Controls.Add(new LiteralControl("Dòng " + j + ", cột " + i));
    r.Cells.Add(c);
}

t1.Rows.Add(r);
t1.Visible = true;
}
}
}

```

Giao diện của chúng ta có hai `HtmlSelect` control, một `HtmlInputButton`. Người dùng có thể chọn số dòng và số cột của table. Khi người dùng click submit button thì hàm xử lý sự kiện button click được thực hiện và sinh ra table với số dòng và số cột chúng ta đã lựa chọn.



15. `HtmlTableCell`

`HtmlTableCell` control được sử dụng để điều khiển thẻ `<td>` và `<th>`. Trong HTML những thẻ này được sử dụng để tạo những cell trong table và những cell header của table.

Các thuộc tính:

Thuộc tính	Mô tả
Align	Sắp xếp theo chiều ngang nội dung của một cell
Attributes	Trả về tất cả tên thuộc tính và giá trị tương ứng của thuộc thẻ
BGColor	Xác định màu nền của cell
Border	Xác định độ dày của viền
ColSpan	Số cột của cell được mở rộng
Disabled	Giá trị boolean xác định control có bị disabled hay không. Mặc định là false
Id	Id duy nhất của control
InnerHTML	Điền vào hay trả về nội dung giữa thẻ đóng và thẻ mở. những ký tự đặc biệt thì không tự động chuyển thành các entities
InnerText	Điền vào hay trả về nội dung giữa thẻ đóng và thẻ mở. Những ký tự đặc biệt tự động chuyển thành các entities
RowSpan	Số dòng của cell được mở rộng
Runat	Xác định rằng control này là server control. Phải được xác định là “server”
Style	Xác định hay trả về thuộc tính CSS được áp dụng cho control
TagName	Trả về tên của thẻ
VAlign	Sắp xếp theo chiều dọc nội dung của cell
Visible	Giá trị boolean xác định control sẽ được hiển thị hay không
Width	Xác định chiều rộng của cell

Chúng ta có file aspx như sau:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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></title>
</head>
<body>
    <form id="Form1" runat="server">
        <table id="table1" border="1" runat="server">
            <tr>
                <td>Cell 1</td>
                <td>Cell 2</td>
            </tr>
            <tr>
                <td>Cell 3</td>
                <td>Cell 4</td>
            </tr>
        </table>
        <br />
        <input id="Button1" type="button" value="Thay đổi nội dung"
OnServerClick="submit" runat="server"/>
    </form>
</body>
</html>

```

Code phía server như sau:

```

public partial class _Default: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected void submit(object sender, EventArgs e)
    {
        int i,j;
        table1.BgColor="yellow";
    }
}

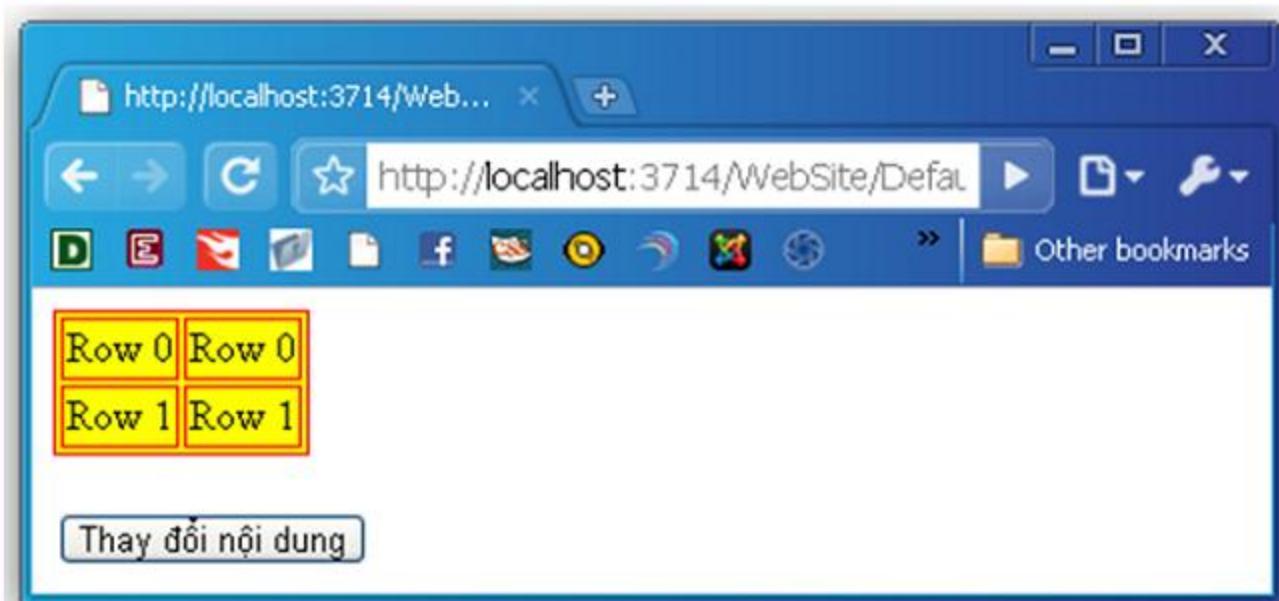
```

```

table1.BorderColor="red";
for (i = 0; i <= table1.Rows.Count - 1; i++)
    for (j = 0; j <= table1.Rows[i].Cells.Count - 1; j++)
        table1.Rows[i].Cells[j].InnerHtml = "Row " + i;
}
}

```

Giao diện của chúng ta có một `HtmlTable` control và một `HtmlInputButton` control. Khi chúng ta click vào submit button thì hàm xử lý sự kiện click của button này được thực hiện. Khi đó, màu nền của table được gán là màu vàng và viền của table được gán lại thành màu đỏ như hình dưới đây:



16. `HtmlTableRow`

`HtmlTableRow` control được sử dụng để điều khiển thẻ `<tr>`. Trong HTML, thẻ `<tr>` được sử dụng để tạo table row.

Các thuộc tính:

Thuộc tính	Mô tả
Align	Sắp xếp theo chiều ngang nội dung của row
Attributes	Trả về tất cả tên thuộc tính và giá trị tương ứng của thuộc thẻ
BGColor	Xác định màu nền của row
Border	Xác định độ dày của viền
Cells	Trả về những cell trong row này
Disabled	Giá trị boolean xác định control có bị disabled hay không. Mặc định là false
Id	Id duy nhất của control

InnerHTML	Điền vào hay trả về nội dung giữa thẻ đóng và thẻ mở. Những ký tự đặc biệt thì không tự động chuyển thành các entities
InnerText	Điền vào hay trả về nội dung giữa thẻ đóng và thẻ mở. Những ký tự đặc biệt tự động chuyển thành các entities
Runat	Xác định rằng control này là server control. Phải được xác định là “server”
Style	Xác định hay trả về thuộc tính CSS được áp dụng cho control
TagName	Trả về tên của thẻ tag
VAlign	Sắp xếp theo chiều dọc nội dung của row
Visible	Giá trị boolean xác định control sẽ được hiển thị hay không

Chúng ta có thể xem thí dụ ở mục HtmlTableCell (15).

17. HtmlTextArea

HtmlTextArea control được sử dụng để điều khiển thẻ **<textarea>**. Trong HTML, thẻ này được sử dụng để tạo text area.

Các thuộc tính:

Thuộc tính	Mô tả
Attributes	Trả về tất cả tên thuộc tính và giá trị tương ứng của thuộc thẻ
Disabled	Giá trị boolean xác định control có bị disabled hay không. Mặc định là false
Id	Id duy nhất của control
InnerHTML	Điền vào hay trả về nội dung giữa thẻ đóng và thẻ mở. Những ký tự đặc biệt thì không tự động chuyển thành các entities
InnerText	Điền vào hay trả về nội dung giữa thẻ đóng và thẻ mở. Những ký tự đặc biệt tự động chuyển thành các entities
Name	Tên duy nhất của text area
OnServerChange	Tên của hàm được thực thi khi nội dung của textarea bị thay đổi
Rows	Số dòng hiển thị của text area
Runat	Xác định rằng control này là server control. Phải được xác định là “server”

Style	Xác định hay trả về thuộc tính CSS được áp dụng cho control
TagName	Trả về tên của thẻ tag
Value	Nội dung của textarea
Visible	Giá trị boolean xác định control sẽ được hiển thị hay không

Chúng ta có file aspx như sau:

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

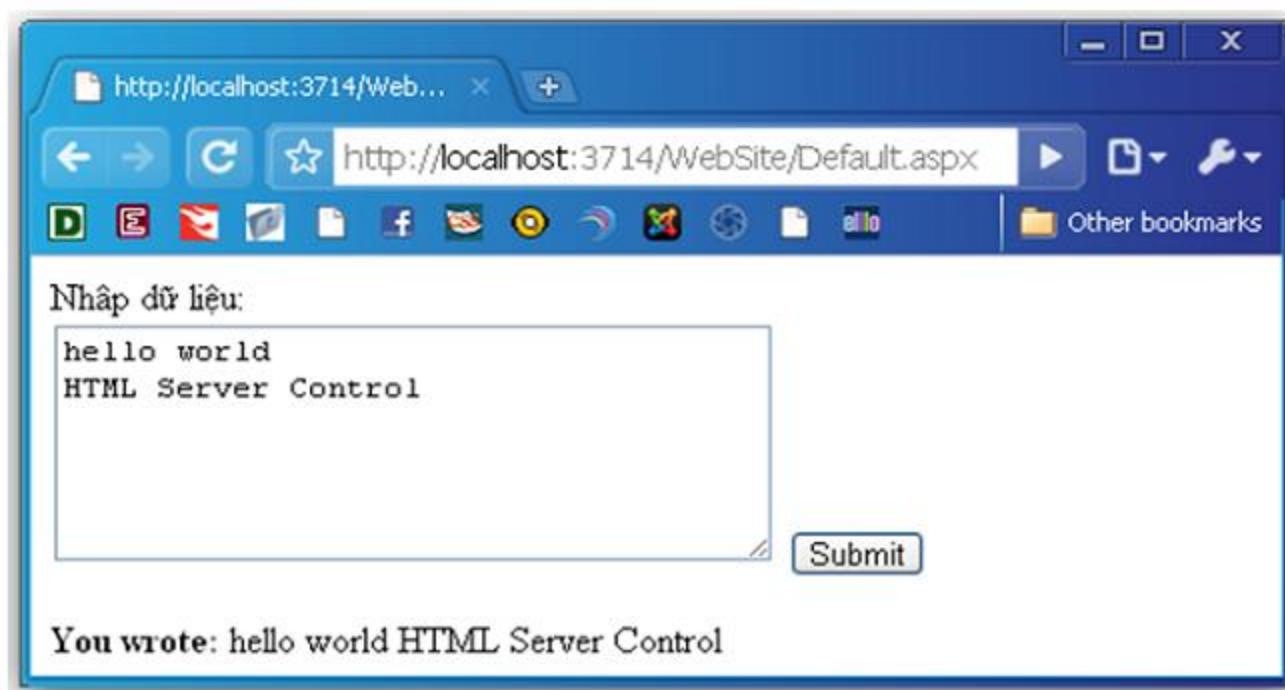
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server"> <title></title> </head>
<body>
    <form id="Form1" runat="server">
        Nhập dữ liệu:<br />
        <textarea id="textarea1" cols="35" rows="6" runat="server" />
        <input id="Submit1" type="submit" value="Submit" OnServerClick="submit"
runat="server" />
        <p id="p1" runat="server" />
    </form>
</body>
</html>
```

Code phía server như sau:

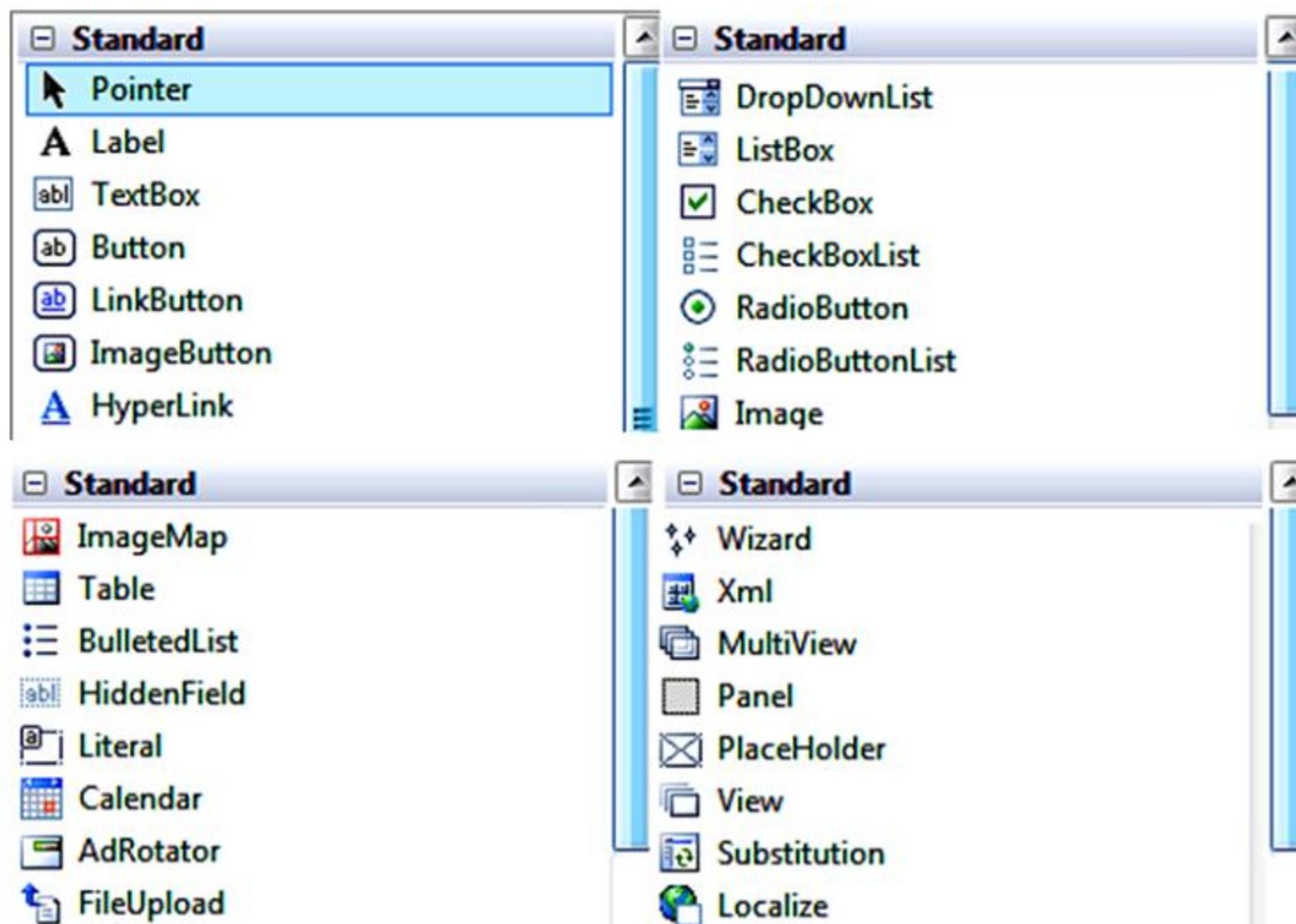
```
public partial class _Default : System.Web.UI.Page
{
    protected void submit(object sender, EventArgs e)
    {
        p1.InnerHtml = "<b>You wrote:</b> " + textarea1.Value;
    }
}
```

Giao diện của chúng ta gồm có một HtmlTextArea control, một HtmlInputButton control. Khi button submit được click thì hàm xử lý sự kiện click của nó được thực hiện, và thông điệp chúng ta ghi trong **textArea** sẽ được in ra trong thẻ **p**.



2.5. ASP.NET SERVER CONTROLS

Cũng giống như HTML server control, các Web server control cũng được tạo trên server và chúng có sẵn thuộc tính **runat="server"** để thực thi tại server. Tuy nhiên, Web server control không nhất thiết phải ánh xạ tới bất kỳ thẻ HTML nào đã tồn tại và chúng có thể hiển thị những thẻ có chức năng phức tạp hơn.



Hình 2.12: Các Web Server Control trên thanh toolbox

Cú pháp để tạo một Web server control là:

<asp:control_name id="some_id" runat="server" />

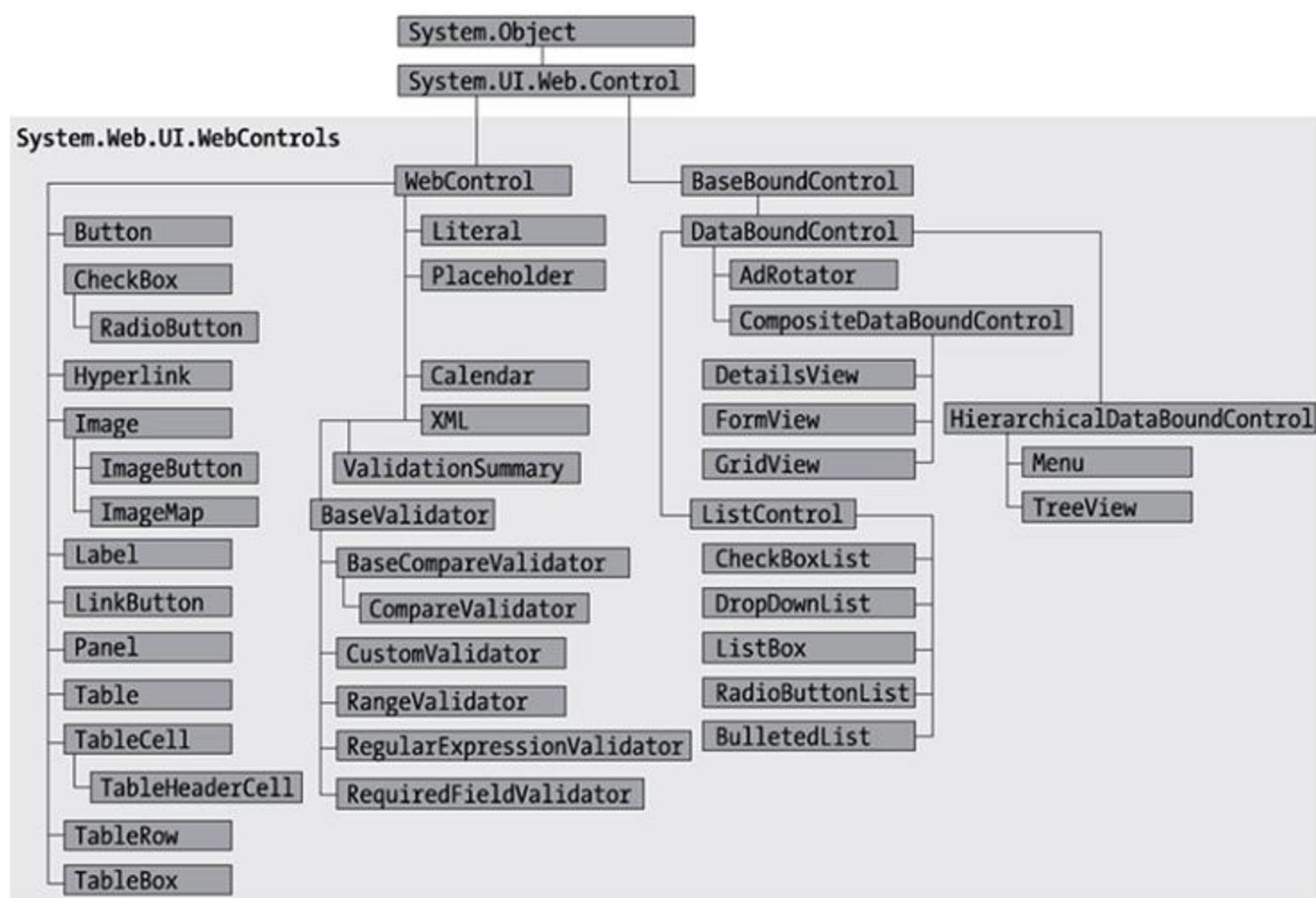
Dưới đây là các lý do bạn nên sử dụng ASP.NET Web Control:

- *Đơn giản, tương tự như các điều khiển trên Windows Form.*
- *Đồng nhất: Các điều khiển Web server có các thuộc tính giống nhau -> dễ tìm hiểu và sử dụng.*
- *Hiệu quả: Các điều khiển Web Server tự động phát sinh ra các tag HTML theo từng loại Browser.*
- *Các controls ASP.NET là các lớp, do đó, ta có thể mở rộng các chức năng của các control (customize control).*

Bảng liệt kê các thuộc tính chung của các Web control

Thuộc tính	Kiểu	Ý nghĩa
(ID)	String	Quy định tên của điều khiển. Tên của điều khiển là duy nhất
AccessKey	String	Quy định ký tự để di chuyển nhanh đến điều khiển - ký tự xử lý phím nóng
Attributes	AttributeCollection	Tập hợp các thuộc tính của điều khiển HTML
BackColor	Color	Quy định màu nền của điều khiển
BorderColor	Color	Quy định màu đường viền của điều khiển
BorderStyle	BorderStyle	Quy định kiểu đường viền của điều khiển
BorderWidth	Unit	Quy định độ rộng của đường viền
CssClass	String	Quy định hình thức hiển thị của điều khiển qua tên CSS
Enabled	Boolean	Quy định điều khiển có được hiển thị hay không. Mặc định là True – được phép hiển thị
Font	FontInfo	Quy định font hiển thị cho điều khiển
ForeColor	Color	Quy định màu chữ hiển thị trên điều khiển
Height	Unit	Quy định chiều cao của điều khiển
ToolTip	String	Dòng chữ sẽ hiển thị khi rê chuột vào điều khiển
Width	Unit	Quy định độ rộng của điều khiển

Các Web Server Controls theo mô hình phân cấp sau:



Hình 2.13: Cấu trúc lớp WebControl

Khảo sát chi tiết các Web Server Controls:

1. AdRotator

AdRotator control được sử dụng để hiện thị tuần tự những hình quảng cáo trên trang web.

Control này sử dụng file XML để lưu trữ thông tin quảng cáo. File XML phải bắt đầu và kết thúc với thẻ `<Advertisements>`. Bên trong thẻ `<Advertisements>` có thể có nhiều thẻ `<Ad>`

Những thẻ nằm trong thẻ `<Ad>` được liệt kê sau đây:

Thẻ	Mô tả
<code><ImageUrl></code>	Tùy chọn. Đường dẫn tới file hình ảnh
<code><NavigateUrl></code>	Tùy chọn. URL tới link nếu người dùng click vào quảng cáo
<code><AlternateText></code>	Tùy chọn. Text thay thế cho hình
<code><Keyword></code>	Tùy chọn. Loại quảng cáo
<code><Impressions></code>	Tùy chọn. Hiển thị tỷ lệ phần trăm của những người click

Các thuộc tính

Thuộc tính	Mô tả
AdvertisementFile	Xác định đường dẫn tới file XML thông tin quảng cáo
AlternateTextField	Xác định trường dữ liệu được sử dụng thay cho Alt text
ImageUrlField	Xác định trường dữ liệu được sử dụng thay cho thuộc tính ImageURL
KeywordFilter	Xác định một bộ lọc để giới hạn những quảng cáo sau những loại
NavigateUrlField	Xác định trường dữ liệu được sử dụng thay cho thuộc tính NavigateUrl cho một quảng cáo
Runat	Xác định control là một server control. Phải được xác định là “server”
Target	Xác định nơi mở URL

Chúng ta có file aspx như sau:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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></title> </head>
<body>
    <form id="Form1" runat="server">
        <asp:AdRotator ID="AdRotator1" AdvertisementFile="Ad1.xml" runat="server"
OnAdCreated="change_url" target="_blank" />
    <p><a href="ad1.xml" target="_blank">View XML file</a></p>
    </form>
</body>
</html>
```

Code phía server như sau:

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e) { }
    protected void change_url(object sender, AdCreatedEventArgs e) {
        e.NavigateUrl = "http://www.w3schools.com";
    }
}
```

File XML như sau:

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements>
    <Ad>
        <ImageUrl>Smile.jpg</ImageUrl>
        <NavigateUrl>http://www.w3schools.com</NavigateUrl>
        <AlternateText>W3Schools Main Site</AlternateText>
        <Impressions>50</Impressions>
        <Keyword>elearning</Keyword>
    </Ad>
    <Ad>
        <ImageUrl>evil.jpg</ImageUrl>
        <NavigateUrl>http://www.w3schools.com/xhtml/default.asp</NavigateUrl>
        <AlternateText>XHTML Tutorial</AlternateText>
        <Impressions>50</Impressions>
        <Keyword>XHTML</Keyword>
    </Ad>
</Advertisements>
```

Kết quả thực thi như hình dưới đây:



2. Button

Button control được sử dụng để hiển thị nút nhấn trên trang web. Push button có thể là submit button hoặc command button. Mặc định control này là một submit button.

Submit button không có tên lệnh và nó post trang web trở lại server khi nó được click. Có thể viết một xử lý sự kiện để điều khiển việc hoạt động khi button submit được click.

Command button có một tên lệnh và cho phép chúng ta tạo nhiều control button trên trang. Có thể viết một quản lý sự kiện để điều khiển việc hoạt động khi command button được click.

Các thuộc tính:

Thuộc tính	Mô tả
CausesValidation	Xác định nếu trang được xác nhận khi button được click
CommandArgument	Bổ sung thông tin về lệnh để thực hiện
CommandName	Chỉ ra lệnh liên quan đến sự kiện lệnh
OnClientClick	Xác định tên của hàm được thực thi khi một button được click
PostBackUrl	Xác định URL của trang để post từ trang hiện tại khi button click
Runat	Xác định control là một server control. Phải được xác định là “server”
Text	Xác định text trên button
UserSubmitBehavior	Xác định có hay không một button sử dụng cơ chế submit của browser hay cơ chế postback của ASP.NET
ValidationGroup	Xác định nhóm những control, một button gây ra sự đánh giá (phê chuẩn) khi nó post back về server

Chúng ta có file aspx như sau:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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></title>

</head>
<body>
    <form id="Form1" runat="server">
        <asp:Button id="button1" Text="Click me!" runat="server" OnClick="submit" />
    </form>
</body>
</html>

```

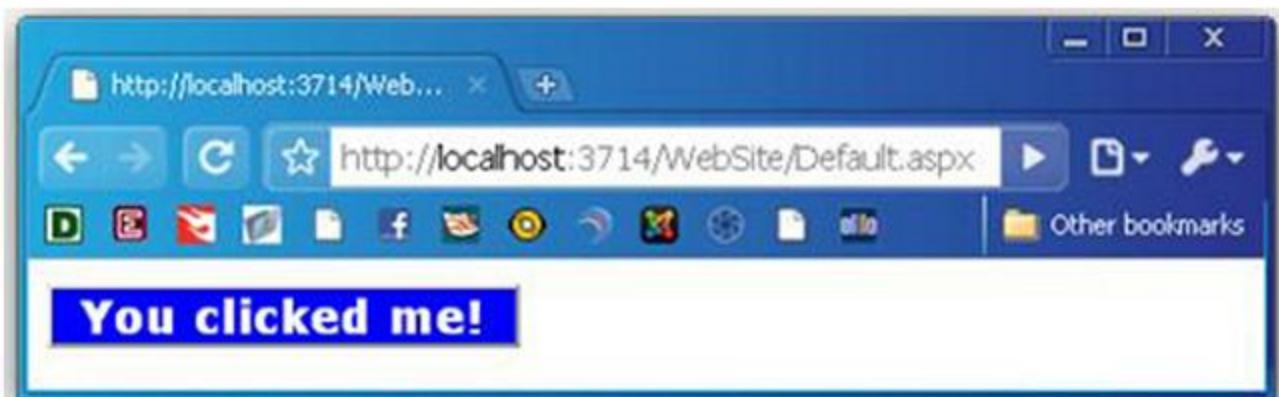
Code phía server như sau:

```

public partial class _Default : System.Web.UI.Page
{
    protected void submit(object sender, EventArgs e)
    {
        button1.Style["background-color"] = "#0000ff";
        button1.Style["color"] = "#ffffff";
        button1.Style["width"] = "200px";
        button1.Style["cursor"] = "pointer";
        button1.Style["font-family"] = "verdana";
        button1.Style["font-weight"] = "bold";
        button1.Style["font-size"] = "14pt";
        button1.Text = "You clicked me!";
    }
}

```

Giao diện của chúng ta có một submit button control. Chúng ta tạo một hàm quản lý sự kiện cho sự kiện Click làm nhiệm vụ thay đổi text và style của button như hình dưới đây:



3. Canlendar

Calendar control được sử dụng để hiển thị một calendar trong trang web.

Control này hiển thị lịch một tháng để người dùng chọn ngày và di chuyển tới những tháng sau và những tháng trước.

Các thuộc tính:

Thuộc tính	Mô tả
Caption	Đầu đề của lịch
CaptionAlign	Sắp xếp đầu đề của lịch
CellPadding	Khoảng cách giữa các cell và nội dung
CellSpacing	Khoảng cách giữa các cell
DayHeaderStyle	Style để hiển thi tên của ngày
DayNameFormat	Định dạng hiển thị những tên của ngày
DayStyle	Style hiển thi ngày
FirstDayOfWeek	Xác định ngày nào là ngày đầu tuần
NextMonthText	Text được hiển thị cho link tới tháng tiếp theo
NextPrevFormat	Định dạng link tới tháng trước và tháng tiếp theo
NextPreStyle	Style để hiển thi link tới tháng trước và tháng tiếp theo
OtherMonthDayStyle	Style để hiển thi ngày mà không phải ngày hiện tại
PrevMonthText	Text hiển thi link tới tháng trước.
runat	Xác định control là một server control. Phải được xác định là “server”
SelectedDate	Ngày được chọn
SelectedDates	Những ngày được chọn
SelectedDayStyle	Style để cho ngày được chọn
SelectionMode	Làm thế nào người dùng chọn một ngày
SelectMonthText	Text hiển thi link chọn tháng

SelectorStyle	Style cho link chọn tuần và tháng
SelectWeekText	Text hiển thị cho link chọn tuần
ShowDayHeader	Giá trị boolean xác định các ngày của tuần có được hiển thị hay không
ShowGridLines	Giá trị boolean xác định lưới phân cách giữa các ngày có hiển thị hay không
ShowNextPrevMonth	Giá trị boolean xác định có hiện link tới tháng trước và tháng tiếp theo hay không
ShowTitle	Giá trị boolean xác định tựa đề của lịch có thực hiện hay không
TitleFormat	Định dạng tựa đề của lịch
TitleStyle	Style của tựa đề của lịch
TodayDayStyle	Style cho ngày hiện tại
TodayDate	Ngày hiện tại
UseAccessibleHeader	Xác định sử dụng thẻ <th> cho những ngày của tuần (thứ) thay cho thẻ <td>
VisibleDate	Xác định ngày hiện tại có hiển thị trong lịch hay không
WeekendDayStyle	Style cho những ngày cuối tuần
OnDayRender	Tên của hàm được thực thi khi mỗi cell ngày được tạo
OnSelectionChanged	Tên của hàm được thực thi khi người dùng chọn một ngày, tuần hoặc tháng
OnVisibleMonthChanged	Tên của hàm được thực thi khi người dùng di chuyển tới một tháng khác

Chúng ta có file aspx như sau:

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

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

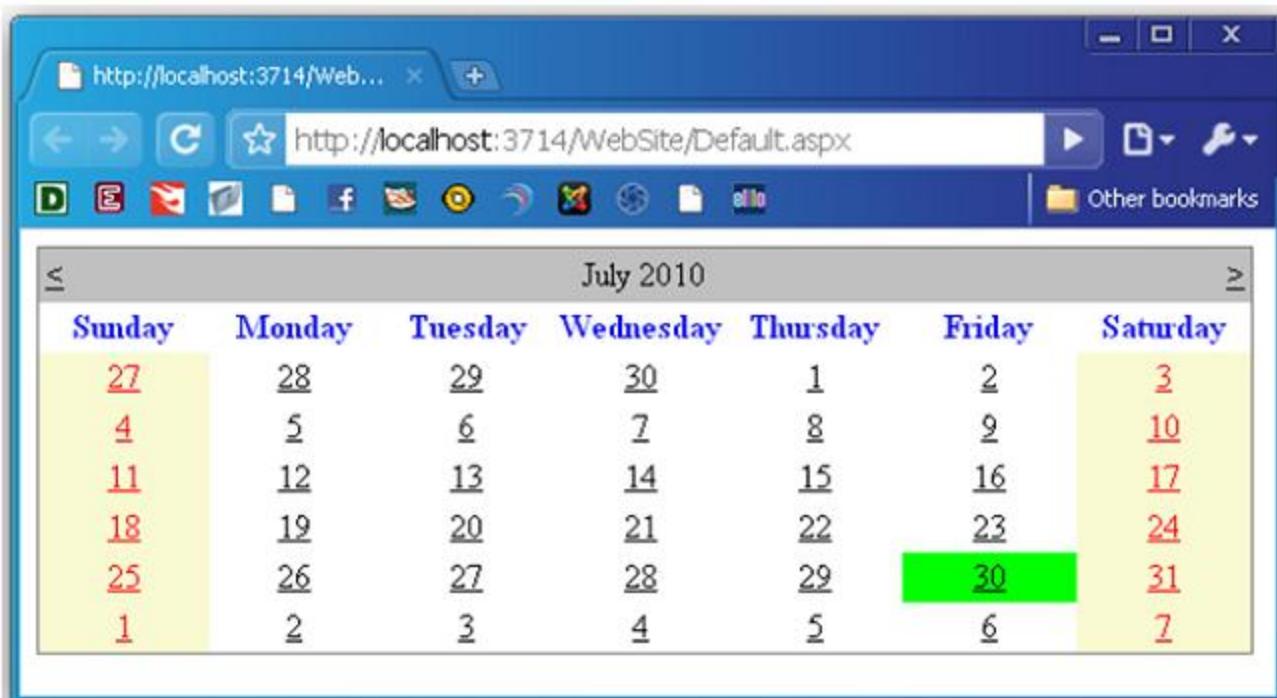
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
```

```

</head>
<body>
    <form id="Form1" runat="server">
        <asp:Calendar ID="Calendar1" DayNameFormat="Full" runat="server">
            <WeekendDayStyle BackColor="#fafad2" ForeColor="#ff0000" />
            <DayHeaderStyle ForeColor="#0000ff" />
            <TodayDayStyle BackColor="#00ff00" />
        </asp:Calendar>
    </form>
</body>
</html>

```

Trang aspx trên hiển thị một lịch, những ngày trong tuần được ghi đầy đủ bằng màu xanh, cuối tuần được ghi bằng màu đỏ trên nền vàng, ngày hiện tại được hiển thị với màu nền là xanh lục như hình sau đây:



Cho file aspx như sau:

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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></title>
</head>
<body>
    <form id="Form1" runat="server">
        <asp:Calendar DayNameFormat="Full" runat="server" />
    </form>

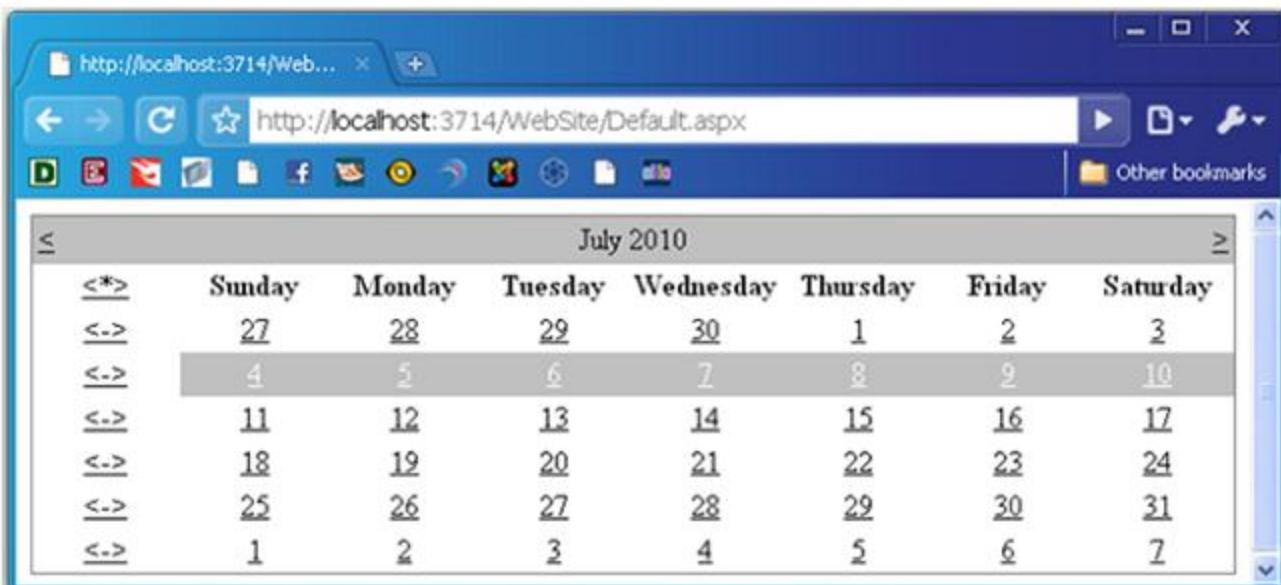
```

```

SelectionMode="DayWeekMonth"
SelectMonthText="<*>"
SelectWeekText="<->"/>
<SelectorStyle BackColor="#f5f5f5" />
</asp:Calendar>
</form>
</body>
</html>

```

Trang aspx bên trên hiển thị lịch với các ngày trong tuần được ghi đầy đủ, người dùng có thể chọn một ngày, một tuần và một tháng, ngày/tuần/tháng được chọn được hiển thị với màu nền là màu xám như hình dưới đây:



4. CalendarDay

CalendarDay control thể hiện một ngày trong một calendar control.

Các thuộc tính:

Thuộc tính	Mô tả
Date	Biến ngày của ngày
DayNumberText	Số ngày (thứ) của ngày
IsOtherMonth	Xác định ngày trong tháng khác có được hiển thị hay không
IsSelectable	Xác định có thể chọn được ngày
IsSelected	Xác định ngày có được chọn hay không
IsToday	Xác định có phải là ngày hiện tại hay không
IsWeekend	Xác định có phải là thứ bảy hay là chủ nhật hay không

5. CheckBox

CheckBox control được sử dụng để hiển thị một check box.

Các thuộc tính:

Thuộc tính	Mô tả
AutoPostBack	Xác định form có tự động post back về server sau khi thuộc tính Checked được thay đổi hay không. Mặc định là không
CausesValidation	Xác định trang có được kiểm chứng khi button control click hay không
Checked	Xác định check box được chọn hay không
InputAttributes	Tên và giá trị thuộc tính được sử dụng cho Input thẻ cho CheckBox control
LabelAttributed	Tên và giá trị thuộc tính được sử dụng cho Label thẻ cho CheckBox control
Runat	Xác định control là một server control. Phải được xác định là “server”
Text	Text bên cạnh CheckBox
 TextAlign	Text sẽ được xuất hiện bên nào check box(trái hay phải)
ValidationGroup	Nhóm những control cho Checkbox control gây ra sự kiểm chứng khi nó post back về server
OnCheckedChanged	Tên của hàm được thực thi khi thuộc tính Checked được thay đổi

Chúng ta có file aspx như sau:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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></title>
</head>
<body>
```

```

<form id="Form1" runat="server">
    <p>Home Phone:<br />
        <asp:TextBox id="home" runat="server" />
    <br />
    Work Phone:<br />
        <asp:TextBox id="work" runat="server" />
        <asp:CheckBox id="check1" Text="Same as home phone" TextAlign="Right"
            AutoPostBack="True" OnCheckedChanged="Check" runat="server" />
    </p>
</form>
</body>
</html>

```

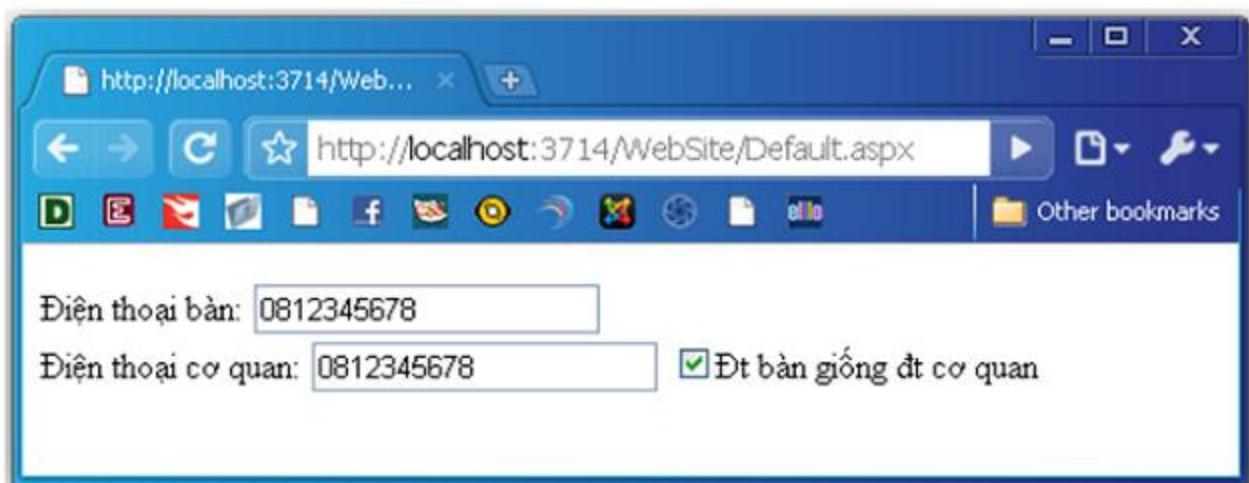
Code phía server như sau:

```

public partial class _Default: System.Web.UI.Page
{
    protected void Check(object sender, EventArgs e)
    {
        if (check1.Checked)
            work.Text = home.Text;
        else
            work.Text = "";
    }
}

```

Giao diện của chúng ta có hai TextBox control và một CheckBox control. Chúng ta tạo một hàm quản lý sự kiện cho sự kiện CheckedChanged để sao chép nội dung của textbox điện thoại bàn vào textbox của điện thoại cơ quan khi checkbox được chọn.



6. CheckBoxList

CheckBoxList control được sử dụng để tạo một nhóm check box với nhiều lựa chọn. Mỗi item có thể được chọn trong CheckBoxList control được định nghĩa bằng thẻ ListItem.

Các thuộc tính:

Thuộc tính	Mô tả
CellPadding	Khoảng cách giữa viền và nội dung
CellSpacing	Khoảng cách giữa các cell
RepeatColumns	Số column sử dụng khi hiển thị nhóm check box
RepeatDirection	Xác định nhóm check box sẽ được lặp lại theo chiều ngang hay chiều dọc
RepeatLayout	Layout của check box group
Runat	Xác định control là một server control. Phải được xác định là “server”
 TextAlign	Xác định text xuất hiện bên nào của check box

Chúng ta có file aspx như sau:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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></title> </head>
<body>

<form id="Form1" runat="server">

<asp:CheckBoxList id="check1" AutoPostBack="True" TextAlign="Right"
OnSelectedIndexChanged="Check" runat="server">

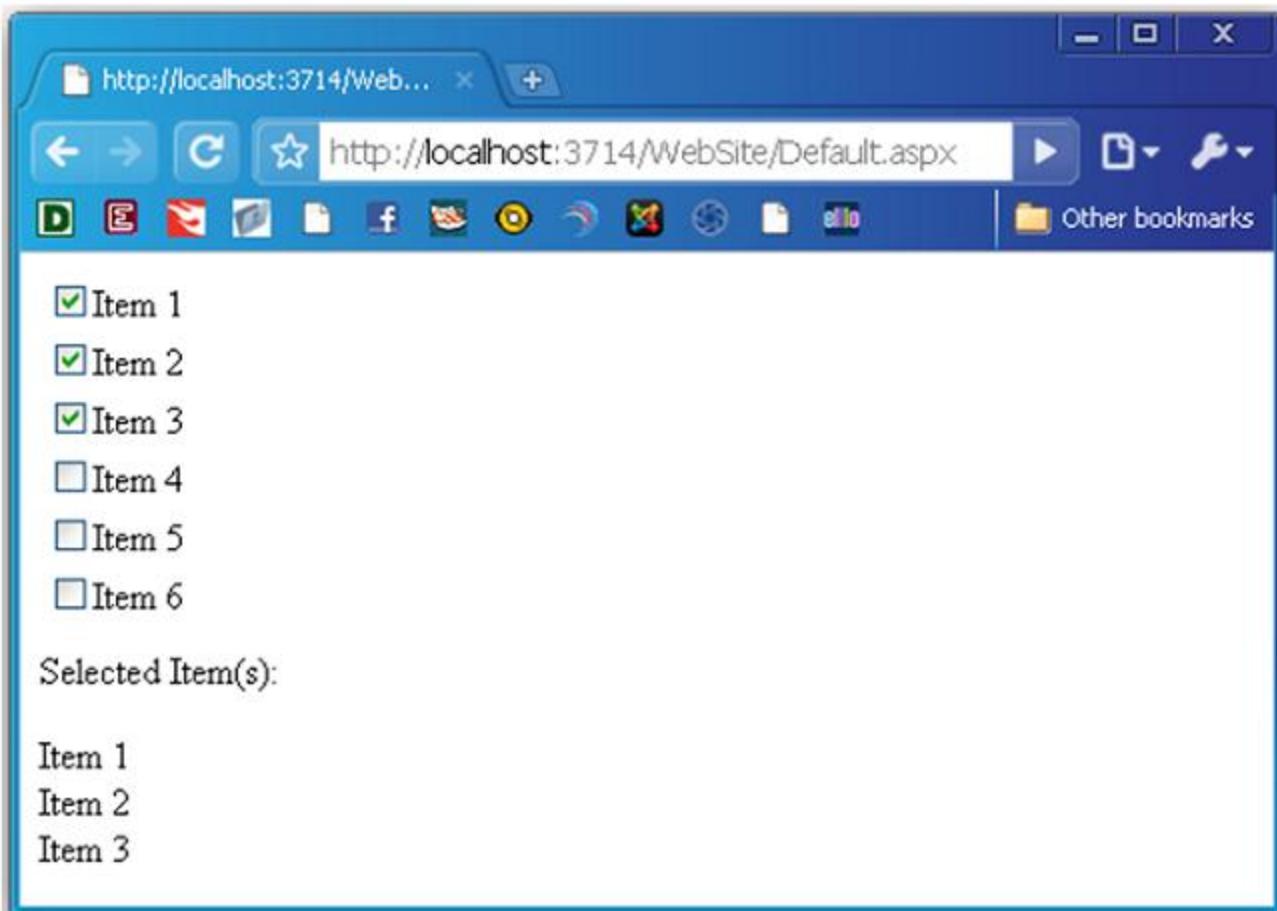
<asp:ListItem>Item 1</asp:ListItem>
<asp:ListItem>Item 2</asp:ListItem>
<asp:ListItem>Item 3</asp:ListItem>
<asp:ListItem>Item 4</asp:ListItem>
<asp:ListItem>Item 5</asp:ListItem>
<asp:ListItem>Item 6</asp:ListItem>
</asp:CheckBoxList> <br />
```

```
<asp:label id="mess" runat="server"/>  
</form>  
</body> </html>
```

Code phía server như sau:

```
public partial class _Default: System.Web.UI.Page  
{  
    protected void Check(object sender, EventArgs e)  
    {  
        int i;  
        mess.Text = "<p>Selected Item(s):</p>";  
        for (i = 0; i <= check1.Items.Count - 1; i++)  
            if (check1.Items[i].Selected)  
                mess.Text += check1.Items[i].Text + "<br />";  
    }  
}
```

Giao diện của chúng ta có một CheckBoxList control. Chúng ta tạo một hàm quản lý sự kiện cho sự kiện SelectedIndexChanged. Danh sách gồm sáu checkbox. Khi chúng ta chọn một trong những checkbox trang sẽ tự động post back về server, và hàm xử lý sự kiện Check được thực thi. Vòng lặp sẽ duyệt qua tất cả các item và hiển thị trong trong label control như hình dưới đây.



7. DropDownList

DropDownList control được sử dụng để tạo một drop-down list.

Mỗi item trong DropDownList control được định nghĩa bởi một thẻ ListItem.

Các thuộc tính:

Thuộc tính	Mô tả
SelectedIndex	Index của item được chọn
OnSelectedIndexChanged	Tên của hàm được thực thi khi index của item được chọn bị thay đổi
runat	Xác định control là một server control. Phải được xác định là “server”

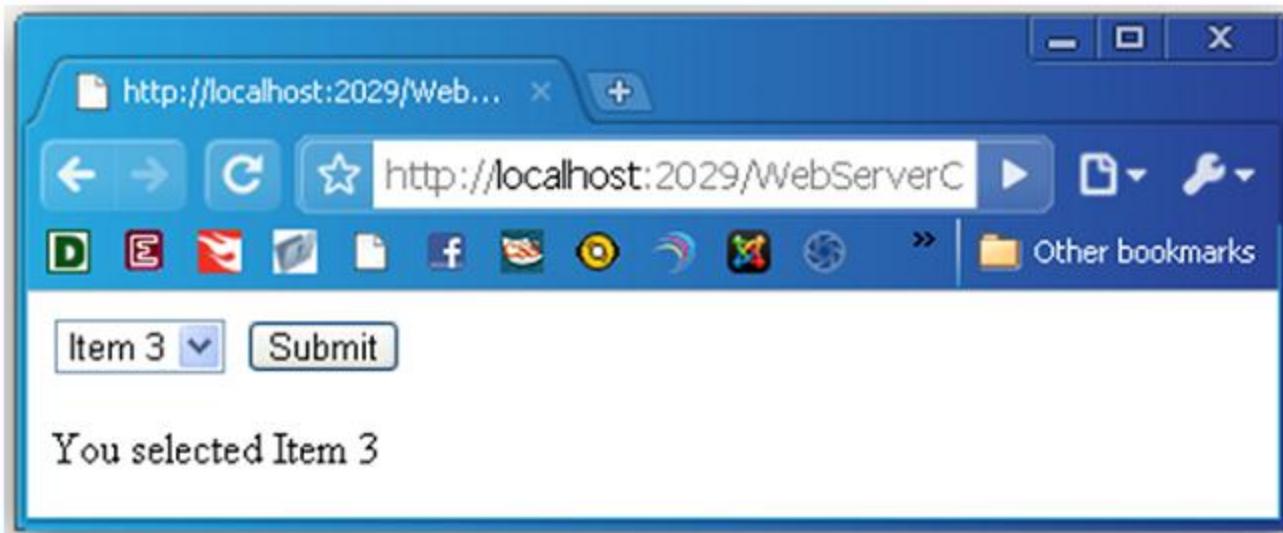
Chúng ta có file aspx như sau:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="DropDownList.aspx.cs" Inherits="DropDownList" %>
<!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></title>
</head>
<body>
    <form id="Form1" runat="server">
        <asp:DropDownList id="drop1" runat="server">
            <asp:ListItem>Item 1</asp:ListItem>
            <asp:ListItem>Item 2</asp:ListItem>
            <asp:ListItem>Item 3</asp:ListItem>
            <asp:ListItem>Item 4</asp:ListItem>
            <asp:ListItem>Item 5</asp:ListItem>
            <asp:ListItem>Item 6</asp:ListItem>
        </asp:DropDownList>
        <asp:Button ID="Button1" Text="Submit" OnClick="submit" runat="server"/>
        <p><asp:label id="mess" runat="server"/></p>
    </form>
</body>
</html>
```

Code phía server như sau:

```
public partial class DropDownList: System.Web.UI.Page
{
    protected void submit(object sender, EventArgs e)
    {
        mess.Text = "You selected " + drop1.SelectedItem.Text;
    }
}
```

Giao diện của chúng ta có một DropDownList control, một button control. Chúng ta tạo một hàm quản lý sự kiện cho sự kiện Click trên button submit, để hiển thị item được chọn trong DropDownList.



8. HyperLink

HyperLink control được sử dụng để tạo một hyperlink.

Các thuộc tính:

Thuộc tính	Mô tả
ImageUrl	Địa chỉ URL để hiện thị cho link
NavigateUrl	Địa chỉ URL của link
Runat	Xác định control là một server control. Phải được xác định là “server”
Target	Xác định cửa sổ sẽ được mở ra
Text	Text để hiển thị cho link

Chúng ta có file aspx như sau:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="HyperLink.aspx.cs"
Inherits="HyperLink" %>
<!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></title> </head>
<body>
<form id="Form1" runat="server">
<asp:HyperLink ID="HyperLink1" ImageUrl="Smile.jpg"
NavigateUrl="http://www.w3schools.com"
Text="Visit W3Schools!" Target="_blank" runat="server" Width="80"/>
</form>
</body>
</html>
```

Trang web của chúng ta có một hình là một hyperlink.



9. Image

Image control là control được sử dụng để hiển thị hình ảnh.

Các thuộc tính:

Thuộc tính	Mô tả
AlternateText	Text thay thế cho hình
DescriptionUrl	Nội dung mô tả chi tiết cho hình
GenerateEmptyAlternateText	Xác định control có tạo ra một chuỗi rỗng cho alternate text hay không
ImageAlign	Xác định cách sắp xếp của hình
ImageUrl	Url của hình để hiển thị
runat	Xác định control là một server control. Phải được xác định là “server”

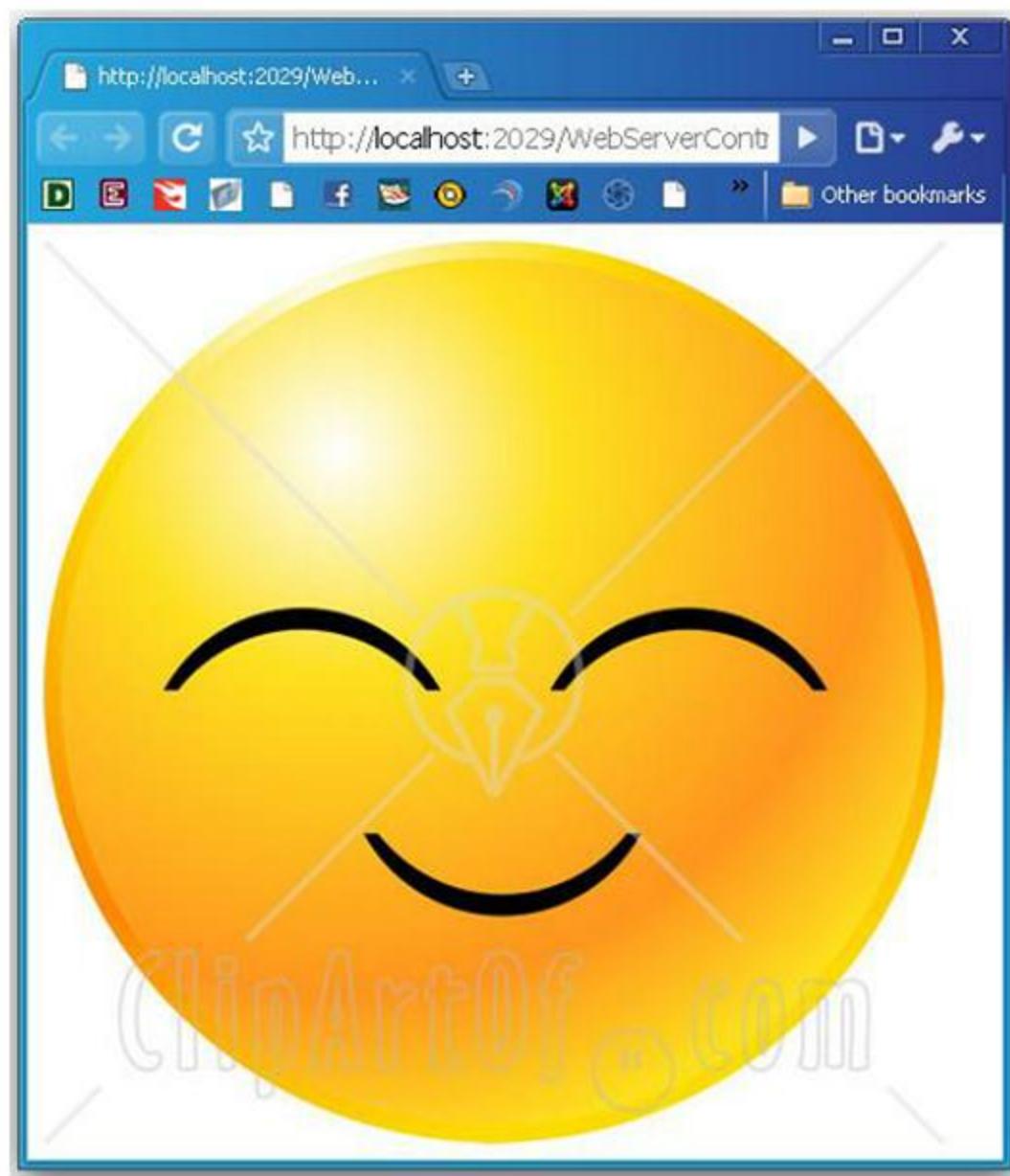
Chúng ta có file asxp hiển thị hình như sau:

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

<!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></title>
</head>
<body>
    <form id="Form1" runat="server">
        <asp:Image ID="Image1" runat="server" AlternateText="W3Schools"
        ImageUrl="Smile.jpg"/>
    </form>
</body>
</html>
```



10. ImageButton

ImageButton control được sử dụng để hiển thị một hình có thể click được

Các thuộc tính:

Thuộc tính	Mô tả
CauseValidation	Xác định trang được kiểm chứng ImageButton control được click
CommandArgument	Thông tin cho lệnh để thực hiện
CommandName	Lệnh được liên kết với sự kiện lệnh
GenerateEmptyAlternateText	Xác định control có được tạo một chuỗi rỗng cho alternate text hay không
OnClientClick	Tên của hàm được thực thi khi hình được click
PostBackUrl	URL của trang để post tới từ trang hiện tại khi ImageButton control được click

Runat	Xác định control là một server control. Phải được xác định là “server”
TagKey	
ValidationGroup	Nhóm những control mà ImageButton control gây nên sự kiểm chứng khi nó post back về server

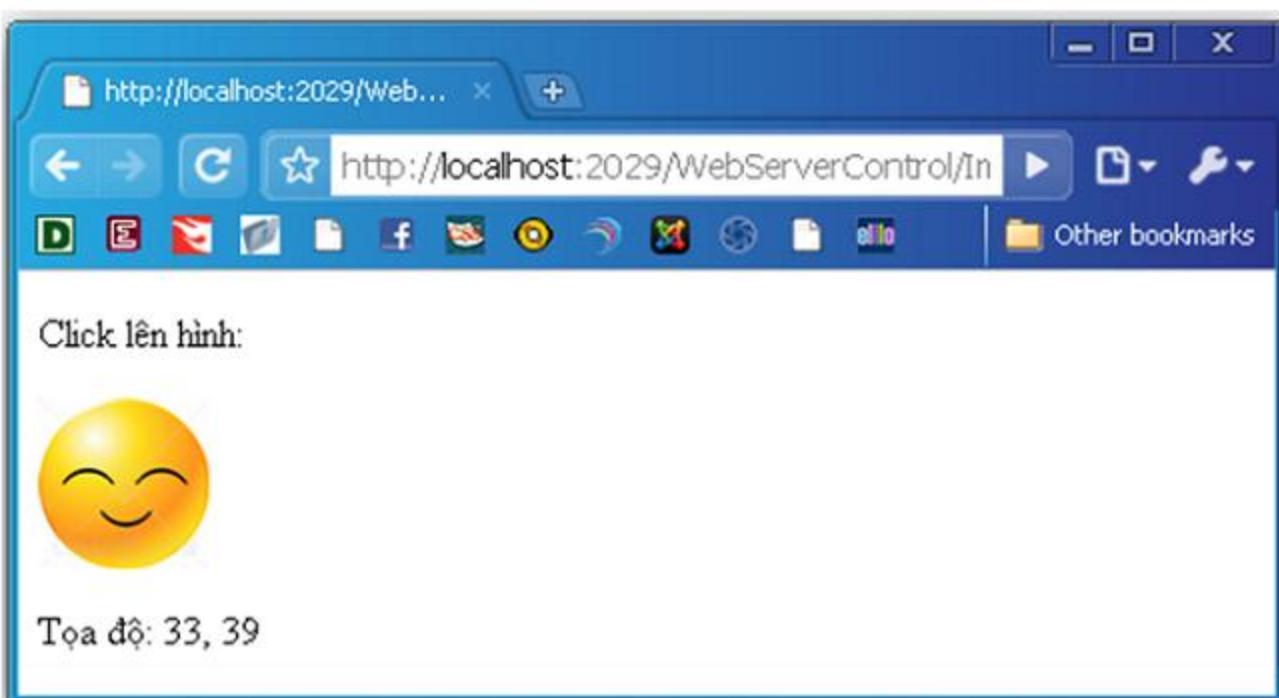
Chúng ta có file aspx như sau:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="ImageButton.aspx.cs" Inherits="ImageButton" %>
<!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></title> </head>
<body>
<form id="form1" runat="server">
<p>Click lên hình:</p>
<asp:ImageButton ID="ImageButton1" runat="server" ImageUrl="Smile.jpg"
OnClick="getCoordinates" Width="70" Height="70"/>
<p><asp:label id="mess" runat="server"/></p>
</form>
</body>
</html>
```

Code phía server như sau:

```
public partial class ImageButton: System.Web.UI.Page
{
    protected void getCoordinates(object sender, ImageClickEventArgs e)
    {
        mess.Text = "Tọa độ: " + e.X + ", " + e.Y; 
    }
}
```

Giao diện chúng ta có một ImageButton control và một label control. Khi người dùng click vào image thì thủ tục con getCoordinates được thực hiện. Thủ tục này trả về thông điệp vị trí của click và thông điệp này được hiển thị lên label control.



11. Label

Label control được sử dụng để hiển thị text trên một trang.

Các thuộc tính:

Thuộc tính	Mô tả
Runat	Xác định control là một server control. Phải được xác định là “server”
Text	Text được hiển thị trên label

Có file aspx như sau:

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

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

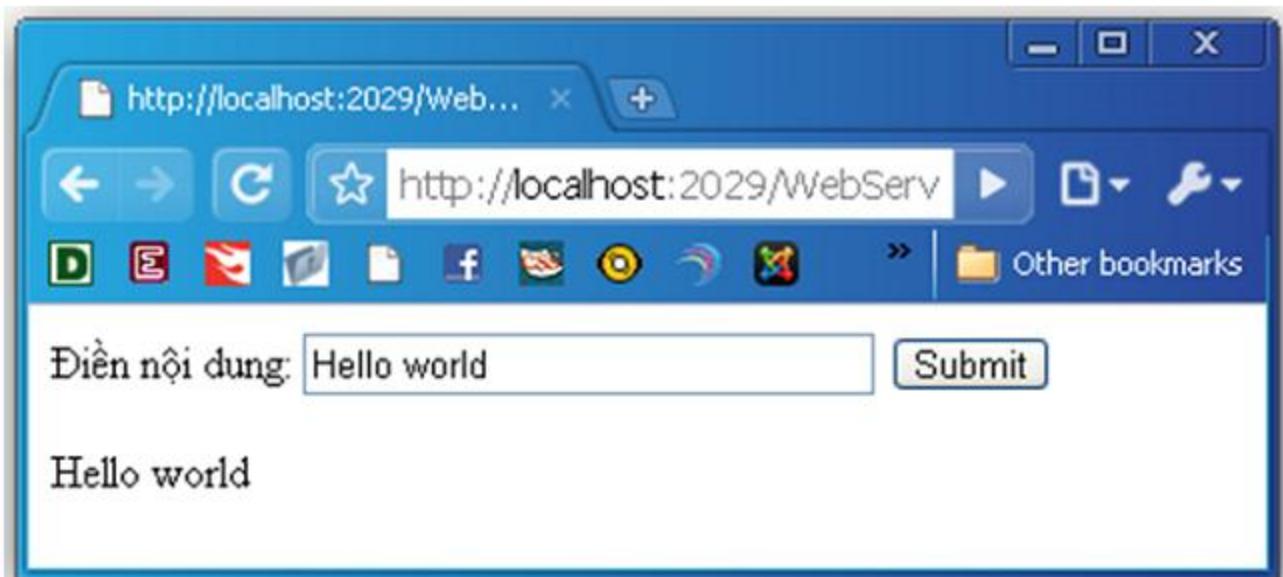
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server"> <title></title> </head>
<body>
<form id="Form1" runat="server">
Điền nội dung:
<asp:TextBox id="txt1" Width="200" runat="server" />
<asp:Button id="b1" Text="Submit" OnClick="submit" runat="server" />
<p><asp:Label id="label1" runat="server" /></p>
</form>
</body>
</html>
```

Code phía server như sau:

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

```
{  
    protected void submit(object sender, EventArgs e) {  
        label1.Text = txt1.Text;  
    }  
}
```

Giao diện của chúng ta có một label control, một textbox control, và một button control. Khi người dùng click vào button, thủ tục submit được thực hiện. Thủ tục này copy nội dung của textbox control gán vào cho label control.



12. LinkButton

LinkButton control được sử dụng để tạo một hyperlink button.

Các thuộc tính:

Thuộc tính	Mô tả
CauseValidation	Xác định trang được kiểm chứng LinkButton control được click
CommandArgument	Thông tin cho lệnh để thực hiện
CommandName	Lệnh được liên kết với sự kiện lệnh
OnClientClick	Tên của hàm được thực thi khi hình được click
PostBackUrl	URL của trang để post tới từ trang hiện tại khi LinkButton control được click
Runat	Xác định control là một server control. Phải được xác định là "server"

Text	Text trên LinkButton
ValidationGroup	Nhóm những control mà LinkButton control gây nên sự kiểm chứng khi nó post back về server

Chúng ta có file aspx như sau:

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

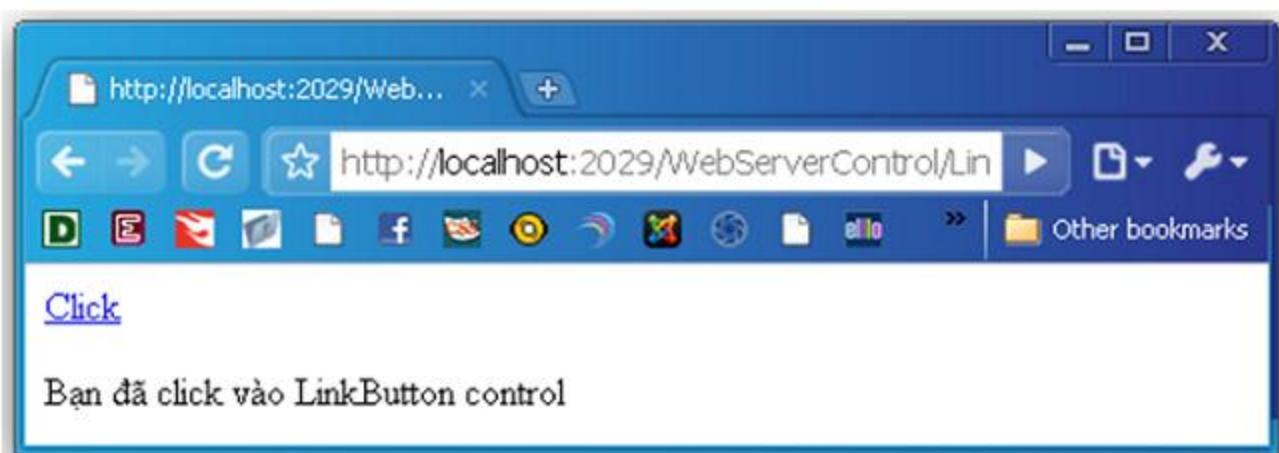
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">    <title></title> </head>
<body>
    <form id="Form1" runat="server">
        <asp:LinkButton ID="LinkButton1" Text="Click" OnClick="lblClick"
runat="server" />
        <p><asp:Label id="Label1" runat="server" /></p>
    </form>
</body>
</html>
```

Code phía server như sau:

```
public partial class LinkButton: System.Web.UI.Page
{
    protected void lblClick(object sender, EventArgs e)
    {
        Label1.Text = "Bạn đã click vào LinkButton control";
    }
}
```

Giao diện của chúng ta gồm có một LinkButton control, một label control. Khi người dùng click và link thì thủ tục lblClick được thực thi. Thủ tục gán thông điệp “Bạn đã click và LinkButton control” vào Label control.



13. ListBox

ListBox control được sử dụng để tạo một drop-down list một hoặc nhiều lựa chọn.

Các thuộc tính:

Thuộc tính	Mô tả
Rows	Số dòng được hiển thị trong list
SelectionMode	Cho phép chọn một hay nhiều item

Chúng ta có file aspx như sau:

```
<%@ 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></title>
</head>
<body>
    <form id="Form1" runat="server">
        <asp:ListBox id="drop1" rows="3" runat="server">
            <asp:ListItem selected="true">Item 1</asp:ListItem>
            <asp:ListItem>Item 2</asp:ListItem>
            <asp:ListItem>Item 3</asp:ListItem>
            <asp:ListItem>Item 4</asp:ListItem>
            <asp:ListItem>Item 5</asp:ListItem>
            <asp:ListItem>Item 6</asp:ListItem>
        </asp:ListBox>
    </form>
</body>
</html>
```

```

<asp:Button ID="Button1" Text="Submit" OnClick="submit" runat="server" />
<p><asp:label id="mess" runat="server" /></p>
</form>
</body>
</html>

```

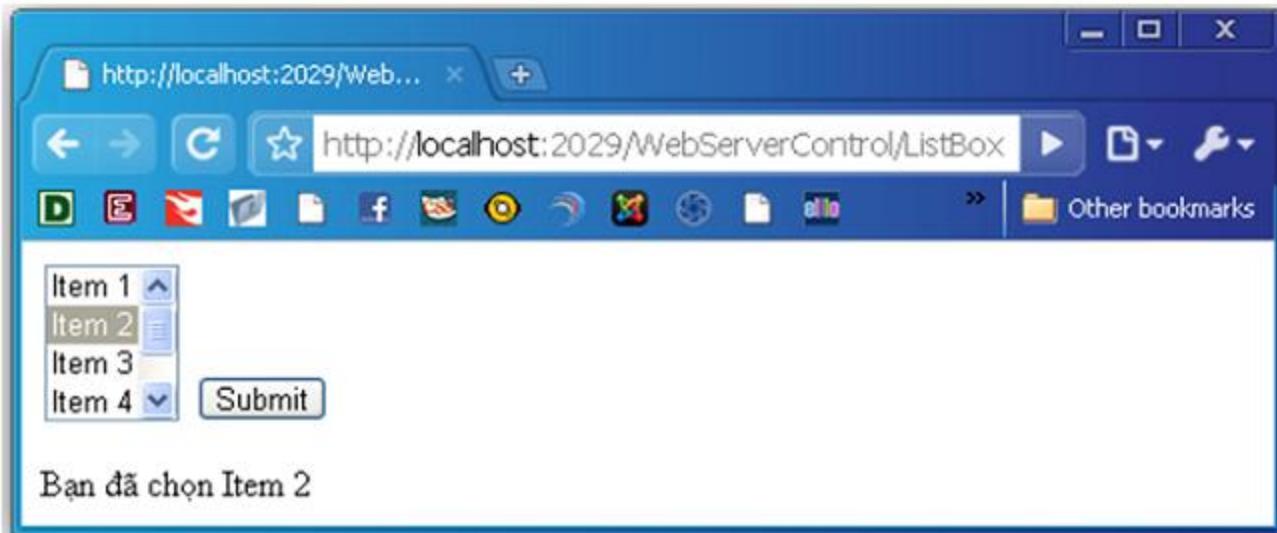
Code phía server như sau:

```

public partial class ListBox: System.Web.UI.Page
{
    protected void submit(object sender, EventArgs e)
    {
        mess.Text = "Bạn đã chọn " + drop1.SelectedItem.Text;
    }
}

```

Giao diện của chúng ta gồm có một ListBox control và một Button control. Tạo một hàm quản lý sự kiện cho sự kiện Click để hiển thị item được chọn ra Label control.



14. ListItem

ListItem control tạo một item trong một list. Control này được dùng cùng với những control như là **<asp:ListBox>**, **<asp:RadioButtonList>**, và **<asp:BulletedList>**

Cấu trúc:

```

<asp:ListItem Enabled="True|False"
Selected="True|False" Text="label" Value="value"/>

```

Các thuộc tính:

Thuộc tính	Mô tả
Attributes	Tùy chọn. Tập hợp của các cặp tên và giá trị thuộc tính tương ứng cho ListItem,
Enable	Tùy chọn. Xác định item enable hay disable
Selected	Tùy chọn. Xác định item có được chọn hay không.
Text	Tùy chọn. Text được hiển thị trong ListItem
Value	Tùy chọn. Giá trị của ListItem.

Chúng ta có thể xem thí dụ ở mục 13 (ListBox)

15. Literal

Literal control được sử dụng để hiển thị text trên một trang.

Các thuộc tính:

Thuộc tính	Mô tả
Mode	
Runat	Xác định control là một server control. Phải được xác định là “server”
Text	Text để hiển thị

Chúng ta có trang aspx như sau:

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

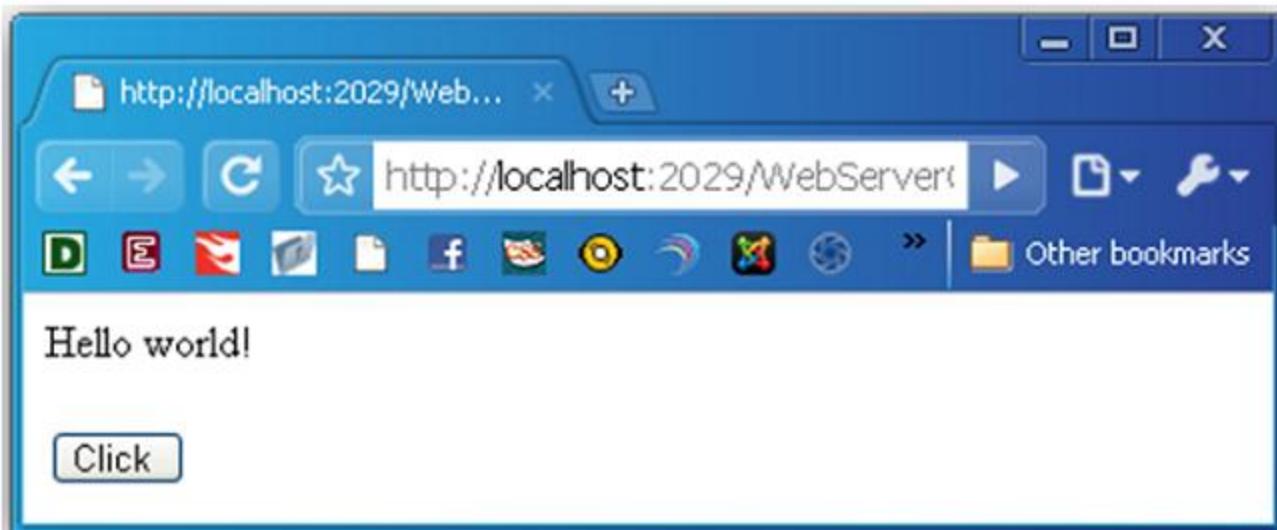
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="Form1" runat="server">
        <asp:Literal id="Literal1" Text="Hello !" runat="server" />
        <br /><br />
        <asp:Button ID="Button1" Text="Click " OnClick="submit" runat="server" />
    </form>
</body>
</html>
```

Code phía server như sau:

```
public partial class Literal: System.Web.UI.Page
{
    protected void submit(object sender, EventArgs e)
    {
        Literal1.Text = "Hello world!";
    }
}
```

Giao diện của trang web chúng ta gồm có một Literal control và một Button control. Khi người dùng click vào button, thì thủ tục submit được thực hiện, thủ tục này thay đổi text của Literal control.



16. Panel

Panel control được sử dụng như là một vật chứa cho các control khác.

Các thuộc tính:

Thuộc tính	Mô tả
BackImageUrl	Xác định URL để file hình được hiển thị như là background cho control này
DefaultButton	Xác định ID của button mặc định trong panel
Direction	Xác định hướng hiển thị nội dung của panel
GroupText	Xác định caption cho nhóm những control trong panel
HorizontalAlign	Xác định việc sắp xếp nội dung theo chiều ngang
Runat	Xác định control là một server control. Phải được xác định là “server”
ScrollBars	Xác định và hiển thị của scroll bar trong panel
Wrap	Xác định nội dung có được bao lại hay không

Chúng ta có file aspx như sau:

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

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

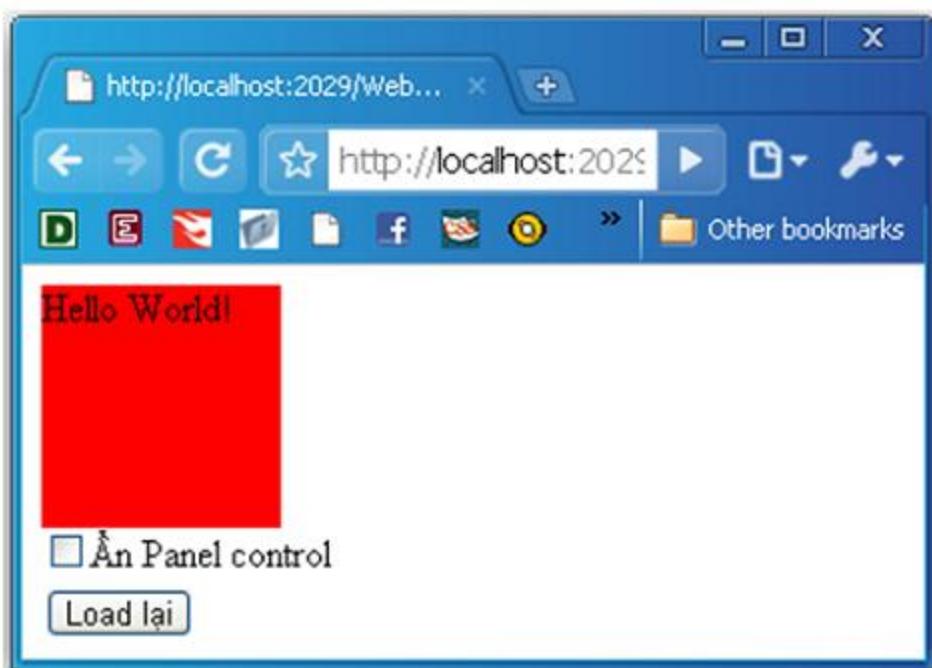
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">  <title></title> </head>
<body>

    <form id="form1" runat="server">
        <asp:Panel id="panel1" runat="server" BackColor="#ff0000" Height="100px"
Width="100px">
            Hello World!
        </asp:Panel>
        <asp:CheckBox id="check1" Text="Ẩn Panel control" runat="server"/> <br
/><br />
        <asp:Button ID="Button1" Text="Load lại" runat="server" />
    </form>
</body>
</html>
```

Code phía server như sau:

```
public partial class Panel : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (check1.Checked)
            panel1.Visible = false;
        else
            panel1.Visible = true;
    }
}
```

Giao diện của chúng ta gồm có một Panel control, một CheckBox control và một Button control. Khi người dùng chọn và CheckBox control và click vào button Load lại thì Panel control sẽ ẩn đi.



17. PlaceHolder

PlaceHolder control được sử dụng để dành khoảng trống cho những control khác để thêm vào bằng code.

Các thuộc tính:

Thuộc tính	Mô tả
Runat	Xác định control là một server control. Phải được xác định là “server”

18. RadioButton

RadioButton control được sử dụng để hiển thị radio button.

Các thuộc tính:

Thuộc tính	Mô tả
AutoPostBack	Giá trị boolean xác định form các được tự động post back về server sau khi thuộc tính Checked bị thay đổi hay không. Mặc định là false
Checked	Giá trị boolean xác định radio button có được chọn hay không
Id	Giá trị duy nhất cho control
GroupName	Tên của group mà radio button thuộc về
OnCheckedChanged	Tên của hàm được thực thi khi thuộc tính Checked bị thay đổi
Runat	Xác định control là một server control. Phải được xác định là “server”
Text	Text bên cạnh radio button
 TextAlign	Xác định text xuất hiện bên nào của radio button

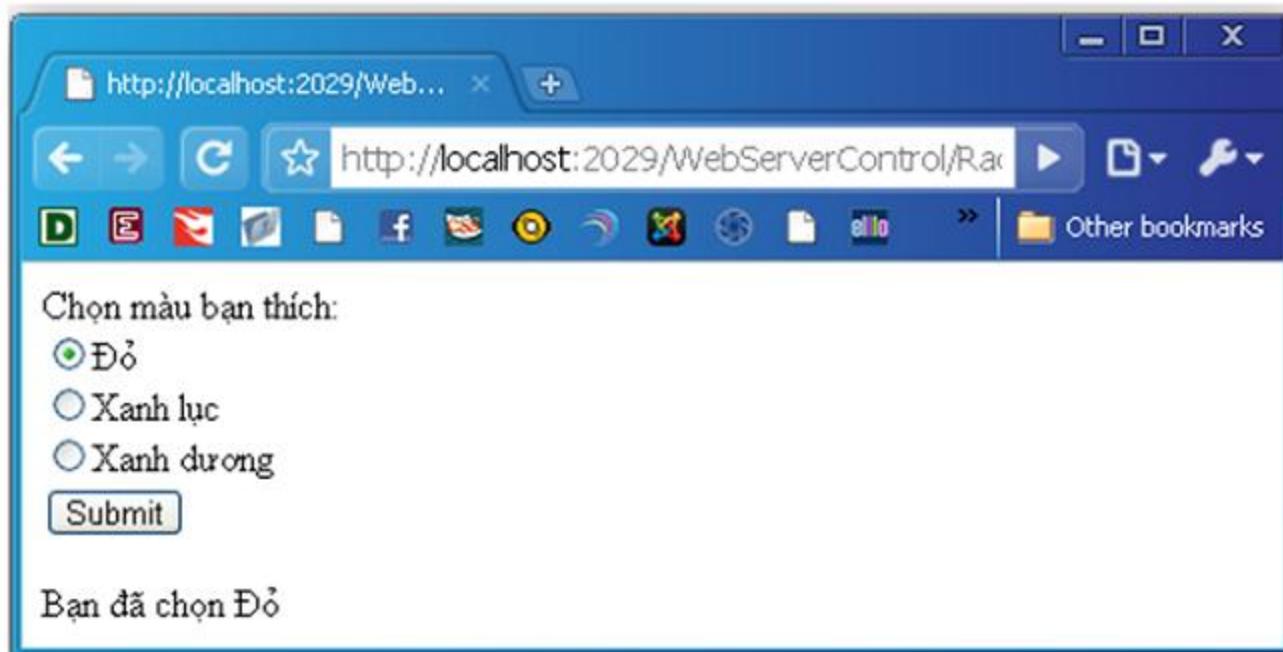
Chúng ta có file aspx như sau:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="RadioButton.aspx.cs"
Inherits="RadioButton" %>
<!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></title>  </head>
<body>
    <form id="form1" runat="server">
        Chọn màu bạn thích:
        <br />
        <asp:RadioButton id="red" Text="Đỏ" Checked="True"
        GroupName="colors" runat="server"/>
        <br />
        <asp:RadioButton id="green" Text="Xanh lục"
        GroupName="colors" runat="server"/>
        <br />
        <asp:RadioButton id="blue" Text="Xanh dương"
        GroupName="colors" runat="server"/>
        <br />
        <asp:Button ID="Button1" text="Submit" OnClick="submit" runat="server"/>
        <p><asp:Label id="Label1" runat="server"/></p>
    </form>
</body>
</html>
```

Code phía server như sau:

```
public partial class RadioButton : System.Web.UI.Page
{
    protected void submit(object sender, EventArgs e)
    {
        if (red.Checked)
            Label1.Text = "Bạn đã chọn " + red.Text;
        else if (green.Checked)
            Label1.Text = "Bạn đã chọn " + green.Text;
        else if (blue.Checked)
            Label1.Text = "Bạn đã chọn " + blue.Text;
    }
}
```

Giao diện của chúng ta có ba RadioButton control, một Button control, và một Label control. Khi người dùng click vào button submit thì thủ tục submit được thực thi, thủ tục sẽ kiểm tra xem radio button nào được chọn và gán giá trị: “Bạn đã chọn: ” cùng với thuộc tính Text của RadioButton được chọn.



19. RadioButtonList

RadioButtonList control được sử dụng để tạo một nhóm những radio button.

Mỗi item trong RadioButtonList control được định nghĩa bởi một thẻ ListItem.

Các thuộc tính:

Thuộc tính	Mô tả
CellPadding	Khoảng cách giữa viền và nội dung bên trong của một cell
CellSpacing	Khoảng cách giữa các cell
RepeatColumns	Số cột sử dụng để hiển thị nhóm radio button
RepeatDirection	Xác định hướng nhóm radio button hiển thị theo chiều dọc hay theo chiều ngang
RepeatLayout	Layout của nhóm các radio button
Runat	Xác định control là một server control. Phải được xác định là “server”
 TextAlign	Xác định text sẽ xuất hiện bên phía nào của radio button

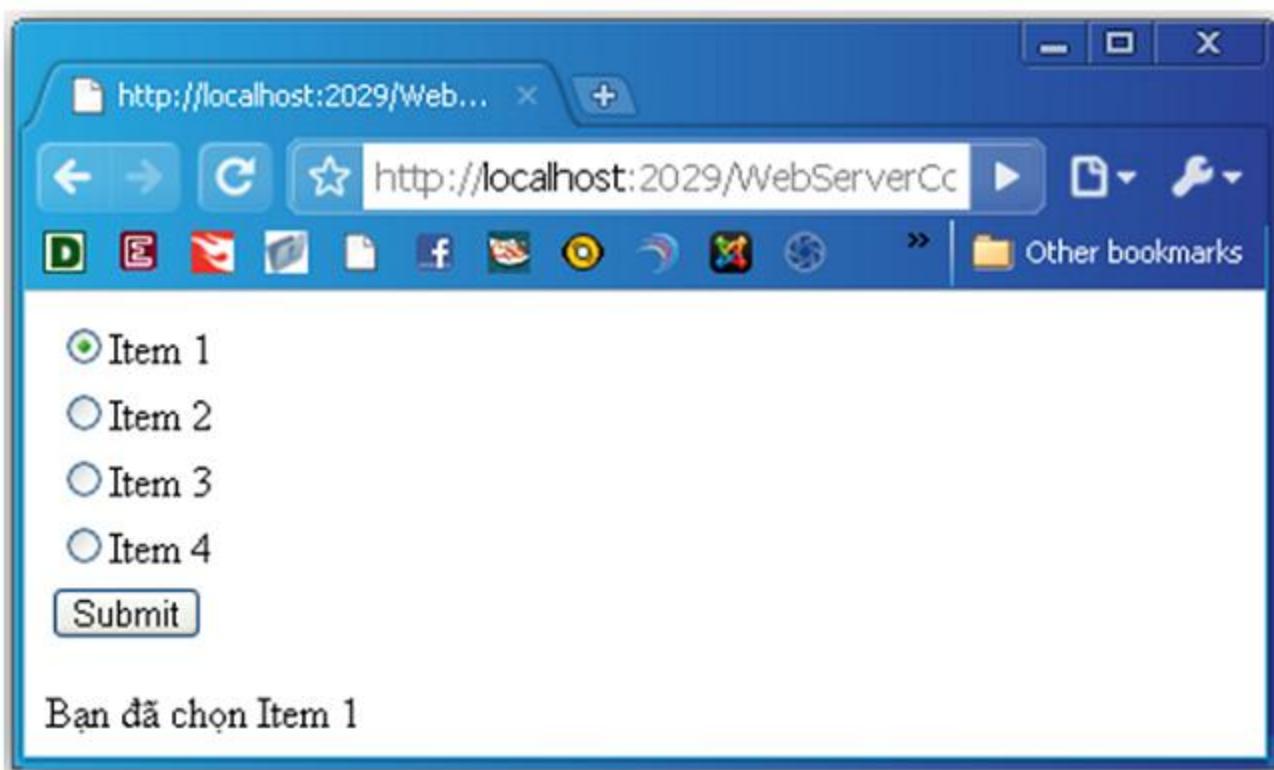
Chúng ta có file aspx như sau:

```
<%@ 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></title></head>
<form id="Form1" runat="server">
    <asp:RadioButtonList id="radiolist1" runat="server">
        <asp:ListItem selected="true">Item 1</asp:ListItem>
        <asp:ListItem>Item 2</asp:ListItem>
        <asp:ListItem>Item 3</asp:ListItem>
        <asp:ListItem>Item 4</asp:ListItem>
    </asp:RadioButtonList>
    <br />
    <asp:Button text="Submit" OnClick="submit" runat="server"/>
    <p><asp:Label id="Label1" runat="server"/></p>
</form>
</html>
```

Code phía server như sau:

```
public partial class RadioButtonList: System.Web.UI.Page
{
    protected void submit(object sender, EventArgs e)
    {
        Label1.Text = "Bạn đã chọn " + radiolist1.SelectedItem.Text;
    }
}
```

Giao diện của chúng ta gồm có một RadioButtonList control và một Button control và một Label control. Chúng ta tạo hàm quản lý sự kiện cho sự kiện Click để hiển thị text và item được chọn lên Label control.



20. BulletedList

BulletedList control tạo một danh sách dạng bullet.

Mỗi item trong BulletedList control được định nghĩa bởi một thẻ **ListItem**.

Các thuộc tính:

Thuộc tính	Mô tả
BulletImageUrl	Xác định URL của 1 custom bullet list image
BulletStyle	Xác định Style của Bullet list
DisplayMode	Xác định loại danh sách được hiển thị
FirstBulletNumber	Xác định số bắt đầu của danh sách các item trong danh sách đã được sắp xếp
runat	Xác định control là một server control. Phải được xác định là “server”
Target	Xác định nơi mở URL

21. Style

Style control được sử dụng để xác định style cho web control.

Các thuộc tính:

Thuộc tính	Mô tả
BackColor	Màu nền của control
BorderColor	Màu viền của control

BorderStyle	Style viền của control
BorderWidth	Độ dày viền của control
CssClass	Class Css được đưa ra bởi control trên client
Font	Thuộc tính Font của control
ForeColor	Màu của chữ
Height	Chiều cao của control
RegisteredCssClass	Class CSS được đăng ký với control
Width	Chiều rộng của control

22. Table

Table control được sử dụng với TableCell và TableRow control để tạo một table

Các thuộc tính:

Thuộc tính	Mô tả
BackImageUrl	URL của hình được sử dụng làm nền của table
Caption	Tên của bảng
CaptionAlign	Sắp xếp caption
CellPadding	Khoảng cách giữa nội dung và viền của cell
CellSpacing	Khoảng cách giữa các cell
GridLines	Định dạng gridline trong table
HorizontalAlign	Sắp xếp bảng theo chiều ngang trên trang
Rows	Tập hợp tất cả các dòng trong table
runat	Xác định control là một server control. Phải được xác định là “server”

Chúng ta có file aspx tạo một table như sau:

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

<!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></title>
```

```

</head>
<body>
    <form id="Form1" runat="server">
        <asp:Table id="Table1" BorderWidth="1" GridLines="Both" runat="server" />
    </form>
</body>
</html>

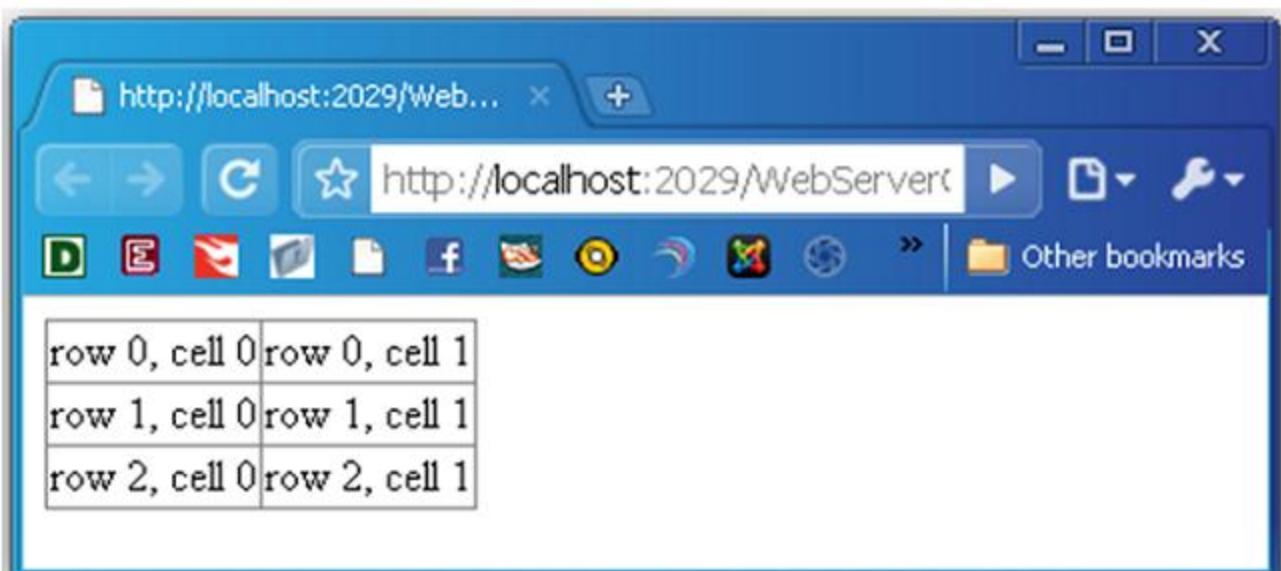
```

Code phía server nhu sau:

```

public partial class Table : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        int rows, cells, j, i;
        rows = 3;
        cells = 2;
        for (j = 0; j <= rows - 1; j++)
        {
            TableRow r = new TableRow();
            for (i = 0; i <= cells - 1; i++)
            {
                TableCell c = new TableCell();
                c.Controls.Add(new LiteralControl("row " + j + ", cell " + i));
                r.Cells.Add(c);
            }
            Table1.Rows.Add(r);
        }
    }
}

```



23. TableCell

TableCell được sử dụng cùng với Table control và TableRow control để tạo một cell trong một table.

Các thuộc tính

Thuộc tính	Mô tả
AssociatedHeaderCellID	Một danh sách các table header cell liên hệ với TableCell control
ColumnSpan	Số cột mà cell mở rộng
HorizontalAlign	Sắp xếp nội dung theo chiều ngang trong table cell
RowSpan	Số dòng mà cell mở rộng
Runat	Xác định control là một server control. Phải được xác định là “server”
Text	Xác định nội dung bên trong của cell
VerticalAlign	Sắp xếp nội dung theo chiều ngang trong table cell
Wrap	Xác định nội dung có được bao hay không

Chúng ta có thể xem thí dụ ở mục 22 (Table)

24. TableRow

TableRow control được sử dụng cùng với TableCell và Table control để tạo một dòng trong table.

Các thuộc tính:

Thuộc tính	Mô tả
Cells	
HorizontalAlign	Sắp xếp nội dung theo chiều ngang trong table row
TableSection	Định vị đối tượng TableRow trong table control
VerticalAlign	Sắp xếp nội dung theo chiều dọc trong table row

25. TextBox

TextBox control được sử dụng để tạo một text box, để người dùng có thể nhập liệu.

Các thuộc tính:

Thuộc tính	Mô tả
AutoCompleteType	Xác định hành vi AutoComplete của TextBox

AutoPostBack	Giá trị boolean xác định control này có tự động post back về server khi nội dung thay đổi hay không. Mặc định là không
CausesValidation	Trang sẽ được kiểm chứng khi Postback xảy ra
Columns	Chiều rộng của textbox
Maxlength	Số lượng ký tự tối đa của textbox
ReadOnly	Xác định nội dung trong textbox có được thay đổi hay không
Rows	Chiều cao của textbox (chỉ được sử dụng khi TextMode="Multiline")
Runat	Xác định control là một server control. Phải được xác định là "server"
TagKey	
Text	Nội dung của textbox
TextMode	Xác định loại TextBox control (SingleLine, MultiLine hay Password)
ValidationGroup	Xác định nhóm những control được kiểm chứng khi Postback xảy ra
Wrap	Giá trị boolean xác định nội dung của text box có được bao hay không
OnTextChanged	Tên hàm được thực thi khi text trong textbox bị thay đổi

Chúng ta có thể xem thí dụ ở mục 11 (Label) có sử dụng TextBox.

2.6. SỰ KIỆN PAGELOAD VÀ THUỘC TÍNH ISPOSTBACK

Phần trình bày sẽ dựa trên việc phân chia thành request lần đầu và request từ lần thứ hai trở đi.

Trường hợp request lần đầu

Nếu ASP.NET 3.5 nhận request HTTP từ Web browser, thì nó sẽ tạo ra form instance. Form instance sẽ tạo ra cây của control instance ở bên trong đó. Sau đó, cho khởi động Load event, và gọi ra event handler mà chúng ta sẽ tạo. Trong event handler, set property của control trong control cây để kiểm soát việc hiển thị màn hình. Cuối cùng, Form sẽ sinh ra HTML và trả về Web browser. Xử lý generate HTML được chuyển đến control, control sẽ sinh ra HTML phù hợp với property của chính

mình. Hơn nữa, có thể lưu giữ trạng thái của control vào trong ViewState khi cần và thẻ `<input type="hidden">` sẽ được tạo ra.

Trường hợp request từ lần thứ hai trở đi

Nội dung cho đến phần nhận request HTTP, tạo cây của form instance và control instance thì giống với request đầu tiên. Trong postback request bao gồm ViewState tạo trong response trước đó, vì thế sẽ set control property giống như khi trả về response lần trước dựa vào ViewState đó.

Sau đó, so sánh trạng thái giữa HTTP request và trạng thái control hiện tại (được phục hồi nhờ ViewState), thì chúng ta sẽ khởi động event thích hợp. Do form được load tương tự với trường hợp request đầu tiên, nên đầu tiên, load event nếu giá trị được input vào textbox khác với ViewState, và nếu TextChanged event, button được click, thì chúng ta cảm giác như là Clicked event.

Đương nhiên, với các event handler của load event mà cả request đầu tiên, cả postback (Page_Load() method) đều được gọi ra vô điều kiện thì sử dụng **IsPostBack** property biểu thị là có postback hay không, để thay thế nội dung xử lý. Thông thường Page_Load() method sẽ như sau:

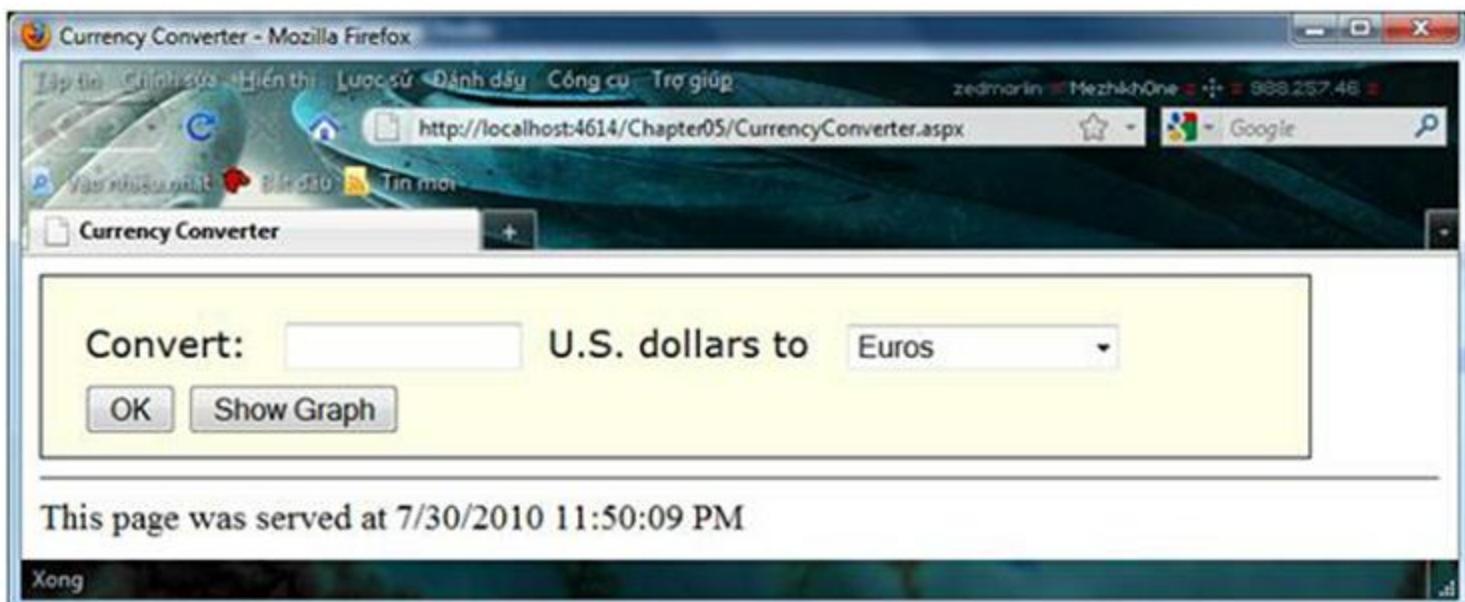
```
protected void Page_Load(object sender, EventArgs e)
{
    // Sử dụng IsPostBack property, phán đoán xem có postback hay không?
    if (!Page.IsPostBack) {
        // Ở đây sẽ mô tả xử lý khởi tạo.

        //
        // Ở đây sẽ mô tả xử lý chung trong tất cả event.

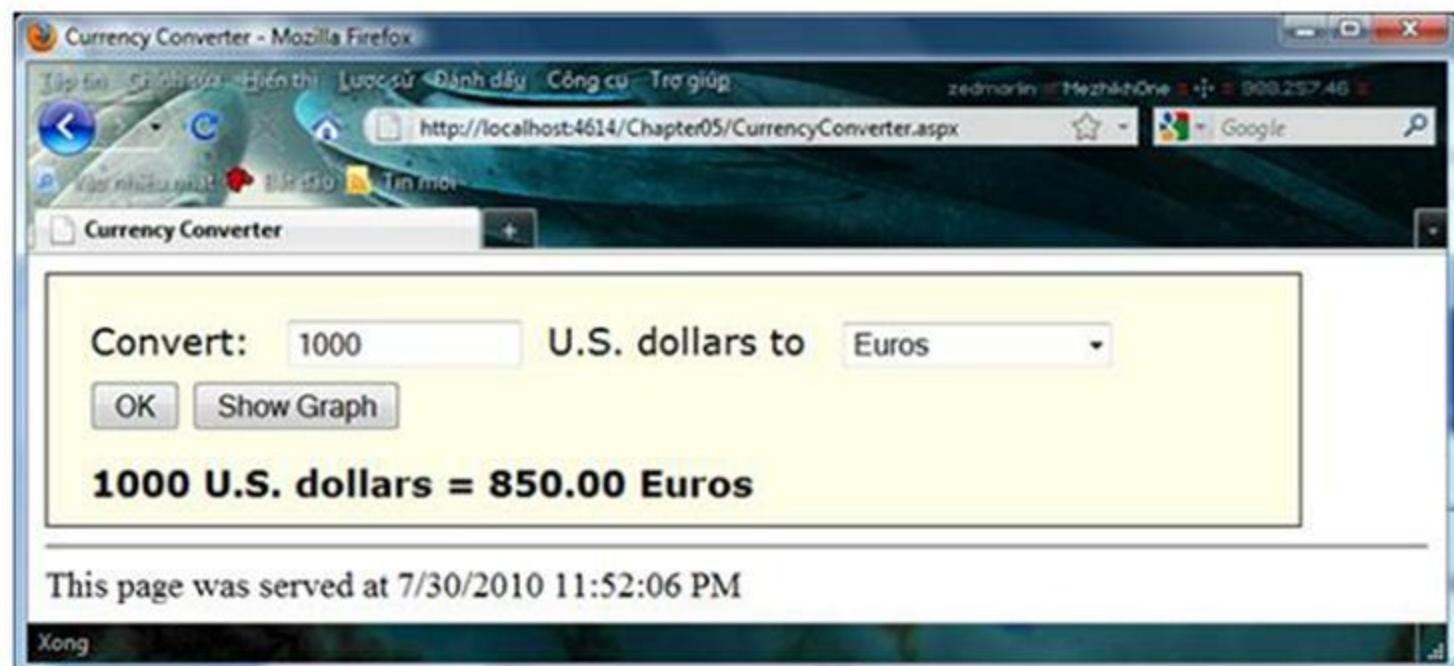
    }
}
```

Các request sau giống với request đầu tiên. Control sẽ tạo ra HTML và ViewState lưu giữ trạng thái sẽ được sinh ra.

Chúng ta xét thí dụ đổi tiền US sang các loại tiền khác, cụ thể như sau:



Khi bạn nhập số tiền trong khung Convert, chọn loại tiền trong ComboBox và nhấp nút OK. Kết quả như sau:



Nhấp nút Show Graph, kết quả như sau:



Trang aspx được thiết kế như sau:

```
<%@ Page Language="C#" AutoEventWireup="true"
    CodeFile="CurrencyConverter.aspx.cs" Inherits="CurrencyConverter" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>    <title>Currency Converter</title>    </head>
    <body>
        <form ID="Form1" method="post" runat="server">
            <div style="border-right: thin ridge; padding-right: 20px; border-top: thin ridge;
padding-left: 20px; padding-bottom: 20px; border-left: thin ridge; width: 531px;
padding-top: 20px; border-bottom: thin ridge; font-family: Verdana;
background-color: #FFFFE8">
                Convert: &ampnbsp
                <input type="text" ID="US" runat="server" style="width: 102px" />&ampnbsp U.S.
dollars to &ampnbsp
                <select ID="Currency" runat="server" />
                <br /><br />
                <input type="submit" value="OK" ID="Convert" runat="server"
OnServerClick="Convert_ServerClick" />
                <input type="submit" value="Show Graph" ID="ShowGraph" runat="server"
OnServerClick="ShowGraph_ServerClick" />    <br /><br />
                <img ID="Graph" alt="Currency Graph" scr="" runat="server" />
                <br /><br />
                <div style="font-weight: bold" ID="Result" runat="server"></div>
            </div>
        </form>
    </body>
</html>
```

Nhận thấy thẻ Select để tạo ComboBox trên trang để chọn loại tiền ban đầu là rỗng:

```
<select ID="Currency" runat="server" />
```

và sẽ được gán nội dung qua sự kiện **Load** của trang **CurrencyConverter.aspx** khi trang được thực thi. Việc gán này chỉ thực hiện một lần duy nhất khi thỏa mãn điều kiện: `this.IsPostBack == false` tương ứng việc request trang lần đầu tiên.

Phần mã lệnh thực thi phía server:

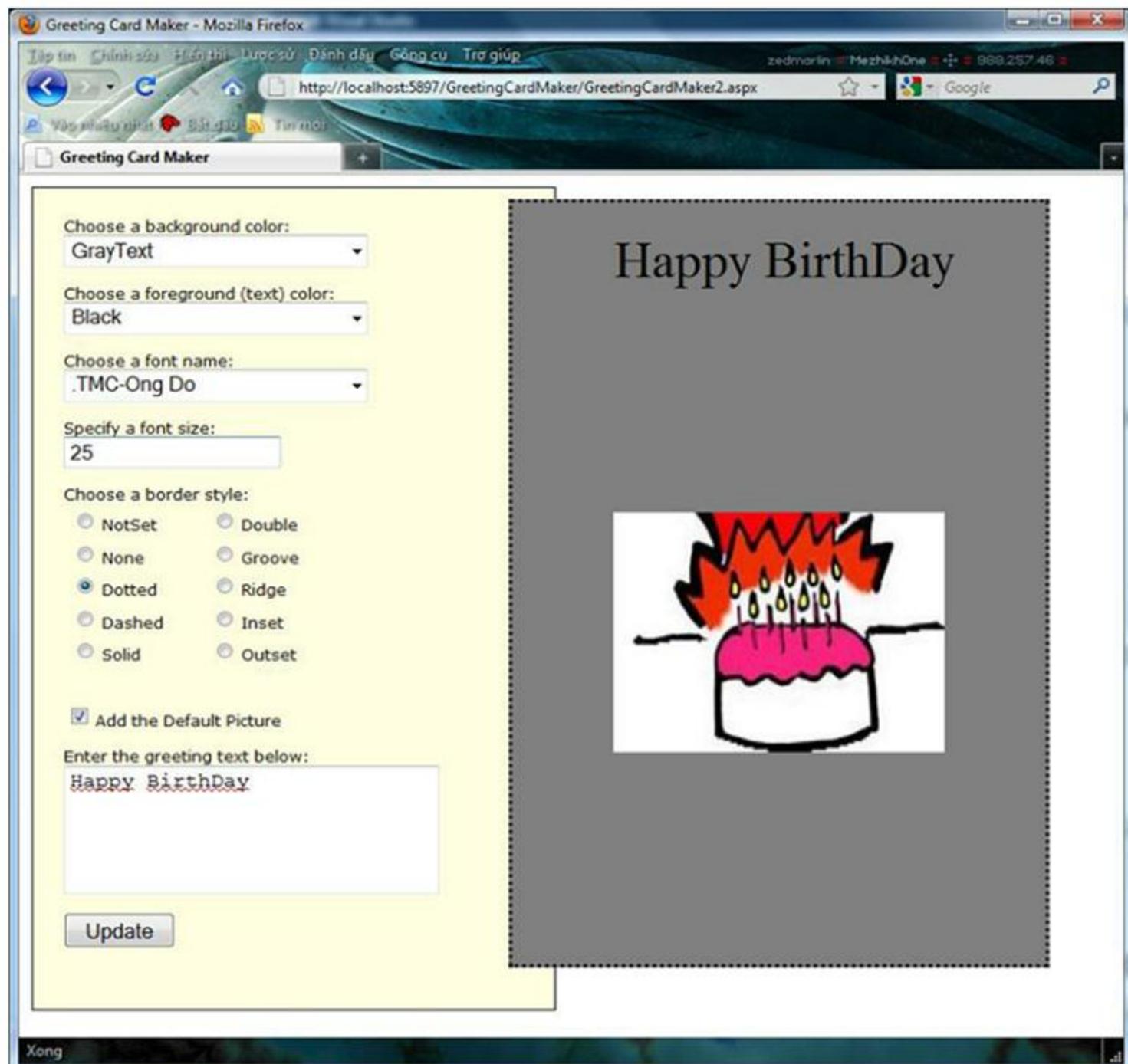
```
public partial class CurrencyConverter: System.Web.UI.Page
{
    protected void Page_Load(Object sender, EventArgs e)      {
        if (this.IsPostBack == false)      {
            // The HtmlSelect control accepts text or ListItem objects.
            Currency.Items.Add(new ListItem("Euros", "0.85"));
            Currency.Items.Add(new ListItem("Japanese Yen", "110.33"));
            Currency.Items.Add(new ListItem("Canadian Dollars", "1.2"));
        }
        Graph.Visible = false;
    }

    protected void Convert_ServerClick(object sender, EventArgs e)
    {
        decimal amount;
        bool success = Decimal.TryParse(US.Value, out amount);
        if (success)  { // Retrieve the selected ListItem object by its index number.
            ListItem item = Currency.Items[Currency.SelectedIndex];
            decimal newAmount = amount * Decimal.Parse(item.Value);
            Result.InnerText = amount.ToString() + " U.S. dollars = ";
            Result.InnerText += newAmount.ToString() + " " + item.Text;
        }
        else  {
            Result.InnerText = "The number you typed in was not in the correct format. ";
            Result.InnerText += "Use only numbers.";}
    }

    protected void ShowGraph_ServerClick(object sender, EventArgs e)
    { Graph.Src = "Pic" + Currency.SelectedIndex.ToString() + ".png";
        Graph.Visible = true;  }
}
```

2.7. THUỘC TÍNH AUTOPOSTBACK CỦA MỘT SỐ WEB SERVER CONTROLS

Chúng ta xét thí dụ thiết kế một thiệp, cụ thể như sau:



Khi bạn thay đổi một vài các giá trị: **background color**, **foreground color**, **font name**, **font size**, **border style**, **default picture** hoặc **greeting text**, nếu thuộc tính **AutoPostBack** của các Web Server Controls tương ứng: **<asp:TextBox>**, **<asp:dropdownlist>**, **<asp:radiobuttonlist>**, **<asp:checkbox>** trên trang đặt giá trị = **true**, bạn sẽ thấy việc hiệu chỉnh thiệp sẽ được cập nhật ngay sau khi bạn thay đổi.

Giải thích

Một web server control có thuộc tính **AutoPostBack=true**, có ý nghĩa là control đó sẽ postback tới server mỗi khi user tương tác với control đó, thuộc tính này support cho một số control của ASP.NET như CheckBox, TextBox, ListControl vì default chúng không tự động postback tới server. Với các ASP.NET control khác như Button ngầm

định chúng sẽ postback tới server mỗi khi user tương tác với chúng. Vì vậy cần phải thiết lập thuộc tính này = true cho checkbox chkAll như đoạn mã khai báo dưới đây, để server có thể nhận được tín hiệu từ control này sau khi user tương tác.

```
<asp:CheckBox ID="chkAll" runat="server"  
OnCheckedChanged="chkAll_CheckedChanged" AutoPostBack="true" />
```

Nội dung trang aspx như sau:

```
<%@ Page language="c#" Inherits="GreetingCardMaker.GreetingCardMaker2"  
CodeFile="GreetingCardMaker2.aspx.cs" %>  
  
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">  
  
<html xmlns="http://www.w3.org/1999/xhtml">  
    <head>  
        <title>Greeting Card Maker</title>  
    </head>  
    <body>  
        <form id="Form1" runat="server">  
            <div>  
                <div style="BORDER-RIGHT: thin ridge; PADDING-  
RIGHT: 20px; BORDER-TOP: thin ridge; PADDING-LEFT: 20px; FONT-SIZE: x-  
small; PADDING-BOTTOM: 20px; BORDER-LEFT: thin ridge; WIDTH: 293px;  
PADDING-TOP: 20px; BORDER-BOTTOM: thin ridge; FONT-FAMILY: Verdana;  
HEIGHT: 508px; BACKGROUND-COLOR: lightyellow">Choose  
                a background color:<br />  
                <asp:dropdownlist ID="lstBackColor"  
runat="server" Height="22px" Width="194px" AutoPostBack="True"  
onselectedindexchanged="ControlChanged"></asp:dropdownlist><br />  
                <br />  
                Choose a foreground (text) color:<br />  
                <asp:dropdownlist ID="lstForeColor"  
runat="server" Height="22px" Width="194px" AutoPostBack="True"  
onselectedindexchanged="ControlChanged"></asp:dropdownlist><br />  
                <br />  
                Choose a font name:<br />  
                <asp:dropdownlist ID="lstFontName"  
runat="server" Height="22px" Width="194px" AutoPostBack="True"  
onselectedindexchanged="ControlChanged"></asp:dropdownlist><br />
```

```

<br />
Specify a font size:<br />
<asp:textbox ID="txtFontSize" runat="server"
AutoPostBack="True" ontextchanged="ControlChanged"></asp:textbox><br />
<br />
Choose a border style:<br />
<asp:radiobuttonlist ID="lstBorder" runat="server"
Height="59px" Width="177px" Font-Size="X-Small" AutoPostBack="True"
RepeatColumns="2"
onselectedindexchanged="ControlChanged"></asp:radiobuttonlist><br />
<br />
<asp:checkbox ID="chkPicture" runat="server"
Text="Add the Default Picture" AutoPostBack="True"
oncheckedchanged="ControlChanged"></asp:checkbox><br />
<br />
Enter the greeting text below:<br />
<asp:textbox ID="txtGreeting" runat="server"
Height="85px" Width="240px" TextMode="MultiLine" AutoPostBack="True"
ontextchanged="ControlChanged"></asp:textbox><br />
<br />
<asp:button ID="cmdUpdate" runat="server"
Height="24px" Width="71px" Text="Update"
onclick="cmdUpdate_Click"></asp:button>&nbsp;</div>
<asp:panel ID="pnlCard" style="Z-INDEX: 101; LEFT:
313px; POSITION: absolute; TOP: 16px" runat="server"
Height="507px" Width="339px"
HorizontalAlign="Center"><br />&nbsp;
<asp:Label ID="lblGreeting" runat="server"
Height="150px" Width="256px"></asp:Label>
<br /><br /><br />
<asp:Image ID="imgDefault" runat="server"
Height="160px" Width="212px" Visible="False"></asp:Image>
</asp:panel>
</div>
</form>
</body>
</html>

```

Code phía server như sau:

```

public partial class GreetingCardMaker2: System.Web.UI.Page
{
    protected void Page_Load(object sender, System.EventArgs e)
    {
        if (this.IsPostBack == false)
        {
            // Get the list of colors.
            string[] colorArray =
Enum.GetNames(typeof(KnownColor));

            lstBackColor.DataSource = colorArray;
            lstBackColor.DataBind();

            lstForeColor.DataSource = colorArray;
            lstForeColor.DataBind();
            lstForeColor.SelectedIndex = 34;
            lstBackColor.SelectedIndex = 163;

            // Get the list of available fonts and add them to the
font list.

            InstalledFontCollection fonts = new
InstalledFontCollection();
            foreach (FontFamily family in fonts.Families)
            {
                lstFontName.Items.Add(family.Name);
            }

            // Set border style options.

            string[] borderStyleArray =
Enum.GetNames(typeof(BorderStyle));
            lstBorder.DataSource = borderStyleArray;
            lstBorder.DataBind();

            // Select the first border option.

            lstBorder.SelectedIndex = 0;

            // Set the picture.

            imgDefault.ImageUrl = "defaultpic.png";
        }
    }
}

```

```
    }

private void UpdateCard()
{
    // Update the color.

    pnlCard.BackColor =
Color.FromName(lstBackColor.SelectedItem.Text);

    lblGreeting.ForeColor =
Color.FromName(lstForeColor.SelectedItem.Text);

    // Update the font.

    lblGreeting.Font.Name =
lstFontName.SelectedItem.Text;

    try
    {
        if (Int32.Parse(txtFontSize.Text) > 0)
        {
            lblGreeting.Font.Size =
FontUnit.Point(Int32.Parse(txtFontSize.Text));
        }
    }
    catch
    {
        // Ignore invalid value.
    }

    try
    {
        if (Int32.Parse(txtFontSize.Text) > 0)
        {
            lblGreeting.Font.Size
=FontUnit.Point(Int32.Parse(txtFontSize.Text));
        }
    }
    catch
    {
        // Ignore invalid value.
    }
}
```

```
// Find the appropriate TypeConverter for the BorderStyle  
enumeration.
```

```
TypeConverter cnvrt =  
TypeDescriptor.GetConverter(typeof(BorderStyle));
```

```
// Update the border style using the value from the converter.  
pnlCard.BorderStyle =  
(BorderStyle)cnvrt.ConvertFromString(lstBorder.SelectedItem.Text);
```

```
// Update the picture.  
if (chkPicture.Checked == true)  
{  
    imgDefault.Visible = true;  
}  
else  
{  
    imgDefault.Visible = false;  
}
```

```
// Set the text.  
lblGreeting.Text = txtGreeting.Text;  
}
```

```
protected void ControlChanged(Object sender, EventArgs e)  
{
```

```
    // Refresh the greeting card (because a control was changed).  
UpdateCard();  
}
```

```
protected void cmdUpdate_Click(object sender, EventArgs e)  
{
```

```
    // Refresh the greeting card (because the button was clicked).  
UpdateCard();  
}
```

BÀI TẬP CHƯƠNG 2

Bài 1: Thực hành lại tất cả các thí dụ về HTML Server Controls và Web Server Controls đã được học. Có thể viết các bài thực hành có nội dung tương đương các thí dụ để nắm vững bài học.

Bài 2: Sử dụng các điều khiển thực hiện bài tập sau:

Tạo trang Default.aspx với nội dung sau:

BÀI TẬP SỬ DỤNG WEB SERVER CONTROLS

- 1. [Nhập x. In ra trị tuyệt đối x.](#)
- 2. [Tính \$S=1+2+3+\dots+n\$ với \$n\$ nguyên \$\geq 1\$.](#)
- 3. [In ra các số có 3 chữ số sao cho \$abc=a^3+b^3+c^3\$.](#)
- 4. [Tính \$\cos\(x\)=1-x^2/2!+x^4/4!-\dots\$ với \$\text{epsilon}=10^{-6}\$.](#)

Khi click vào các LinkButton 1, 2, 3 hoặc 4, sẽ chuyển đến trang thực hiện các bài tập tương ứng:

1. Nhập x và in ra trị tuyệt đối x (

TRỊ TUYỆT ĐỐI X

Nhập x :

Tri Tuyet Doi X

2. Tính $S = 1 + 2 + 3 + \dots + n$ với n số nguyên và $n \geq 1$

3. In ra các số có ba chữ số sao cho $abc = a^3 + b^3 + c^3$.

IN CÁC SỐ CÓ 3 CHỮ SỐ BẰNG TỔNG LẬP PHƯƠNG TÙNG CHỮ SỐ

In số

Các số : 153 , 370 , 371 , 407

4. Tính $\cos(x) = 1 - x^2/2! + x^4/4! - \dots$ với $\text{epsilon} = 10^{-6}$.

TÍNH $\cos(x)=1-X^2/2!+X^4/4!-\dots$ dùng với $\text{epsilon}=10^{-6}$

Nhập góc x (độ) cần tính $\cos(x)$:

Tính Cos(X)

$\cos(30)=0.866025404210352$

Hướng dẫn

Nội dung trang Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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>Untitled Page</title>

    <style type="text/css">

        .style1
        {
            width: 100%;
        }

        .style2
        {
            text-align: center;
            font-weight: bold;
            font-size: x-large;
        }
    </style>

</head>

<body>

    <form id="form1" runat="server">
        <div>
            <table class="style1">
                <tr>
                    <td class="style2">
                        BÀI TẬP SỬ DỤNG WEB SERVER CONTROLS
                    </td>
                </tr>
                <tr>
                    <td>
                        <asp:BulletedList ID="BulletedList1" runat="server" BorderStyle="Dotted">

```

```

        BulletStyle="Numbered" DisplayMode="LinkButton"
        onclick="BulletedList1_Click">
            </asp:BulletedList>
        </td>
    </tr>
    <tr>
        <td>
            &nbsp;</td>
        </tr>
    </table>
</div>
</form>
</body>
</html>

```

Phần mã lệnh trang Default.aspx.cs

```

public partial class _Default: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            BulletedList1.Items.Add("Nhập x. In ra trị tuyêt đồi x.");
            BulletedList1.Items.Add("Tính S=1+2+3+...+n với n nguyên >=1");
            BulletedList1.Items.Add("In ra các số có 3 chữ số sao cho
abc=a^3+b^3+c^3.");
            BulletedList1.Items.Add("Tìm Cos(x)=1-x^2/2!+x^4/4!-... với epsilon=10^-6");
        }
    }
    protected void BulletedList1_Click(object sender, BulletedListEventArgs e)
    {
        int item = e.Index;
        if (item==0)

```

```

Response.Redirect("TrituyetdoiX.aspx");
else
if (item==1)
    Response.Redirect("TinhS.aspx");
else
if (item==2)
    Response.Redirect("BaChuSo.aspx");
else
    Response.Redirect("Cosx.aspx");
}
}

```

- Tạo trang BaChuSo.aspx với nội dung thiết kế sau:

```

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

<!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>
    <style type="text/css">
        .style1
        {
            width: 100%;
        }
        .style2
        {
            font-size: large;
            font-weight: bold;
            text-align: center;
        }
    </style>
</head>
<body>
    <div style="text-align: center; background-color: #cccccc; width: 100%; height: 100px;>
        <div style="width: 100%; height: 100%; background-color: #cccccc; border: 1px solid black; border-radius: 10px; padding: 10px; margin: auto; position: relative; top: -50%; left: -50%;>
            <div style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; background-color: #cccccc; border: 1px solid black; border-radius: 10px; z-index: -1; opacity: 0.5; filter: alpha(opacity=50);>
            </div>
            <div style="position: absolute; top: 50%; left: 50%; width: 200px; height: 100px; background-color: white; border: 1px solid black; border-radius: 10px; padding: 10px; text-align: center; transform: rotate(-45deg);>
                <h1>Chu So</h1>
                <p>Chu So</p>
            </div>
        </div>
    </div>
</body>

```

```

</style>
</head>
<body>
<form id="form1" runat="server">
<div>

<table class="style1">
<tr>
<td class="style2">
IN CÁC SỐ CÓ BA CHỮ SỐ BẰNG TỔNG LẬP PHƯƠNG TÙNG CHỮ
SỐ</td>
</tr>
<tr>
<td>
<asp:Button ID="BtnIn3So" runat="server" onclick="BtnIn3So_Click"
style="font-size: medium; font-weight: 700" Text="In số" />
</td>
</tr>
<tr>
<td>
<asp:Label ID="lblKQ" runat="server"
style="text-align: center; font-weight: 700; font-size:
large"></asp:Label>
</td>
</tr>
</table>

</div>
</form>
</body>
</html>

```

Phần mã lệnh trang BaChuSo.aspx.cs như sau:

```
public partial class BaChuSo: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e) { }

    protected void BtnIn3So_Click(object sender, EventArgs e)
    {
        int a, b, c;
        string kq = "Cac so: ";
        for (int i = 100; i <= 999; i++)
        {
            a = i / 100;
            b = (i % 100) / 10;
            c = i % 10;
            if (Math.Pow(a, 3) + Math.Pow(b, 3) + Math.Pow(c, 3) == i)
                kq += i.ToString() + ",";
        }
        lblKQ.Text = kq.Substring(0, kq.Length - 1);
    }
}
```

- Tạo trang Cosx.aspx với nội dung thiết kế sau:

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

<!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>
    <style type="text/css">
        .style1
    {
        width: 100%;
    }

```

```

.style2
{
    text-align: center;
    font-weight: bold;
    font-size: large;
}

.style3
{
    font-size: large;
    font-weight: bold;
}

</style>
</head>
<body>

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

<table class="style1">
    <tr>
        <td class="style2" colspan="3">
TÍNH COS(X)=1-X^2/2!+X^2/4!-.... dùng với epsilon=10^-6</td>
    </tr>
    <tr>
        <td>
            <span class="style3">&nbsp;Nhập góc x (độ) cần tính<br/>
Cos(x)&nbsp;;</span>&nbsp;</td>
        <td>
            <asp:TextBox ID="txtX" runat="server"></asp:TextBox>
        </td>
        <td>
            &nbsp;</td>
    </tr>
    <tr>

```

```

<td>
    <asp:Button ID="btnSinx" runat="server" Font-Bold="True"
    onclick="btnCox_Click"
        Text="Tính Cos(X)" />
</td>
<td>
    &nbsp;</td>
<td>
    &nbsp;</td>
</tr>
<tr>
    <td colspan="3" style="font-size: large; font-weight: 700; text-align:
    center">
        <asp:Label ID="lblKQ" runat="server"></asp:Label>
    </td>
</tr>
</table>

</div>
</form>
</body>
</html>

```

Phần mã lệnh Cosx.aspx.cs

```

public partial class Cosx : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (this.IsPostBack == false) txtX.Text = "0.00";
    }
    protected void btnCox_Click(object sender, EventArgs e)
    {
        double x,p,cosx;

```

```
const double esp=1e-06;
int i;
try
{
    x = Double.Parse(txtX.Text)*Math.PI/180; // doi tu do sang radian
    cosx = 1;
    p = -Math.Pow(x, 2) / 2;
    i = 2;
    while(Math.Abs(p)>=esp){
        cosx+=p;
        p*=-Math.Pow(x,2)/((i+1)*(i+2));
        i+=2;
    }
    cosx+=p;
    lblKQ.Text="Cos("+txtX.Text+)"="+cosx;
}
catch(Exception ex){
}
}
```

Chương 3

MASTER PAGE – WEBSITE NAVIGATION

Các vấn đề chính sẽ được đề cập:

- ✓ *Cấu trúc một Master Page*
- ✓ *Các dạng tạo Master Page*
- ✓ *Cách tạo SiteMap kết hợp với trang Master Page*
- ✓ *Cách tạo Web User Control*
- ✓ *Các đối tượng Request, Response và Server*

Kết thúc bài này các bạn có thể:

- *Nắm vững cách tạo Master Page và Web User Control, sử dụng các đối tượng ASP.NET gồm Request, Response và Server.*

3.1. MASTER PAGE

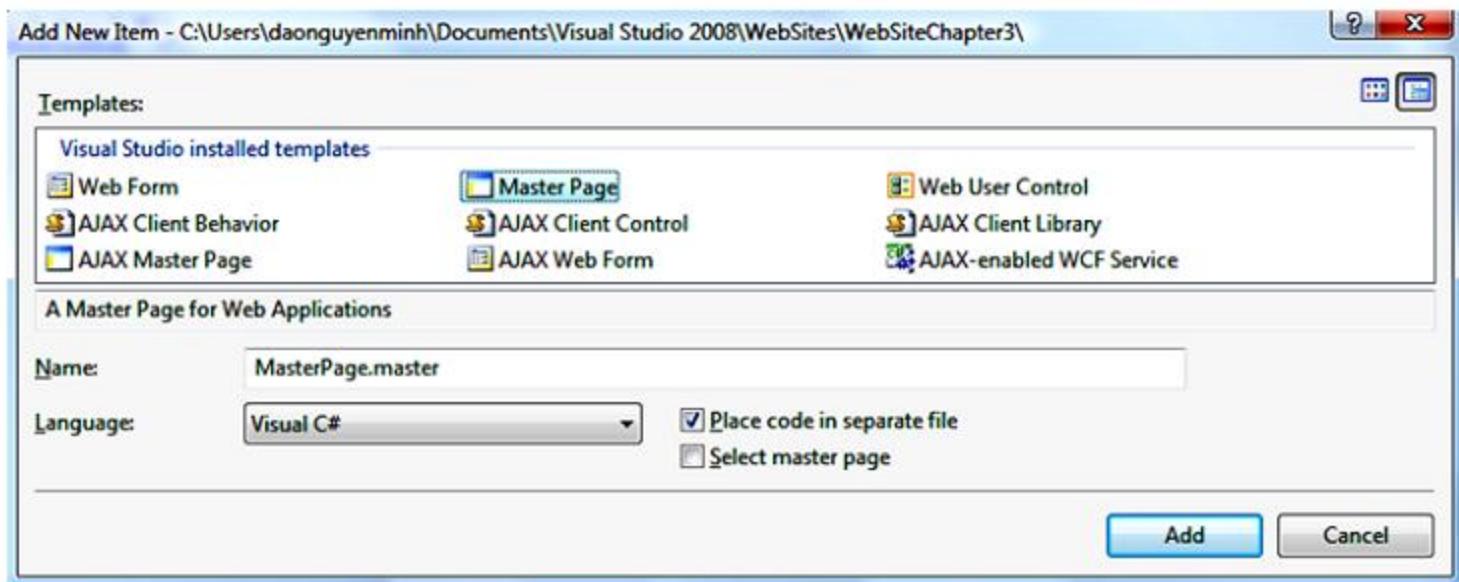
Các trang Master tương tự như các trang ASP.NET. Giống như các trang thông thường, các trang master là những tập tin văn bản có thể chứa HTML, điều khiển web và mã lệnh. Tuy nhiên, các trang master có phần mở rộng tập tin là .master thay vì .aspx, và chúng không thể được xem trực tiếp bởi một trình duyệt. Thay vào đó, các trang master phải được sử dụng bởi các trang khác, được biết đến như là nội dung trang. Về cơ bản, trang master xác định cấu trúc trang và các thành phần chung.

Các trang nội dung thông qua cấu trúc này và chỉ cần điền vào nó với nội dung thích hợp.

Thí dụ, nếu một trang web bán sách được tạo ra bằng cách sử dụng ASP.NET, một trang master đơn có thể xác định bố trí cho toàn bộ trang web. Mỗi trang sẽ sử dụng trang master này, và kết quả là, mỗi trang sẽ có cùng một tổ chức cơ bản và tiêu đề tương tự (header, footer,...). Tuy nhiên, mỗi trang sẽ cũng chèn thông tin cụ thể của nó, chẳng hạn như sách bán chạy, sách mới, sách giảm giá, hoặc kết quả tìm kiếm vào trong mẫu này.

3.1.1. Tạo Master Page đơn giản

Để tạo một Master Page trong một ứng dụng web có sẵn, bạn chọn **R-Click** tại ứng dụng web trong cửa sổ **Solution Explorer** và chọn **Add New Item...**, xuất hiện hộp thoại, bạn chọn **Master Page** như trong hình 3.1



Hình 3.1: Hộp thoại tạo Master Page

Nhập tên master page vào trong khung **Name:** **DefaultContent.master** và nhấp nút **Add**, trang DefaultContent.master sẽ được thêm vào ứng dụng, và nội dung sau:

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeFile="DefaultContent.master.cs" Inherits="DefaultContent" %>

<!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">

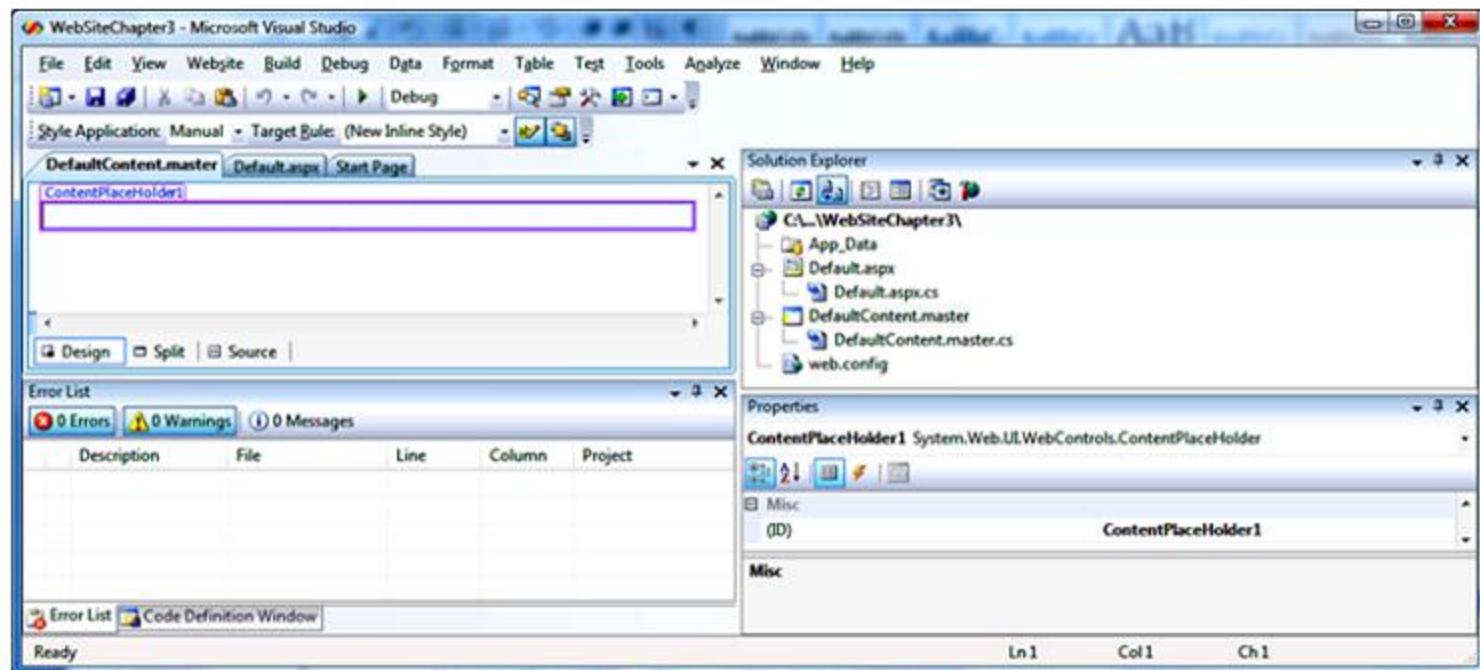
<head runat="server">
    <title>Untitled Page</title>
    <asp:ContentPlaceHolder id="head" runat="server">
        </asp:ContentPlaceHolder>
</head>

<body>
    <form id="form1" runat="server">
        <div>
            <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
                </asp:ContentPlaceHolder>
        </div>
    </form>
</body>
</html>
```

Bạn để ý dòng đầu tiên của trang sẽ bắt đầu bằng chỉ dẫn **Master**:

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeFile="DefaultContent.master.cs" Inherits="DefaultContent" %>
```

Chuyển qua chế độ hiển thị là Design, trên trang master vừa tạo đã có sẵn hai thành phần là ContentPlaceHolder:



Hình 3.2: Trang DefaultContent.master vừa được thêm vào ứng dụng

ContentPlaceholder thứ nhất được đưa vào trong thẻ Head, mang đến cho các trang nội dung các trang thêm siêu dữ liệu (meta data), chẳng hạn như các từ khóa tìm kiếm và liên kết trang tính (search keywords and style sheet links)

```
<head runat="server">
    <title>Untitled Page</title>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
```

ContentPlaceholder thứ hai được đưa vào trong thẻ Body, đại diện các nội dung hiển thị của trang. Nó xuất hiện trên các trang như là một khung rỗng chờ sẵn nội dung sẽ được điền vào từ trang đó. Nếu bạn nhấp vào bên trong hoặc di chuột qua nó, tên của ContentPlaceholder xuất hiện trong một tooltip, như trong hình 3.3. Phần kích thước của khung sẽ mở rộng để phù hợp với nội dung bạn đặt bên trong.



Hình 3.3: Trang DefaultContent.master chứa ContentPlaceholder bên trong thẻ Body

Bạn thay đổi nội dung trang DefaultContent.master như sau:

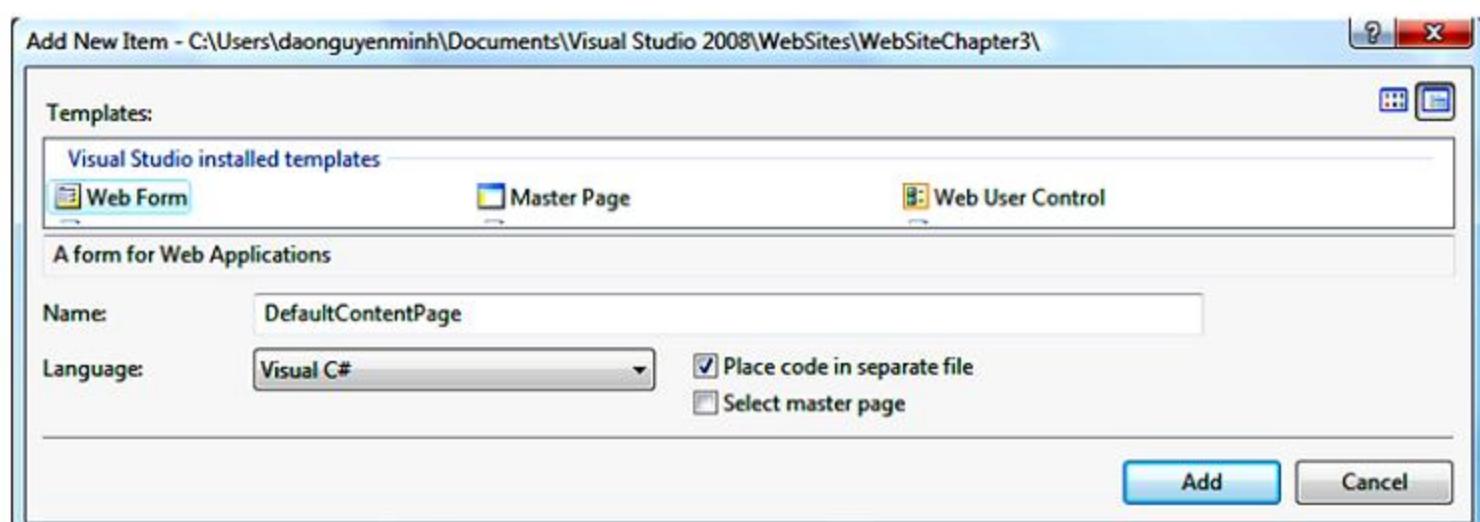
```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <br />
            <asp:contentplaceholder id="ContentPlaceholder1" runat="server">
                This is default content.<br />
            </asp:contentplaceholder>
            <i>This is a simple footer.</i>
        </div>
    </form>
</body>
</html>
```

Chuyển qua chế độ hiển thị là Design, trên trang master vừa thay đổi có nội dung sau:



Hình 3.4: Trang DefaultContent.master sau khi cập nhật nội dung

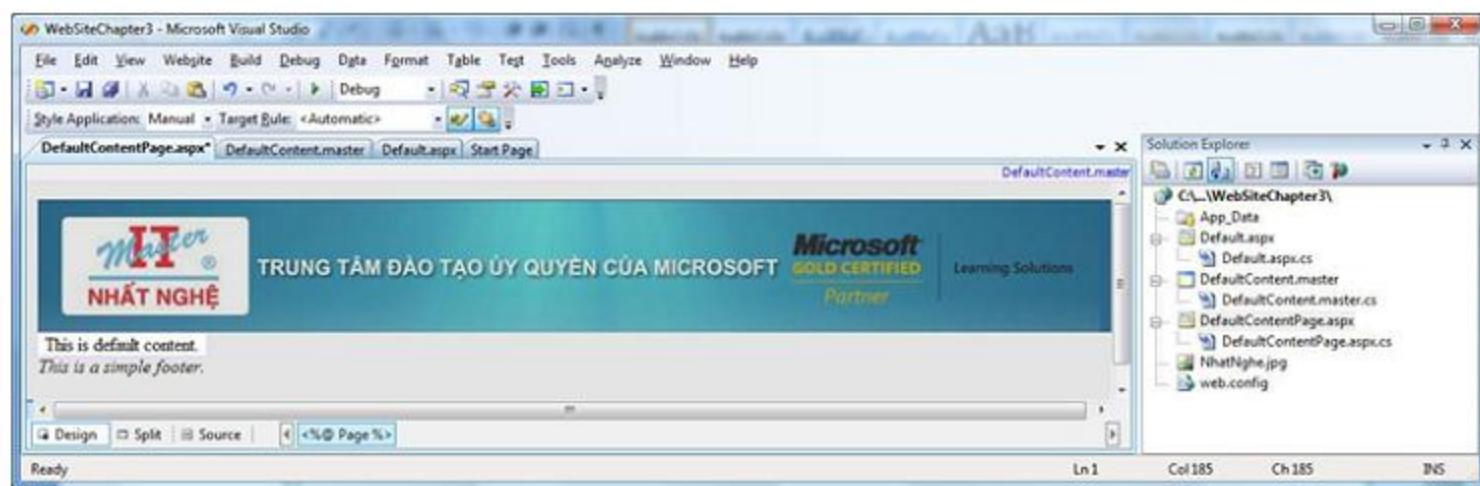
Tạo trang DefaultContentPage.aspx sử dụng trang master là DefaultContent.master.



Chỉnh sửa nội dung trang DefaultContentPage.aspx chỉ còn một dòng mã lệnh như sau, chú ý thuộc tính MasterPageFile trong thẻ chỉ dẫn Page:

```
<%@ Page Language="C#" MasterPageFile="~/DefaultContent.master"
AutoEventWireup="true" CodeFile="DefaultContentPage.aspx.cs"
Inherits="DefaultContentPage_aspx" Title="Untitled Page" %>
```

Kết quả hiển thị khi chuyển sang chế độ Design, nội dung của trang giống như nội dung trang master được khai báo trong nó.



Hình 3.5: Trang DefaultContentPage.aspx sau khi cập nhật nội dung

- Đưa nội dung vào trong khung ContentPlaceHolder:

Tạo thêm một trang master cho ứng dụng là SiteTemplate.master có nội dung sau:

```

<%@ Master Language="C#" AutoEventWireup="true"
CodeFile="SiteTemplate.master.cs" Inherits="SiteTemplate" %>
<!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 id="Head1" runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <br />
            <asp:contentplaceholder id="ContentPlaceHolder1" runat="server">
            </asp:contentplaceholder>
            <i>This is a simple footer.</i>
        </div>
    </form>
</body>
</html>

```

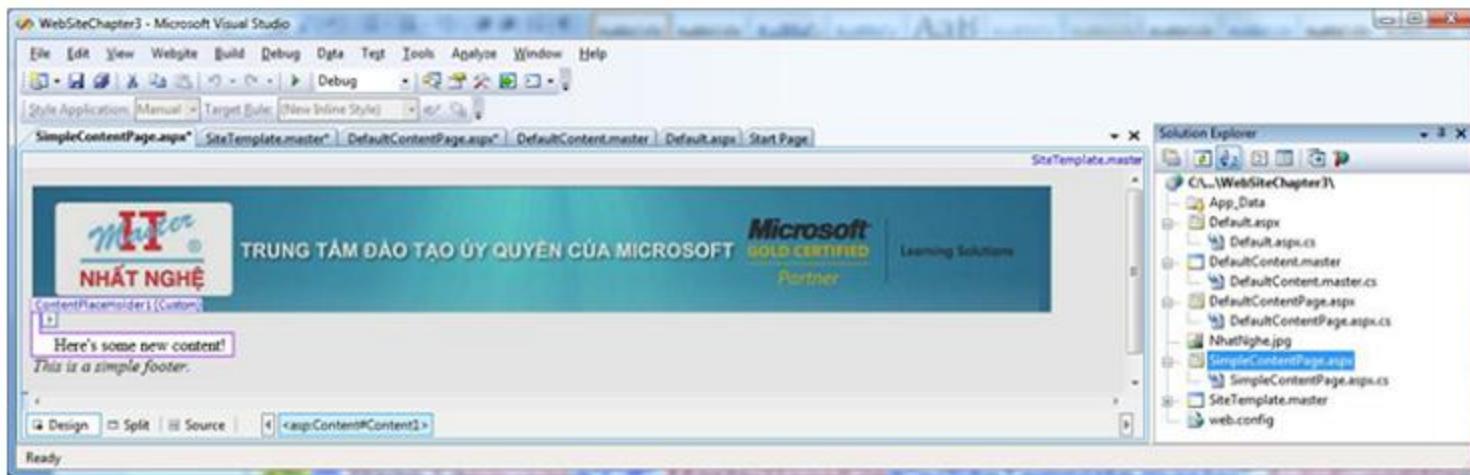
Và tạo trang **SimpleContentPage.aspx** có **MasterPageFile="~/SiteTemplate.master"**, trong trang này có đưa nội dung vào trong **ContentPlaceHolder** qua thẻ **asp:Content** như sau:

```

<%@ Page Language="C#" MasterPageFile="~/SiteTemplate.master"
AutoEventWireup="true" CodeFile="SimpleContentPage.aspx.cs"
Inherits="SimpleContentPage_aspx" Title="Content Page" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"
Runat="Server">
    <br />
    &nbsp;&nbsp;&nbsp; Here's some new content!<br />
</asp:Content>

```

Kết quả hiển thị ở chế độ Design, nội dung của trang SimpleContentPage.aspx như sau:



Hình 3.6: Trang SimpleContentPage.aspx có nội dung hiển thị trong ContentPlaceHolder

Bạn để ý thấy nội dung:

```
<br />
    &nbsp;&nbsp;&nbsp;&nbsp; Here's some new content!<br />
```

được hiển thị trong khung ContentPlaceHolderID="ContentPlaceHolder1" khai báo trong trang SiteTemplate.master

3.1.2. Tạo trang Mater Page có nhiều khu vực chứa nội dung

Trang Master không giới hạn chỉ có một ContentPlaceHolder. Tất cả bạn cần làm là thêm nhiều điều khiển ContentPlaceHolder và sắp xếp chúng một cách thích hợp trên trang Master này.

Tạo thêm một trang master vào ứng dụng có tên **MultipleContent.master** với nội dung sau:

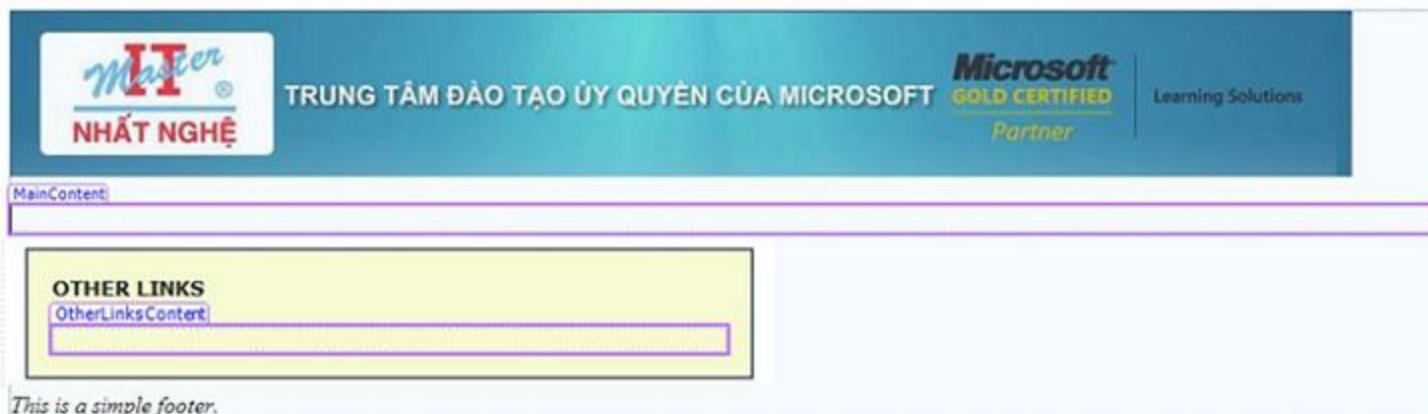
```
<%@ Master Language="C#" AutoEventWireup="true"
CodeFile="MultipleContent.master.cs" Inherits="MultipleContent" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <br /><br />
            <asp:contentplaceholder id="MainContent" runat="server">
            </asp:contentplaceholder>
    </form>
</body>
</html>
```

```

<div style="border-right: 1px solid; padding-right: 15px; border-top: 1px solid;
padding-left: 15px; font-size: small; padding-bottom: 15px; margin: 15px;
border-left: 1px solid;
width: 417px; padding-top: 15px; border-bottom: 1px solid; font-family:
Verdana;
background-color: #ffffcc">
<strong>
OTHER LINKS<br />
</strong>
<asp:ContentPlaceHolder id="OtherLinksContent" runat="server">
</asp:ContentPlaceHolder>
</div>
<i>This is a simple footer.</i>
</div>
</form>
</body>
</html>

```

Kết quả hiển thị khi chuyển sang chế độ Design, nội dung của trang MultipleContent.master như sau:



Hình 3.7: Trang MultipleContent.master có hai khung ContentPlaceholder

Trang master này có chứa hai ContentPlaceholder trong thẻ Body là:

```

<asp:contentplaceholder id="MainContent" runat="server">
</asp:contentplaceholder>

```

và

```

<asp:ContentPlaceHolder id="OtherLinksContent" runat="server">
</asp:ContentPlaceHolder>

```

Sau đó, bạn tạo trang **MultipleContentPage.aspx** sử dụng trang **MultipleContent.master** trên với nội dung sau:

```
<%@ Page Language="C#" MasterPageFile="~/MultipleContent.master"
AutoEventWireup="true" CodeFile="MultipleContentPage.aspx.cs"
Inherits="MultipleContentPage_aspx" Title="Untitled Page" %>
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" Runat="Server">
    This is the generic content for this page. Here you might provide some site
    specific
    text. This is the generic content for this page. Here you might provide some site
    specific text. This is the generic content for this page. Here you might provide
    some site specific text. This is the generic content for this page. Here you might
    provide some site specific text.<br />
    <br />
    This is the generic content for this page. Here you might provide some site
    specific
    text. This is the generic content for this page. Here you might provide some site
    specific text. This is the generic content for this page. Here you might provide
    some site specific text.</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="OtherLinksContent"
Runat="Server">
    Here's a <a href="http://www.prosetech.com">link</a>.
    <br />
    Here's a <a href="http://www.prosetech.com">link</a>.<br />
    Here's a <a href="http://www.prosetech.com">link</a>.<br />
    Here's a <a href="http://www.prosetech.com">link</a>.</asp:Content>
```

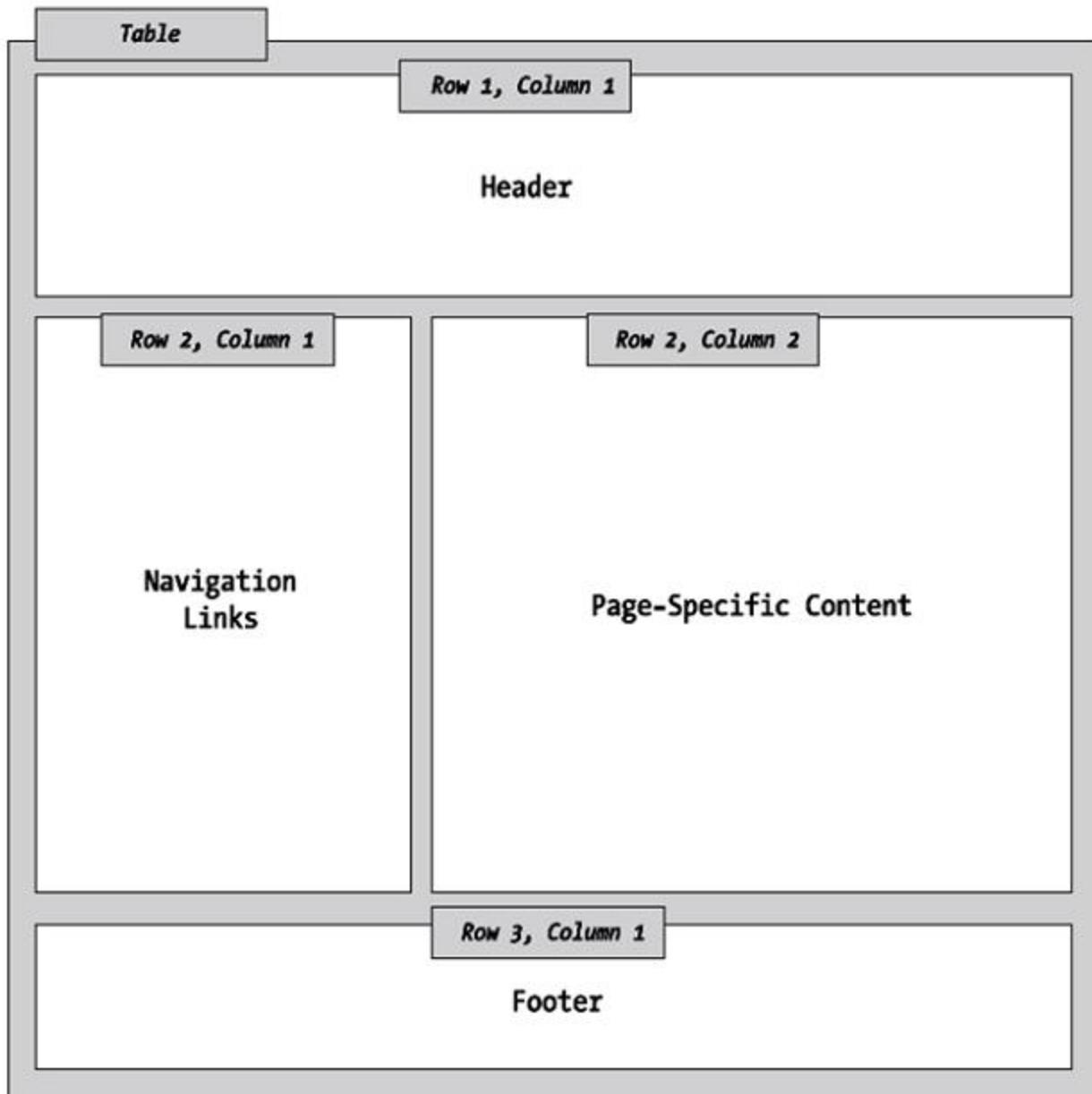
Kết quả hiển thị khi chuyển sang chế độ Design, nội dung của trang **MultipleContentPage.aspx** như sau:

The screenshot shows a Microsoft MultipleContentPage.aspx page. At the top left is the "Master NHẤT NGHỆ" logo. To its right is the text "TRUNG TÂM ĐÀO TẠO ỦY QUYỀN CỦA MICROSOFT". Further right is the Microsoft "GOLD CERTIFIED Partner" badge and the text "Learning Solutions". Below the header, there are two content placeholders. The first placeholder contains the text "This is the generic content for this page. Here you might provide some site specific text. This is the generic content for this page. Here you might provide some site specific text. This is the generic content for this page. Here you might provide some site specific text. This is the generic content for this page. Here you might provide some site specific text." The second placeholder contains the same generic text. Below these placeholders is a yellow box labeled "OTHER LINKS" containing four links. At the bottom of the page is a footer note: "This is a simple footer."

Hình 3.8: Trang MultipleContentPage.aspx có nội dung đưa vào cho hai khung ContentPlaceholder

3.1.3. Tạo trang Master Page với cấu trúc bảng

Phần này hướng dẫn các bạn tạo một trang Master theo thiết kế bảng (**Table-Based Layouts**). Chúng ta thiết kế các trang ASP.NET theo cùng một mẫu trang master có dạng như hình sau:



Hình 3.9: Trang Master thiết kế theo table layout

Tạo trang master có tên **TableMaster.master** có nội dung sau:

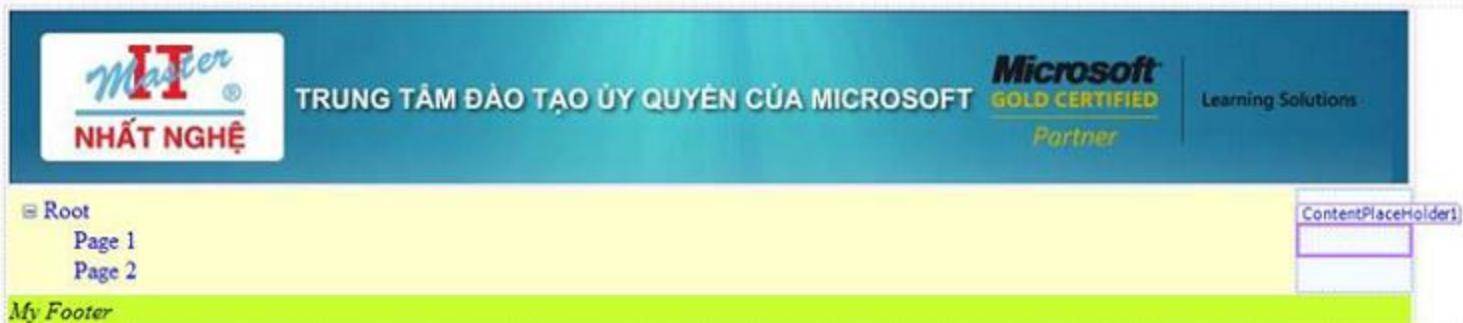
```
<%@ Master Language="C#" AutoEventWireup="true"
CodeFile="TableMaster.master.cs" Inherits="TableMaster" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <table width="80%">
            <tr>
                <td colspan="2" style="background: #ffccff">
                    <!-- MyHeader -->
                    <br />
                </td>
            </tr>
            <tr>
                <td style="background: #ffffcc" >
                    <asp:treeview ID="Treeview1" runat="server" Width="150px">
                        <Nodes>
                            <asp:TreeNode Text="Root" Value="New Node">
                                <asp:TreeNode Text="Page 1" Value="Page 1"></asp:TreeNode>
                                <asp:TreeNode Text="Page 2" Value="Page 2"></asp:TreeNode>
                            </asp:TreeNode>
                        </Nodes>
                    </asp:treeview>
                </td>
                <td>
                    <asp:contentplaceholder id="ContentPlaceHolder1" runat="server">
                    </asp:contentplaceholder>
                </td>
            </tr>
            <tr>
                <td colspan="2" style="background: #ccff33">
                    <i>My Footer</i>
                </td>
            </tr>
        </table>
    </form>
</body>
</html>
```

```

</tr>
</table>
</form>
</body>
</html>

```

Kết quả hiển thị khi chuyển sang chế độ Design, thiết kế của trang **TableMaster.master** như sau:



Hình 3.10: Trang TableMaster thiết kế theo table layout

Khai báo một property **ShowNavigationControls** trong trang TableMaster.master.cs để bật hoặc tắt điều khiển Treeview trên trang master.

```

public partial class TableMaster : System.Web.UI.MasterPage
{
    protected void Page_Load(object sender, EventArgs e) { }

    public bool ShowNavigationControls {
        set { Treeview1.Visible = value; }
        get { return Treeview1.Visible; }
    }
}

```

Sau đó, bạn tạo trang **TableContentPage.aspx** sử dụng trang **TableMaster.master** trên với nội dung sau:

```

<%@ Page Language="C#" MasterPageFile="~/TableMaster.master"
AutoEventWireup="true" CodeFile="TableContentPage.aspx.cs"
Inherits="TableContentPage_aspx" Title="Untitled Page" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"
Runat="Server"> Your content goes in this cell.<br /><br />
    <asp:Button ID="cmdShow" runat="server" Text="Show"
    OnClick="cmdShow_Click" />
    <asp:Button ID="cmdHide" runat="server" Text="Hide" OnClick="cmdHide_Click" />
</asp:Content>

```

Kết quả hiển thị khi chuyển sang chế độ Design, nội dung của trang TableContentPage.aspx như sau:



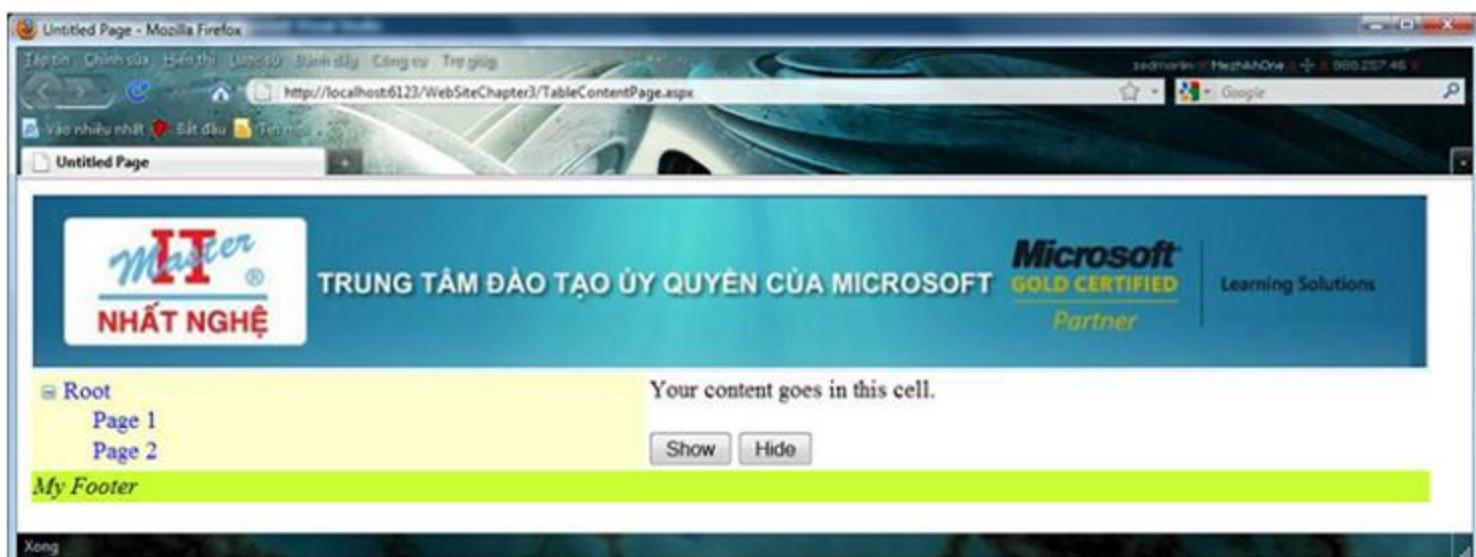
Viết phần mã lệnh thực thi trong trang TableContentPage.aspx.cs:

```
public partial class TableContentPage : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e) { }

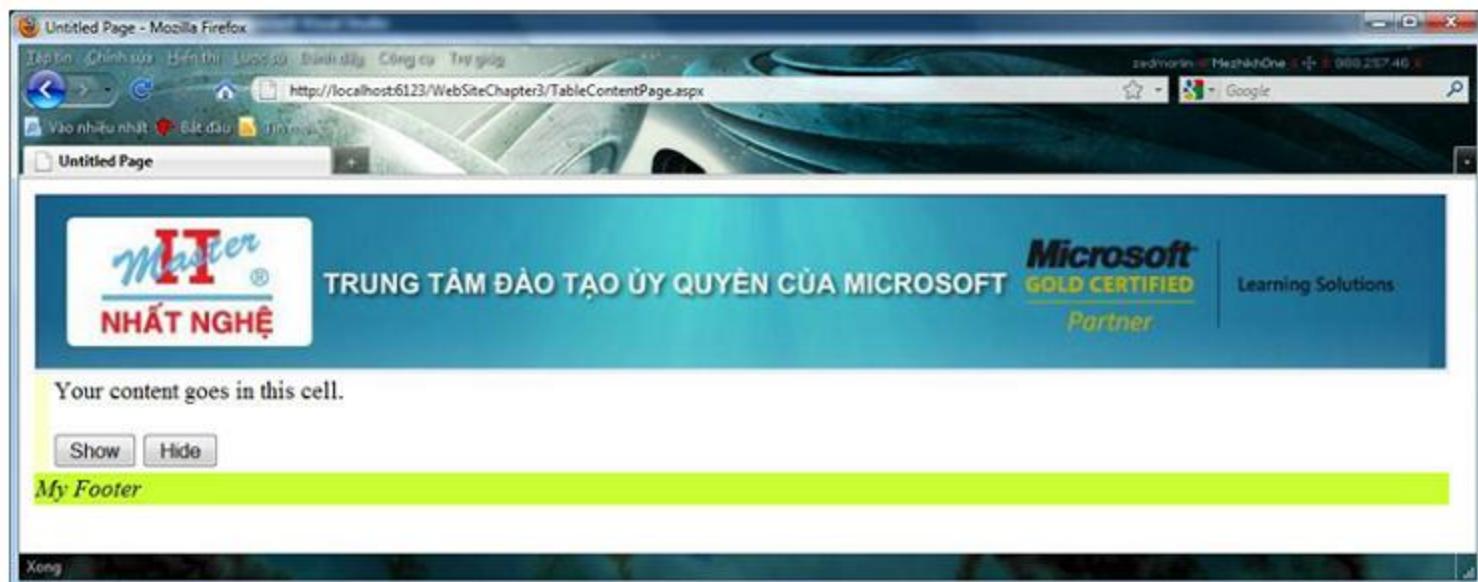
    // Khi click nút Hide, che điền điều khiển Treeview trên trang master
    protected void cmdHide_Click(object sender, EventArgs e) {
        TableMaster master = (TableMaster)this.Master;
        master.ShowNavigationControls = false;
    }

    // Khi click nút Show, hiển thị điều khiển Treeview trên trang master
    protected void cmdShow_Click(object sender, EventArgs e) {
        TableMaster master = (TableMaster)this.Master;
        master.ShowNavigationControls = true;
    }
}
```

Thực thi trang TableContentPage.aspx, ta có kết quả:



Hình 3.11a: Thực thi trang TableContentPage (click nút Show)

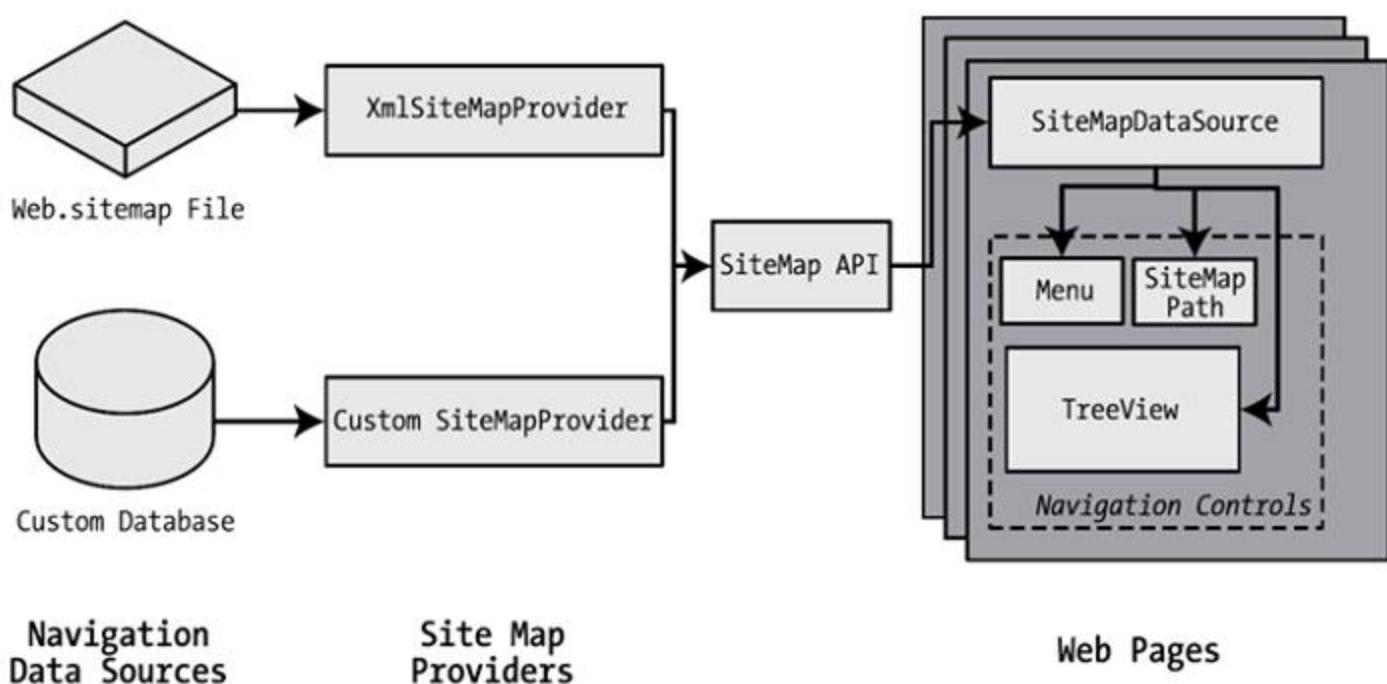


Hình 3.11b: Thực thi trang TableContentPage (click nút Hide)

3.2. WEBSITE NAVIGATION

Bạn đã học được cách đơn giản để cho một người truy cập trang web từ trang này sang trang khác. Thí dụ, bạn có thể thêm các liên kết HTML (hoặc điều khiển HyperLink) trong trang của bạn để cho phép người dùng lướt web thông qua trang web của bạn. Nếu bạn muốn thực hiện chuyển hướng trang, bạn có thể gọi **Response.Redirect()** hoặc **Server.Transfer()** trong trang mã lệnh của bạn. Nhưng trong ứng dụng web chuyên nghiệp, các yêu cầu chuyển hướng chuyên sâu hơn. Các ứng dụng này cần một hệ thống cho phép người dùng lướt web thông qua một hệ thống các trang, mà không buộc bạn viết đoạn code chuyển hướng cùng buồn tẻ trong mỗi trang.

ASP.NET cung cấp một mô hình chuyển hướng dễ dàng để cho phép người dùng lướt web thông qua các ứng dụng web của bạn. Trước khi bạn có thể sử dụng mô hình này, bạn cần phải xác định hệ thống cấp bậc của trang web của bạn, nói cách khác, làm thế nào các trang được tổ chức thành các nhóm một cách logic. Bạn xác định rằng cấu trúc trong một tập tin chuyên dụng và kết nối thông tin đó để chuyển hướng với các điều khiển menu bao gồm **TreeView** và **Menu**.

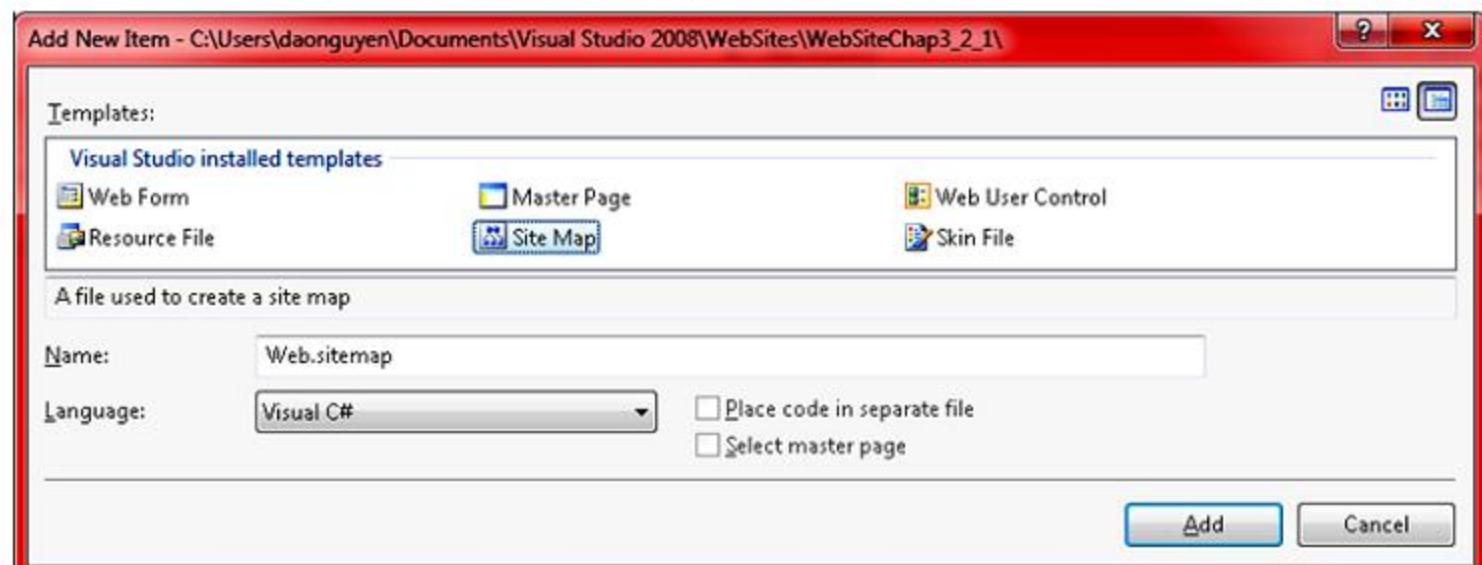


Hình 3.12: ASP.NET navigation với SiteMap

3.2.1. Site Maps

Thực hiện việc tạo một Website **SimpleSiteMap**:

Trong đó để tạo trang **Web.sitemap**, bạn R-Click tại tên Project trong cửa sổ **Solution Explorer**, Click mục **Add New Item..**, xuất hiện hộp thoại **Add New Item**, chọn mục **Site Map** trong hình và click nút **Add**:



Hình 3.13: Tạo trang Web.sitemap với mục chọn SiteMap

Nội dung trang **Web.sitemap** ban đầu như sau:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
    <siteMapNode url="" title="" description="">
        <siteMapNode url="" title="" description="" />
        <siteMapNode url="" title="" description="" />
    </siteMapNode>
</siteMap>
```

Các nguyên tắc tạo trang SiteMap:

- **Nguyên tắc 1:** Một trang SiteMap bắt đầu bằng thẻ **<siteMap>**

```
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
    .....
</siteMap>
```

- **Nguyên tắc 2:** Mỗi trang web trong SiteMap được khai báo trong thẻ **<siteMapNode>**

```
<siteMapNode title="RevoStock"
    description="Investment software for stock charting"
    url="~/product1.aspx" />
```

- **Nguyên tắc 3:** Một siteMapNode có thể chứa các siteMapNode khác.

```
<siteMapNode title="Products" description="Learn about our products"
    url="~/products.aspx">
    <siteMapNode title="RevoStock"
        description="Investment software for stock charting"
        url="~/product1.aspx" />
    <siteMapNode title="RevoAnalyze"
        description="Investment software for yield analysis"
        url="~/product2.aspx" />
</siteMapNode>
```

- **Nguyên tắc 4:** Lặp lại địa chỉ URL trong các siteMapNode là không cho phép.

Các bạn thiết kế trang sitemap với nội dung sau:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
    <siteMapNode title="Home" description="Home" url="~/default.aspx">

        <siteMapNode title="Information" description="Learn about our company">
            <siteMapNode title="About Us" description="How RevoTech was founded"
                url="~/aboutus.aspx" />
            <siteMapNode title="Investing"
                description="Financial reports and investor analysis" />
        </siteMapNode>
    </siteMapNode>
</siteMap>
```

```

    url="~/financial.aspx" />
</siteMapNode>
<siteMapNode title="Products" description="Learn about our products"
    url="~/products.aspx">
    <siteMapNode title="RevoStock"
        description="Investment software for stock charting"
        url="~/product1.aspx" />
    <siteMapNode title="RevoAnalyze"
        description="Investment software for yield analysis"
        url="~/product2.aspx" />
</siteMapNode>

</siteMapNode>
</siteMap>

```

Tạo tiếp trang **MasterPage.master** có nội dung sau:

```

<%@ Master Language="C#" AutoEventWireup="true"
CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Navigation Test</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <table>
            <tr>
                <td style="width: 228px; vertical-align: top;">
                    <asp:SiteMapPath ID="SiteMapPath1" runat="server" Width="264px" Font-
Size="10pt">
                        <PathSeparatorTemplate>
                            <asp:Image ID="Image1" ImageUrl="~/arrowright.gif" runat="server" />
                        </PathSeparatorTemplate>
                        <RootNodeTemplate>

```

```

<b>Root</b>
</RootNodeTemplate>
<CurrentNodeTemplate>
    <%# Eval("title") %> <br /><small><i>&nbsp;<%# Eval("description") %></i></small>
</CurrentNodeTemplate>
</asp:SiteMapPath>
<br />
<asp:TreeView ID="TreeView1" runat="server"
DataSourceID="SiteMapDataSource1" ImageSet="Faq" NodeIndent="0" >
    <ParentNodeStyle Font-Bold="False" />
    <HoverNodeStyle Font-Underline="True" ForeColor="Purple" />
    <SelectedNodeStyle Font-Underline="True" />
    <NodeStyle Font-Names="Tahoma" Font-Size="8pt" ForeColor="DarkBlue" HorizontalPadding="5px" 
NodeSpacing="0px" VerticalPadding="0px" />
</asp:TreeView>
</td>
<td style="vertical-align: top; padding: 10px; width: 301px;" bgcolor="LightGoldenrodYellow">
    <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server" />
</td>
</tr>
</table>
<asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server" />
</form>
</body>
</html>

```

Phần thiết kế trang **MasterPage** như sau:

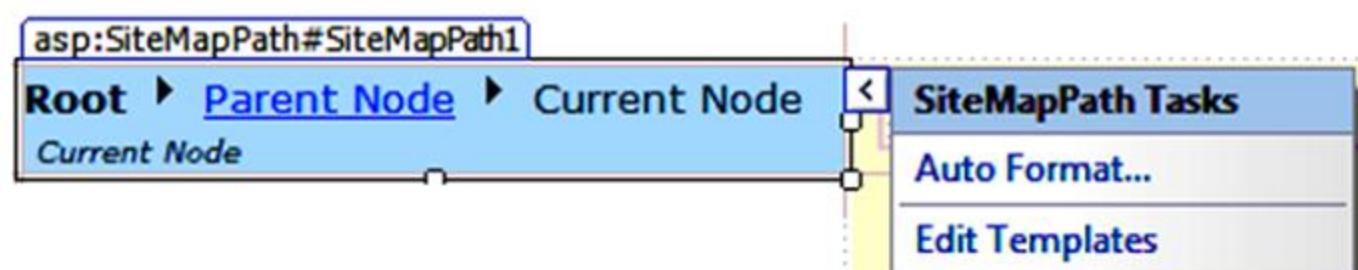
Điều khiển SitemapDataSource:

```
<asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server" />
```

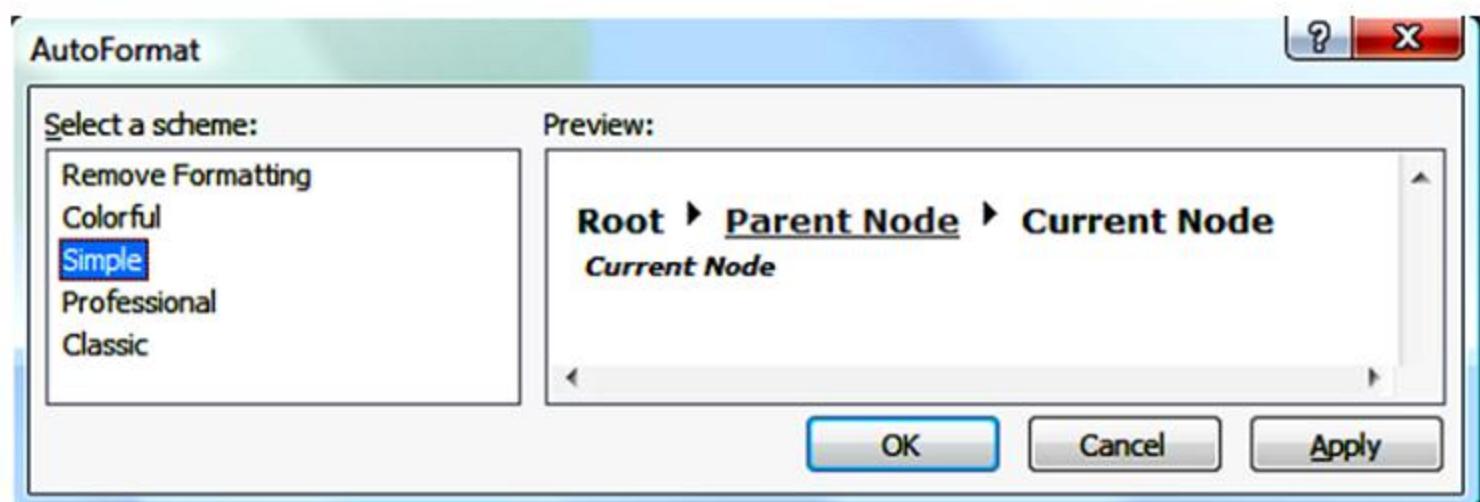


Điều khiển SiteMapPath:

```
<asp:SiteMapPath ID="SiteMapPath1" runat="server" Width="264px" Font-Size="10pt">  
    <PathSeparatorTemplate>  
        <asp:Image ID="Image1" ImageUrl="~/arrowright.gif" runat="server" />  
    </PathSeparatorTemplate>  
    <RootNodeTemplate>  
        <b>Root</b>  
    </RootNodeTemplate>  
    <CurrentNodeTemplate>  
        <%# Eval("title") %> <br /><small><i>&ampnbsp<%# Eval("description") %></i></small>  
    </CurrentNodeTemplate>  
</asp:SiteMapPath>
```



Chọn mục **Auto Format..**, chọn một lựa chọn scheme thích hợp trong danh sách và click **OK**.



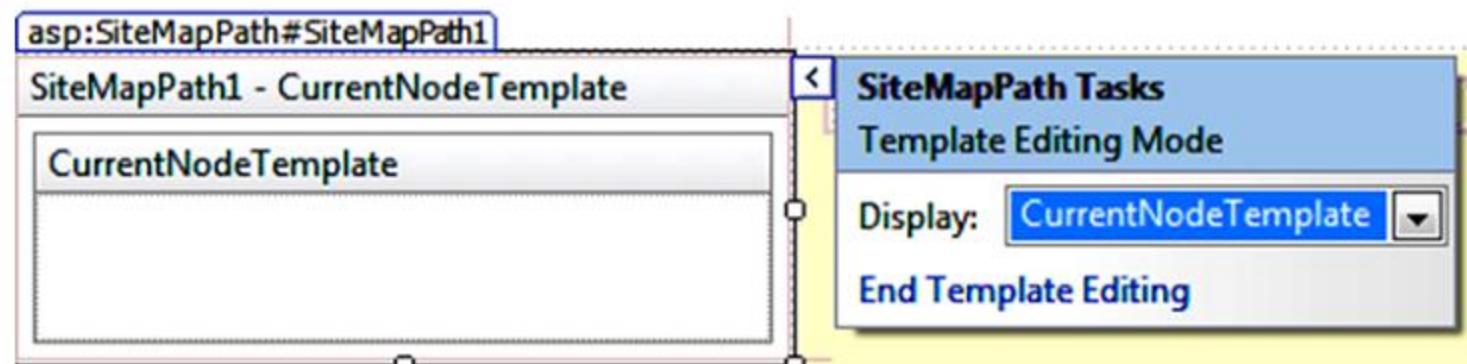
Hình 3.14: Thay đổi định dạng của SiteMappath

Sau đó, bạn chọn mục **Edit Templates** để thiết kế nội dung cho SiteMapPath: có bốn thành phần thẻ bên trong gồm **RootNodeTemplate**, **NodeTemplate**, **CurrentNodeTemplate**, và **PathSeparatorTemplate**.

Nhập nội dung thẻ **Current NodeTemplate** để hiển thị nội dung thẻ **title** và thẻ **description**. Trong đó, **<%# Eval("title") %>** là một cách viết

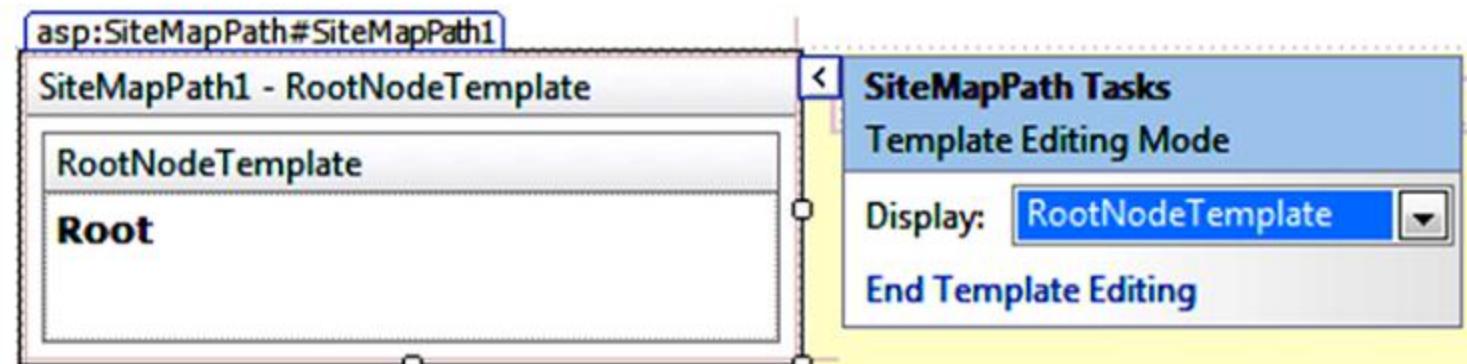
theo dạng kết nối dữ liệu đơn giản (simple data binding) sẽ được học trong bài học sau, dùng để lấy nội dung **title**.

```
<CurrentNodeTemplate>  
<%# Eval("title") %> <br /><small><i>&nbsp;<%# Eval("description")  
%></i></small>  
</CurrentNodeTemplate>
```

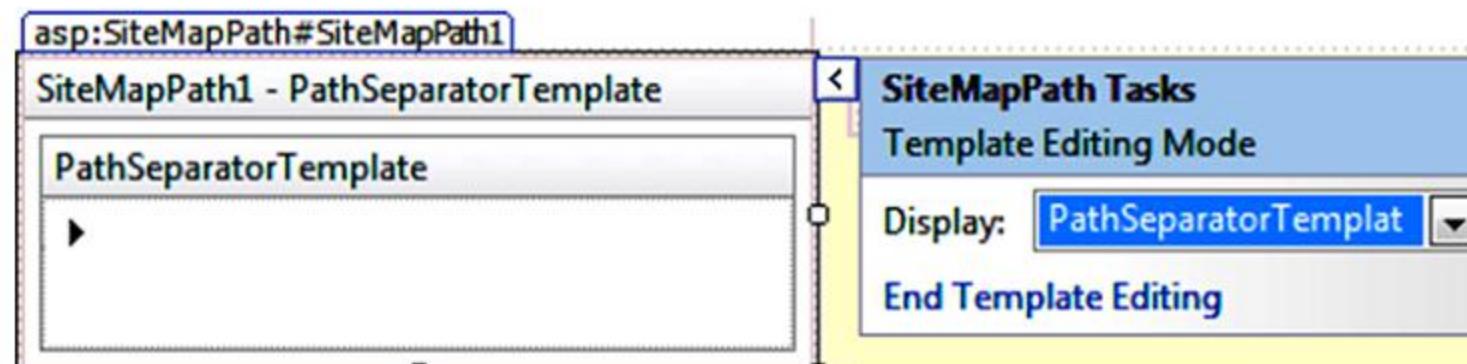


Nhập nội dung thẻ **RootNodeTemplate**: Hiển thị chữ **Root** in đậm

```
<RootNodeTemplate> <b>Root</b> </RootNodeTemplate>
```



```
<PathSeparatorTemplate>  
    <asp:Image ID="Image1" ImageUrl="~/arrowright.gif" runat="server" />  
</PathSeparatorTemplate>
```

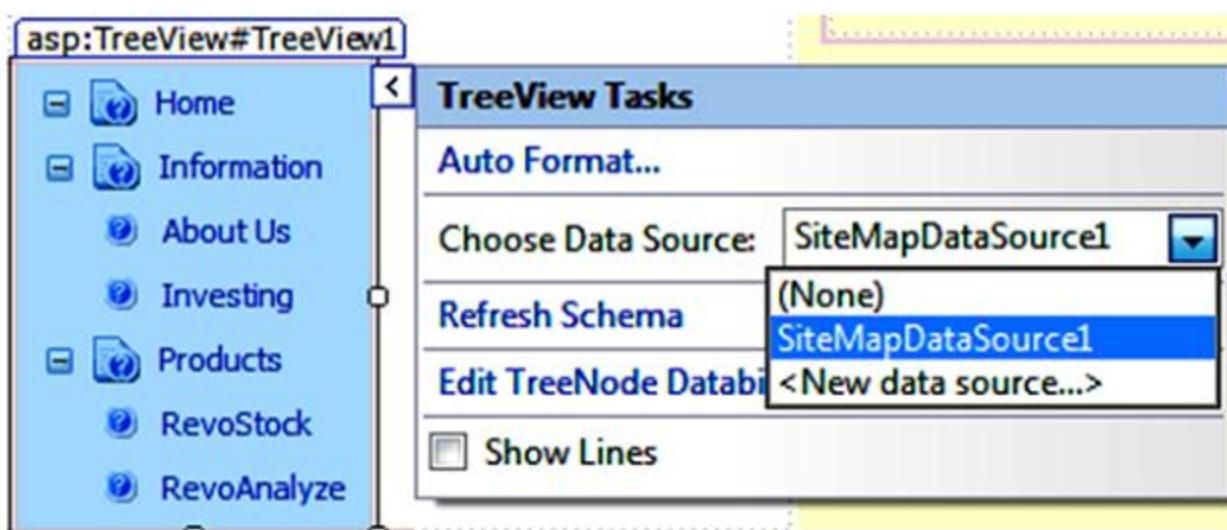


Kết quả: khi chọn mục Investing, nội dung SiteMappath hiện ra như sau:

Root ▶ Information ▶ Investing
Financial reports and investor analysis

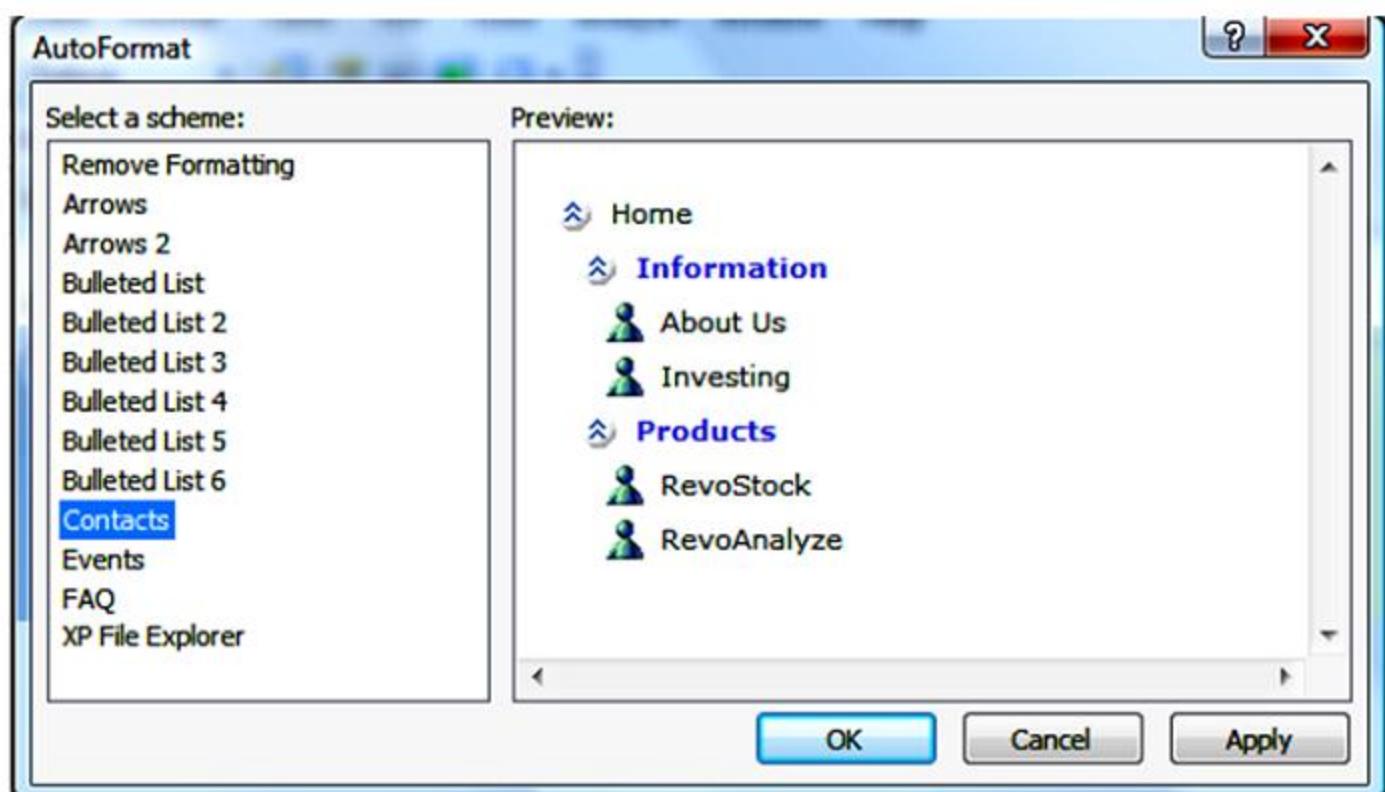
Hình 3.15: Nội dung SiteMapPath hiển thị theo Web.sitemap

Điều khiển TreeView:



Đầu tiên chọn thuộc tính **DataSource** của **TreeView** chỉ đến **SiteMapDataSource1**, sau đó click vào mục **Refresh Schema**, nội dung **TreeView** sẽ được cập nhật theo **SiteMapDataSource1**.

Click vào mục **Auto Format..** để định dạng hiển thị cho **TreeView**, click nút **Apply** để xác nhận việc thay đổi dạng hiển thị:



Hình 3.16: Định dạng hiển thị cho TreeView

Nội dung **TreeView** sau khi cấu hình:

```
<asp:TreeView ID="TreeView1" runat="server">  
    <DataSourceID>SiteMapDataSource1</DataSourceID>  
    <ParentNodeStyle Font-Bold="False" />  
    <HoverNodeStyle Font-Underline="True" ForeColor="Purple" />  
    <SelectedNodeStyle Font-Underline="True" />  
    <NodeStyle Font-Names="Tahoma" Font-Size="8pt" ForeColor="DarkBlue"  
        HorizontalPadding="5px" />
```

```
NodeSpacing="0px" VerticalPadding="0px" />
```

```
</asp:TreeView>
```

Tạo một trang StyleSheet.css để định dạng hiển thị cho các thẻ HTML:

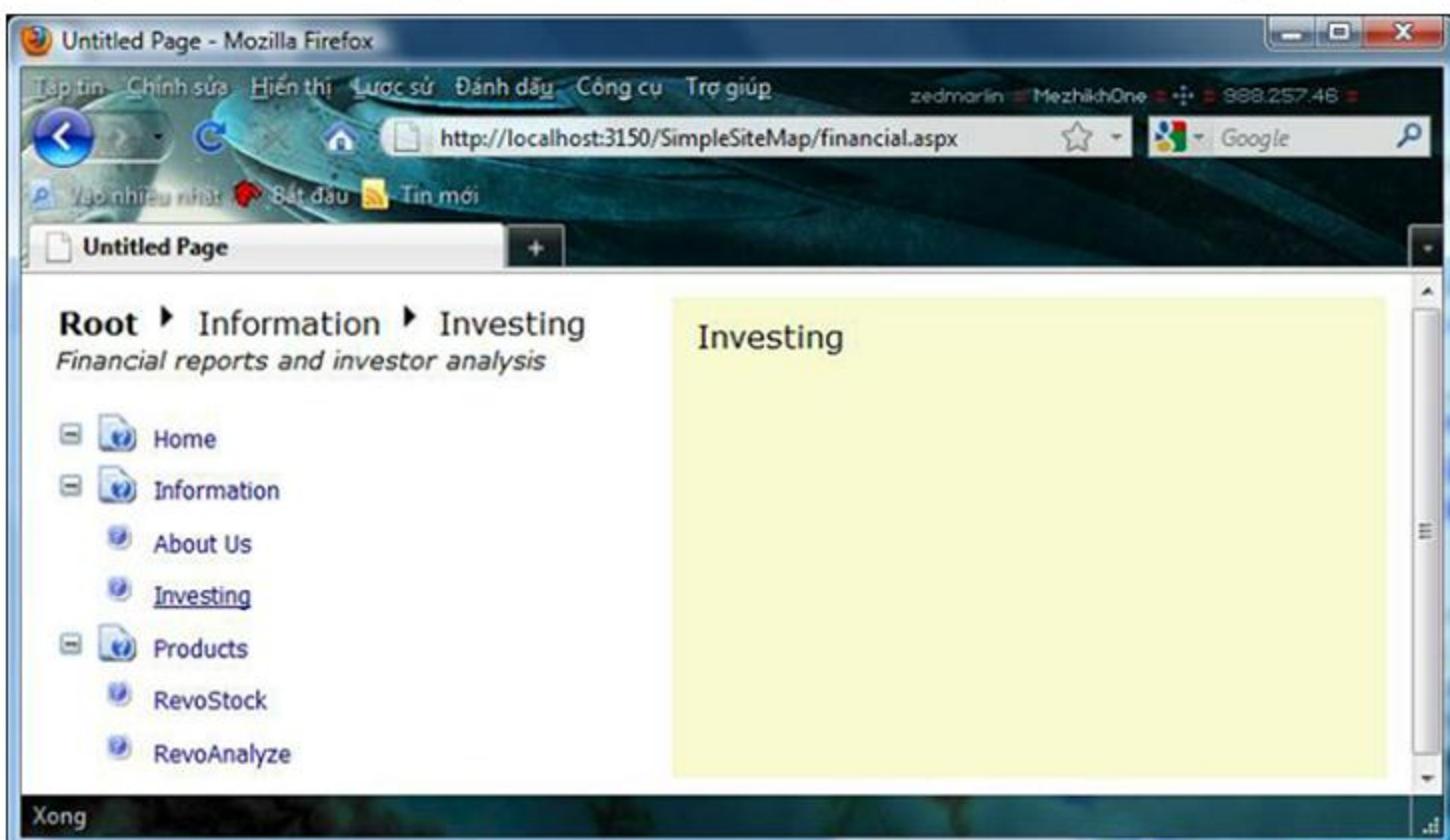
```
body {  
    font-family: Verdana;  
    font-size: 83%;}  
  
div.Box {  
    padding: 5px;  
    border-width: 2px;  
    border-style: ridge;  
    background-color: Lime;  
}
```

Các trang thành phần trong Web.sitemap, chỉ thiết kế với nội dung đơn giản minh họa cho bài tập và sẽ được gọi khi được lựa chọn trên TreeView theo sự điều hướng của Web.sitemap

Trang financial.aspx

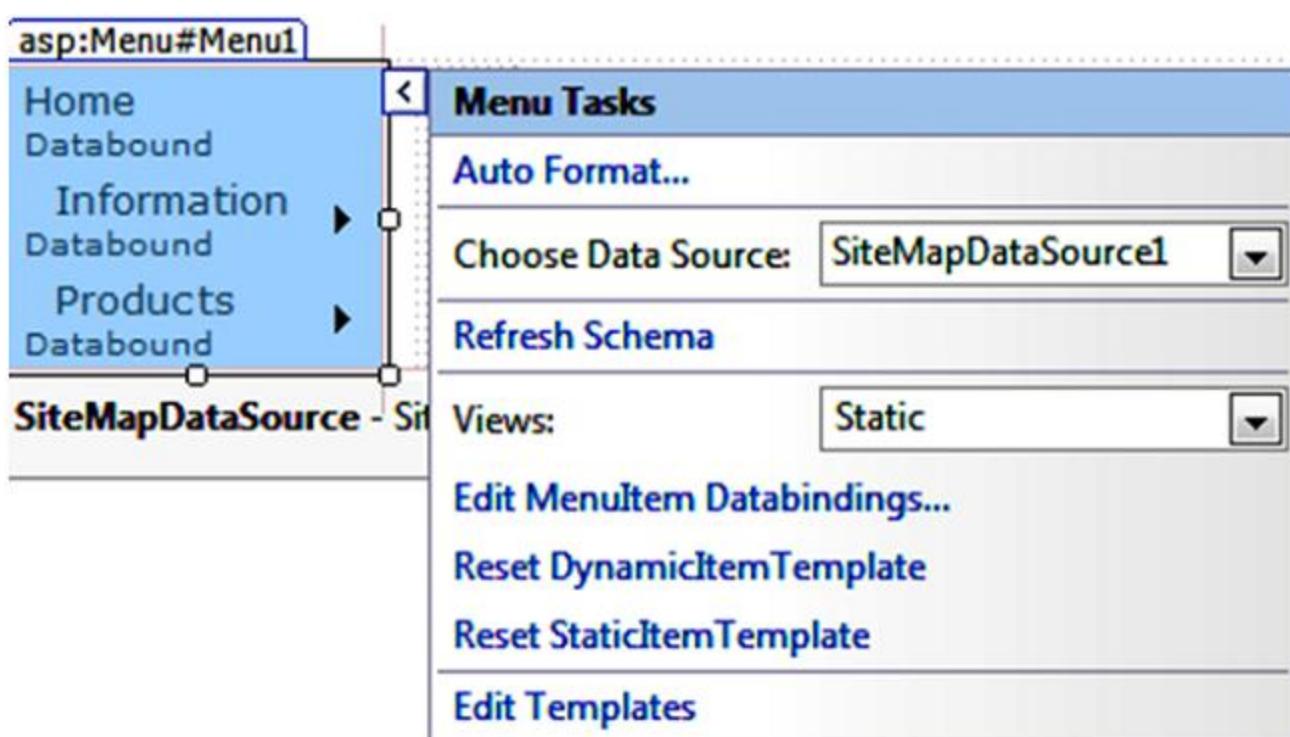
```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master"  
AutoEventWireup="true" CodeFile="financial.aspx.cs" Inherits="financial"  
Title="Untitled Page" %>  
  
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"  
Runat="Server"> Investing </asp:Content>
```

Kết quả chọn Investing trên TreeView sẽ gọi trang financial.aspx:



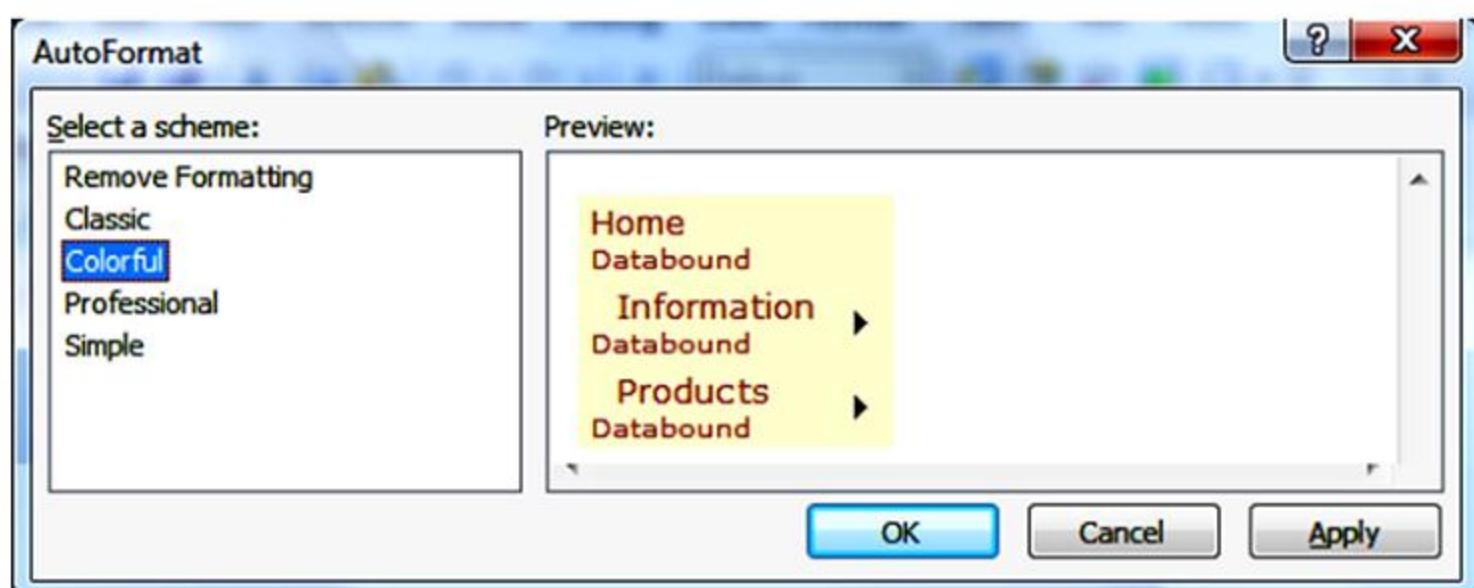
Hình 3.17: Chọn mục Investing trên TreeView

Tương tự với thí dụ trên được sử dụng Menu trên MasterPage có điều hướng theo Web.sitemap



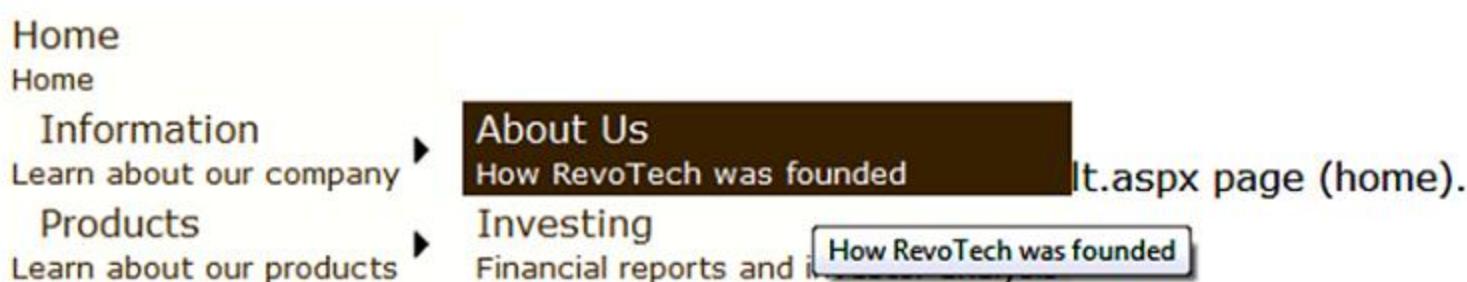
Hình 3.18: Menu sử dụng theo điều hướng của Web.sitemap

Chọn **Auto Format..** để định dạng hiển thị cho Menu, click nút OK để xác nhận việc thay đổi.



Hình 3.19: Định dạng hiển thị cho Menu

Trang **MasterPage.master** có chứa thẻ **<asp:Menu..>** được thiết kế theo hình dưới đây:



Hình 3.20: Menu theo điều hướng của Web.sitemap

Nội dung trang MasterPage.master như sau:

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Navigation Test</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
<form id="form1" runat="server">
    <table>
        <tr>
            <td style="width: 131px; vertical-align: top;">
                <asp:Menu ID="Menu1" runat="server" DataSourceID="SiteMapDataSource1"
BackColor="#F7F6F3"
DynamicHorizontalOffset="2" Font-Names="Verdana" ForeColor="#7C6F57"
StaticDisplayLevels="2" StaticSubMenuIndent="10px">
                    <StaticMenuItemStyle HorizontalPadding="5px" VerticalPadding="2px" />
                    <DynamicHoverStyle BackColor="#7C6F57" ForeColor="White" />
                    <DynamicMenuStyle BackColor="#F7F6F3" />
                    <StaticSelectedStyle BackColor="#F7F6F3" />
                    <DynamicSelectedStyle BackColor="#5D7B9D" />
                    <DynamicMenuItemStyle HorizontalPadding="5px" VerticalPadding="2px" />
                    <StaticHoverStyle BackColor="#7C6F57" ForeColor="White" />

                    <StaticItemTemplate>
                        <%# Eval("Text") %><br />
                        <small>
                            <%# GetDescriptionFromTitle(((MenuItem)Container.DataItem).Text)
%>
                        </small>
                    </StaticItemTemplate>
                    <DynamicItemTemplate>
```

```

<%# Eval("Text") %><br />
<small>
    <%# GetDescriptionFromTitle(((MenuItem)Container.DataItem).Text)>
<%>
</small>
</DynamicItemTemplate>
</asp:Menu>
</td>
<td style="vertical-align: top; padding: 10px" >
    <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server" />
</td>
</tr>
</table>
<asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server" />
</form>
</body>
</html>

```

Phần mã lệnh của trang MasterPage.master cung cấp phương thức **GetDescriptionFromTitle(string Title)** hiện thông tin thẻ chọn trên menu.

```

public partial class MasterPage : System.Web.UI.MasterPage
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected string GetDescriptionFromTitle(string title)
    {
        // This assumes there's only one node with this title.
        SiteMapNode startingNode = SiteMap.RootNode;
        SiteMapNode matchNode = SearchNodes(startingNode, title);
        if (matchNode == null)
        {
            return null;
        }
        else
    }
}

```

```

    {
        return matchNode.Description;
    }
}

private SiteMapNode SearchNodes(SiteMapNode node, string title)
{
    if (node.Title == title)
    {
        return node;
    }
    else
    {
        // Perform recursive search.
        foreach (SiteMapNode child in node.ChildNodes)
        {
            SiteMapNode matchNode = SearchNodes(child, title);
            // Was a match found?
            // If so, return it.
            if (matchNode != null) return matchNode;
        }
        // All the nodes were examined, but no match was found.
        return null;
    }
}
}

```

3.3. WEB USER CONTROLS

3.3.1. Giới thiệu User Custom Control

MS Visual Studio.NET cung cấp rất nhiều các điều khiển để phát triển ứng dụng gọi là điều khiển nội tại (Instrict control). Ngoài ra, nó còn cung cấp cho chúng ta khả năng tự xây dựng các điều khiển tùy biến, nếu các điều khiển hiện hành không đáp ứng được yêu cầu thiết kế trang web với những chức năng riêng biệt.

Thí dụ: Nếu ứng dụng của bạn cần chiếc máy tính (Calculator) ở rất nhiều trang thì giải pháp tốt nhất là nên tạo một điều khiển Calculator riêng thay việc kết hợp các điều khiển truyền thống, khi đó ta có thể sử dụng điều khiển này trong toàn bộ ứng dụng.

Đề mục này sẽ hướng dẫn cách tạo và sử dụng điều khiển do người dùng tự xây dựng – hay còn gọi là điều khiển tùy biến (Web User Controls).

Thực chất của User Control (UC) chính là một "trang con", trong đó có thể chứa **bất kỳ nội dung nào** (trừ các thẻ <HTML> <BODY>, <FORM>, vì một trang chỉ có duy nhất một lần xuất hiện các thẻ này). "Trang con" này sau đó có thể được khai báo vào các trang khác để sử dụng. Khi muốn cập nhật nội dung ở tất cả các trang, ta chỉ việc sửa đổi duy nhất UC ban đầu. Khả năng này của ASP.NET giúp chúng ta xây dựng ứng dụng nhanh hơn, dễ bảo trì hơn.

Mỗi một UC được đặt trong một trang có phần mở rộng là *.ascx. File này có đặc điểm là không truy cập trực tiếp từ trình duyệt mà chỉ được chèn vào các trang aspx. Nội dung trang User Control được khai báo với thẻ <%@ Control..%> như sau:

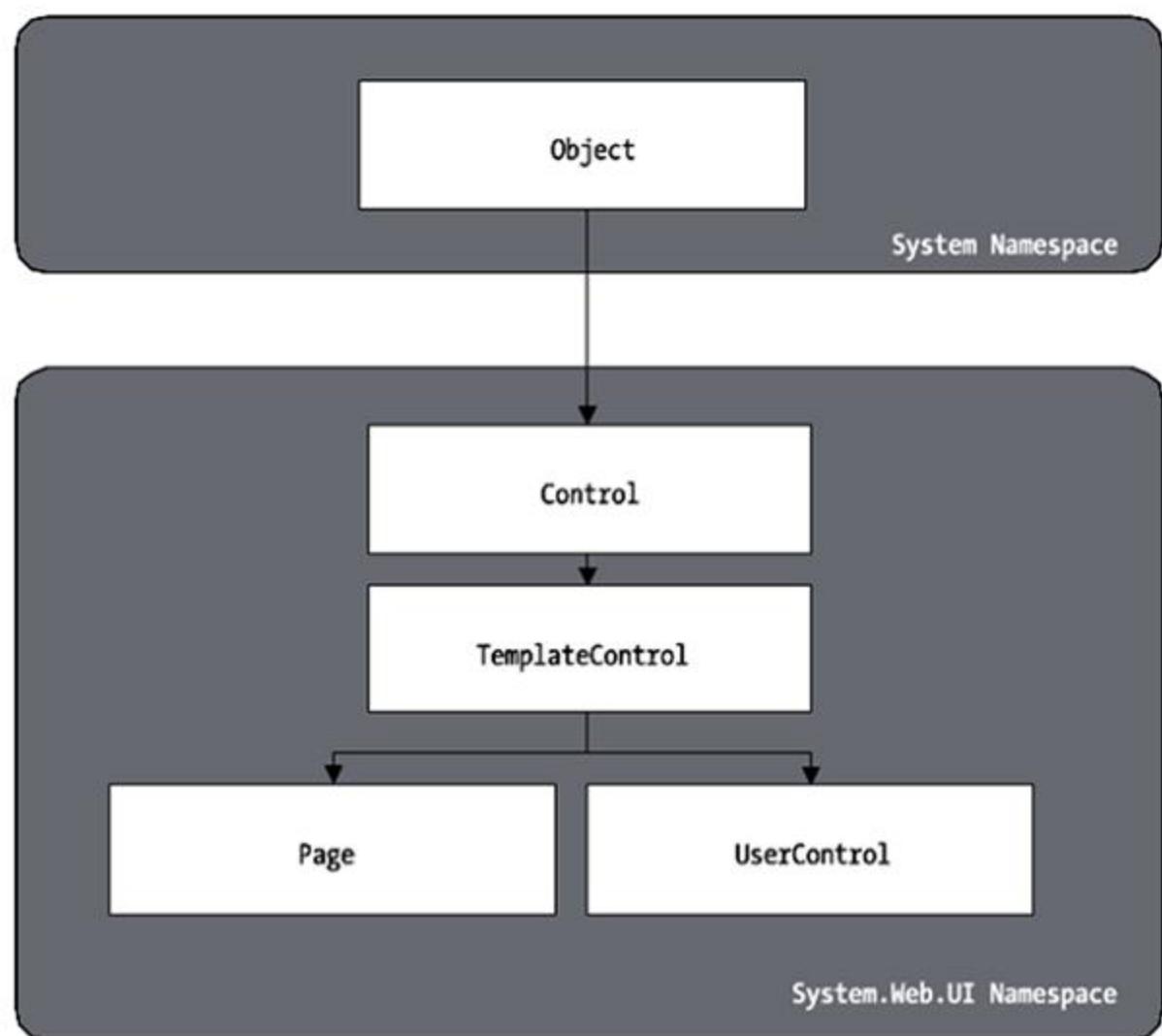
```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="WebUserControl.ascx.cs" Inherits=" WebUserControl " %>
```

Phần mã lệnh trang WebUserControl.aspx.cs:

```
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.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class WebUserControl: System.Web.UI.UserControl { }
```

Các lớp User Control kế thừa từ **System.Web.UI.UserControl**, hình 3.21 cho thấy sự thừa kế của hai đối tượng Page (trang.aspx) và User Control (trang.ascx).

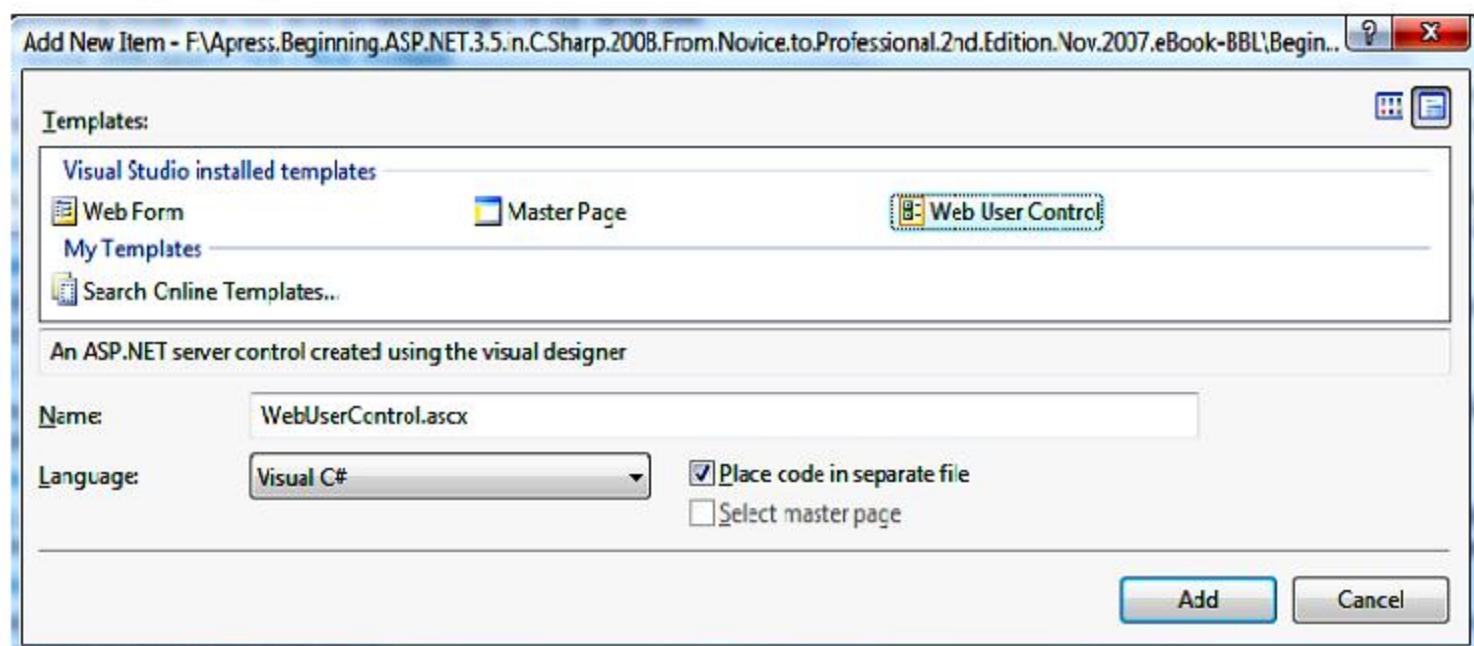


Hình 3.21: Cây thừa kế của hai đối tượng Page và User Control

3.3.2. Các bước tạo User Custom control

Gồm ba bước sau:

Bước 1: R-Click tại ứng dụng web trong cửa sổ **Solution Explorer** và chọn **Add New Item..**, xuất hiện hộp thoại, bạn chọn **Web User Control**, nhập tên User Control vào khung **Name**, thí dụ tên **CalculatorControl.ascx**, click nút **Add**, trang User Control sẽ được thêm vào ứng dụng.



Hình 3.22: Menu theo điều hướng của Web.sitemap

Bước 2: Soạn nội dung của trang.

Bước 3: Lưu lại nội dung của trang.

3.3.3. Các loại User Control

Có hai loại User Control: User Control hoạt động độc lập và tích hợp. User Control độc lập (**Independent User Control**) là loại điều khiển không tương tác với phần còn lại của các mã lệnh trên trang web và ngược lại nếu có sự tương tác sẽ là User Control tích hợp (**Integrated User Control**).

Thí dụ 1: Tạo User Control độc lập đơn giản.

Tạo một trang Footer. Ascx, với nội dung hiển thị sau:



Bước 1:

Tạo trang Footer.ascx và thiết kế nội dung trang như sau

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="Footer.ascx.cs"
Inherits="Footer" %>

<div id="footer" class="clearfix">
    <p class="col1 style2">
        <strong>TRUNG TÂM ĐÀO TẠO MẠNG MÁY TÍNH NHẤT NGHỆ
NGHỆ</strong>
        <br/>
        <strong>Địa chỉ:</strong>
        <b>105 Bà Huyện Thanh Quan, Quận 3, TP. HCM</b>
        <br/>
        <strong>Điện thoại:</strong><span class="style3">(08) 3 9322 735 - (08) 3
9322734</span><br>
        <strong>Email:</strong>
    </p>
</div>
```

```

<a href="mailto:info@nhatnghe.com">info@nhatnghe.com</a>
<br/>
<strong>Website:</strong>
<a href="http://www.nhatnghe.com">http://www.nhatnghe.com</a></p>
</div>
<asp:Label id="lblFooter" runat="server" />
```

Điều khiển nhãn **lblFooter** sẽ hiển thị nội dung ngày hoặc giờ khi trang web có sử dụng User Control **Footer** trên được triệu gọi. Thiết kế nội dung mã lệnh cho trang như sau:

```

public partial class Footer: System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {
        lblFooter.Text = "<br/> Trang này được triệu gọi lúc ";
        if (format == FooterFormat.LongDate)
        {
            lblFooter.Text += DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
        }
        else if (format == FooterFormat.ShortTime)
        {
            lblFooter.Text += DateTime.Now.ToString("HH:mm:ss");
        }
    }

    public enum FooterFormat
    {
        LongDate, ShortTime
    }

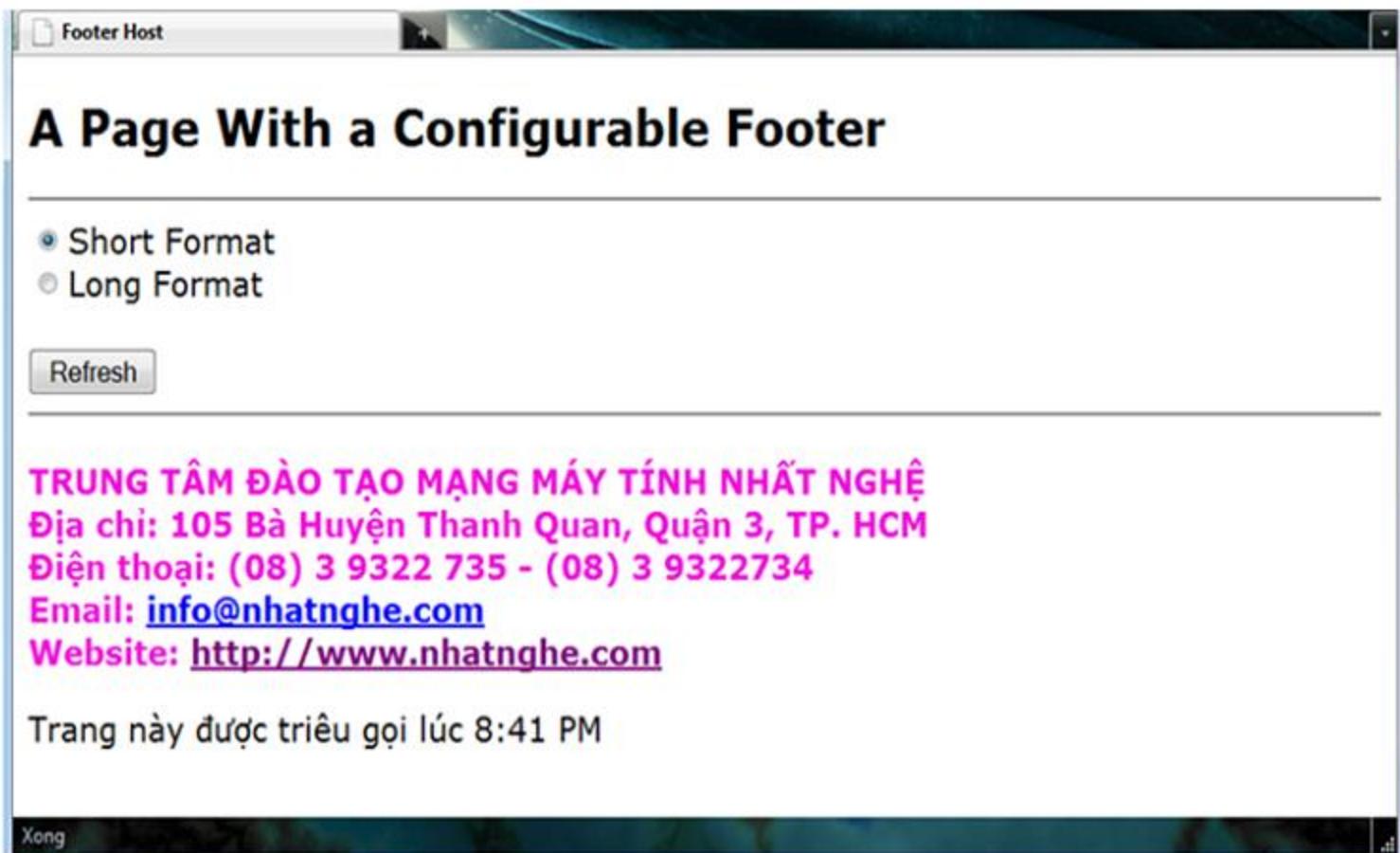
    private FooterFormat format = FooterFormat.LongDate;
    public FooterFormat Format
    {
        get { return format; }
        set { format = value; }
    }
}
```

Tạo trang **FooterHost.aspx** có sử dụng User Control- **Footer** trên bằng cách khai báo thẻ **<%@ Register.. %>** có dạng sau:

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

<%@ Register TagPrefix="uc" TagName="Footer" Src="Footer.ascx" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Footer Host</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <h2>A Page With a Configurable Footer</h2>
            <br /><br /><br />
            <hr />
            <asp:RadioButton id="optShort" runat="server"
GroupName="Format" Text="Short Format"></asp:RadioButton>
            <br />
            <asp:RadioButton id="optLong" runat="server"
GroupName="Format" Text="Long Format"></asp:RadioButton>
            <br /><br />
            <asp:Button id="cmdRefresh" runat="server"
Text="Refresh"></asp:Button>
            <hr />
            <br />
            <uc:Footer id="Footer1" runat="server"></uc:Footer>
        </div>
    </form>
</body>
</html>
```

Kết quả thực thi:



Thí dụ 2: Tạo User Control tích hợp

Tạo một User Control với tên LinkMenu.ascx có nội dung thiết kế như sau:

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="LinkMenu2.ascx.cs" Inherits="LinkMenu2" %>
<div style="border-right: 2px groove; padding-right: 5px; border-top: 2px groove;
padding-left: 5px; font-weight: bold; font-size: small; padding-bottom: 5px;
border-left: 2px groove; width: 100px; padding-top: 5px; border-bottom: 2px
groove;
font-family: Verdana; height: 120px; background-color: #FFFFD9">
Products:
<asp:LinkButton ID="lnkBooks" runat="server"
CommandArgument="Menu2Host.aspx?product=Books"
OnCommand="lnk_Command" >Books
</asp:LinkButton><br />
<asp:LinkButton ID="lnkToys" runat="server"
CommandArgument="Menu2Host.aspx?product=Toys"
OnCommand="lnk_Command">Toys
</asp:LinkButton><br />
<asp:LinkButton ID="lnkSports" runat="server"
CommandArgument="Menu2Host.aspx?product=Sports"
OnCommand="lnk_Command">Sports
</asp:LinkButton><br />
```

```
<asp:LinkButton ID="lnkFurniture" runat="server"
CommandArgument="Menu2Host.aspx?product=Furniture"
OnCommand="lnk_Command">Furniture
</asp:LinkButton>
</div>
```



Khi click tại LinkButton **Books** có khai báo các thuộc tính của điều khiển **lnkBooks** với **OnCommand="lnk_Command"** và **CommandArgument="Menu2Host.aspx?product=Books"** cho phép chuyển đến trang Menu2Host.aspx với chuỗi (query string) nội dung là **product=Books** tương tự lệnh **Response.Redirect()**.

Products:

```
<asp:LinkButton ID="lnkBooks" runat="server"
CommandArgument="Menu2Host.aspx?product=Books"
OnCommand="lnk_Command" >Books
</asp:LinkButton><br />
```

Nội dung trang mã lệnh có khai báo phương thức **lnk_Command(object sender, CommandEventArgs e)** có cấu trúc khai báo giống như khai báo hình thức của một hàm ủy thác (delegate):

```
public delegate void LinkClickedEventHandler(object sender,
LinkClickedEventArgs e);
```

đáp ứng sự kiện **LinkClicked** xảy ra khi click trên một LinkButton nào đó trên trang:

```
public event LinkClickedEventHandler LinkClicked;
```

Như vậy, User Control **LinkMenu2** đã gắn vào một sự kiện **LinkClicked**, và khi sự kiện được xảy ra trên **LinkButton** thì hàm ủy thác chỉ định thực thi sẽ được triệu gọi:

```
<asp:LinkButton ID="lnkBooks" runat="server"
CommandArgument="Menu2Host.aspx?product=Books"
OnCommand="lnk_Command" >Books
</asp:LinkButton>
```

Nội dung phần mã lệnh trang LinkMenu2.ascx.cs như sau:

```
public partial class LinkMenu2: System.Web.UI.UserControl
{
    public event LinkClickedEventHandler LinkClicked;
    protected void lnk_Command(object sender, CommandEventArgs e)
    {
        // One of the LinkButton controls has been clicked.
        // Raise an event to the page.
        if (LinkClicked != null)
        {
            // Pass along the link information.
            LinkClickedEventArgs args = new
            LinkClickedEventArgs((string)e.CommandArgument);
            LinkClicked(this, args);

            // Perform the redirect.
            if (!args.Cancel)
            {
                // Notice we use the Url from the LinkClickedEventArgs
                // object, not the original link. That means the web page
                // can change the link if desired before the redirect.
                Response.Redirect(args.Url);
            }
        }
    }
}
```

Nội dung trang xây dựng lớp sự kiện **LinkClickedEventArgs**, chứa một property **Url** trong khai báo constructor của lớp.

```

public class LinkClickedEventArgs: EventArgs
{
    private string url;
    public string Url
    {
        get { return url; }
        set { url = value; }
    }

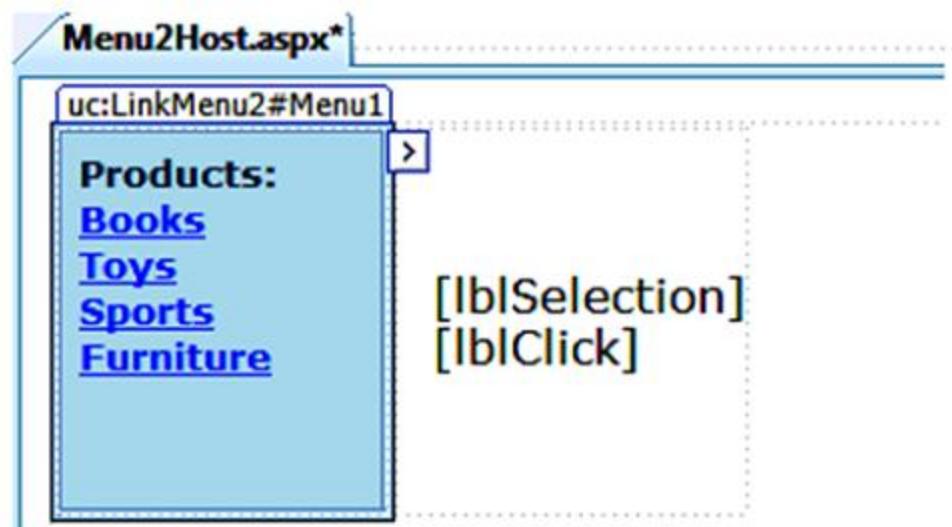
    private bool cancel = false;
    public bool Cancel
    {
        get { return cancel; }
        set { cancel = value; }
    }

    public LinkClickedEventArgs(string url)
    {
        Url = url;
    }

    public delegate void LinkClickedEventHandler(object sender,
        LinkClickedEventArgs e);
}

```

Tạo trang **Menu2Host.aspx** có sử dụng User Control **LinkMenu2** vừa thiết kế trên:



Nội dung trang như sau:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Menu2Host.aspx.cs" Inherits="Menu2Host"%>
<%@ Register TagPrefix="uc" TagName="LinkMenu2" Src="LinkMenu2.ascx" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

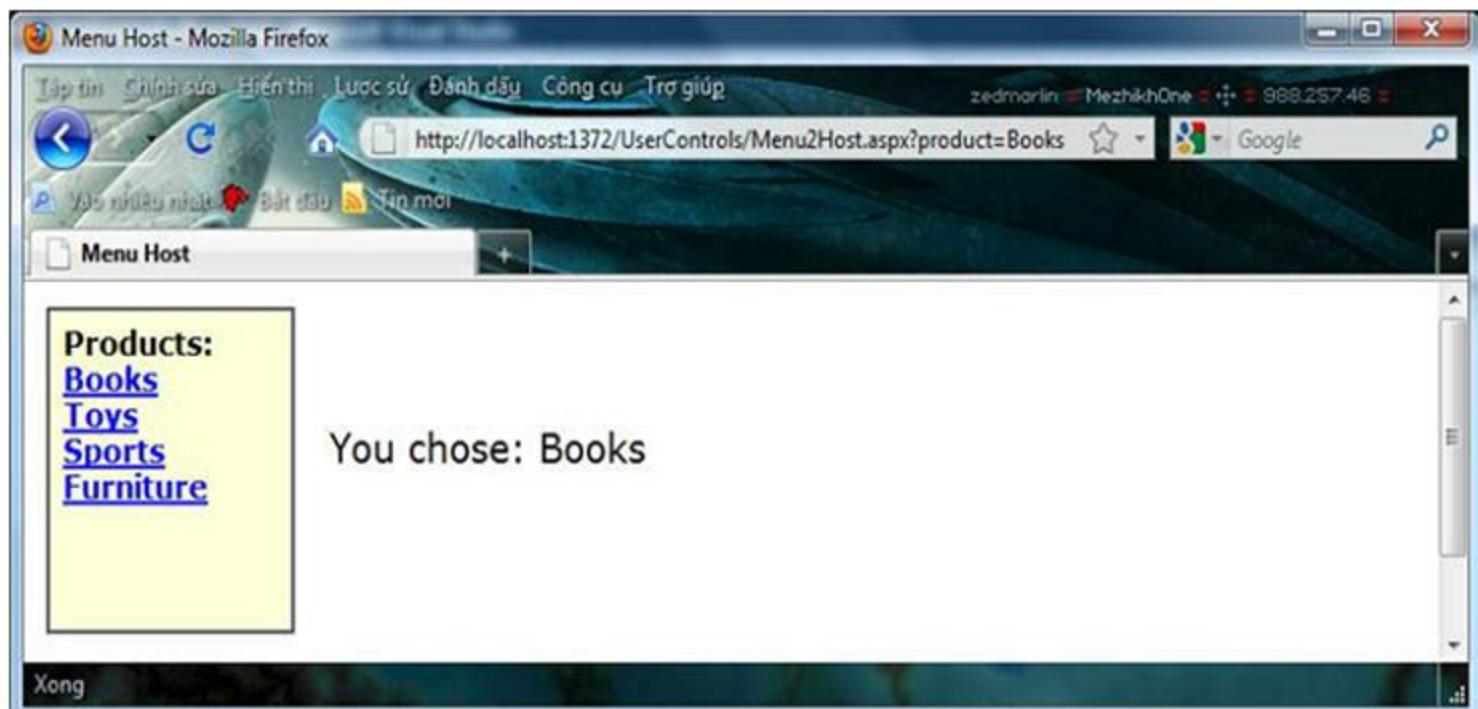
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Menu Host</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <table>
                <tr>
                    <td style="width: 100px">
                        <uc:LinkMenu2 id="Menu1" runat="server"
OnLinkClicked="LinkClicked" />
                    </td>
                    <td>
                        &nbsp;&nbsp;<asp:Label id="lblSelection" runat="server"
EnableViewState="false"/><br />
                        &nbsp;&nbsp;<asp:Label id="lblClick" runat="server"
EnableViewState="false"/>
                    </td>
                </tr>
            </table>
            <br />
            &nbsp;
        </div>
    </form>
</body>
</html>
```

Nội dung trang mã lệnh **Menu2Host.aspx.cs** có viết hàm ủy thác **LinkClicked(object sender, LinkClickedEventArgs e)** được gọi khi sự kiện **LinkClicked** xảy ra trên **uc:LinkMenu2**:

```
public partial class Menu2Host: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            if (Request.Params["product"] != null)
            {
                lblSelection.Text = "You chose: ";
                lblSelection.Text += Request.Params["product"];
            }
        }
    }

    protected void LinkClicked(object sender, LinkClickedEventArgs e)
    {
        if (e.Url == "Menu2Host.aspx?product=Furniture")
        {
            lblClick.Text = "This link is not allowed.";
            e.Cancel = true;
        }
        else
        {
            // Allow the redirect, and don't make any changes to the URL.
        }
    }
}
```

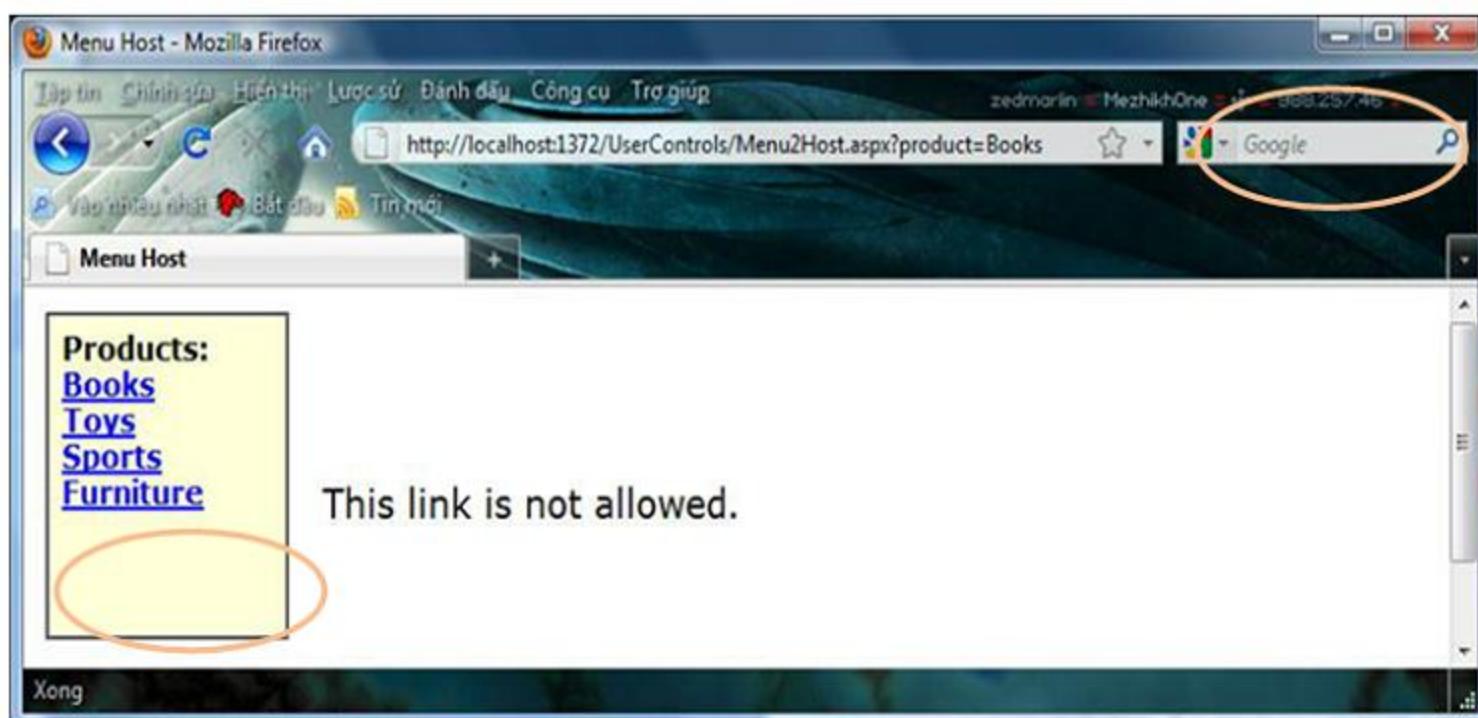
Kết quả thực thi:



Khi click lên LinkButton **Furniture**, thông báo sẽ hiện ra tương ứng nội dung trong hàm ủy thác, lúc này không có thực hiện việc chuyển trang.

```
if (e.Url == "Menu2Host.aspx?product=Furniture")
{
    lblClick.Text = "This link is not allowed.";
    e.Cancel = true;
}
```

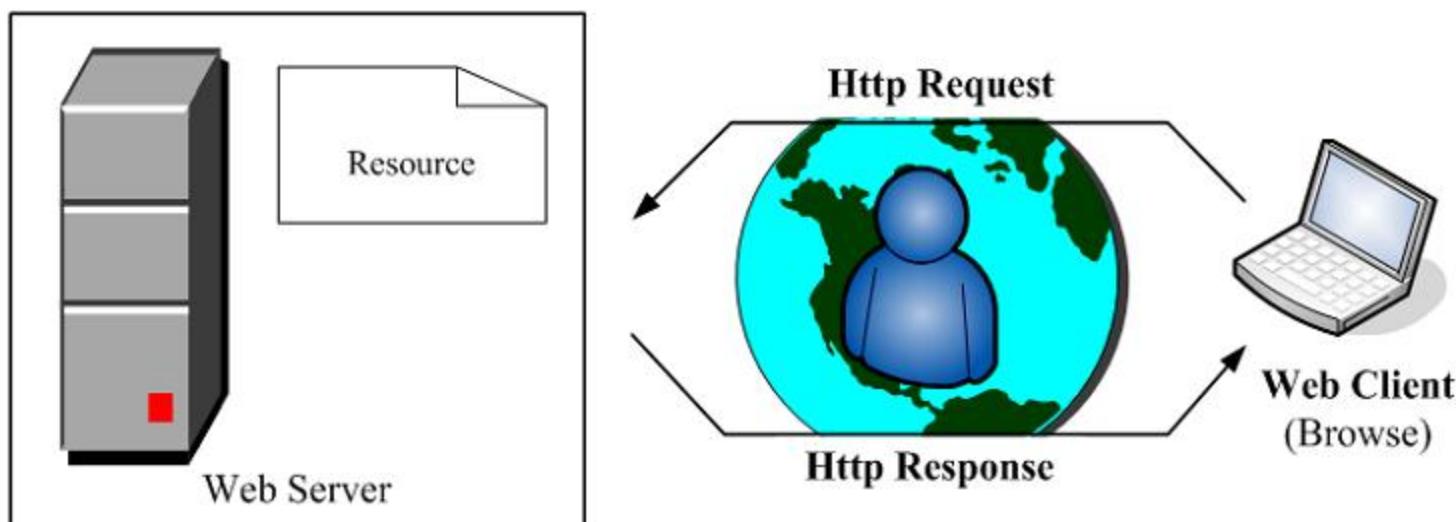
Kết quả:



3.4. CÁC ĐỐI TƯỢNG REQUEST, RESPONSE VÀ SERVER

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 phần này, chúng ta cũng tìm hiểu thêm một đối tượng khác cũng rất hay dùng khi xây dựng ứng dụng là đối tượng Server.



Hình 3.23: Quá trình Request – Response của HTTP

Đối tượng Request

- Đối tượng Request được dùng để nhận những thông tin từ trình duyệt của người dùng gửi về cho Web Server. Những thông tin này gồm các thông số của Form khi được Submit dùng phương thức POST hoặc GET hay các tham số được ghi cùng với trang ASP.NET trong lời gọi đến trang đó.
- Dùng đối tượng Request có thể chia sẻ thông tin qua lại giữa các trang ASP.NET trong một ứng dụng và để lấy giá trị các Cookie lưu trữ trên máy Client.
- Bảng thuộc tính và phương thức của Request

Phương thức / thuộc tính	Điễn giải
AccepTypes	Trả về mảng chuỗi của MIME được hỗ trợ bằng trình khách
ApplicationPath	Trả về đường dẫn ảo của ứng dụng
BinayRead	Trả về mảng Byte chứa đựng thông tin nhị phân gửi đến Server
Browser	Trả về đối tượng <code>HttpBrowserCapabilities</code> trình bày thông tin của trình duyệt
ClientCertificate	Trả về đối tượng <code>HttpClientCertificate</code>
ContentEncoding	Tập ký tự của thực thể Body
ContentLength	Chiều dài tính bằng byte của yêu cầu

ContentType	Loại MILE của yêu cầu
Cookies	Trả về đối tượng HttpCookiesCollection
Filepath	Trả về đường dẫn ảo của yêu cầu
Files	Trả về HttpFileCollection của tập nhiều tập tin được tải lên Server
Form	Trả về tập dữ liệu của nội dung từ Form
Header	Trả về tập dữ liệu của nội dung từ HTTP Header
HttpMethod	Trả về phương thức HTTP sử dụng cho yêu cầu
InputStream	Là luồng dữ liệu chưa đựng các Input của Form
Params	Lấy giá trị của phương thức trong Form, QueryString
Path	Đường dẫn ảo của yêu cầu
PathInfo	Đường dẫn ảo của yêu cầu
PhysicalPath	Đường dẫn vật lý của yêu cầu
QueryString	Trả về một tập dữ liệu của nội dung từ QueryString
RawURL	URL của yêu cầu
RequestType	Phương thức HTTP sử dụng cho Request
TotalByte	Dung lượng của Stream trong luồng dữ liệu
URL	Đối tượng URL chưa đựng chi tiết của yêu cầu
UserHostName	Tên DNS của người sử dụng
MapPath	Chuyển đổi đường dẫn ảo thành đường dẫn vật lý
SaveAs	Lưu yêu cầu HTTP vào đĩa

- Có hai cách để gửi thông tin đặc biệt từ trình duyệt đến Web server, đó là thông tin từ phần <FORM> trong trang được đưa vào HTTP header hay được đưa trực tiếp vào chuỗi truy vấn trong địa chỉ liên kết URL. Đó chính là vai trò của hai tập hợp QueryString và Form của đối tượng Request.
- Tập hợp QueryString:
- Để lấy được giá trị từ chuỗi truy vấn ta dùng:
- Request.QueryString (“tên biến”).
- Khi đó, thuộc tính của <FORM> phải được dùng là METHOD = GET

- Tập hợp Form:
- Để lấy được giá trị từ phần Header của HTTP ta dùng:
Request.QueryString (“tên biến”).
- Khi đó, thuộc tính của <FORM> phải được dùng là METHOD = POST
- So sánh giữa hai tập hợp Form và QueryString:
- Theo phương pháp dùng QueryString có hạn chế đó là giới hạn chiều dài của chuỗi địa chỉ URL (khoảng 1000 ký tự) đây chính là yêu cầu của giao thức HTTP. Do đó sẽ rất phiền phức khi có yêu cầu gửi đi quá dài. Mặc khác, giá trị mà ta gửi đi được hiển thị rõ ràng trong ô địa chỉ URL trên trình duyệt máy Client nên dễ dàng bị người khác đọc được.
- Phương pháp dùng POST đã khắc phục được nhược điểm trên bằng cách đưa dữ liệu vào trong phần Header của HTTP.

Đối tượng Response

- Trong hệ thống các đối tượng xây dựng sẵn của ASP.NET thì đối tượng Response đóng vai trò rất quan trọng. Khi mà đối tượng Request bao gồm những thông tin gửi đến Web server từ trình duyệt thì đối tượng Response nắm giữ những gì mà Web server phải gửi trả lại cho trình duyệt. Tóm lại, ta dùng đối tượng Response để gửi thông tin ra User, gồm có ghi thông tin trực tiếp ra Browser, chuyển Browser đến địa chỉ URL khác hay để thiết lập các Cookie trên máy Client.
- Bảng các thuộc tính và phương thức của đối tượng Response

Thuộc tính / phương thức	Diễn giải
BufferOutput	Có sử dụng hay không bộ nhớ đệm cho kết xuất dữ liệu
Cache	Trả về đối tượng HttpCachePolicy chứa đựng thông tin về quy định Cache của phúc đáp hiện hành
CacheControl	Mặc dù còn hỗ trợ nhưng phương thức này còn đối nghịch trong phương thức của HttpCachePolicy
ContentEncoding	Tập nhận dạng kết xuất, là một trong các giá trị như UnicodeEncoding, UTF7Encoding, UTF8Encoding
Cookies	Trả về một tập của đối tượng HttpCookies
Expires	Mặc dù còn hỗ trợ nhưng phương thức này còn đối nghịch trong phương thức của HttpCachePolicy

ExpiresAbsolute	Mặc dù còn hỗ trợ nhưng phương thức này còn đối nghịch trong phương thức của <code>HttpCachePolicy</code>
Filter	Đối tượng Stream dùng làm bộ lọc dữ liệu kết xuất
Output	Trả về đối tượng <code>TextWriter</code>
OutputStream	Đối tượng Stream dùng để trình bày hàng dữ liệu của body
Status	Gán trạng thái HTTP trả về cho trình khách
StatusCode	Trạng thái HTTP Response
StatusDescription	Gán diễn giải trạng thái HTTP và trả về cho trình khách, thuộc tính này được ưu tiên hơn thuộc tính <code>Status</code>
ClearContent	Xóa nội dung từ Buffer Stream
ClearHeaders	Xóa header từ Buffer Stream
Close	Đóng kết nối với Client
Redirect	Chuyển hướng đến địa chỉ file trong cùng ứng dụng hay URL khác trong lúc thi hành
Writeln	Ghi một luồng dữ liệu ra tập tin chỉ định
Write	Ghi thông tin từ các kiểu dữ liệu như Char, Object, String, Array ra trang web

Thí dụ: Minh họa sử dụng đối tượng Request và Response

Bước 1: Tạo hai trang ASP.NET gồm trang NhapTen.aspx và trang XemChiTiet.aspx.

Bảng mô tả các thuộc tính của các controls trang NhapTen.aspx.

Control	Tên thuộc tính	Giá trị thuộc tính
Lable	Text	Nhập tên
Lable	Text	Ngày Sinh
TextBox	ID	txtTen
TextBox	ID	txtNgaySinh
Button	Text	Xem chi tiết
	ID	btnXemChiTiet

Bảng mô tả các thuộc tính của các controls trang XemChiTiet.aspx.

Control	Tên thuộc tính	Giá trị thuộc tính
Lable	ID	lblXemChiTiet

Bước 2: Viết lệnh xử lý cho các trang như sau:

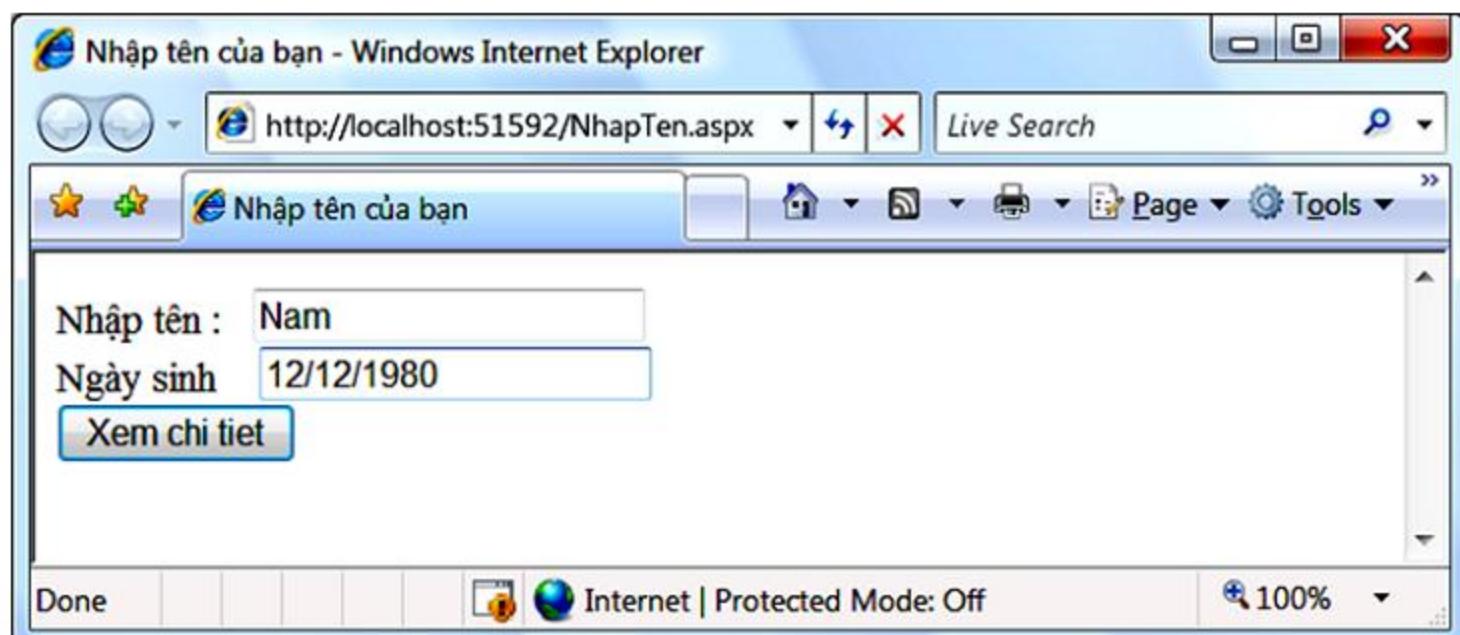
```
public partial class NhapTen: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e) { }
    protected void btnXemChiTiet_Click(object sender, EventArgs e)
    {
        //Lấy các giá trị đã nhập vào các TextBox
        string strHoTen = txtTen.Text;
        string strNgaySinh = txtNgaySinh.Text;
        //Chuyển quan trang XemChiTiet.aspx
        Response.Redirect("XemChiTiet.aspx?Ten=" + strHoTen + "&NgaySinh=" + strNgaySinh);
    }
}
```

Minh họa phần mã của trang NhapTen.aspx

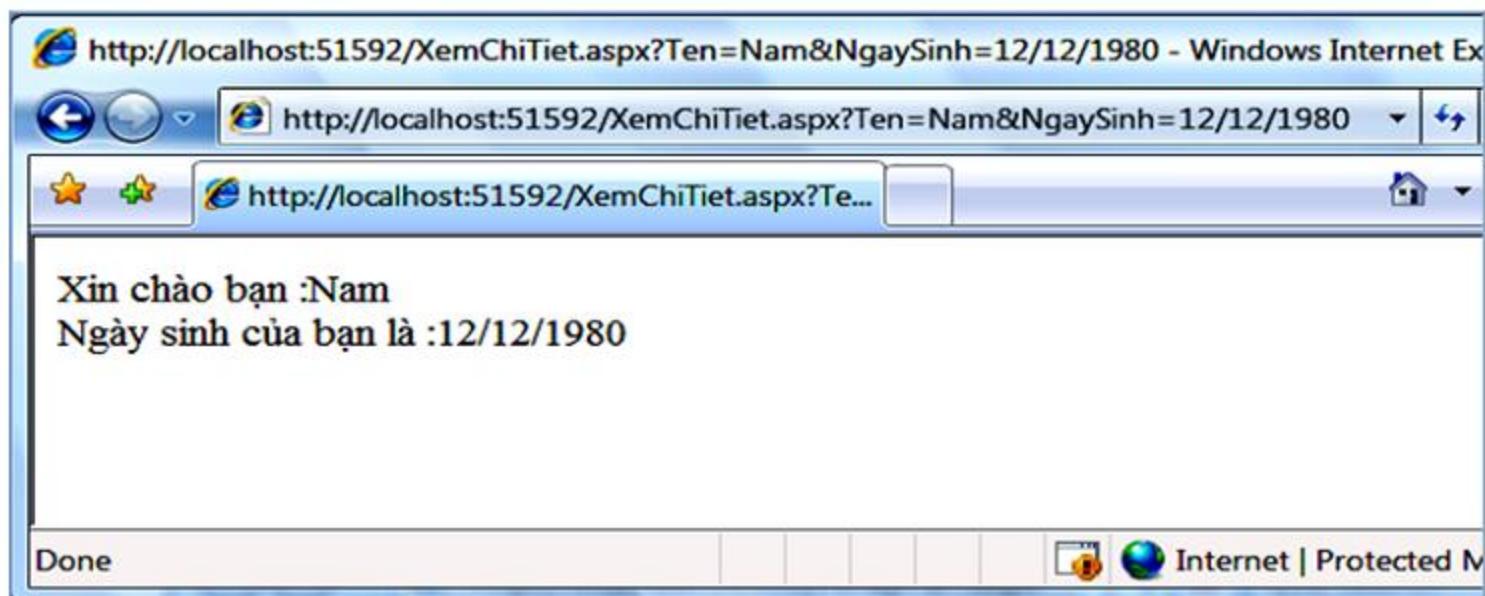
```
public partial class XemChiTiet: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        //Lấy các giá trị từ trang NhapTen.aspx
        string strHoTen = Request.QueryString["Ten"];
        string strNgaySinh = Request.QueryString["NgaySinh"];
        lblXemChiTiet.Text = "Xin chào bạn:" + strHoTen + "<br>" +
            "Ngày sinh của bạn là:" + strNgaySinh;
    }
}
```

Minh họa phần mã của trang XemChiTiet.aspx

Bước 3: Nhấn Ctrl+F5 để thi hành ứng dụng. Nhập tên và ngày sinh và nhấn nút **Xem chi tiết**. Kết quả như hình 6.2 và 6.3.



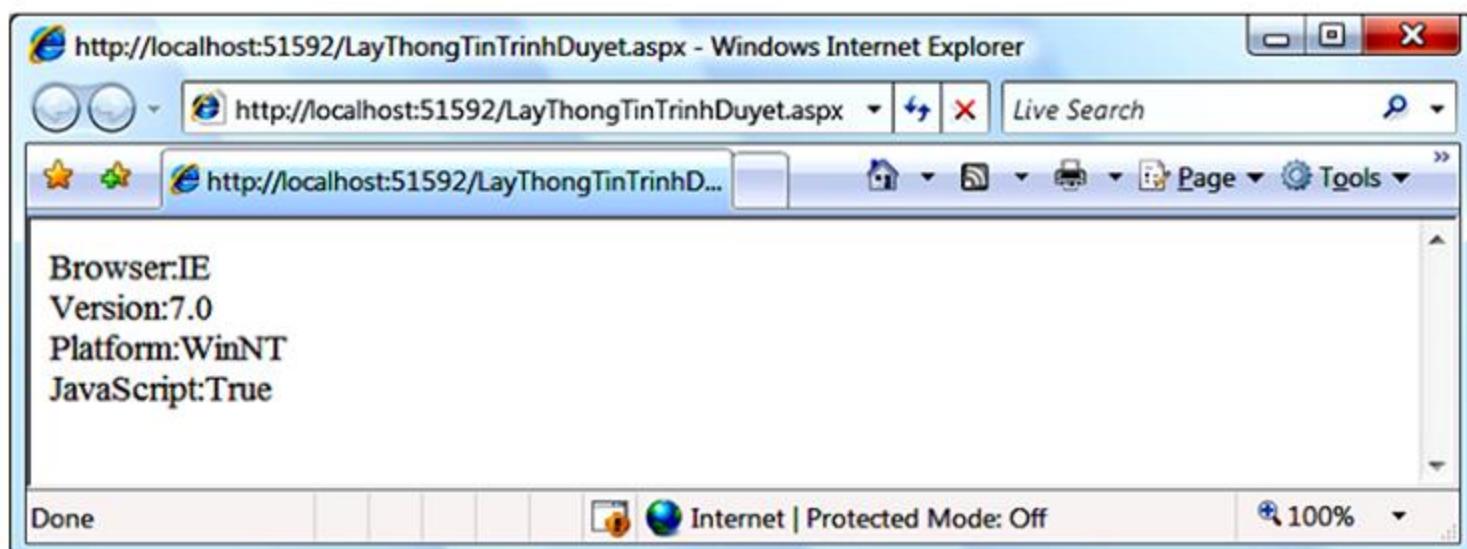
Hình 3.24: Khi thi hành trang XemChiTiet.aspx



Hình 3.25: Khi thi hành trang XemChiTiet.aspx

Ví dụ: Tạo trang LayThongTinTrinhDuyet.aspx hiển thị các thông tin các trình duyệt của người dùng. Viết lệnh xử lý như sau:

```
public partial class LayThongTinTrinhDuyet: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Response.Write("Browser:" + Request.Browser + "<br>");
        Response.Write("Version:" + Request.Browser.Version + "<br>");
        Response.Write("Platform:" + Request.Browser.Platform +
        "<br>");
        Response.Write("JavaScript:" + Request.Browser.JavaScript);
    }
}
```



Hình 3.26: Kết quả thi hành trang LayThongTinTrinhDuyet.aspx

Đối tượng Server

Đối tượng Server được sử dụng để cung cấp thông tin của Server cho ứng dụng.

- **Thuộc tính MachineName**

Thuộc tính này được dùng để lấy tên của Web Server.

- **Phương thức MapPath**

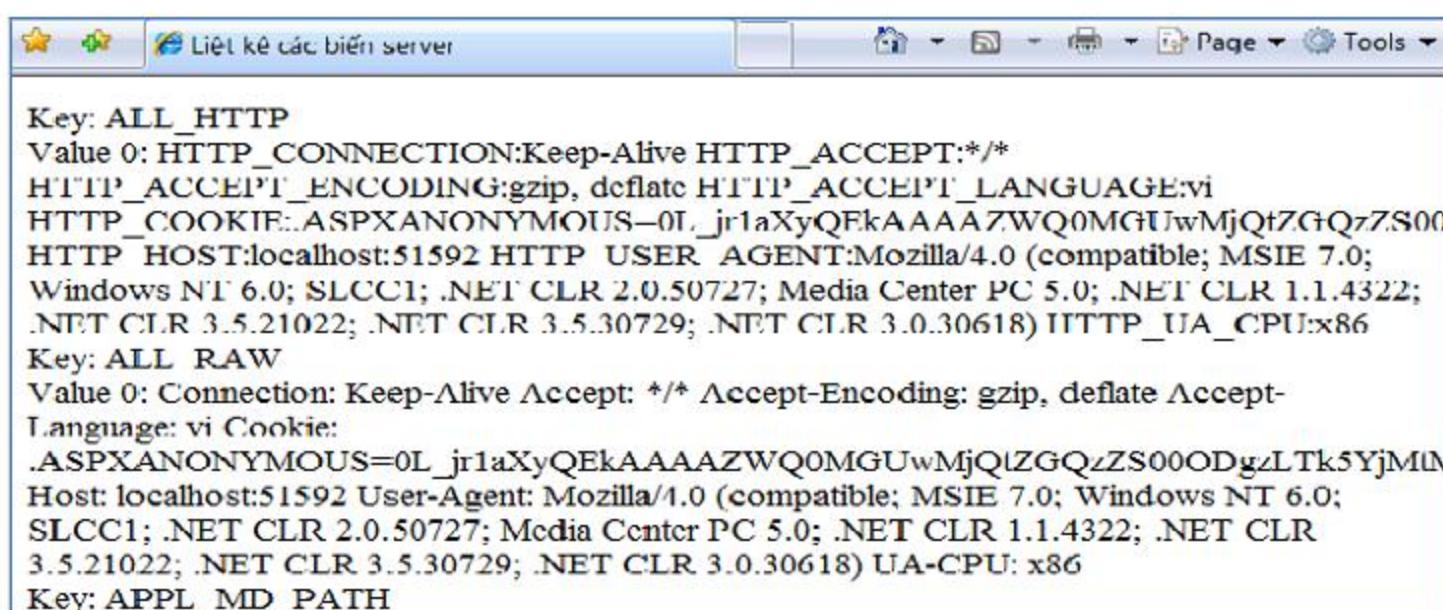
Phương thức MapPath được dùng để lấy đường dẫn vật lý hoặc đường dẫn ảo đến một thư mục trên Server.

- **Phương thức Transfer (<Đường dẫn đến trang cần yêu cầu>)**

Ngừng thi hành trang hiện hành, gửi yêu cầu mới đến trang được gọi thực hiện.

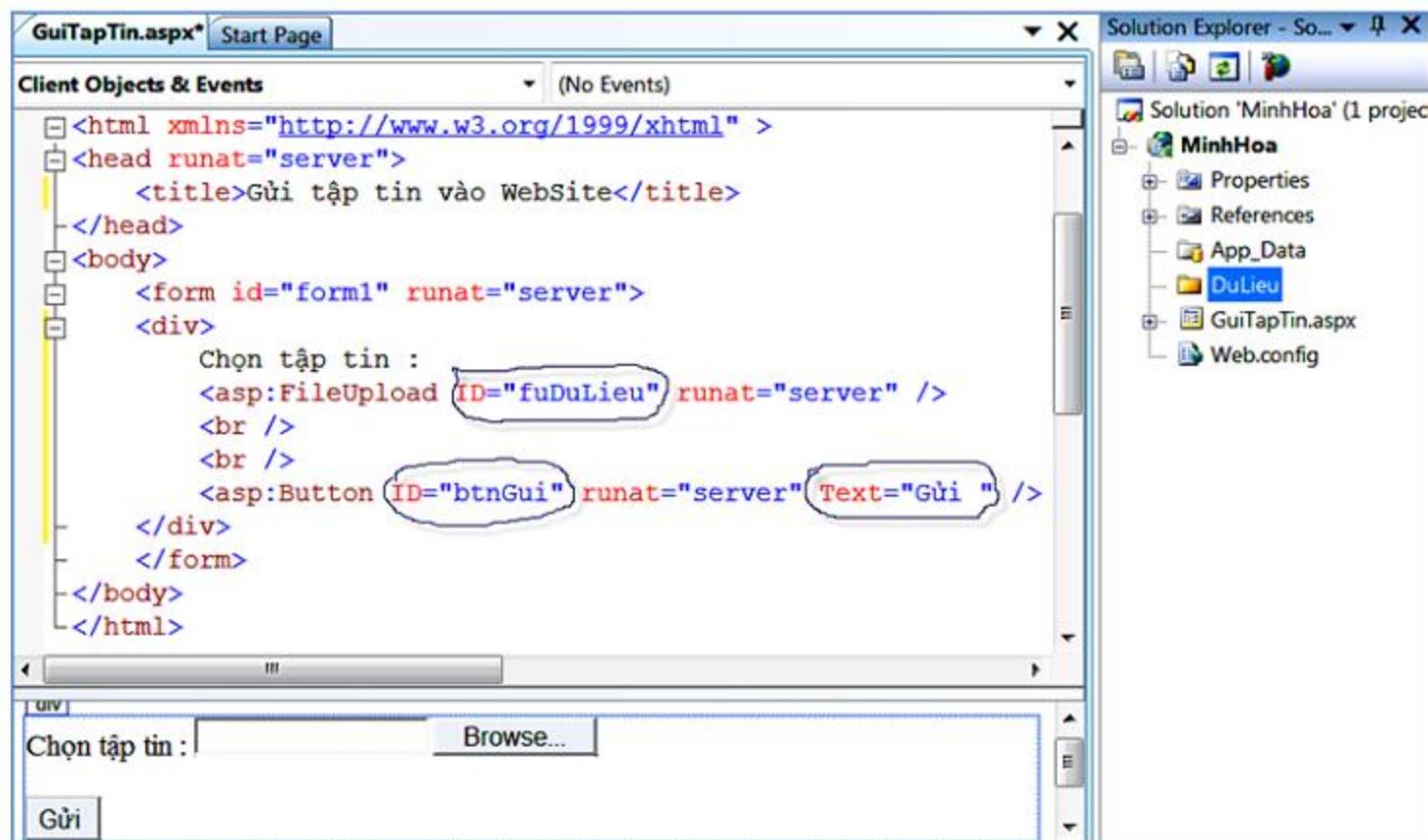
Thí dụ: Tạo trang LietKeCacBienServer, lấy danh sách các biến server

```
public partial class LietKeCacBienServer : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        int i, j;
        NameValueCollection coll = Request.ServerVariables;
        String[] arr1 = coll.AllKeys;
        for (i = 0; i < arr1.Length; i++)
        {
            Response.Write("Key: " + arr1[i] + "<br>");
            String[] arr2 = coll.GetValues(arr1[i]);
            for (j = 0; j < arr2.Length; j++)
            {
                Response.Write("Value " + j + ": " + Server.HtmlEncode(arr2[j])
                + "<br>");
            } } }
}
```



Thí dụ: Minh họa upload tập tin về Sever

Nội dung trang GuiTapTin.aspx, cho phép gửi một tập tin từ Client đến Server.

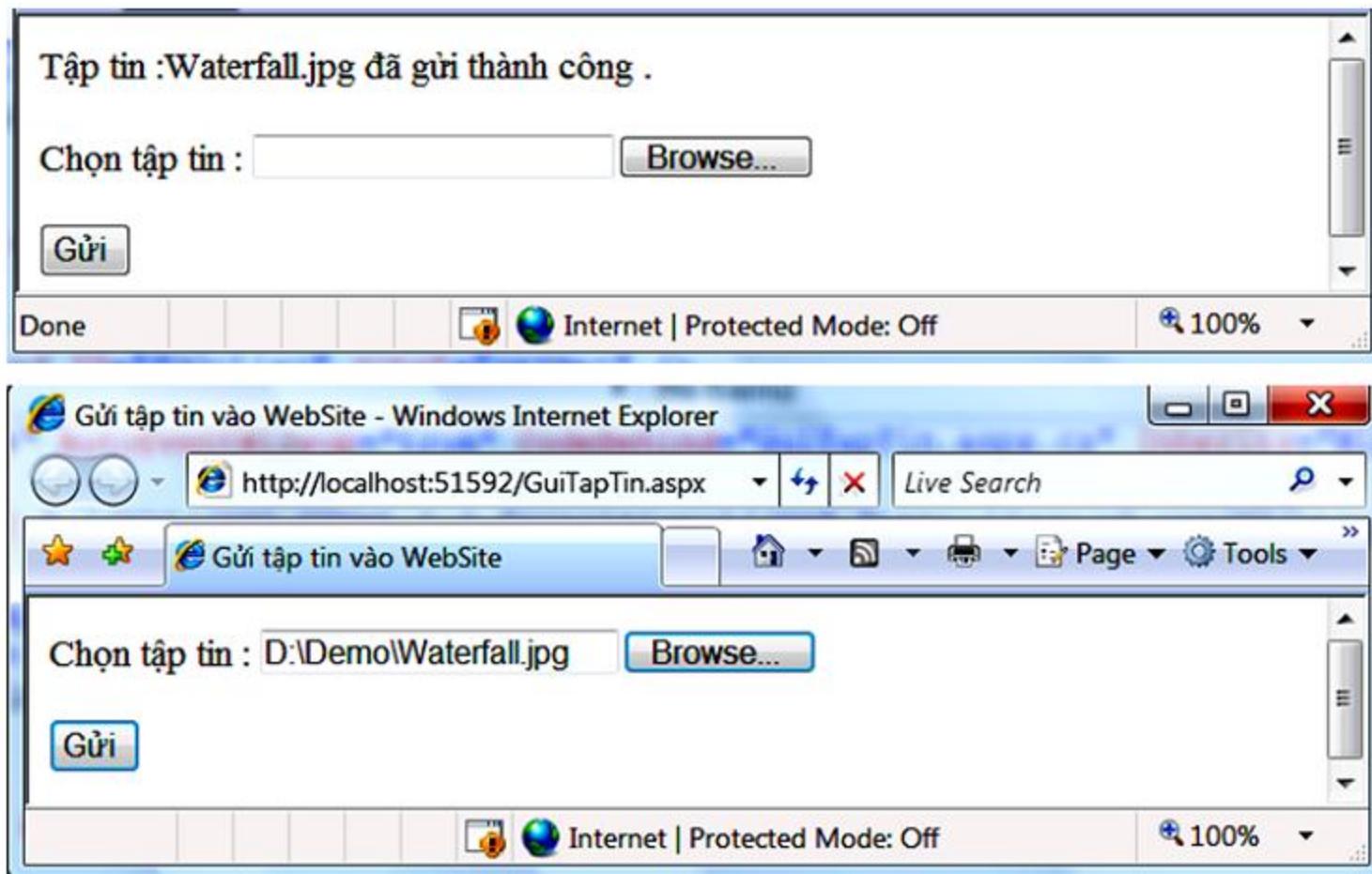


Phần mã lệnh:

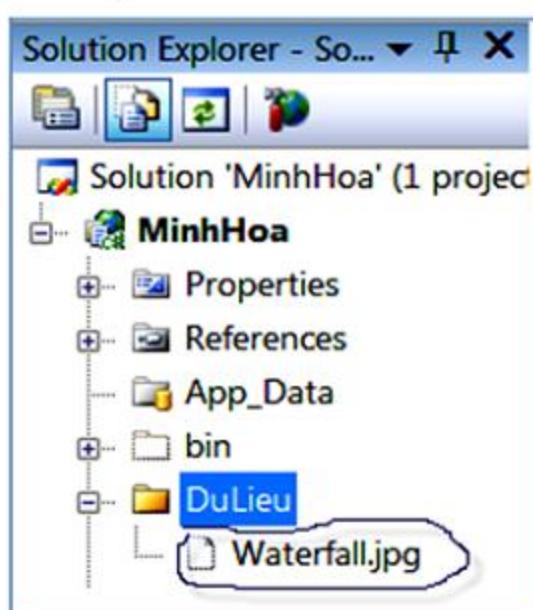
```
public partial class GuiTapTin: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected void btnGui_Click(object sender, EventArgs e)
    {
        //Kiểm tra có tập tin nào được chọn ?
        if (fuDuLieu.HasFile)
        {
            //thiết lập đường dẫn cho tập tin
            string filename = "~/DuLieu/" + fuDuLieu.FileName;
            //Lấy đường dẫn vật lý cho tập tin trên ứng dụng
            string s = Server.MapPath(filename);
            //Lưu tập tin về thư mục DuLieu của ứng dụng
            fuDuLieu.SaveAs(s);
            Response.Write("Tập tin:" + fuDuLieu.FileName +
                "đã gửi thành công.");
        }
    }
}
```

Kết quả thực thi:



Bước 5: Kiểm tra kết quả. Nhấp vào biểu tượng **Show All Files** trên cửa sổ Solution, sau đó nhấp biểu tượng **Refresh** thấy kết quả như hình bên hoặc các bạn có thể kiểm tra bằng cách mở thư mục DuLieu trên đĩa để xem kết quả.



BÀI TẬP CHƯƠNG 3

Bài 1: Thực hiện lại các bài tập thí dụ trong chương 3.

Bài 2: Tạo một Web User Control Login như sau:

The image shows a window titled "Đăng nhập" (Login) with a purple header bar. Inside, there are two text input fields: one for "User name:" and one for "Password:", both with black placeholder text. Below the fields are two buttons: "Login" on the left and "Cancel" on the right. The entire window has a light gray background and a thin black border.

Mã lệnh trang Login.ascx

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="Login.ascx.cs"
Inherits="Login" %>
<style type="text/css">
.style1
{
    width: 174px;
}
.style2
{
    width: 48%;
}
</style>
<table runat="server" id="NoiDungLogin"
    style="border:solid 1px purple; border-collapse:collapse; width: 307px;">
<tr>
<td colspan="2" style="text-align:center; background-color:purple; color:White">
Đăng nhập
</td>
</tr>
<tr>
<td style="text-align:center;" class="style2">User name:</td>
<td class="style1">
<asp:TextBox runat="server" ID="txtUserID" Width="79%"></asp:TextBox>
</td>
</tr>
```

```

<tr>
<td style="text-align:center; " class="style2">Password:</td>
<td class="style1">
<asp:TextBox runat="server" ID="txtPassword" Width="78%" 
TextMode="Password"></asp:TextBox>
</td>
</tr>
<tr>
<td class="style2">
<asp:Button runat="server" ID="cmdLogin" Text="Login" />
<asp:Button runat="server" ID="cmdCancel" Text="Cancel" />
</td>
<td class="style1">
    &nbsp;</td>
</tr>
</table>

```

Sau đó, thêm vào trong phần mã lệnh

```

public partial class Login: System.Web.UI.UserControl
{
    private string dorong;
    public string DoRong
    {
        set
        {
            dorong = value; NoiDungLogin.Width = dorong;
        }
        get
        {
            return dorong;
        }
    }
    public Boolean CheckAccount()
    {
        return (txtUserID.Text == "ASP.NET" && txtPassword.Text == "123456");
    }
    // Lấy User name trong ô User name

```

```

public string GetUserName()
{
    return txtUserID.Text;
}

// lấy Password trong ô Password
public string GetPassword()
{
    return txtPassword.Text;
}

protected void Page_Load(object sender, EventArgs e)
{
}

}

```

Tạo trang **LoginDemo.aspx**, sử dụng Web User Control **Login** trên

```

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

<%@ Register src="Login.ascx" tagname="Login" tagprefix="uc1" %>
<!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>
            <uc1:Login ID="Login1" runat="server" DoRong="350px" />
        <br />
        <asp:Label runat="server" ID="lblThongBao"></asp:Label>
        </div>
    </form>
</body>
</html>

```

Phần mã lệnh thực thi trang LoginDemo.aspx.cs

```
public partial class LoginDemo : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (this.IsPostBack == true) //Nếu không phải lần đầu nạp trang
        {
            lblThongBao.Text = "Bạn đã nhập User name=" + Login1.GetUserName() + "
Mật khẩu: " + Login1.GetPassword();
        }
    }
}
```

Thực thi ứng dụng ta có kết quả:



Bài 3: Tạo phần Menu như sau:



Bài 4: Tạo phần Header và Footer để có thể chèn vào các trang.
Giao diện như sau:

Nội dung của điều khiển Header.ascx

TRUNG TÂM ĐÀO TẠO MẠNG MÁY TÍNH NHẤT NGHỆ

Địa chỉ: 105 Bà Huyện Thanh Quan, Quận 3, TP. HCM

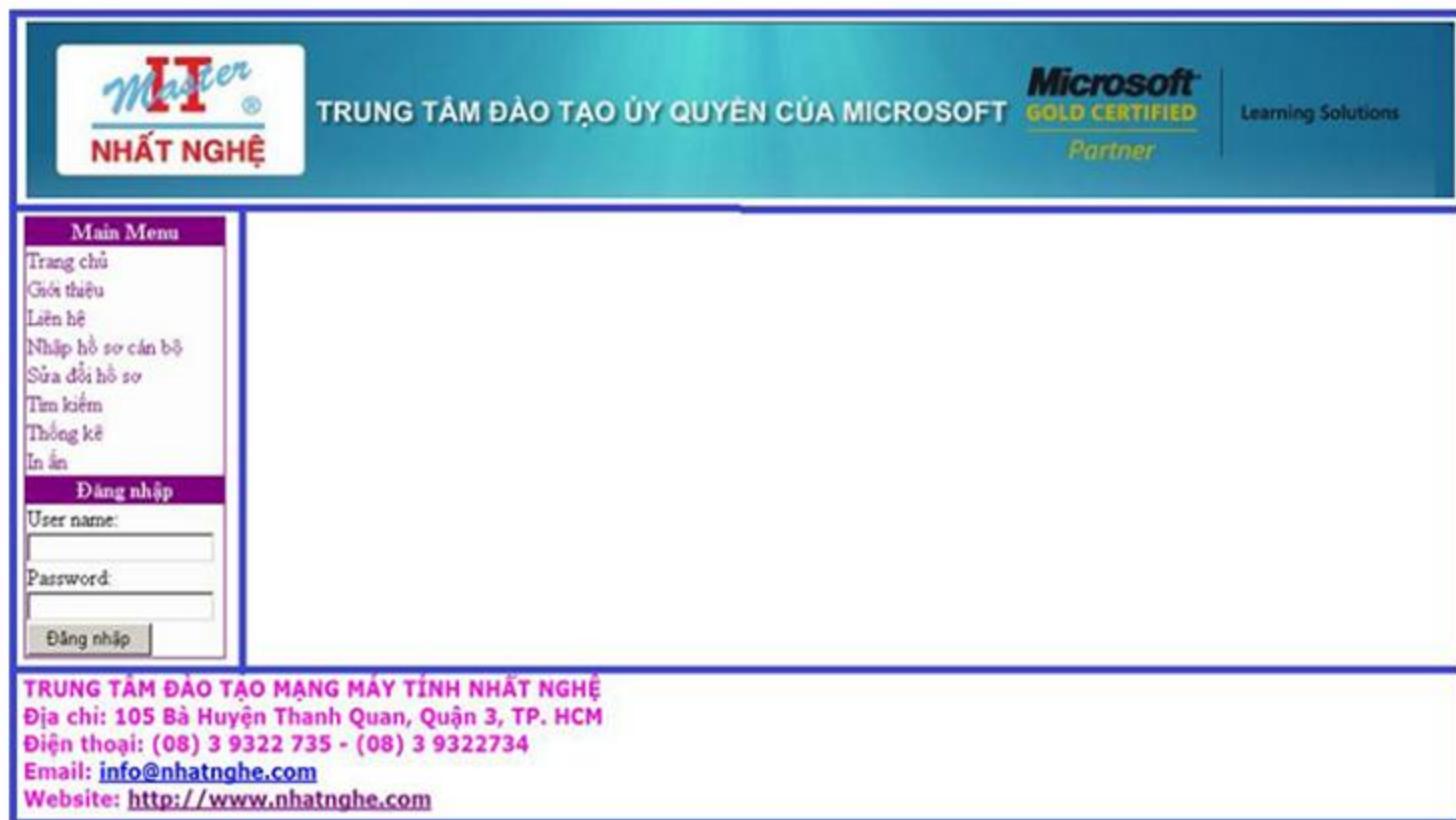
Điện thoại: (08) 3 9322 735 - (08) 3 9322734

Email: info@nhatnghe.com

Website: <http://www.nhatnghe.com>

Nội dung của điều khiển Footer.ascx

Bài 5: Thiết kế trang MasterPage có sử dụng các Web User Controls trên để tạo trang.



The screenshot shows a Master Page layout. On the left is a vertical navigation menu with items: Trang chủ, Giới thiệu, Liên hệ, Nhập hồ sơ cán bộ, Sửa đổi hồ sơ, Tên kiếm, Thông kê, In ấn, Đăng nhập. The 'Đăng nhập' item is highlighted with a purple background. Below it is a login form with fields for User name and Password, and a 'Đăng nhập' button. The main content area on the right contains the contact information from the question: TRUNG TÂM ĐÀO TẠO MẠNG MÁY TÍNH NHẤT NGHỆ, Địa chỉ: 105 Bà Huyện Thanh Quan, Quận 3, TP. HCM, Điện thoại: (08) 3 9322 735 - (08) 3 9322734, Email: info@nhatnghe.com, Website: <http://www.nhatnghe.com>.

Bài 6: Tạo một ứng dụng ASP.NET thực hành với đối tượng Request và Response.

Tạo trang RequestFormGetPage.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"  
CodeFile="RequestFormGetPage.aspx.cs" Inherits="RequestFormGetPage" %>  
<!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>
            <div>
                <table style="width: 292px">
                    <tr>
                        <td colspan="2">
                            <asp:Label ID="Label1" runat="server" Font-Bold="True" Text="Quick
Create New User">
                                Width="404px"></asp:Label></td>
                        </tr>
                        <tr>
                            <td style="width: 79px; height: 26px;">
                                <asp:Literal ID="Literal1" runat="server" Text="Email:"></asp:Literal>
                            </td>
                            <td style="width: 351px; height: 26px;">
                                <input id="Email" runat="server" style="width: 183px" type="text"
/></td>
                        </tr>
                        <tr>
                            <td style="width: 79px; height: 21px">
                                <asp:Literal ID="Literal2" runat="server"
Text="Password:"></asp:Literal></td>
                            <td style="height: 21px; width: 351px;">
                                <input id="Password" runat="server" style="width: 181px"
type="password" /></td>
                            </tr>
                            <tr>
                                <td style="width: 79px; height: 21px">
                                    <asp:Literal ID="Literal3" runat="server"
Text="Address:"></asp:Literal></td>
                                <td style="height: 21px; width: 351px;">

```

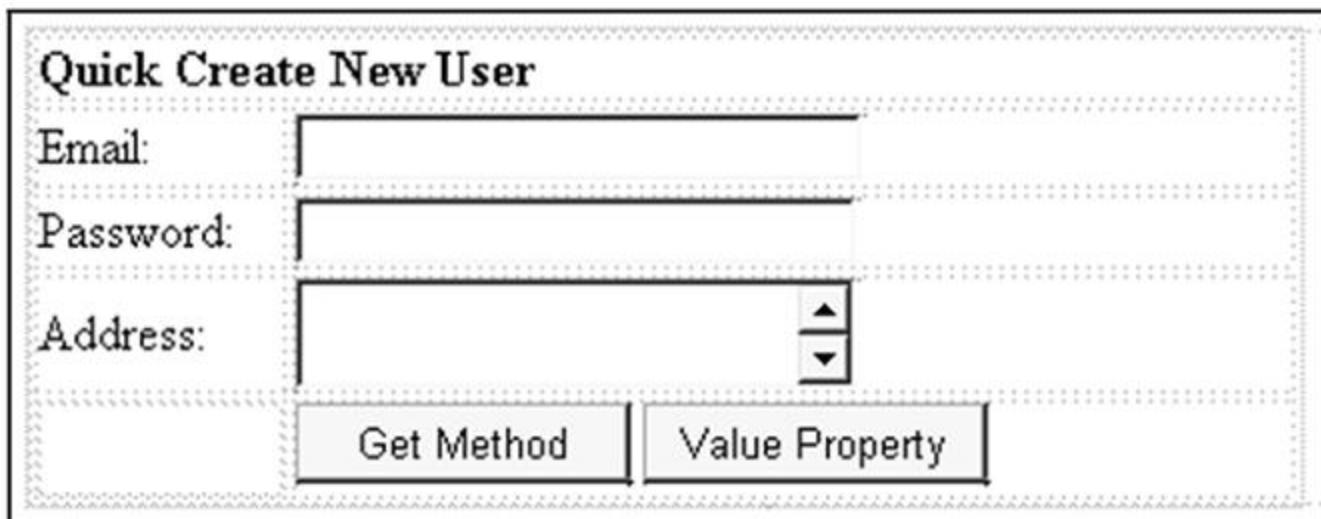
```

        <textarea id="Address" runat="server" rows="2" style="width: 181px"></textarea></td>
    </tr>
    <tr>
        <td style="width: 79px">
        </td>
        <td style="width: 351px">
            <input id="Submit1" style="width: 108px" type="submit" value="Get Method" language="javascript" onclick="return Submit1_onclick()" onserverclick="Submit1_ServerClick" runat="server" />
            <asp:Button ID="Button1" runat="server" Text="Value Property" Width="111px" OnClick="Button1_Click" /></td>
        </tr>
    </table>
</div>

</div>
</form>
</body>
</html>

```

Phần giao diện thiết kế trang RequestFormGetPage.aspx



Phần mã lệnh thực thi trang RequestFormGetPage.aspx.cs

```

public partial class RequestFormGetPage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

```

```

}

protected void Submit1_ServerClick(object sender, EventArgs e)
{
    Response.Write("Email: " +
        Request.Form.Get("Email"));

    Response.Write("<br>Password: " +
        Request.Form.Get("Password"));

    Response.Write("<br>Address: " +
        Request.Form.Get("Address"));
}

protected void Button1_Click(object sender, EventArgs e)
{
    Response.Write("Email: " + Email.Value);
    Response.Write("<br>Password: " + Password.Value);
    Response.Write("<br>Address: " + Address.Value);
}
}

```

Kết quả thực thi:

Nhập dữ liệu và click nút **[Get Method]**

Email: annguyen@gmail.com
 Password: 123456
 Address: Việt Nam

Quick Create New User

Email:

Password:

Address:

Nhập dữ liệu và click nút **[Value Property]**. Cho biết kết quả hiển thị.

Chương 4

QUẢN LÝ TRẠNG THÁI

Các vấn đề chính sẽ được đề cập:

- ✓ Quản lý trạng thái ViewState
- ✓ Chuyển thông tin giữa các trang
- ✓ Các đối tượng của ASP.NET gồm Cookies, Session và Application

Kết thúc bài này các bạn có thể:

- *Vận dụng các kỹ thuật trên để quản lý trạng thái thông tin giữa các trang ASP.NET*
- *Mô tả được các thuộc tính và phương thức của các đối tượng Session, Request Response, Server, Cookie và Application*
- *Trình bày và sử dụng được các sự kiện trong tập tin Global.asax*
- *Mô tả và sử dụng được các thành phần cơ bản trong tập tin Web.config*

Sự khác biệt đáng kể nhất giữa lập trình web form và lập trình winform là quản lý trạng thái (state management), làm thế nào bạn lưu trữ thông tin qua các vòng đời của ứng dụng của bạn. Thông tin này có thể đơn giản như tên của người dùng, hoặc phức tạp như một giỏ hàng trong một website thương mại điện tử.

Trong một ứng dụng Windows truyền thống, ít cần phải suy nghĩ về quản lý trạng thái vì do luôn sẵn có bộ nhớ đệm và bạn chỉ cần phải quan tâm về một người dùng duy nhất. Trong một ứng dụng web, đó là một vấn đề khác. Hàng ngàn người sử dụng có thể đồng thời chạy cùng một ứng dụng trên cùng một máy tính (máy chủ web), mỗi một giao tiếp với HTTP không lưu trạng thái (stateless protocol). Các điều kiện này làm cho nó không thể thiết kế một ứng dụng web như một ứng dụng Windows truyền thống.

Sự hiểu biết về các hạn chế trạng thái là chìa khóa để tạo ra các ứng dụng web hiệu quả. Trong bài này, bạn sẽ thấy làm thế nào bạn có thể quản lý trạng thái của ASP.NET với các tính năng để lưu trữ thông tin bảo mật và nhất quán. Bạn sẽ có các tùy chọn lưu trữ khác nhau, bao gồm xem trạng thái (view state), trạng thái phiên giao dịch (session), và tùy chỉnh cookies. Bạn cũng sẽ xem xét làm thế nào để chuyển thông tin từ trang này sang trang khác bằng cách sử dụng cross-page (post) và chuỗi truy vấn (query string).

4.1. VẤN ĐỀ CỦA TRẠNG THÁI

Trong một chương trình Windows truyền thống, người dùng tương tác với một ứng dụng liên tục chạy. Một phần của bộ nhớ trên máy tính để bàn được phân bổ để lưu trữ các thiết lập hiện hành của thông tin làm việc.

Trong một ứng dụng web, một trang web ASP.NET chuyên nghiệp có thể trông giống như một ứng dụng liên tục chạy, nhưng đó là thực sự chỉ là một ảo ảnh thông minh. Trong một yêu cầu web điển hình, khách hàng kết nối đến máy chủ web và yêu cầu một trang. Khi trang được chuyển giao, kết nối được cắt đứt, và các máy chủ web từ bỏ bất cứ thông tin nó có về khách hàng. Với thời gian người sử dụng nhận được một trang, do mã trang web này đã ngừng chạy nên không có thông tin còn lại trong bộ nhớ của web server.

Thiết kế này có một lợi thế đáng kể. Bởi vì, khách hàng cần phải được kết nối chỉ trong một vài giây là nhiều nhất, do đó một máy chủ web có thể xử lý một số lượng lớn các yêu cầu gần như đồng thời. Tuy nhiên, nếu bạn muốn giữ lại thông tin đủ thời gian để nó có thể được sử dụng trên nhiều postbacks hoặc trên nhiều trang, bạn cần phải thực hiện các bước bổ sung.

4.2. XEM TRẠNG THÁI (VIEW STATE)

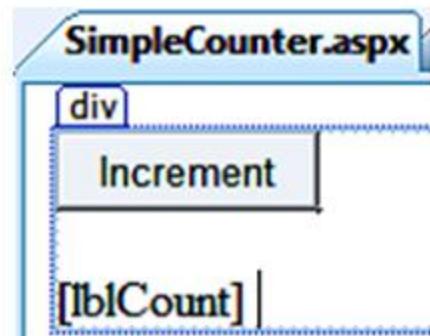
Các điều khiển Web Server Controls lưu trữ hầu hết các giá trị của chúng vào view state, khi thuộc tính EnableViewState được thiết lập với giá trị true (mặc định). Thuộc tính ViewState của trang cung cấp thông tin view state hiện hành. Thuộc tính này là một thẻ hiện của lớp StateBag. StateBag là một từ điển, có mỗi mục được lưu trong một khe "riêng" bằng cách sử dụng một tên chuỗi duy nhất.

Thí dụ, hãy xem xét mã này:

```
this.ViewState["Counter"] = 1;
```

Lệnh này đặt giá trị 1 (hay đúng hơn, một số nguyên có chứa giá trị 1) vào trong ViewState. Nếu hiện tại không có mục có tên “Counter”, một mục mới sẽ được thêm tự động. Nếu một mục đã được lưu trữ dưới cái tên “Counter”, nó sẽ được thay thế.

Thí dụ 1: Tạo một trang **SimpleCounter.aspx** có một Button và Label.



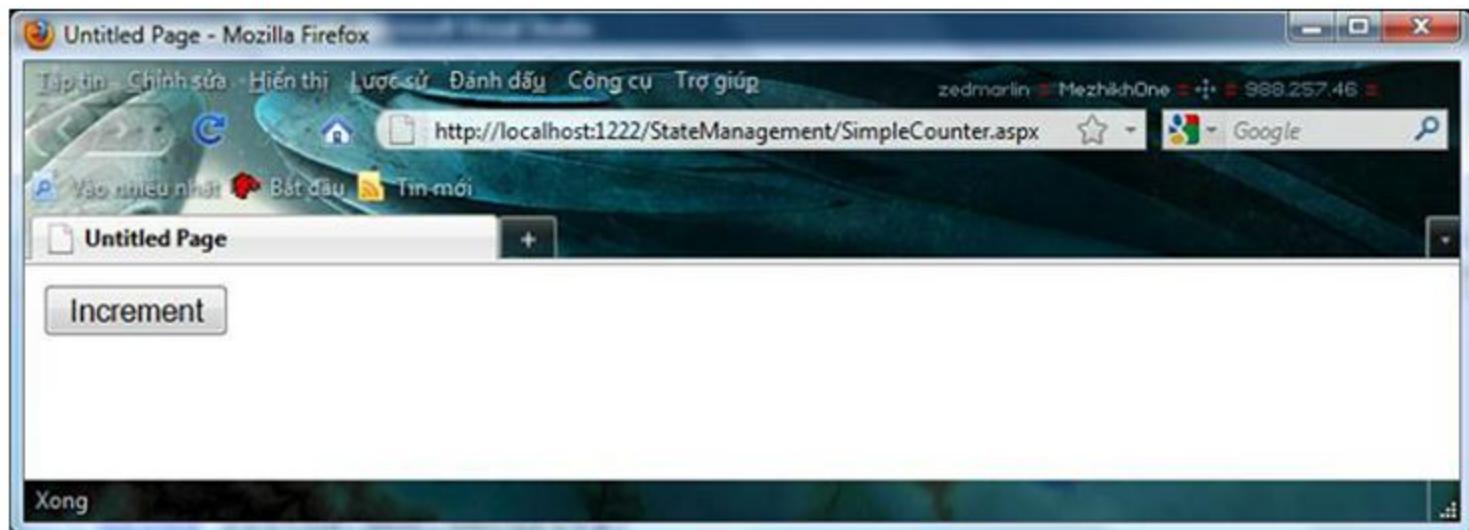
```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="SimpleCounter.aspx.cs" Inherits="SimpleCounter" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">  <title>Untitled Page</title> </head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Button ID="cmdIncrement" runat="server" OnClick="cmdIncrement_Click"
Text="Increment" /><br />      <br />
            <asp:Label ID="lblCount" runat="server"></asp:Label>&ampnbsp</div>
        </form>
    </body>
</html>
```

Phần mã lệnh trang **SimpleCounter.aspx.cs**:

```
public partial class SimpleCounter : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e) { }

    protected void cmdIncrement_Click(object sender, EventArgs e)
    {
        int counter;
        if (ViewState["Counter"] == null) {
            counter = 1;
        }
        else {
            counter = (int)ViewState["Counter"] + 1;
        }
        ViewState["Counter"] = counter;
        lblCount.Text = "Counter: " + counter.ToString();
    }
}
```

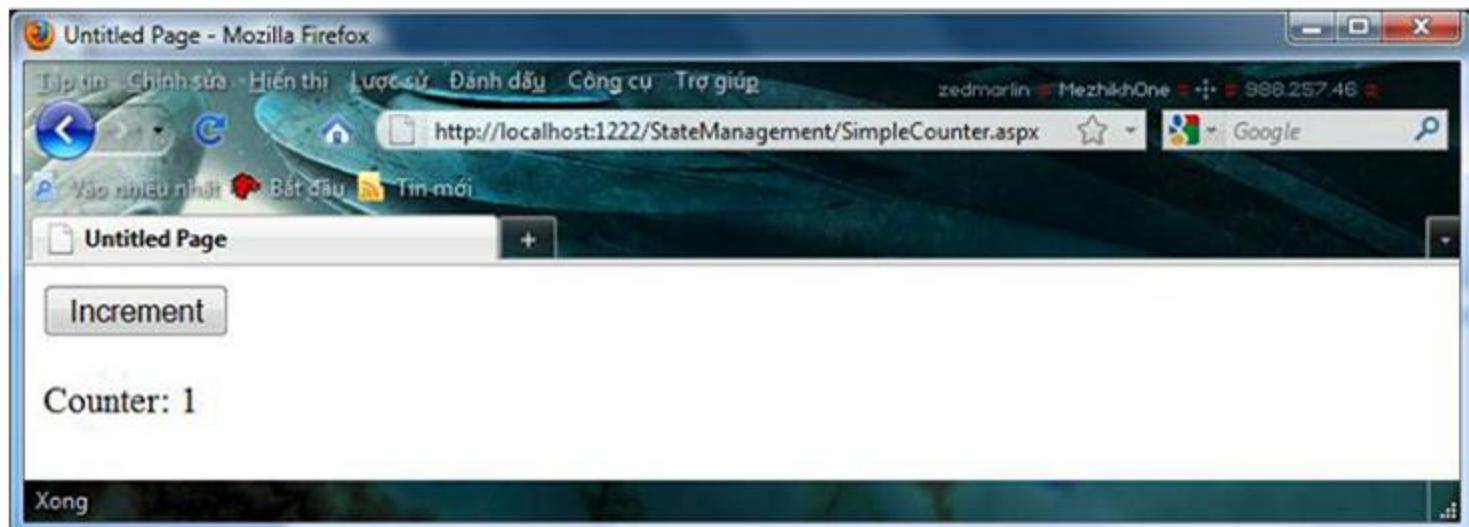
Khi thực thi chương trình, đầu tiên chỉ xuất hiện button như hình sau:



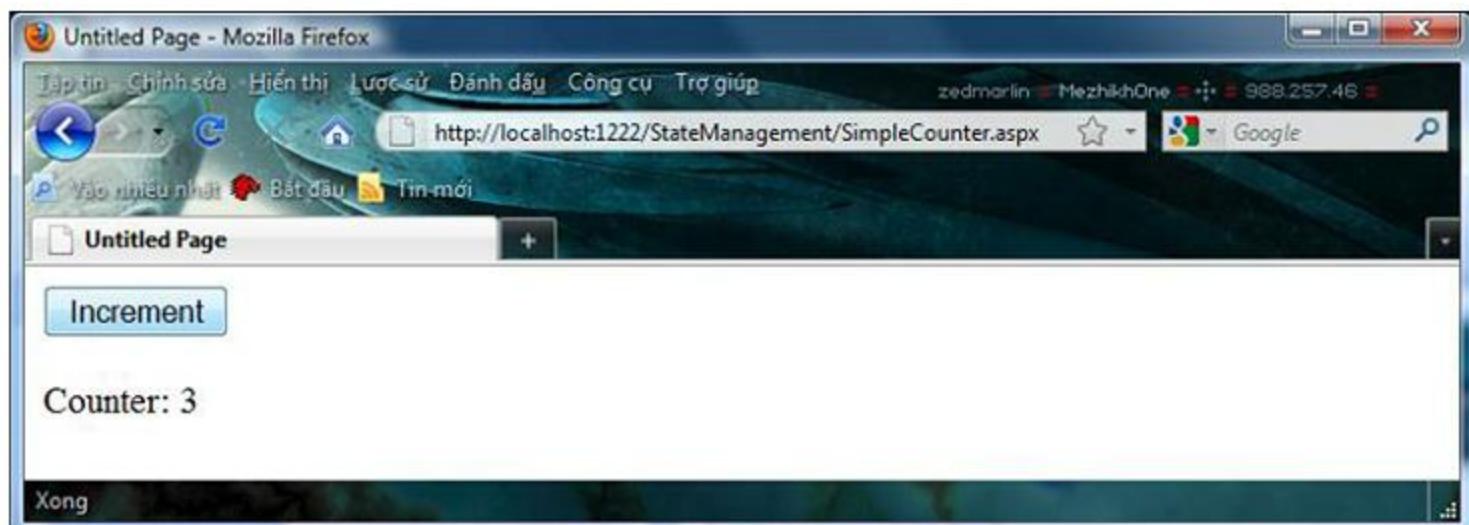
Khi click vào button increment, giá trị counter được khởi tạo bằng 1 và gán cho ViewState với tên “Counter”, và hiển thị giá trị trên trang qua điều khiển nhãn lblCount:

```
ViewState["Counter"] = counter;  
lblCount.Text = "Counter: " + counter.ToString();
```

Kết quả hiển thị:

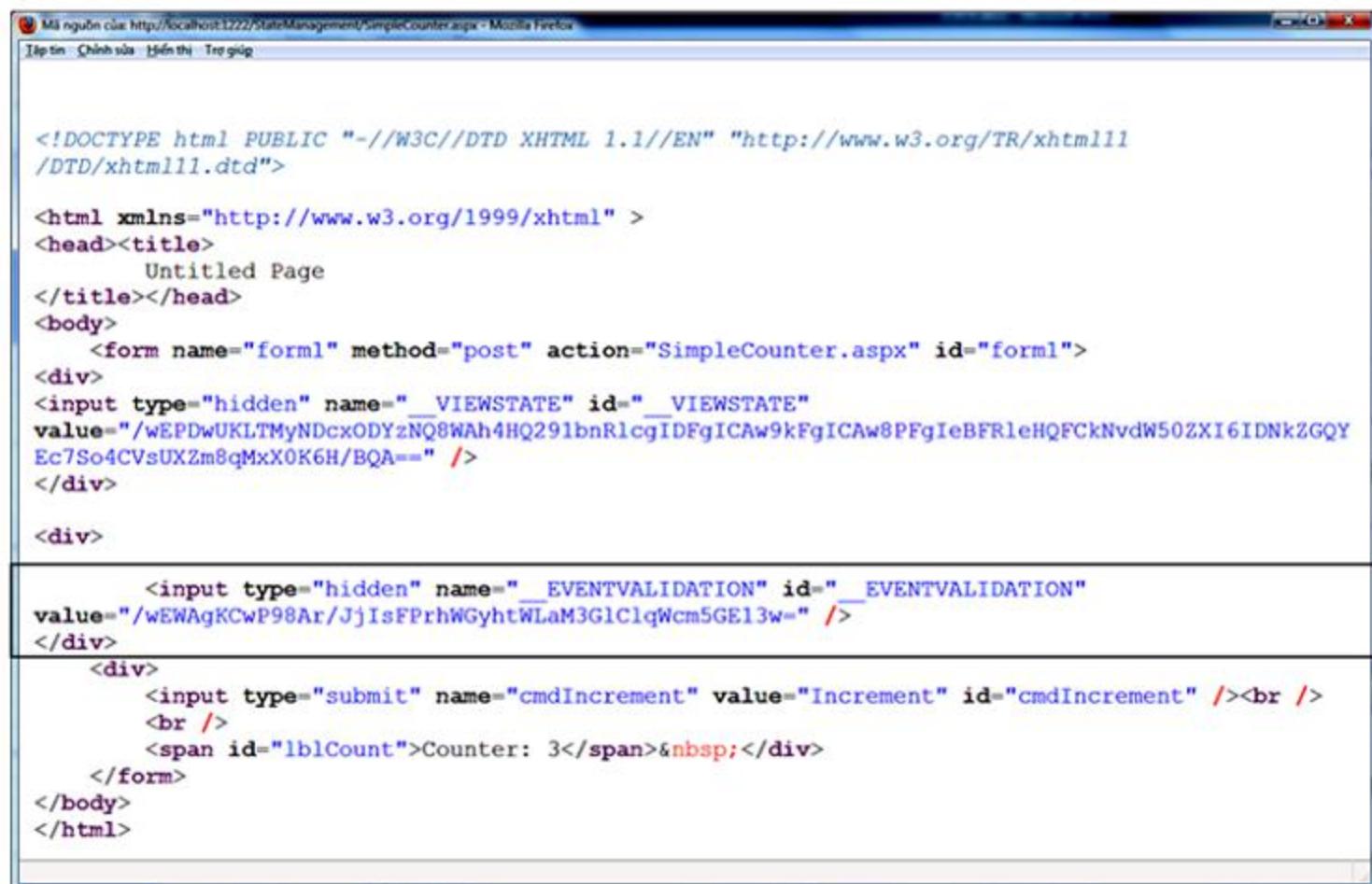


Khi click tiếp hai lần trên button, giá trị của biến counter được tăng lên là 3, như vậy giá trị counter được duy trì qua các lần request nhờ vào ViewState[“Counter”]



Khi xem mã nguồn của trang SimpleCounter qua trình duyệt, nội dung trang có chứa khai báo _VIEWSTATE với giá trị lưu dạng mã Base64 (kiểu mã hóa đặc biệt của chuỗi đó và luôn luôn được chấp nhận trong một tài liệu HTML bởi vì nó không bao gồm bất kỳ ký tự mở rộng nào) như sau:

```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwUKLTMyNDcxODYzNQ8WAh4HQ291bnRlcgIDFgICAw9kFgIC
Aw8PFgIeBFRleHQFCkNvdW50ZXI6IDNkZGQYEc7So4CVsUXZm8qMxX0K6
H/BQA==" />
```



```
Mã nguồn của: http://localhost:1222/StateManagement/SimpleCounter.aspx - Mozilla Firefox
Tệp tin Chính sửa Hiển thị Trợ giúp

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11
/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head><title>
    Untitled Page
</title></head>
<body>
    <form name="form1" method="post" action="SimpleCounter.aspx" id="form1">
<div>
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwUKLTMyNDcxODYzNQ8WAh4HQ291bnRlcgIDFgICAw9kFgICAw8PFgIeBFRleHQFCkNvdW50ZXI6IDNkZGQYEc7So4CVsUXZm8qMxX0K6H/BQA==" />
</div>
<div>
    <input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION"
value="/wEWAjKCwP98Ar/JjIsFPrhWGyhtWLam3G1ClqWcm5GE13w=" />
</div>
    <div>
        <input type="submit" name="cmdIncrement" value="Increment" id="cmdIncrement" /><br />
        <br />
        <span id="lblCount">Counter: 3</span>&nbsp;</div>
    </form>
</body>
</html>
```

Hình 4.1: Mã nguồn trang SimpleCounter.aspx có biến _VIEWSTATE

4.3. CHUYỂN THÔNG TIN GIỮA CÁC TRANG

Phần này trình bày hai cách thức chuyển thông tin giữa các trang bằng cách sử dụng Cross-page Post và chuỗi truy vấn (query string)

4.3.1. Sử dụng Cross-Page Post

Thí dụ 2

Tạo trang CrossPage1.aspx, với hai nút lệnh Cross-Page Postback và Manual Transfer cho phép chuyển thông tin đến trang CrossPage2.aspx bằng kỹ thuật crosspage post với khai báo thuộc tính PostBackUrl="CrossPage2.aspx" trong khai báo button [Cross-Page

[Postback] và cách chuyển thông thường với Server.Transfer("CrossPage2.aspx", true); khi click button [Manual Transfer]

Nội dung trang **CrossPage1.aspx**:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="CrossPage1.aspx.cs"
Inherits="CrossPage1" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">  <title>CrossPage1</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server" >
        <div>
            First Name:
            <asp:TextBox ID="txtFirstName" runat="server"></asp:TextBox>      <br />
            Last Name:
            <asp:TextBox ID="txtLastName" runat="server"></asp:TextBox>      <br />
<br />
            <asp:Button runat="server" ID="cmdPost"
                PostBackUrl="CrossPage2.aspx" Text="Cross-Page Postback" /><br />
            <asp:Button runat="server" ID="cmdTransfer" Text="Manual Transfer"
                OnClick="cmdTransfer_Click" Visible="False" />
        </div>
    </form>
</body>
</html>
```

Mã lệnh trang **CrossPage2.aspx.cs**:

```
public partial class CrossPage1 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)  {
        if (Request.QueryString["err"] != null)
            Page.Validate();
    }

    protected void cmdTransfer_Click(object sender, EventArgs e)  {
        Server.Transfer("CrossPage2.aspx", true);
    }
}
```

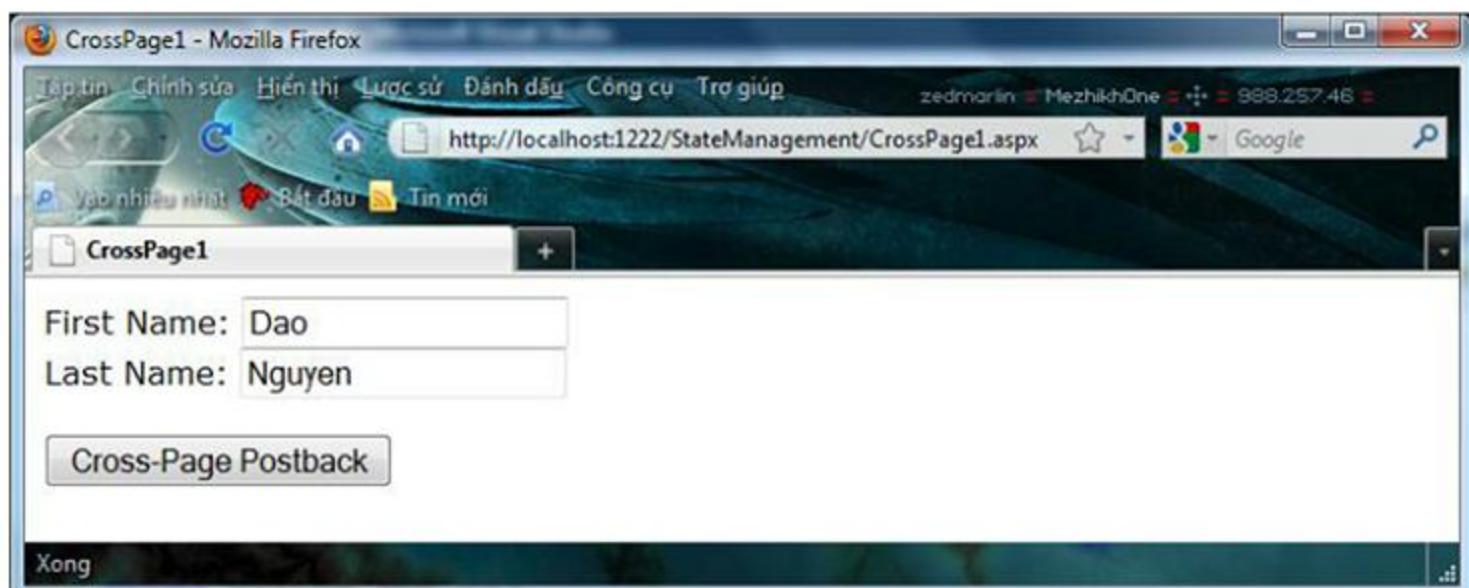
```

public string FullName {
    get { return txtFirstName.Text + " " + txtLastName.Text; }
}

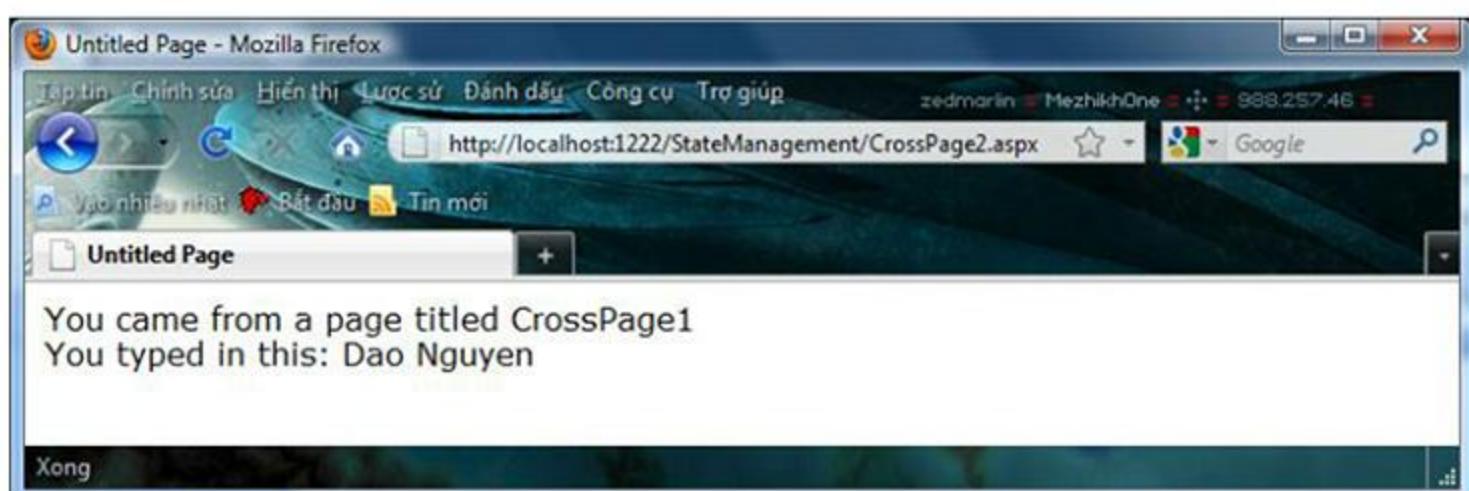
```

Thực thi trang CrossPage1.aspx, nội dung hiển thị như sau:

Nhập thông tin cho **FirstName** và **LastName**, sau đó click một trong hai nút lệnh đều chuyển đến trang **CrossPage2.aspx**



Sau khi click nút [**Cross-Page Postback**]



Trang **CrossPage2.aspx**:

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="CrossPage2.aspx.cs"
Inherits="CrossPage2" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>

```

```

<form id="form1" runat="server">
<div>
    <asp:Label ID="lblInfo" runat="server"></asp:Label></div>
</form>
</body>
</html>

```

Nội dung mã lệnh trang **CrossPage2.aspx.cs**:

```

public partial class CrossPage2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (PreviousPage != null)
        {

            lblInfo.Text = "You came from a page titled " +
                PreviousPage.Title + "<br />";

            CrossPage1 prevPage = PreviousPage as CrossPage1;
            if (prevPage != null)
            {
                lblInfo.Text += "You typed in this: " + prevPage.FullName +
                    "<br />";
            }
        }
    }
}

```

Trong mã lệnh trên nội dung FullName chỉ hiển thi khi trang CrossPage2.aspx được chuyển đến từ trang CrossPage1.aspx với một trong hai cách trên.

4.3.2. Sử dụng chuỗi truy vấn (Query String)

Thí dụ 3:

Tạo trang QueryStringSender.aspx, khi thực thi sẽ chuyển giá trị các điều khiển trên trang đến trang QuerStringRecipient.aspx qua chuỗi truy vấn gắn vào chuỗi url.

Nội dung trang **QueryStringSender.aspx**:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="QueryStringSender.aspx.cs" Inherits="QueryStringSender" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">  <title>Untitled Page</title> </head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:ListBox ID="lstItems" runat="server" Height="155px" Width="165px">
            </asp:ListBox><br /> <br />
            <asp:CheckBox ID="chkDetails" runat="server" Text="Show full details" /><br />
            <br />
            <asp:Button ID="cmdGo" runat="server" OnClick="cmdGo_Click" Text="View
Information" Width="165px" /><br />      <br />
            <asp:Label ID="lblError" runat="server" EnableViewState="False"></asp:Label>
        </div>
    </form>
</body>
</html>
```

Phần mã lệnh **QueryStringSender.aspx.cs**:

```
public partial class QueryStringSender: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!this.IsPostBack){
            // Add sample values.
            lstItems.Items.Add("Econo Sofa");
            lstItems.Items.Add("Supreme Leather Drapery");
            lstItems.Items.Add("Threadbare Carpet");
            lstItems.Items.Add("Antique Lamp");
            lstItems.Items.Add("Retro-Finish Jacuzzi"); }
    }
}
```

```

}

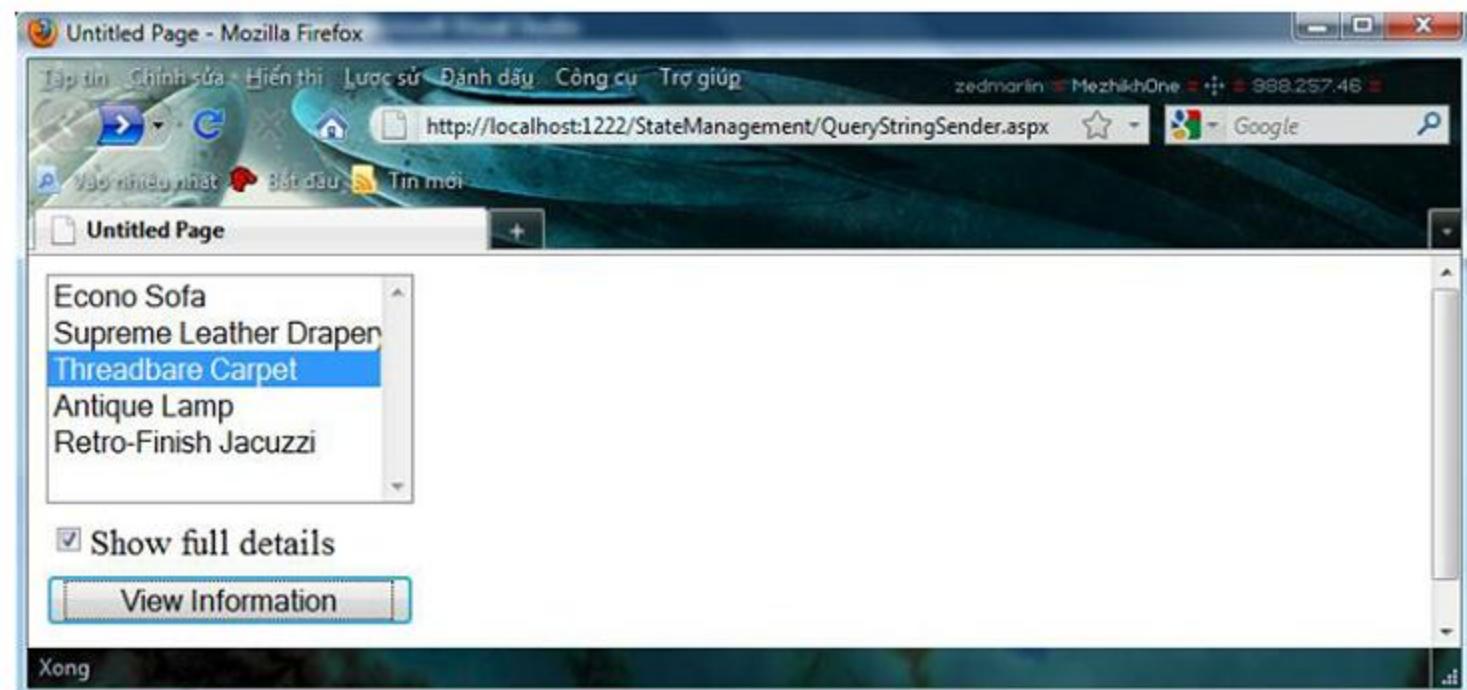
protected void cmdGo_Click(object sender, EventArgs e)
{
    if (lstItems.SelectedIndex == -1) {
        lblError.Text = "You must select an item.";
    }
    else
    {
        // Forward the user to the information page, with the query string
        data.

        string url = "QueryStringRecipient.aspx?";
        url += "Item=" + Server.UrlEncode(lstItems.SelectedItem.Text) +
        "&";

        url += "Mode=" + chkDetails.Checked.ToString();
        Response.Redirect(url);
    }
}

```

Khi click nút **[View Information]** sẽ chuyển đến trang `QueryStringRecipient.aspx` cùng với chuỗi truy vấn như sau:
`http://localhost:1222/StateManager/QueryStringRecipient.aspx?Item=Threadbare + Carpet&Mode=True`



Kết quả hiển thị của trang `QueryStringRecipient.aspx`:



Nội dung trang **QueryStringRecipient.aspx**:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="QueryStringRecipient.aspx.cs" Inherits="QueryStringRecipient" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div style="border-right: 2px solid; padding-right: 30px; border-top: 2px solid;
padding-left: 30px; font-weight: bold; font-size: 140%; padding-bottom: 30px; border-
left: 2px solid; padding-top: 30px; border-bottom: 2px solid; font-family: Verdana;
background-color: #FFFFD9">
            <asp:Label ID="lblInfo" runat="server" EnableViewState="False" ></asp:Label>
        </div>
    </form>
</body>
</html>
```

Phần mã lệnh của trang **QueryStringRecipient.aspx.cs**:

```
public partial class QueryStringRecipient : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        lblInfo.Text = "Item: " + Request.QueryString["Item"];
        lblInfo.Text += "<br />Show Full Record: ";
        lblInfo.Text += Request.QueryString["Mode"];
    }
}
```

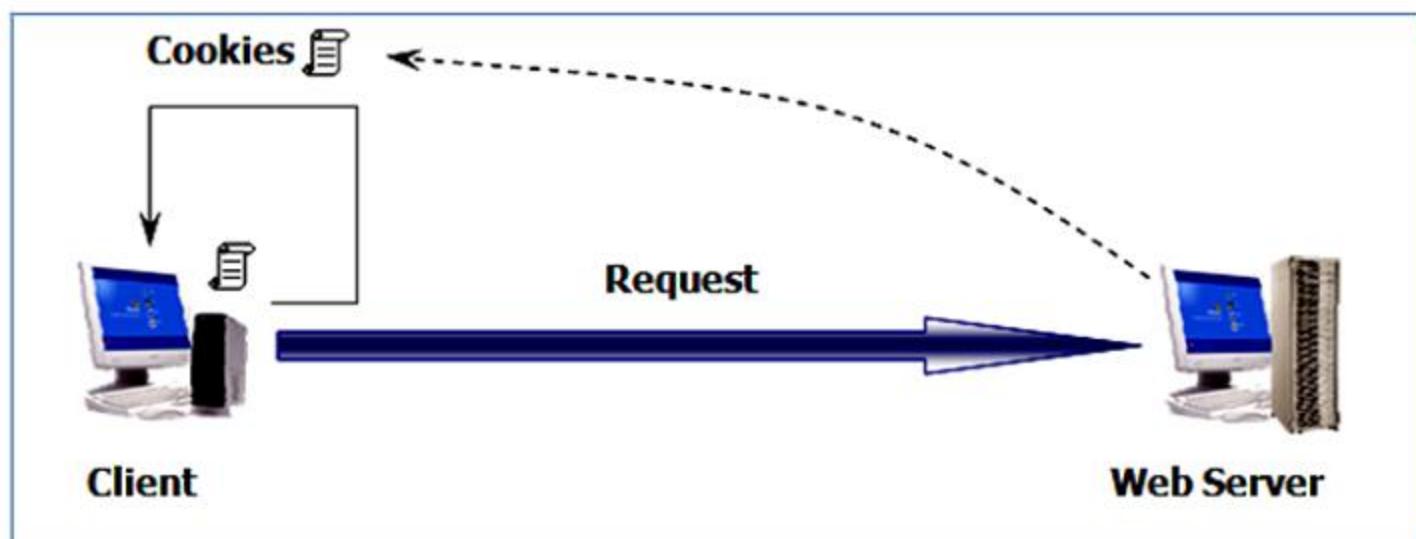
4.4. ĐỐI TƯỢNG COOKIES

Có lẽ bạn cũng đã từng đăng ký là một thành viên của một trang web hay một forum nào đó, và chắc cũng không ít lần ngạc nhiên khi bạn vừa yêu cầu đến một trang web hay forum mà bạn đã đăng ký trước đó, trang web nhận ngay ra, bạn chính là thành viên của họ và gửi ngay lời chào đến bạn, chẳng hạn: Chào Nguyễn Văn An.

Những thông tin của bạn khi đăng nhập hay khi đăng ký được lưu ngay chính tại máy của bạn. Những thông tin này được Web Server lưu tại máy Client được gọi là Cookies.

Không giống như đối tượng Session, đối tượng Cookies cũng được dùng để lưu trữ thông tin của người dùng, tuy nhiên, thông tin này được lưu ngay tại máy gửi yêu cầu đến Web Server.

Có thể xem một Cookie như một tập tin (với kích thước khá nhỏ) được Web Server lưu tại máy của người dùng. Mỗi lần có yêu cầu đến Web Server, những thông tin của Cookies cũng sẽ được gửi theo về Server.



Hình 4.2: Minh họa Cookies

- **Làm việc với Cookies**
 - **Thêm Cookies**

```
Response.Cookies.Add(<HttpCookie>)
```

Thí dụ:

```
HttpCookie ck = new HttpCookie("TenDangNhap");
ck.Value = txtTenDangNhap.Text;
ck.Expires = DateTime.Now.AddDays(15);
Response.Cookies.Add(ck);
```

Trong thí dụ trên, chúng ta đã tạo ra Cookies có tên là TenDangNhap lưu trữ tên đăng nhập của người dùng. Thông tin này sẽ được lưu trữ trên Cookies 15 ngày kể từ ngày hiện hành trên Web Server.

- **Lấy giá trị từ Cookies**

```
HttpCookie ck = Request.Cookies["TenDangNhap"];
string s = ck.Value;
```

Trong trường hợp Cookies chưa được lưu hoặc đã hết thời hạn duy trì tại Client, giá trị nhận được là null.

Thí dụ: Tạo trang

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="CookieExample.aspx.cs" Inherits="CookieExample" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
```

```
<body>
    <form id="form1" runat="server">
        <div>
            <div style="border-right: 2px solid; padding-right: 30px; border-top: 2px solid;
padding-left: 30px; font-weight: bold; font-size: 140%; padding-bottom: 30px; border-
left: 2px solid; padding-top: 30px; border-bottom: 2px solid; font-family: Verdana;
background-color: #FFFFD9">
                <asp:Label ID="lblWelcome" runat="server" EnableViewState="False"
                ></asp:Label>
            </div>    <br />
            Name:
            <asp:TextBox ID="txtName" runat="server" Width="178px"></asp:TextBox>
            <asp:Button ID="cmdStore" runat="server" OnClick="cmdStore_Click"
Text="Create Cookie" />
        </div>
    </form>
</body>
</html>
```

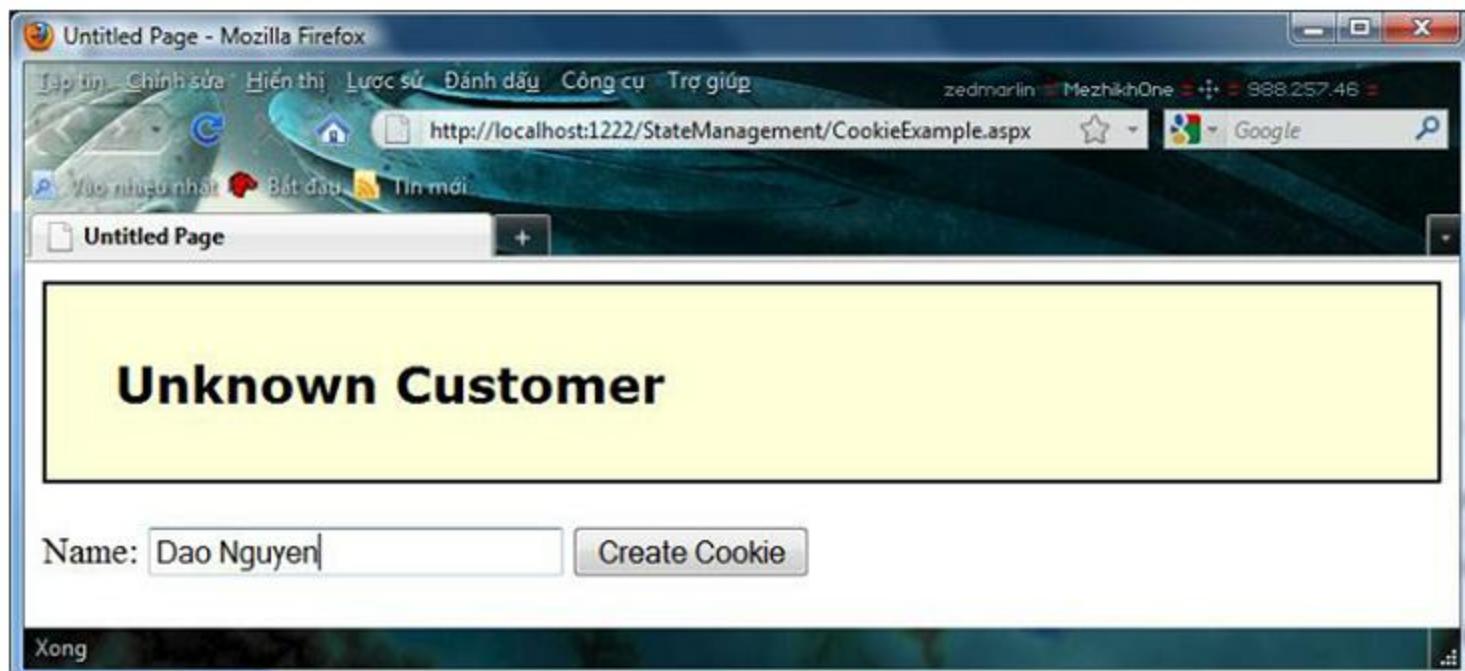
Khi click nút [Create Cookie], phần mã lệnh thực thi như sau:

```
public partial class CookieExample : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        HttpCookie cookie = Request.Cookies["Preferences"];
        if (cookie == null)
        {
            lblWelcome.Text = "<b>Unknown Customer</b>";
        }
        else
        {
            lblWelcome.Text = "<b>Cookie Found.</b><br><br>";
            lblWelcome.Text += "Welcome, " + cookie["Name"];
        }
    }

    protected void cmdStore_Click(object sender, EventArgs e)
    {
        // Check for a cookie, and only create a new one if
        // one doesn't already exist.
        HttpCookie cookie = Request.Cookies["Preferences"];
        if (cookie == null)
        {
            cookie = new HttpCookie("Preferences");
        }
        cookie["Name"] = txtName.Text;
        cookie.Expires = DateTime.Now.AddYears(1);
        Response.Cookies.Add(cookie);

        lblWelcome.Text = "<b>Cookie Created.</b><br><br>";
        lblWelcome.Text += "New Customer: " + cookie["Name"];
    }
}
```

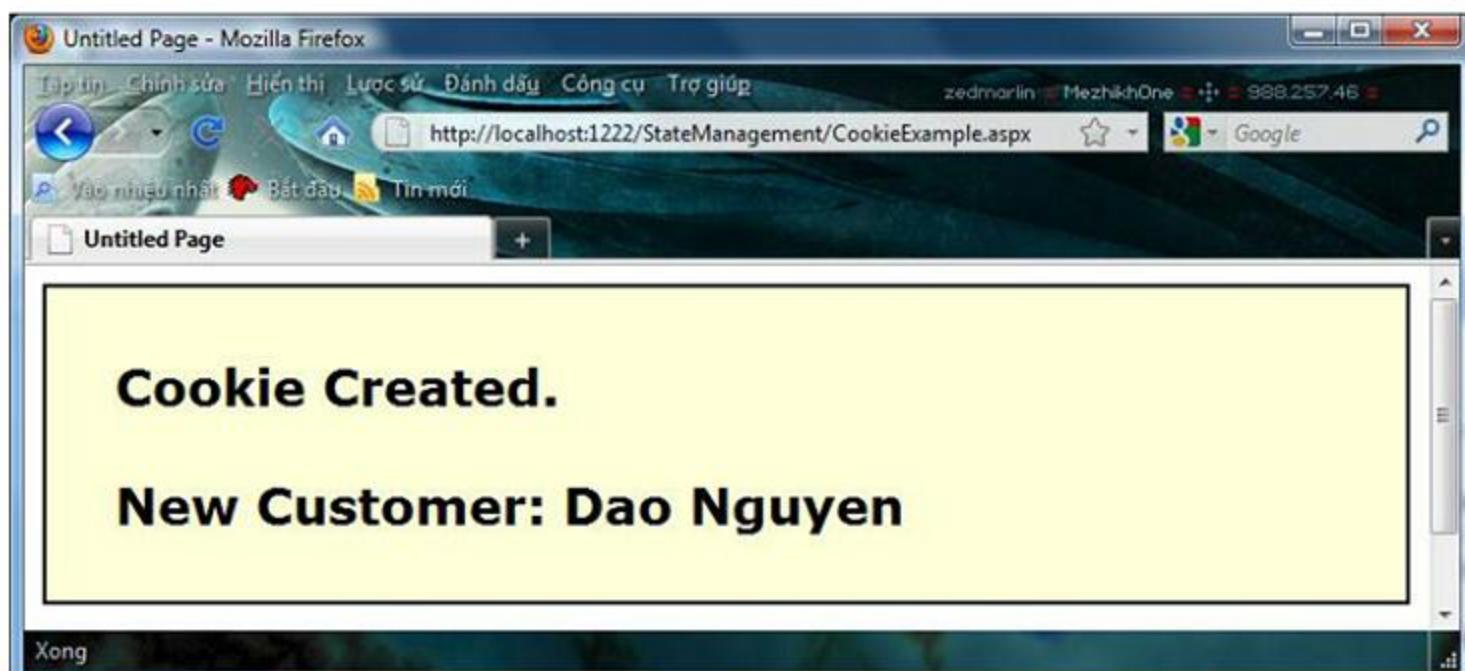
Khi thực thi trang CookieExample.aspx:



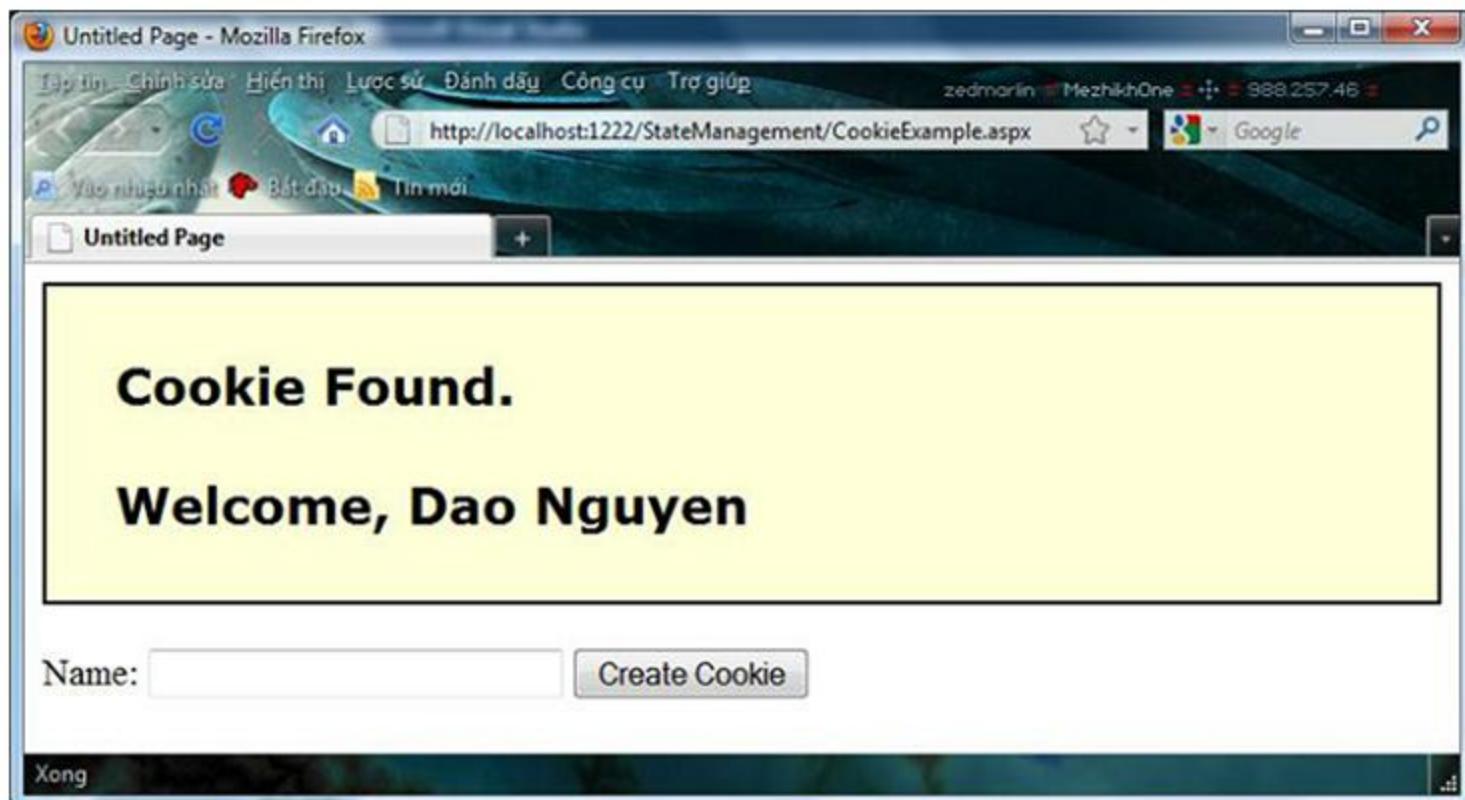
Click nút [Create Cookie], trình chủ (web server) sẽ tạo một Cookie tên Name có giá trị là Dao Nguyen ghi lại trên máy khách với thời hạn lưu trú là một năm.

```
cookie["Name"] = txtName.Text;  
cookie.Expires = DateTime.Now.AddYears(1);  
Response.Cookies.Add(cookie);
```

Kết quả hiển thị sau khi click nút [Create Cookie]:



Gọi thực thi lại trang CookieExample.aspx lần thứ hai trở đi, kết quả hiển thị trang:



4.5. ĐỐI TƯỢNG SESSION

Đối tượng Session được dùng để lưu trữ thông tin của người dùng trong ứng dụng. Thông tin được lưu trữ trong Session là của một người dùng trong một phiên làm việc cụ thể. Web Server sẽ tự động tạo một đối tượng Session cho mỗi người dùng mới kết nối vào ứng dụng và tự động hủy chúng nếu người dùng còn không làm việc với ứng dụng nữa.

Tuy nhiên, không giống như đối tượng Application, đối tượng Session không thể chia sẻ thông tin giữa những lần làm việc của người dùng, nó chỉ có thể cung cấp, trao đổi thông tin cho các trang trong lần làm việc tương ứng.

Trong ứng dụng web, đối tượng Session giữ vai trò khá quan trọng. Do sử dụng giao thức HTTP, một giao thức phi trạng thái, Web Server hoàn toàn không ghi nhớ những gì giữa những lần yêu cầu của Client. Đối tượng Session tỏ ra khá hữu hiệu trong việc thực hiện "lưu vết và quản lý thông tin của người dùng".

Thuộc tính & Phương thức

- **Thuộc tính Timeout**

Quy định khoảng thời gian (tính bằng phút) mà Web Server duy trì đối tượng Session nếu người dùng không gửi yêu cầu nào về lại Server. Giá trị mặc định của thuộc tính này là 20.

Nếu không có yêu cầu nào kể từ lần yêu cầu sau cùng một khoảng thời gian là `<Timeout>` phút, đối tượng Session mà Web server cấp cho lần làm việc đó sẽ tự động được giải phóng. Những yêu cầu sau đó được

Web server coi như là một người dùng mới, và đương nhiên sẽ được cấp một đối tượng Session mới.

▪ Phương thức Abandon

Như các bạn đã biết, trong khoảng thời gian <Timeout> phút kể từ lần yêu cầu sau cùng của Client, đối tượng Session vẫn được duy trì dù cho không có sự tương tác nào của Client. Điều này đồng nghĩa với việc Web server phải sử dụng một vùng nhớ để duy trì đối tượng Session trong một khoảng thời gian tương ứng.

Phương thức Abandon của đối tượng Session sẽ giải phóng vùng nhớ được dùng để duy trì đối tượng Session trên Web Server ngay khi được gọi thực hiện. Những yêu cầu sau đó được Web server coi như là một người dùng mới.

Sử dụng biến toàn cục với Session

Tạo biến Session

```
Session["Tên biến"] = <giá trị>
```

Lấy giá trị từ biến Session

```
<biến> = Session["Tên biến"]
```

Thí dụ: Minh họa sử dụng biến session lưu thông tin của người dùng khi đăng nhập

Tạo ứng dụng Web gồm các trang ASP.NET được mô tả như sau:

Trang DangNhap.aspx

Control	Tên thuộc tính	Giá trị thuộc tính
Lable	Text	Tên đăng nhập
Lable	Text	Mật khẩu
TextBox	ID	txtTenDangNhap
TextBox	ID	txtMatKhau
Button	Text	Đăng nhập
	ID	btnDangNhap

Bảng mô tả các thuộc tính của các controls trang DangNhap.aspx.

Tên đăng nhập :

Mật khẩu :

Đăng nhập

Thiết kế giao diện trang DangNhap.aspx

Xử lý sự kiện

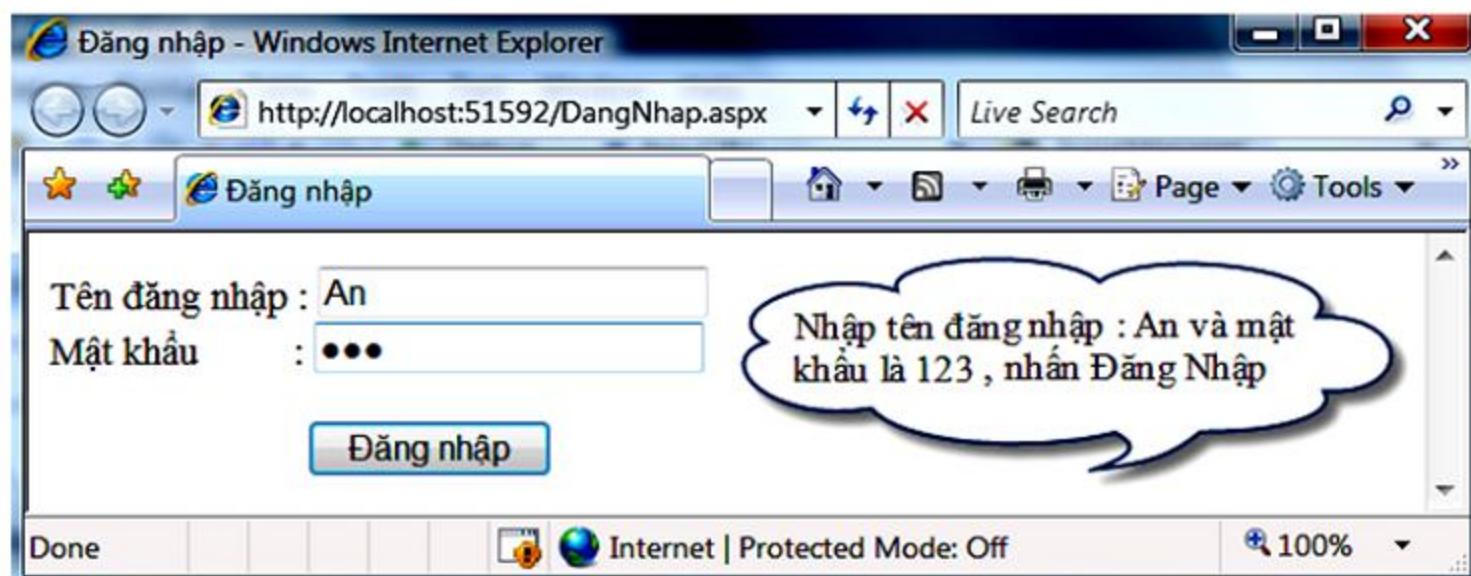
```
public partial class DangNhap: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e) { }

    protected void btnDangNhap_Click(object sender, EventArgs e)
    {
        if (txtTenDangNhap.Text == "An" && txtMatKhau.Text == "123")
        {
            Session["TenDangNhap"] = txtTenDangNhap.Text;
            Response.Redirect("ChaoMung.aspx");
        }
    }
}
```

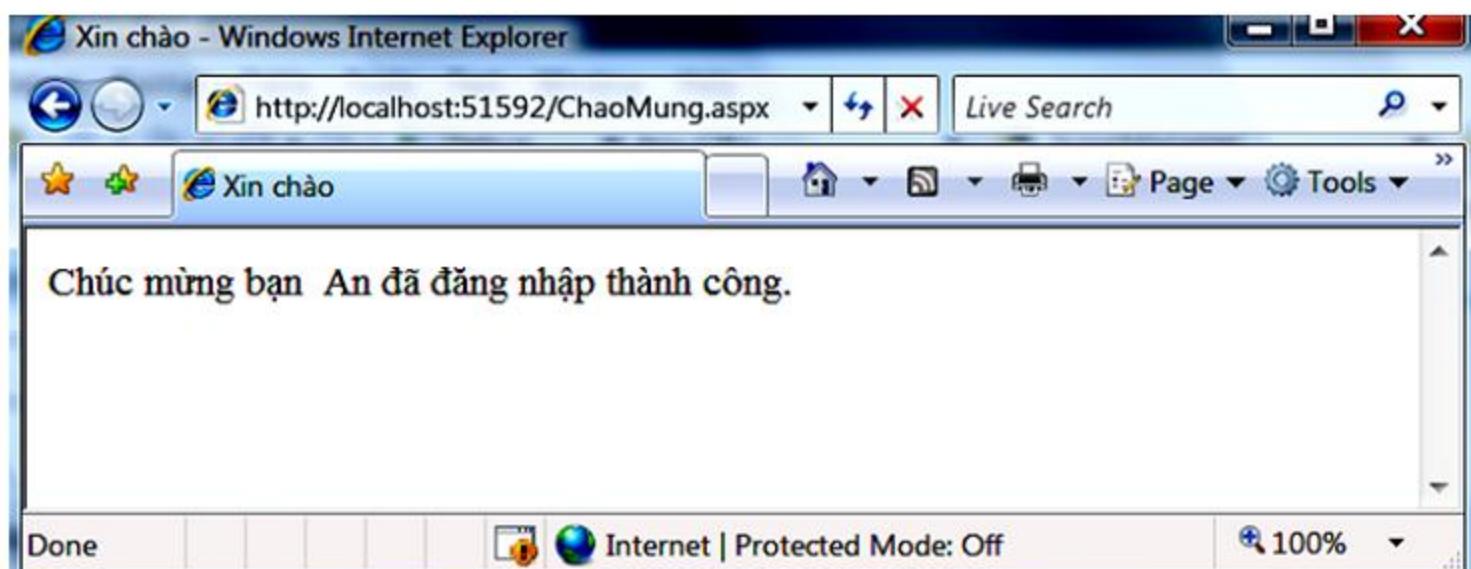
Trang ChaoMung.aspx:

```
public partial class ChaoMung: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        //Lấy giá trị từ biến Session
        string s = Session["TenDangNhap"].ToString();
        Response.Write("Chúc mừng bạn &nbsp;" + s + "&nbsp;đã đăng
nhập thành công.");
    }
}
```

Kết quả thi hành các trang ASP.NET

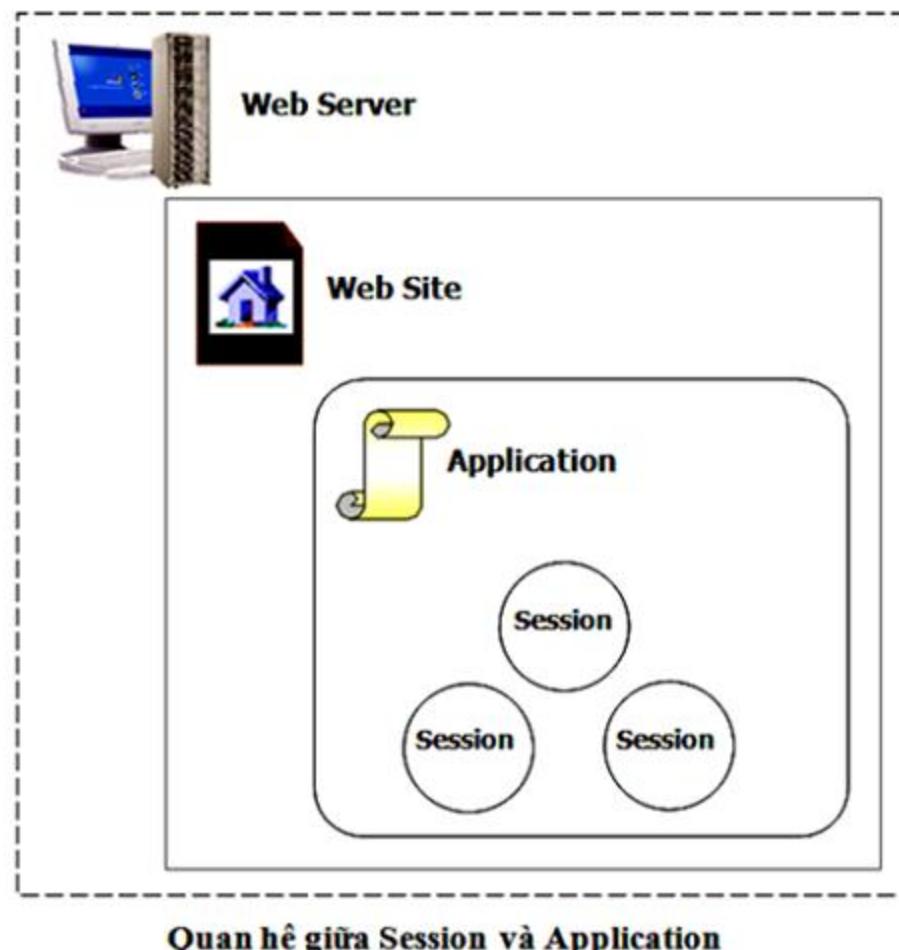


Kết quả sau khi nhập thông tin và click nút [Đăng nhập]



4.6. ĐỐI TƯỢNG APPLICATION

Application và Session là hai đối tượng khá quan trọng trong ứng dụng web, giúp các trang aspx có thể liên kết và trao đổi dữ liệu cho nhau. Trong phần này, chúng ta sẽ tìm hiểu và sử dụng hai đối tượng này trong ứng dụng.



Hình 4.3: Minh họa quan hệ của đối tượng Session và Application

Đối tượng Application được sử dụng để quản lý tất cả các thông tin của một ứng dụng web. Thông tin được lưu trữ trong đối tượng Application có thể được xử lý trong bất kỳ trang aspx nào trong suốt chu kỳ sống của ứng dụng.

▪ Sử dụng biến Application

Tạo biến Application

```
Application["Tên biến"] = <giá trị>
```

Lấy giá trị từ biến Application

```
<biến> = Application["Tên biến"]
```

Thí dụ:

```
Application.Lock()  
Application["SoLanTruyCap"] = 0  
Application["SoNguoiOnLine"] = 0  
Application.UnLock()
```

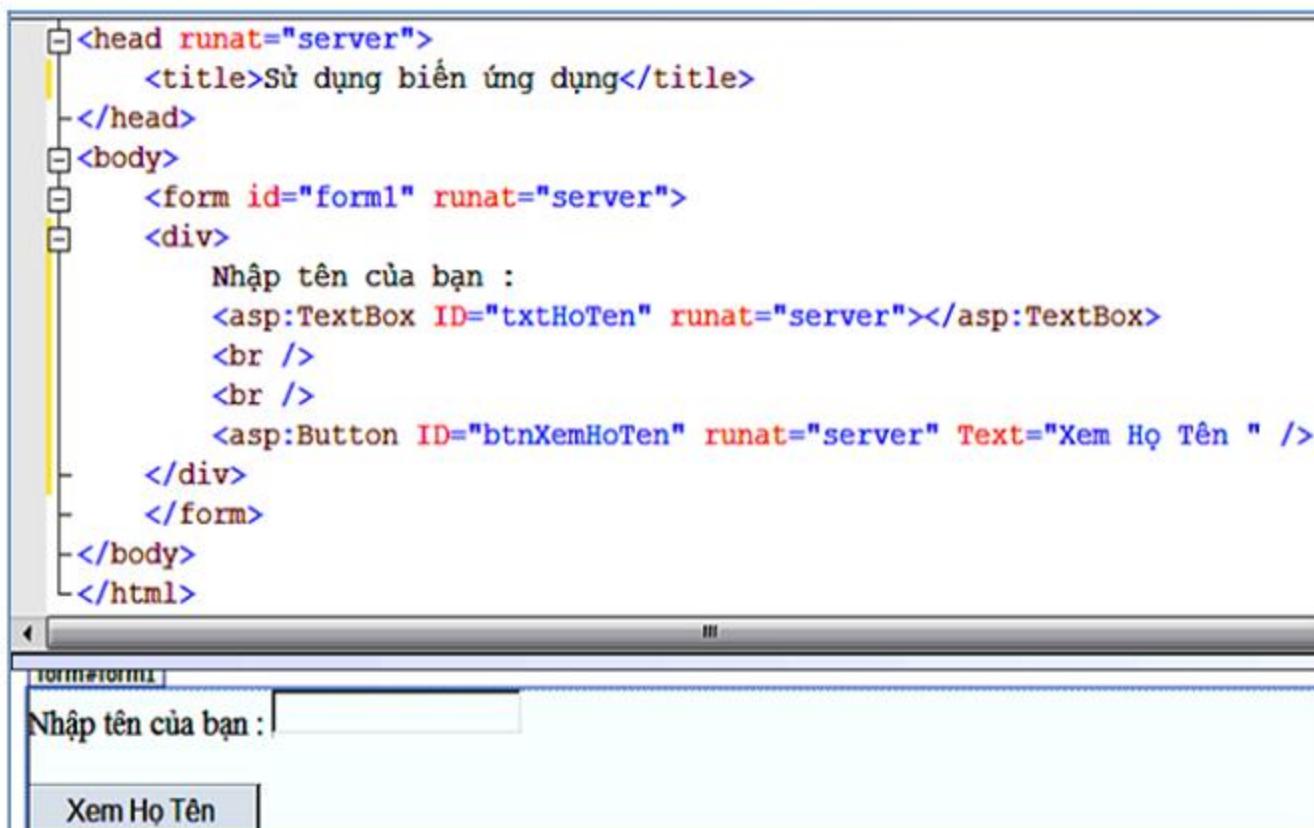
Chú ý:

Do tại một thời điểm có thể có nhiều người cùng lúc truy cập và thay đổi giá trị của các thông tin được lưu trong đối tượng Application, chúng ta nên sử dụng bộ lệnh Lock và UnLock ngay trước và sau khi cập nhật giá trị của biến Application.

Biến Application có thể được sử dụng ở bất kỳ trang nào và được duy trì trong suốt chu kỳ sống của ứng dụng.

▪ Thí dụ minh họa sử dụng biến Application

Tạo một ứng dụng Web gồm hai trang ASP.NET sau đây:



Hình 4.4: Thiết kế trang SuDungApplication.aspx

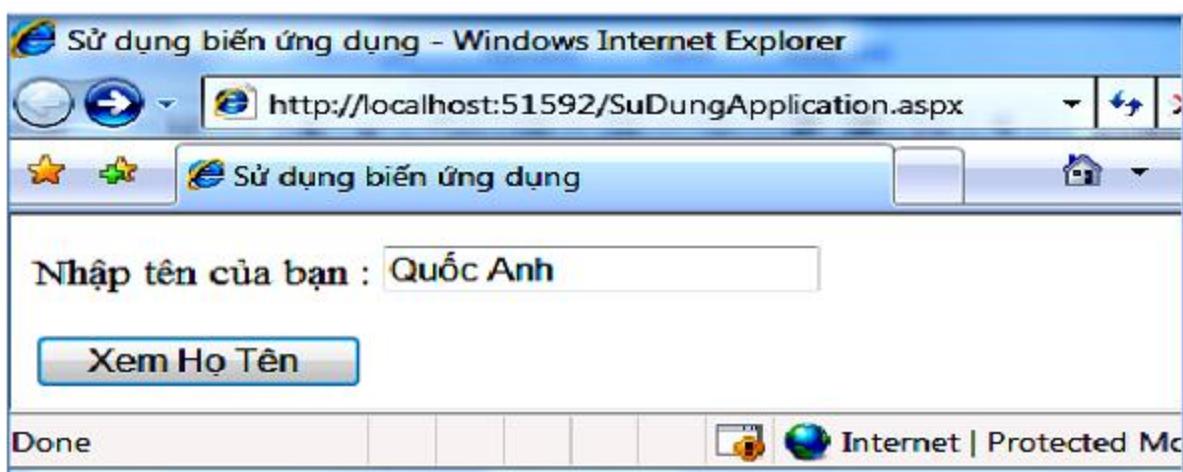
Xử lý sự kiện trang **SuDungApplication.aspx**

```
public partial class SuDungApplication: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e){ }
    protected void btnXemHoTen_Click(object sender, EventArgs e)
    {
        //Khai báo biến ứng dụng
        Application["HoTen"] = txtHoTen.Text;
        Response.Redirect("~/LayGiaTri.aspx");
    }
}
```

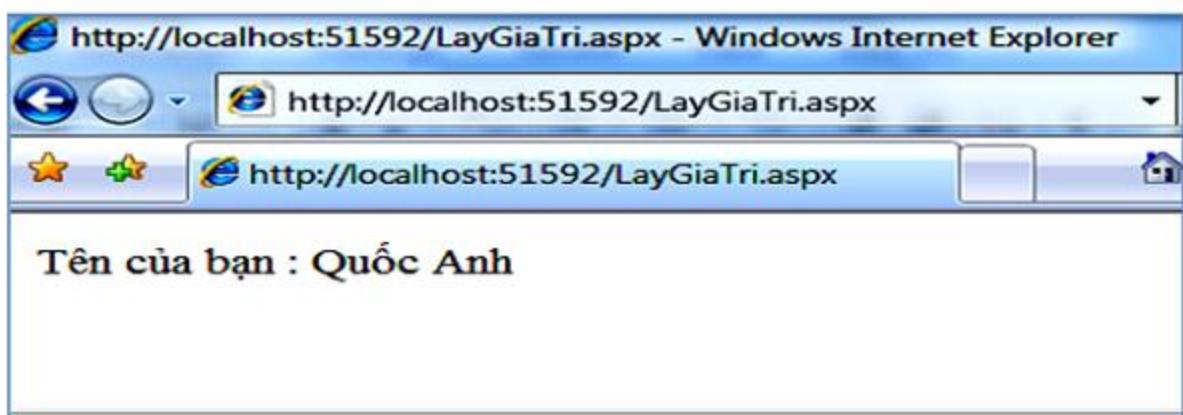
Tạo trang **LayGiaTri.aspx** và viết lệnh xử lý sự kiện như sau:

```
public partial class LayGiaTri: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string s = Application["HoTen"].ToString();
        Response.Write("Tên của bạn: "+s);
    }
}
```

Kết quả thực hiện:



Trang *SuDungApplication.aspx*



Trang *LayGiaTri.aspx*

4.7. TẬP TIN GLOBAL.ASAX

Tập tin Global.asax được dùng để:

- Khai báo và khởi tạo giá trị cho các biến Application, Session.
- Viết xử lý cho các sự kiện của hai đối tượng Application và Session.

➤ Các sự kiện trong tập tin Global.asax

Application_Start: Chỉ xảy ra một lần đầu tiên khi bất kỳ trang nào trong ứng dụng được gọi.

```
protected void Application_Start(object sender, EventArgs e)
{
    //Khai báo biến Application đếm số người truy cập
    Application["SoNguoiTruyCap"] = 0;
    Application["SoNguoiOnline"] = 0;
}
```

Session_Start: Xảy ra khi có một người dùng mới yêu cầu đến bất kỳ trang aspx của ứng dụng. Khi Session_Start xảy ra, một giá trị duy nhất (SessionID) sẽ được tạo cho người dùng, và giá trị này được sử dụng để quản lý người dùng trong quá trình làm việc với ứng dụng.

```
protected void Session_Start(object sender, EventArgs e)
{
    //Tăng giá trị biến Application
    Application["SoNguoiTruyCap"]=(int)Application["SoNguoiTruyCap"]+1;
    Application["SoNguoiOnline"]=(int)Application["SoNguoiOnline"] +1 ;
}
```

Application_BeginRequest: Xảy ra khi mỗi khi có Postback về Server.

Application_Error: Xảy ra khi có lỗi phát sinh trong quá trình thi hành.

Session_End: Xảy ra khi phiên làm việc không có gửi yêu cầu hoặc làm tươi trang aspx của ứng dụng web trong một khoảng thời gian (mặc định là 20 phút).

```
protected void Session_End (object sender, EventArgs e)
{
    //giảm giá trị biến Application
    Application["SoNguoiOnline"]=(int)Application["SoNguoiOnline"]-1 ;
}
```

Application_End: Xảy ra khi dừng hoạt động của WebServer. Ví dụ xử lý ghi nhận thông tin Số lượt truy cập vào cơ sở dữ liệu (nếu cần).

4.8. TẬP TIN WEB.CONFIG

➤ Cấu trúc tập tin web.config

Web.config là một tập tin văn bản được sử dụng để lưu trữ thông tin cấu hình của một ứng dụng, được tự động tạo ra khi chúng ta tạo mới ứng dụng web. Tập tin web.config được viết theo định dạng XML.

Web.config được tạo kế thừa các giá trị từ tập tin
Windows\Microsoft.NET\Framework\
[FrameworkVersion]\CONFIG\machine.config

➤ Các cấu hình mặc định

<compilation debug="false"/>: quy định ngôn ngữ mặc định của ứng dụng C#. debug:

Bật/tắt chế độ debug của ứng dụng

<customErrors mode="RemoteOnly"/>

Đây là một cấu hình khá cần thiết cho ứng dụng Web. Hiệu chỉnh cấu hình này cho phép chúng ta quản lý việc xử lý lỗi khi có lỗi phát sinh trong ứng dụng.

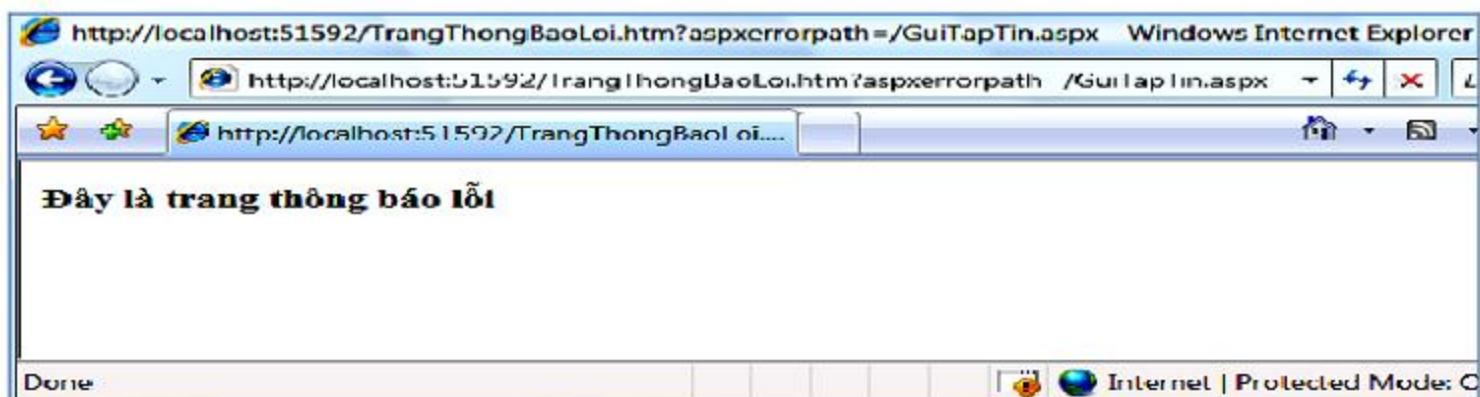
- Thuộc tính mode có các giá trị: RemoteOnly, On và Off.

RemoteOnly: Cho phép người dùng thấy thông báo lỗi của hệ thống hoặc trang thông báo lỗi được chỉ định qua defaultRedirect (nếu có).

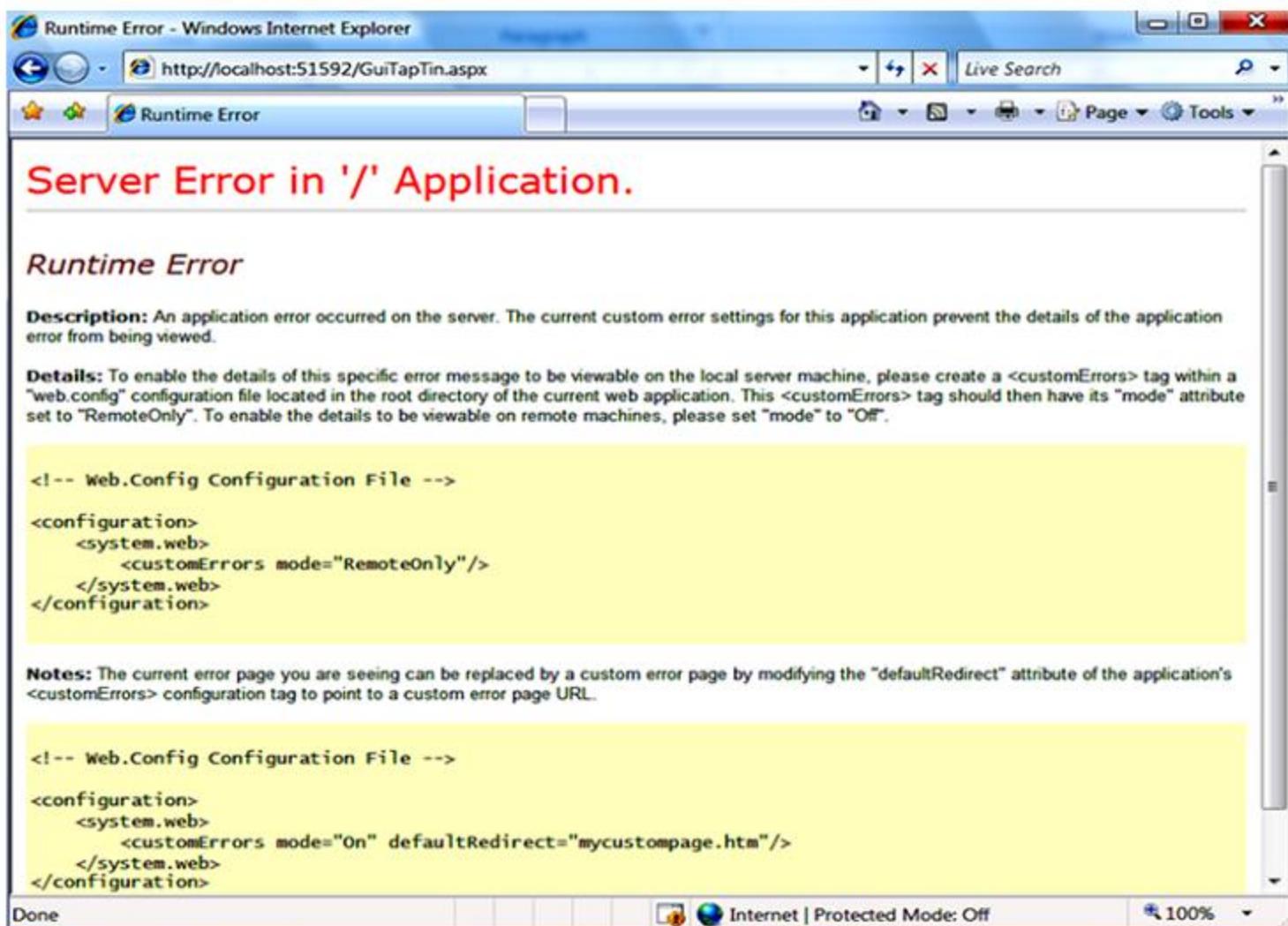
Thông báo lỗi gồm: Mã lỗi và mô tả lỗi tương ứng

```
<customErrors mode="RemoteOnly"  
defaultRedirect="TrangThongBaoLoi.htm"/>
```

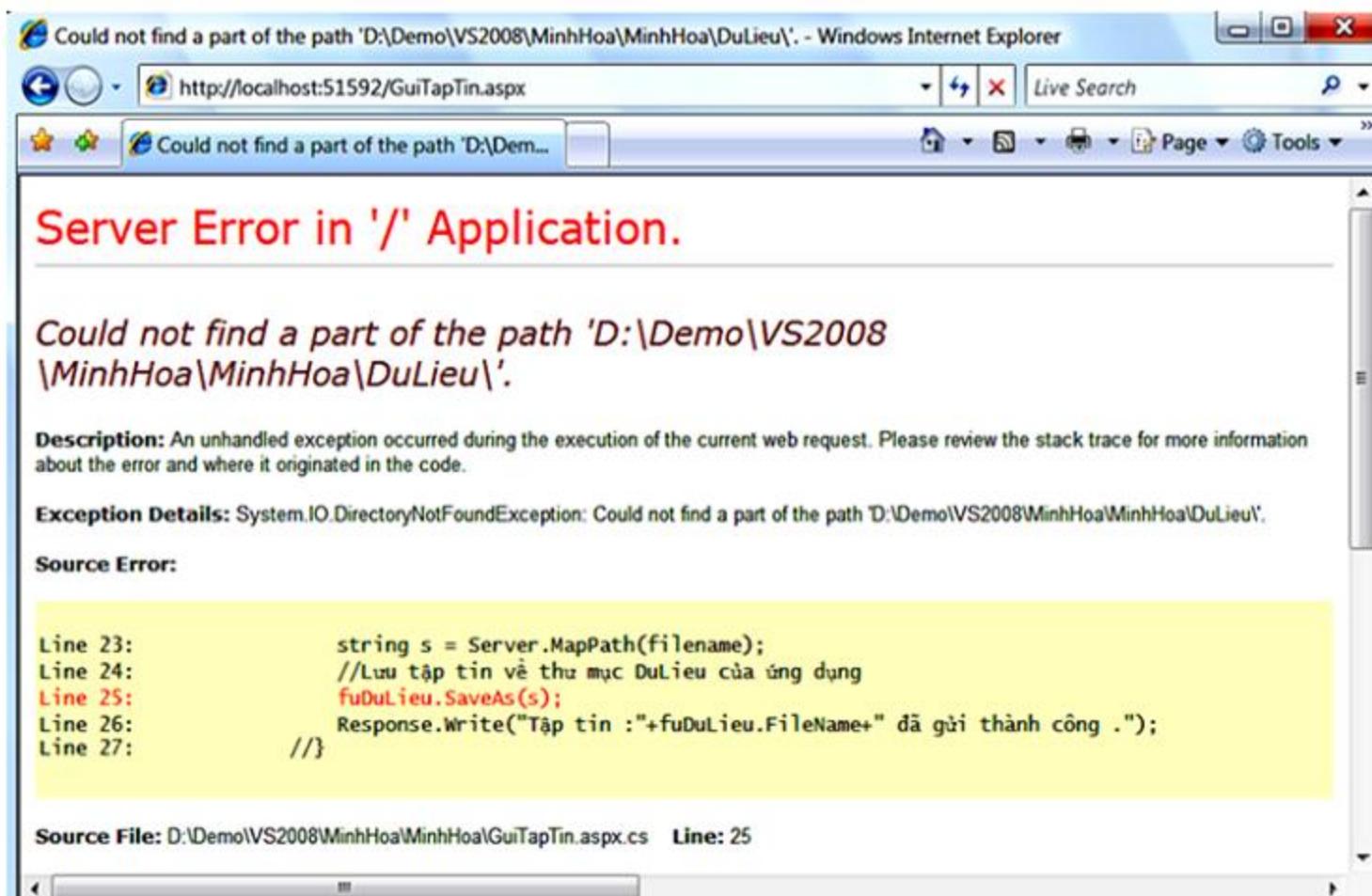
- On**: Tùy theo giá trị của defaultRedirect mà có các trường hợp tương ứng:
 - Có quy định trang thông báo lỗi qua defaultRedirect: Hiển thị trang thông báo lỗi.



Không có thuộc tính defaultRedirect: Hiển thị trang báo lỗi nhưng không có hiển thị mã lỗi và mô tả lỗi.



- **Off:** Hiển thị thông báo lỗi của trang aspx (nếu xảy ra lỗi).



<sessionState/>

```
<sessionState mode="InProc"
stateConnectionString="tcpip=127.0.0.1:42424"
sqlConnectionString="data source=127.0.0.1;Trusted_Connection=yes"
cookieless="false" timeout="20" />
```

- **mode**: Thuộc tính này có ba giá trị: InProc, sqlserver (lưu trong database), và stateserver (lưu trong bộ nhớ)
- **stateConnectionString**: Cấu hình địa chỉ và cổng (port) của máy để lưu trữ thông tin của Session trong vùng nhớ (nếu chức năng này được chọn).
- **sqlConnectionString**: Cấu hình kết nối đến SQL Server được dùng để lưu thông tin Session (nếu chức năng này được chọn).
- **cookieless**: Nếu giá trị của thuộc tính này = True, thông tin cookie sẽ được lưu trữ trong URL, ngược lại, nếu = False, thông tin cookies sẽ được lưu trữ tại client (nếu client có hỗ trợ)
- **timeout**: Khoảng thời gian (tính bằng phút) mà đối tượng Session được duy trì. Sau khoảng thời gian này, đối tượng Session sẽ bị hủy. Giá trị mặc định của thuộc tính này là 20.

BÀI TẬP CHƯƠNG 4

Bài 1: Thực hành lại các bài tập thí dụ trong bài học, có thể hiệu chỉnh lại theo ý học viên.

Bài 2: Đọc và giải thích nội dung của

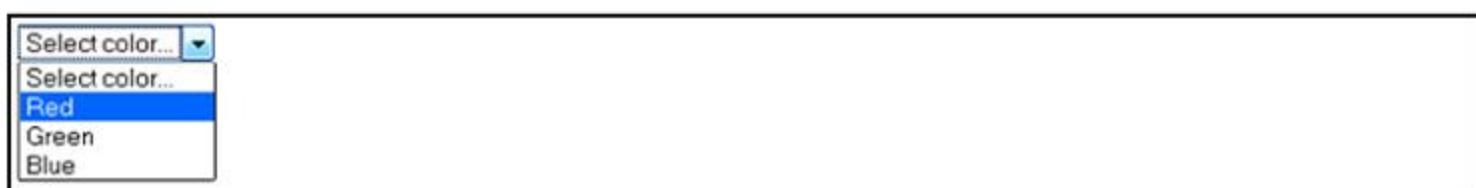
Phần mã lệnh trang SessionEx1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="SessionEx1.aspx.cs"
Inherits="SessionEx1" %>

<!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 id="Head1" runat="server">
    <title>Sessions</title>
</head>
<body runat="server" id="BodyTag">
<div>
    <form id="form1" runat="server">
        <asp:DropDownList runat="server" id="ColorSelector" autopostback="true"
onselectedindexchanged="ColorSelector_IndexChanged">
            <asp:ListItem value="White" selected="True">Select color...</asp:ListItem>
            <asp:ListItem value="Red">Red</asp:ListItem>
            <asp:ListItem value="Green">Green</asp:ListItem>
            <asp:ListItem value="Blue">Blue</asp:ListItem>
        </asp:DropDownList>
    </form>
</div>
</body>
</html>
```

Trang thiết kế của SessionEx1.aspx



Phần mã lệnh trang SessionEx1.aspx.cs

```
public partial class SessionEx1 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["BackgroundColor"] != null)
        {
            ColorSelector.SelectedValue = Session["BackgroundColor"].ToString();
            BodyTag.Style["background-color"] = ColorSelector.SelectedValue;
        }
    }

    protected void ColorSelector_IndexChanged(object sender, EventArgs e)
    {
        BodyTag.Style["background-color"] = ColorSelector.SelectedValue;
        Session["BackgroundColor"] = ColorSelector.SelectedValue;
    }
}
```

Khi chọn màu Green từ DropDownList, kết quả thực thi trang:



Kết thúc ứng dụng và chạy lại ứng dụng nà trong cùng một cửa sổ trình duyệt đang mở, trang hiển thị ra sao? Giải thích.

Nếu đóng cửa sổ trình duyệt và thực thi lại ứng dụng, trang hiển thị ra sao? Giải thích.

Bài 3: Tạo một ứng dụng ASP.NET về đối tượng Application và tập tin Global.asax

- Tạo trang Global.asax

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

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

```
void Application_Start(object sender, EventArgs e)
{
    int SiteHitCounter=500;
    int CurrentUsers=100;
    Application["SiteHitCounter"] = SiteHitCounter;
    Application["CurrentUsers"] = CurrentUsers;
}

void Application_End(object sender, EventArgs e)
{
    HitCountClass cls = new HitCountClass();
}

void Application_Error(object sender, EventArgs e)
{
    // Code that runs when an unhandled error occurs
}

void Session_Start(object sender, EventArgs e)
{
    Application["SiteHitCounter"] =
        (int)Application["SiteHitCounter"] + 1;
    Application["CurrentUsers"] =
        (int)Application["CurrentUsers"] + 1;
}

void Session_End(object sender, EventArgs e)
{
    Application["CurrentUsers"] =
        (int)Application["CurrentUsers"] - 1;
    // Code that runs when a session ends.
    // Note: The Session_End event is raised only when the sessionstate mode
    // is set to InProc in the Web.config file. If session mode is set to StateServer
    // or SQLServer, the event is not raised.
}

</script>
```

- Tạo trang ApplicationPage.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="ApplicationPage.aspx.cs" Inherits="ApplicationPage" %>
<!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>
            <asp:Literal ID="Literal1" runat="server"></asp:Literal><br />
            <br />
            <asp:Literal ID="Literal2" runat="server"></asp:Literal>&ampnbsp</div>
        </form>
    </body>
</html>
```

- Tạo trang ApplicationPage.aspx.cs

```
public partial class ApplicationPage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Literal1.Text = "Hit Counter: " +
        Application["SiteHitCounter"].ToString();

        Literal2.Text = "Current users: " +
        Application["CurrentUsers"].ToString();
    }
}
```

Kết quả thực thi trang ApplicationPage.aspx

Hit Counter: 501

Current users: 101

Chương 5

SỬ DỤNG CÁC VALIDATION CONTROLS

Các vấn đề chính sẽ được đề cập:

- ✓ *Mô tả được các thuộc tính và sử dụng các Validation Controls*

Kết thúc bài này các bạn có thể:

- *Vận dụng các Validation Controls để kiểm chứng thông tin các trang ASP.NET.*

Khi xây dựng ứng dụng, chúng ta nên kiểm tra dữ liệu nhập từ người dùng để hạn chế các sai sót dữ liệu nhập nhằm đảm bảo việc thực hiện xử lý dữ liệu được chính xác theo các yêu cầu nghiệp vụ. Nếu chúng ta viết mã để kiểm tra phải mất nhiều thời gian (sử dụng JavaScript hoặc VBScript). Webform hỗ trợ các validation controls để kiểm tra dữ liệu nhập từ người dùng trong các sever controls, mục đích là tránh để người dùng nhập sai hoặc không được bỏ trống các thông tin quan trọng bắt buộc,...

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 ASP.NET đã cung cấp cho ta những điều khiển kiểm tra tính hợp lệ của các điều khiển nhập liệu trên Form. Trong bài 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 thí dụ bạn sẽ tạo một Ajax Validator để kiểm tra nhập liệu phía Client.

Có sáu điều khiển Validation trong .NET framework 3.5:

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

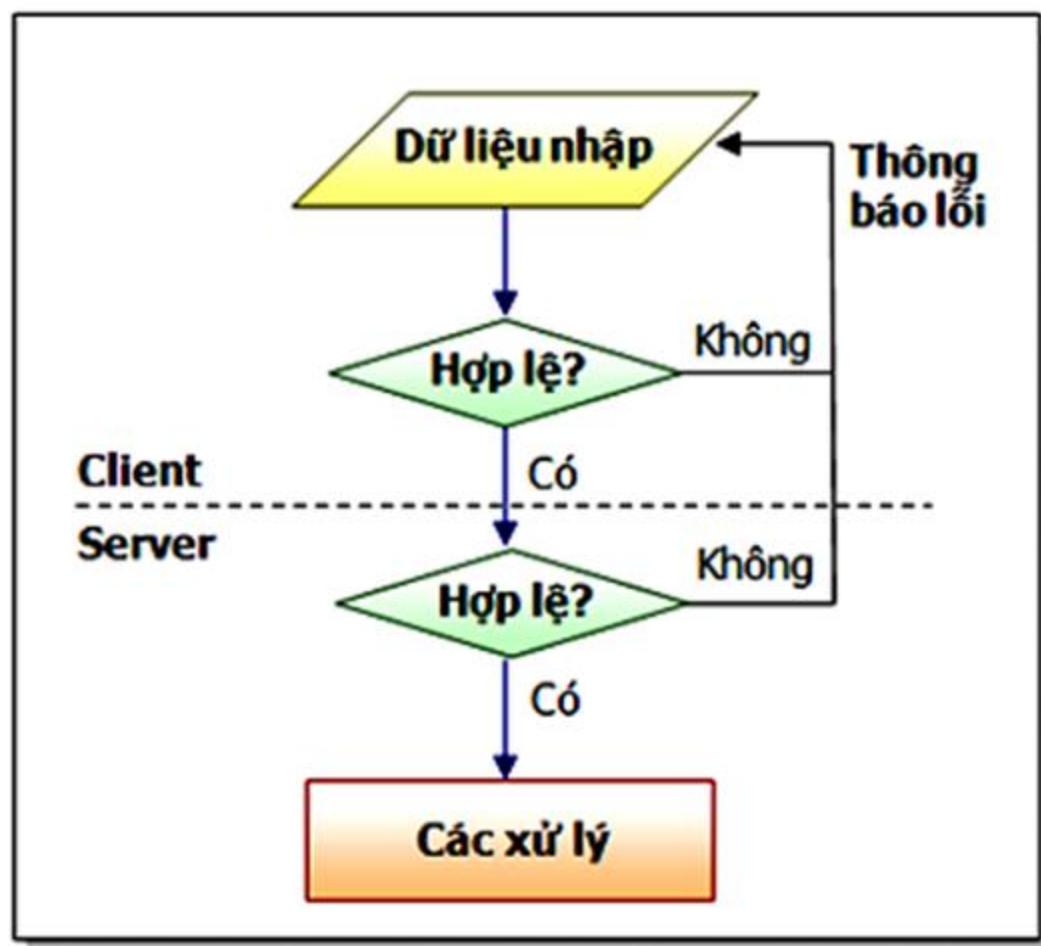
RangeValidator: Kiểm tra giá trị nhập vào có nằm trong một khoảng nhỏ nhất và lớn nhất định trước hay không.

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 một biểu thức quy tắc nào đấy có thể là email, số điện thoại, số tài khoản ngân hàng,...

CustomValidator: Bạn có thể tùy chỉnh đối tượng Validator theo ý của mình.

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



Hình 5.1: Sơ đồ kiểm tra dữ liệu nhập trên Form tại Client và Server

Như các bạn đã biết, mỗi khi PostBack về Server, trang Web luôn kiểm tra tính hợp lệ dữ liệu (nếu có yêu cầu khi thiết kế). Nếu dữ liệu không hợp lệ (bỏ trống, vi phạm miền giá trị, mật khẩu nhập lại không đúng, ...), trang web sẽ không thể PostBack về Server.

Các thuộc tính chung của các validation control

Thuộc tính	Ý nghĩa
ControlToValidate	Tên điều khiển cần kiểm tra. Đây là thuộc tính mà các bạn phải xác định khi sử dụng Validation Control.
Text	Chuỗi thông báo xuất hiện khi có lỗi
ErrorMessage	Chuỗi thông báo xuất hiện trong điều khiển Validation Summary Giá trị này sẽ được hiển thị tại vị trí của điều khiển nếu chúng ta không gán giá trị cho thuộc tính Text
Display	Quy định hình thức hiển thị: <ul style="list-style-type: none">▪ None: Không hiển thị thông báo lỗi (vẫn có kiểm tra dữ liệu)

	<ul style="list-style-type: none"> ▪ Static: Trong trường hợp không có vi phạm dữ liệu, điều khiển không có hiển thị nhưng vẫn chiếm vị trí như trong lúc thiết kế ▪ Dynamic: Trong trường hợp không có vi phạm dữ liệu, điều khiển không chiếm dụng vị trí trên màn hình
EnableClientScript	Có cho phép thực hiện kiểm tra ở phía Client hay không? Giá trị mặc định là true - có kiểm tra

5.1. REQUIREDFIELDVALIDATOR

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

2. Cách sử dụng: Đưa điều khiển **RequiredFieldValidator** từ ToolBox (trong phần Validation) vào trong Form và thêm vào cho nó hai thuộc tính:

- **ControlToValidate:** chỉ đến điều khiển nhập liệu sẽ được kiểm tra.
- **Text(hoặc ErrorMessage):** Thông báo lỗi khi kiểm tra có lỗi xảy ra.

Thí dụ 1

Tạo trang RequiredFieldValidate1.aspx có sử dụng điều khiển kiểm chứng RequiredFieldValidator cho điều khiển nhập TextBox:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="RequiredFieldValidate1.aspx.cs" Inherits="RequiredFieldValidate1" %>

<!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>
<script runat="server">
    void ValidateBtn_Click(Object sender, EventArgs e)
    {
        if (Page.IsValid)
    }
</script>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:RequiredFieldValidator ID="RequiredFieldValidator1" ControlToValidate="txtName" ErrorMessage="Nhập tên" runat="server"/>
            <asp:TextBox ID="txtName" runat="server" />
            <br/>
            <asp:Button ID="btnSubmit" Text="Submit" OnClick="ValidateBtn_Click" runat="server" />
        </div>
    </form>
</body>
</html>
```

```

{
    lblOutput.Text = "Required field is filled!";
}
else
{
    lblOutput.Text = "Required field is empty!";
}
}

</script>
</head>
<body>

<form id="form1" runat="server">
<div>
<h3>RequiredField Validator Example</h3>
<table style="background-color:#eeeeee; padding:10">
<tr valign="top">
<td colspan="3">
<asp:Label ID="lblOutput"
    Text="Fill in the required field below"
    runat="server"
    AssociatedControlID="TextBox1"/>
<br />
</td>
</tr>
<tr>
<td colspan="3">
<b>Credit Card Information</b>
</td>
</tr>
<tr>
<td align="right">
Card Number:
</td>
<td>

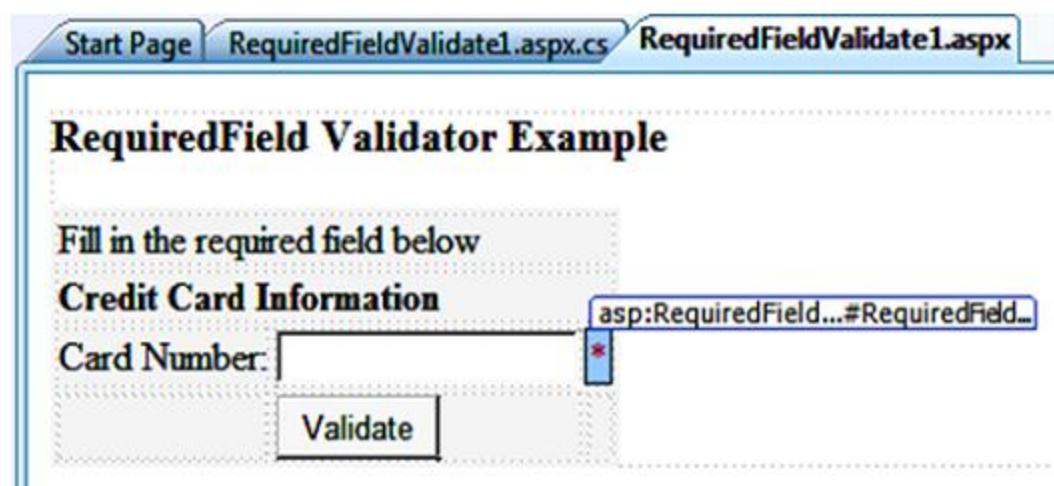
```

```

<asp:TextBox id="TextBox1"
    runat="server"/>
</td>
<td>
    <asp:RequiredFieldValidator id="RequiredFieldValidator2"
        ControlToValidate="TextBox1"
        Display="Static"
        ErrorMessage="*"
        runat="server"/>
</td>
</tr>
<tr>
    <td></td>
    <td>
        <asp:Button id="Button1"
            Text="Validate"
            OnClick="ValidateBtn_Click"
            runat="server"/>
    </td>
    <td></td>
</tr>
</table>
</div>
</form>
</body>
</html>

```

Giao diện trang



Khi bạn không nhập dữ liệu cho khung nhập **Card Number**, và click nút **Validate**, sẽ xuất hiện thông báo lỗi bên cạnh khung nhập (*):

RequiredField Validator Example

Fill in the required field below

Credit Card Information

Card Number: *

Thông báo lỗi sẽ không xuất hiện khi bạn nhập dữ liệu trong khung nhập.

RequiredField Validator Example

Required field is filled!

Credit Card Information

Card Number:

5.2. ĐIỀU KHIỂN RANGEVALIDATOR

1. Ý nghĩa

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

2. Cách sử dụng

Bạn đưa điều khiển **RangeValidator** 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 Integer, String, Double, Date và Currency.

Thí dụ 2

Tạo trang RangeValidator.aspx sử dụng điều khiển kiểm tra dữ liệu RangeValidator kiểm tra phạm vi nhập một số nguyên có giá trị từ một đến mươi trong khung nhập.

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

<!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 Example</title>
    <script runat="server">
        void ButtonClick(Object sender, EventArgs e)
        {
            if (Page.IsValid)
            {
                Label1.Text="Page is valid.";
            }
            else
            {
                Label1.Text="Page is not valid!!";
            }
        }
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <h3>RangeValidator Example</h3>
            Enter a number from 1 to 10:
            <br />
            <asp:TextBox id="TextBox1" runat="server"/>
            <br />
        </div>
    </form>
</body>

```

```

<asp:RangeValidator id="Range1"
    ControlToValidate="TextBox1"
    MinimumValue="1"
    MaximumValue="10"
    Type="Integer"
    EnableClientScript="false"
    Text="The value must be from 1 to 10!"
    runat="server"/>

<br /><br />

<asp:Label id="Label1"
    runat="server"/>

<br /><br />

<asp:Button id="Button1" Text="Submit" OnClick="ButtonClick"
runat="server"/>

</div>

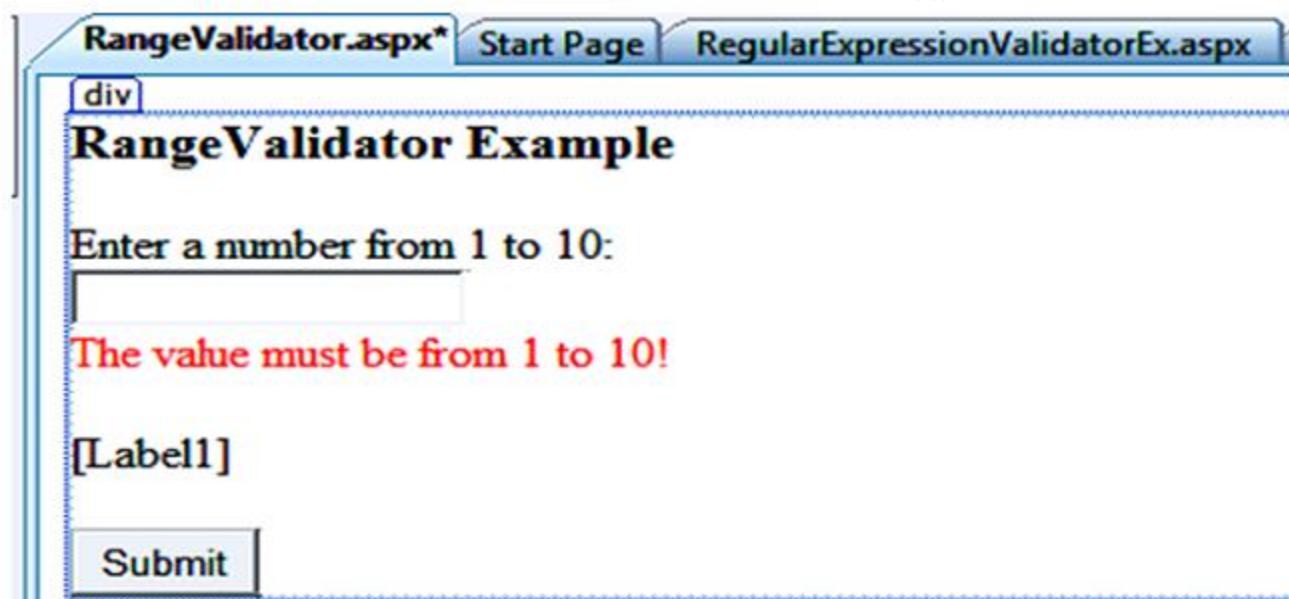
</form>

</body>

</html>

```

Giao diện thiết kế trang **RangeValidator.aspx**



Kết quả thực thi khi click nút lệnh Submit, phương thức xử lý đáp ứng sự kiện OnClick trên nút lệnh như sau:

```

void ButtonClick(Object sender, EventArgs e)
{
    if (Page.IsValid) {
        Label1.Text = "Page is valid.";
    }
}

```

```

else {
    Label1.Text="Page is not valid!!";
}
}

```

Kết quả hiển thị:

RangeValidator Example Enter a number from 1 to 10: <input type="text" value="15"/> <div style="border: 2px solid red; padding: 2px; color: red;">The value must be from 1 to 10!</div> <div style="border: 1px solid blue; padding: 2px; background-color: #e0f2ff;">Page is not valid!!</div> <input type="button" value="Submit"/>	RangeValidator Example Enter a number from 1 to 10: <input type="text" value="8"/> <div style="border: 1px solid blue; padding: 2px; background-color: #e0f2ff;">Page is valid.</div> <input type="button" value="Submit"/>
---	--

5.3. ĐIỀU KHIỂN COMPAREVALIDATOR

1. 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.

2. Cách sử dụng

Thông qua thuộc tính Operator, chúng ta có thể thực hiện các phép so sánh như: =, <>, >, >=, <, <= hoặc dùng để kiểm tra kiểu dữ liệu (DataTypeCheck).

Sử dụng điều khiển này để kiểm tra ràng buộc miền giá trị, kiểu dữ liệu, liên thuộc tính.

Lưu ý: Trong trường hợp không nhập dữ liệu, điều khiển sẽ không thực hiện kiểm tra vi phạm.

➤ Các thuộc tính

- **ControlToCompare:** Tên điều khiển cần so sánh giá trị. Nếu bạn chọn giá trị của thuộc tính Operator = DataTypeCheck thì không cần phải xác định giá trị cho thuộc tính này.
- **Operator:** Quy định phép so sánh, kiểm tra kiểu dữ liệu
 - Equal: = (Đây là giá trị mặc định)

- GreaterThan: >
- GreaterThanEqual: >=
- LessThan: <
- LessThanEqual: <=
- NotEqual: ◊
- DataTypeCheck: Kiểm tra kiểu dữ liệu
- **Type:** Quy định kiểu dữ liệu để kiểm tra hoặc so sánh.
 - String
 - Integer
 - Double
 - Date
 - Currency
- **ValueToCompare:** Giá trị cần so sánh. Trong trường hợp bạn xác định giá trị của cả hai thuộc tính ControlToCompare và ValueToCompare thì giá trị của điều khiển được quy định bởi thuộc tính ControlToCompare được ưu tiên dùng để kiểm tra.

Thí dụ 3

Tạo trang CompareValidatorEx.aspx với nội dung thiết kế sử dụng điều khiển **CompareValidator** để kiểm chứng việc so sánh bằng (mặc định) giá trị chuỗi nhập trong hai khung nhập String1 (TextBox1) và String2 (TextBox2):

```
<asp:CompareValidator id="Compare1"
    ControlToValidate="TextBox1" ControlToCompare="TextBox2"
    EnableClientScript="False" Type="String" runat="server"/>
```

Nội dung trang CompareValidatorEx.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="CompareValidatorEx.aspx.cs" Inherits="CompareValidatorEx" %>
<!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 Example</title>
<script runat="server">
```

```

void Button_Click(Object sender, EventArgs e)
{
    if (Page.IsValid) {
        lblOutput.Text = "Result: Valid!";
    }
    else {
        lblOutput.Text = "Result: Not valid!";
    }
}

void Operator_Index_Changed(Object sender, EventArgs e)
{
    Compare1.Operator = (ValidationCompareOperator)
ListOperator.SelectedIndex;
    Compare1.Validate();
}

void Type_Index_Changed(Object sender, EventArgs e)
{
    Compare1.Type = (ValidationDataType) ListType.SelectedIndex;
    Compare1.Validate();
}

</script>
</head>
<body>

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

<h3>CompareValidator Example</h3><br />
Enter a value in each textbox. Select a comparison operator<br />
and data type. Click "Validate" to compare values.

<table style="background-color:#eeeeee; padding:10">
    <tr valign="top">
        <td>
            <h5>String 1:</h5>
            <asp:TextBox id="TextBox1" runat="server"/>
        </td>
    </tr>
</table>

```

```

</td>
<td>
    <h5>Comparison Operator:</h5>
    <asp:ListBox id="ListOperator"
        OnSelectedIndexChanged="Operator_Index_Changed"
        runat="server">
        <asp:ListItem Selected="True" Value="Equal">Equal</asp:ListItem>
        <asp:ListItem Value="NotEqual">NotEqual</asp:ListItem>
        <asp:ListItem Value="GreaterThan">GreaterThan</asp:ListItem>
        <asp:ListItem
            Value="GreaterThanOrEqualTo">GreaterThanOrEqualTo</asp:ListItem>
        <asp:ListItem Value="LessThan">LessThan</asp:ListItem>
        <asp:ListItem Value="LessThanOrEqualTo">LessThanOrEqualTo</asp:ListItem>
        <asp:ListItem Value="DataTypeCheck">DataTypeCheck</asp:ListItem>
    </asp:ListBox>
</td>
<td>
    <h5>String 2:</h5>
    <asp:TextBox id="TextBox2" runat="server"/>
    <br />
    <asp:Button id="Button1" Text="Validate" OnClick="Button_Click"
        runat="server"/>
</td></tr>
<tr>
    <td colspan="3" align="center">
        <h5>Data Type:</h5>
        <asp:ListBox id="ListType"
            OnSelectedIndexChanged="Type_Index_Changed"
            runat="server">
            <asp:ListItem Selected="true" Value="String" >String</asp:ListItem>
            <asp:ListItem Value="Integer" >Integer</asp:ListItem>
            <asp:ListItem Value="Double" >Double</asp:ListItem>
            <asp:ListItem Value="Date" >Date</asp:ListItem>
            <asp:ListItem Value="Currency" >Currency</asp:ListItem>
        </asp:ListBox>
    </td>
</tr>

```

```

</asp:ListBox>
</td>
</tr>
</table>
<asp:CompareValidator id="Compare1"
    ControlToValidate="TextBox1" ControlToCompare="TextBox2"
    EnableClientScript="False" Type="String"
    runat="server"/> <br />
<asp:Label id="lblOutput" Font-Names="verdana" Font-Size="10pt"
runat="server"/>
</div>
</form>
</body>
</html>

```

Phần mã thực thi đáp ứng sự kiện xảy ra khi thay đổi nội dung trong hai ListBox (sự kiện OnSelectedIndexChanged) và Button (sự kiện OnClick), cho phép cập nhật lại giá trị **Operator** cho điều khiển **CompareValidator**.

```

<script runat="server">
    void Button_Click(Object sender, EventArgs e)
    {
        if (Page.IsValid) {
            lblOutput.Text = "Result: Valid!";
        }
        else {
            lblOutput.Text = "Result: Not valid!";
        }
    }
    void Operator_Index_Changed(Object sender, EventArgs e)
    {
        Compare1.Operator = (ValidationCompareOperator)
ListOperator.SelectedIndex;
        Compare1.Validate();
    }

```

```

void Type_Index_Changed(Object sender, EventArgs e)
{
    Compare1.Type = (ValidationDataType) ListType.SelectedIndex;
    Compare1.Validate();
}
</script>

```

Giao diện trang CompareValidatorEx.aspx với hai lần thực thi so sánh hai chuỗi với phép toán so sánh bằng (Equal) hay khác (Not Equal):

<p>CompareValidator Example</p> <p>Enter a value in each textbox. Select a comparison operator and data type. Click "Validate" to compare values.</p> <p>String 1: Comparison Operator: String 2:</p> <p>ASP.NET Equal ASP.NET Validate</p> <p>NotEqual GreaterThan GreaterThanEqual</p> <p>Data Type:</p> <p>String Integer Double Date</p> <p>Result: Valid!</p>	<p>CompareValidator Example</p> <p>Enter a value in each textbox. Select a comparison operator and data type. Click "Validate" to compare values.</p> <p>String 1: Comparison Operator: String 2:</p> <p>ASP.NET Equal ASP.NET Validate</p> <p>NotEqual GreaterThan GreaterThanEqual</p> <p>Data Type:</p> <p>String Integer Double Date</p> <p>Result: Not valid!</p>
---	---

5.4. ĐIỀU KHIỂN REGULAREXPRESSIONVALIDATOR

1. Ý nghĩa

Điều khiển **RegularExpressionValidator** cho phép bạn so sánh giá trị nhập tại một 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, hoặc product codes...

2. Cách sử dụng

Bạn đưa điều khiển **RegularExpressionValidator** 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 email, số điện thoại,...

Bảng mô tả các ký hiệu thường sử dụng trong Validation Expression

Ký hiệu	Mô tả
A	Ký tự chữ cái (đã được xác định). Ở đây là chữ a
1	Ký tự số (đã được xác định). Ở đây là số 1
[0-n]	Một ký tự số từ 0 đến 9
[abc]	Một ký tự: hoặc a hoặc b hoặc c
	Lựa chọn mẫu này hoặc mẫu khác
\w	Ký tự thay thế phải là một ký tự chữ cái
\d	Ký tự thay thế phải là một ký tự số
\	Thể hiện các ký tự đặc biệt sau.
\.	Ký tự thay thế phải là dấu chấm câu (.)
?	Quy định số lần xuất hiện: 0 hoặc 1 lần
*	Quy định số lần xuất hiện: 0 hoặc nhiều lần
+	Quy định số lần xuất hiện: 1 hoặc nhiều lần (ít nhất là 1)
{n}	Quy định số lần xuất hiện: đúng n lần

Thí dụ 3

Tạo trang RegularExpressionValidatorEx.aspx sử dụng điều khiển kiểm tra RegularExpressionValidator quy định dữ liệu nhập cho khung nhập **ZipCode** là số và phải là năm chữ số.

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="RegularExpressionValidatorEx.aspx.cs"
Inherits="RegularExpressionValidatorEx" %>
<!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>RegularExpression Validator Example</title>
<script runat="server">
    void ValidateBtn_Click(Object sender, EventArgs e)
    {
        if (Page.IsValid)
        {

```

```

lblOutput.Text = "Page is Valid.";

}

else

{

lblOutput.Text = "Page is InValid.";

}

}

</script>

</head>

<body>

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

<div>

<h3>RegularExpressionValidator Example</h3>

<table style="background-color:#eeeeee; padding:10">

<tr valign="top">

<td colspan="3">

<asp:Label ID="lblOutput"

    Text="Enter a 5-digit ZIP Code"

    runat="server"

    AssociatedControlID="TextBox1"/>

</td>

</tr>

<tr>

<td colspan="3">

<b>Personal Information</b>

</td>

</tr>

<tr>

<td align="right">

Zip Code:

</td>

<td>

<asp:TextBox id="TextBox1"

    runat="server"/>

</td>

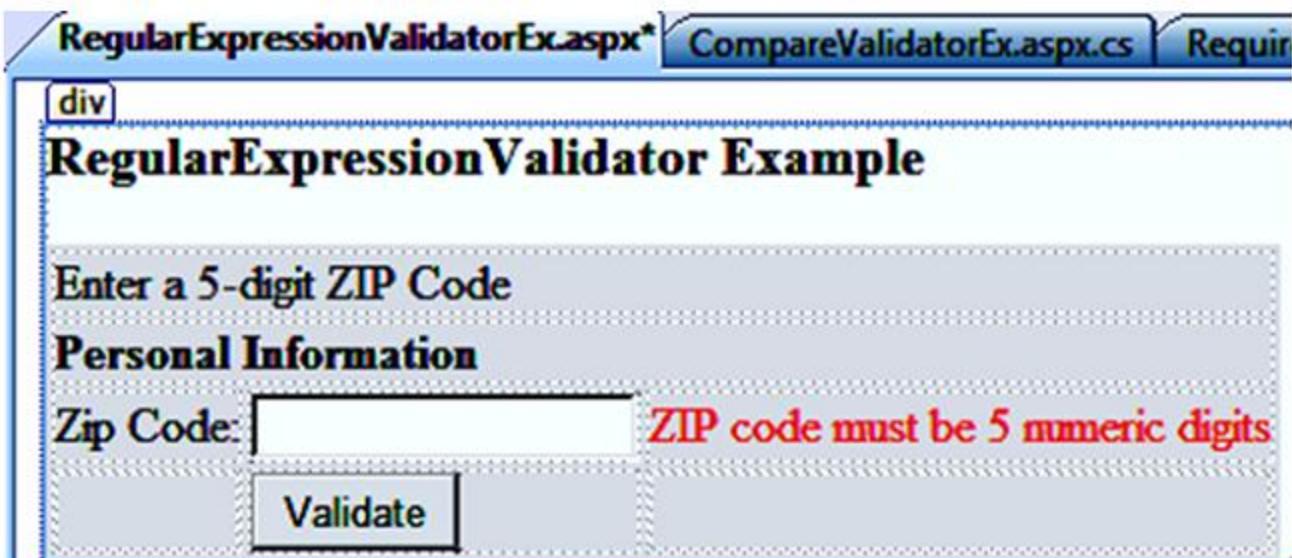
```

```

<td>
    <asp:RegularExpressionValidator id="RegularExpressionValidator1"
        ControlToValidate="TextBox1"
        ValidationExpression="\d{5}"
        Display="Static"
        ErrorMessage="ZIP code must be 5 numeric digits"
        EnableClientScript="False"
        runat="server"/>
</td>
</tr>
<tr>
    <td></td>
    <td>
        <asp:Button ID="Button1" text="Validate"
            OnClick="ValidateBtn_Click"
            runat="server" />
    </td>
    <td></td>
</tr>
</table>
</div>
</form>
</body>
</html>

```

Giao diện trang RegularExpressionValidatorEx.aspx:



Khi click nút Validate, phương thức đáp ứng sự kiện click **ValidateBtn_Click** có nội dung như sau:

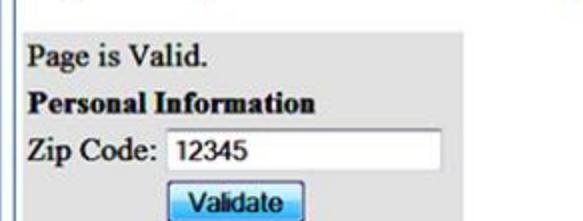
```

void ValidateBtn_Click(Object sender, EventArgs e)
{
    if (Page.IsValid)
    {
        lblOutput.Text = "Page is Valid.";
    }
    else
    {
        lblOutput.Text = "Page is InValid.";
    }
}

```

Dùng thuộc tính **Page.IsValid** để kiểm tra dữ liệu nhập trên Form nhập có hợp lệ hay không.

Kết quả thực thi cho hai trường hợp dữ liệu nhập không hợp lệ và hợp lệ:

RegularExpressionValidator Example  <p>Page is InValid. Personal Information Zip Code: <input type="text" value="1234"/> ZIP code must be 5 numeric digits <input type="button" value="Validate"/></p>	RegularExpressionValidator Example  <p>Page is Valid. Personal Information Zip Code: <input type="text" value="12345"/> <input type="button" value="Validate"/></p>
---	--

5.5. ĐIỀU KHIỂN CUSTOMVALIDATOR

1. 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 sử dụng điều khiển CustomValidator, bạn có thể kết hợp CustomValidator với một hàm.

2. Cách sử dụng

CustomValidator có ba thuộc tính hay sử dụng là:

- **ControlToValidator**: điều khiển của Form sẽ được kiểm tra
- **Text(ErrorMessage)**: hiển thị nội dung thông báo lỗi khi có lỗi
- **ClientValidationFunction**: tên của một hàm client-side để thực hiện kiểm tra trên client-side

CustomValidator hỗ trợ một sự kiện:

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

Thí dụ 5:

Tạo trang CustomValidatorEx.aspx có sử dụng điều khiển kiểm tra dữ liệu nhập trong khung nhập phải là số chẵn, có nội dung dưới đây:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="CustomValidatorEx.aspx.cs" Inherits="CustomValidatorEx" %>
<!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 ServerValidate Example</title>
<script runat="server">
    void ValidateBtn_OnClick(object sender, EventArgs e)
    {
        // Display whether the page passed validation.
        if (Page.IsValid)
        {
            Message.Text = "Page is valid.";
        }
        else
        {
            Message.Text = "Page is not valid!";
        }
    }
    void ServerValidation(object source, ServerValidateEventArgs args)
    {
        try
        {
            // Test whether the value entered into the text box is even.
            int i = int.Parse(args.Value);
            args.IsValid = ((i%2) == 0);
        }
        catch(Exception ex)
    }
</script>
<body>
    <form>
        <input type="text" id="txtInput" />
        <input type="button" value="Validate" onclick="ValidateBtn_OnClick(this,EventArgs.Empty)" />
        <div>Message:</div>
        <div>Message.Text</div>
    </form>
</body>
</html>
```

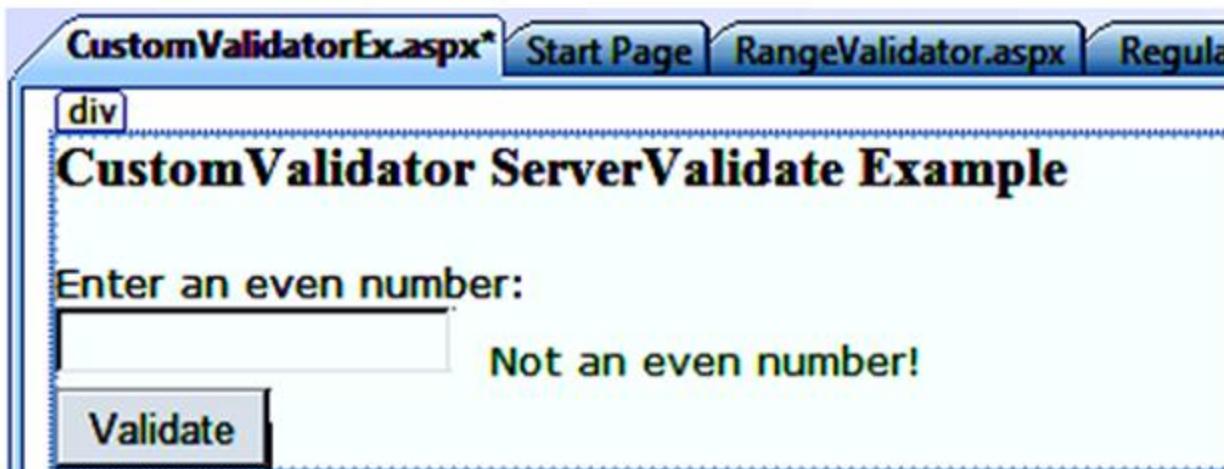
```
{  
    args.IsValid = false;  
}  
}  
</script>  
</head>  
<body>  
    <form id="form1" runat="server">  
        <div>  
            <h3>CustomValidator ServerValidate Example</h3>  
            <asp:Label id="Message"  
                Text="Enter an even number:"  
                Font-Names="Verdana"  
                Font-Size="10pt"  
                runat="server"  
                AssociatedControlID="Text1"/>  
            <br />  
            <asp:TextBox id="Text1"  
                runat="server" />  
            &nbsp;&nbsp;  
            <asp:CustomValidator id="CustomValidator1"  
                ControlToValidate="Text1"  
                Display="Static"  
                ErrorMessage="Not an even number!"  
                ForeColor="green"  
                Font-Names="verdana"  
                Font-Size="10pt"  
                OnServerValidate="ServerValidation"  
                runat="server"/>  
            <br />  
            <asp:Button id="Button1"  
                Text="Validate"  
                OnClick="ValidateBtn_OnClick"
```

```

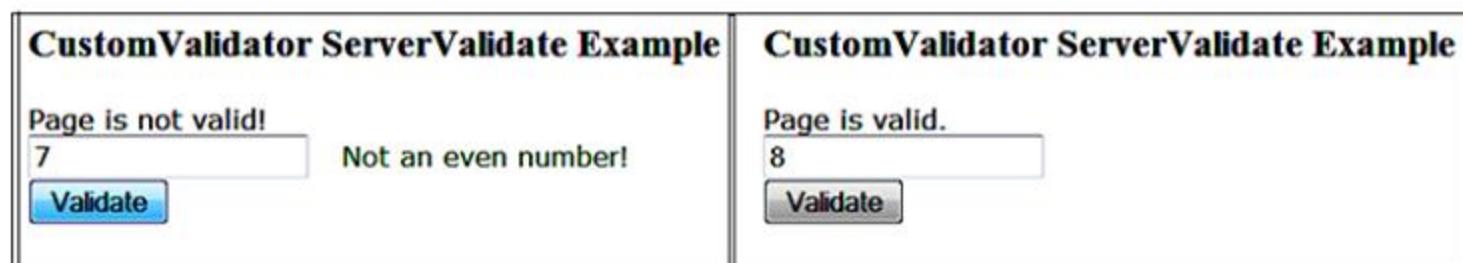
    runat="server"/>
</div>
</form>
</body>
</html>

```

Giao diện thiết kế trang CustomValidatorEx.aspx như hình sau:



Kết quả thực thi sau khi nhập giá trị trong khung nhập và click nút Validate, cho hai trường hợp dữ liệu nhập không hợp lệ và hợp lệ.



5.6. ĐIỀU KHIỂN VALIDATIONSUMMARY

1. Ý nghĩa

ValidationSummary cho phép bạn liệt kê tất cả 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.

2. 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:** Quy định bảng thông báo lỗi có được phép hiển thị như cửa sổ MessageBox hay không. Giá trị mặc định của thuộc tính này là False - không hiển thị.



Thông báo lỗi xuất hiện qua cửa sổ MessageBox

- **ShowSummary:** Quy định bảng thông báo lỗi có được phép hiển thị hay không. Giá trị mặc định của thuộc tính này là True - được phép hiển thị.



Thông báo lỗi hiển thị trực tiếp trên trang Web

Thí dụ 6:

Tạo trang ValidationSummaryEx.aspx có sử dụng một số điều khiển kiểm tra dữ liệu trên trang, trong đó, có điều khiển ValidationSummary để liệt kê tất cả các lỗi kiểm tra trên trang từ những điều khiển kiểm chứng vào một vị trí (cell <td>) thích hợp.

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

<!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 Sample</title>
</head>
<body>
```

```

<h3>ValidationSummary Sample</h3>
<br />
<form id="form1" runat="server">
<div>
<table cellpadding="10">
<tr>
<td>
<table style="background-color:#eeeeee; padding:10">
<tr>
<td colspan="3">
<b>Credit Card Information</b>
</td>
</tr>
<tr>
<td align="right">
Card Type:
</td>
<td>
<asp:RadioButtonList id="RadioButtonList1"
RepeatLayout="Flow"
runat="server">
<asp:ListItem>MasterCard</asp:ListItem>
<asp:ListItem>Visa</asp:ListItem>
</asp:RadioButtonList>
</td>
<td align="center" rowspan="1">
<asp:RequiredFieldValidator
id="RequiredFieldValidator1"
ControlToValidate="RadioButtonList1"
ErrorMessage="Card Type. "
Display="Static"

```

```

    InitialValue="" Width="100%" runat="server">
    *
</asp:RequiredFieldValidator>
</td>
</tr>
<tr>
<td align="right">
Card Number:
</td>
<td>
<asp:TextBox id="TextBox1" runat="server" />
</td>
<td>
<asp:RequiredFieldValidator
    id="RequiredFieldValidator2"
    ControlToValidate="TextBox1"
    ErrorMessage="Card Number."
    Display="Static"
    Width="100%" runat="server">
    *
</asp:RequiredFieldValidator>
</td>
</tr>

<tr>
<td></td>
<td>
<asp:Button
    id="Button1"
    text="Validate"
    runat="server" />
</td>
<td></td>

```

```

        </tr>
    </table>
</td>
<td valign="top">
    <table cellpadding="20">
        <tr>
            <td>
                <asp:ValidationSummary
                    id="valSum"
                    DisplayMode="BulletList"
                    runat="server"
                    HeaderText="You must enter a value in the following fields:"
                    Font-Names="verdana"
                    Font-Size="12"/>
            </td>
        </tr>
    </table>
</td>
</tr>
</table>
</body>
</html>

```

Giao diện trang ValidationSummary.aspx như hình sau:

The screenshot shows a web browser window with the title "ValidationSummary Sample". The page contains a form titled "Credit Card Information" with two radio buttons for "Card Type" (MasterCard and Visa) and a text input field for "Card Number". Below the form, a red error message is displayed: "You must enter a value in the following fields:" followed by a bulleted list: "• Error message 1." and "• Error message 2.". The browser's navigation bar is visible at the top.

Kết quả thực thi cho hai trường hợp dữ liệu nhập không hợp lệ và hợp lệ:

ValidationSummary Sample Credit Card Information Card Type: <input checked="" type="radio"/> MasterCard <input type="radio"/> Visa Card Number: <input type="text"/> <input type="button" value="Validate"/>	You must enter a value in the following fields: <ul style="list-style-type: none">• Card Type.• Card Number.	ValidationSummary Sample Credit Card Information Card Type: <input checked="" type="radio"/> MasterCard <input type="radio"/> Visa Card Number: 1234567890 <input type="button" value="Validate"/>
--	---	--

Thí dụ 6: Minh họa sử dụng các validation control

Tạo một trang ASP.NET tên DangKy.aspx với giao diện và điều kiện ràng buộc như sau:

Phiếu Đăng Ký Thành Viên Điện Đàn Tin Học

Tên đăng nhập	<input type="text"/>	Bạn phải nhập tên !
Tuổi	<input type="text" value="1"/>	18<= Tuổi <=40
Mật khẩu	<input type="text" value="123"/>	Mật khẩu không hợp lệ
Nhập lại mật khẩu:	<input type="text" value="456"/>	
Email:	<input type="text" value="a.com"/>	Email không hợp lệ

Các ràng buộc dữ liệu được mô tả như sau:

- Tên đăng nhập không được bỏ trống
- Tuổi phải nằm trong khoảng từ 18 đến 40
- Mật khẩu nhập phải chính xác (nhập hai lần)
- Email phải hợp lệ (ví dụ: abc@yahoo.com)

Bảng mô tả các thuộc tính của các controls

STT	Control	Tên thuộc tính	Giá trị thuộc tính
1	Label	Text	Tên đăng nhập
2	Label	Text	Tuổi
3	Label	Text	Mật khẩu
4	Label	Text	Nhập lại mật khẩu
5	Label	Text	Email
6	Label	Text	Thông báo
7	TextBox	ID	txtTenDangNhap
8	TextBox	ID	txtTuoi
9	TextBox	ID	txtMatKhau
10	TextBox	ID	txtNhapLaiMatKhau
11	TextBox	ID	txtEmail
12	RequiredFieldValidator	ControlToValidate	txtTenDangNhap
		ErrorMessage	Bạn phải nhập tên !
13	RequiredFieldValidator	ControlToValidate	txtNhapLaiMatKhau
		ErrorMessage	Bạn phải nhập lại mật khẩu !
14	RangeValidator	ControlToValidate	txtTuoi
		ErrorMessage	18<= tuổi <=40
		MaximumValue	40
		MinimumValue	18
		Type	Integer
15	CompareValidator	ControlToCompare	txtNhapLaiMatKhau
		ControlToValidate	txtMatKhau
		ErrorMessage	Mật khẩu không hợp lệ
16	RegularExpressionValidator	ErrorMessage	Email không hợp lệ
		ValidationExpresstion	Nhập vào nút ... và chọn Internet e-mail address.
		ControlToValidate	txtEmail

- Sau khi chạy trang DangKy.aspx, nếu ta nhập vào giá trị không hợp lệ và nhấn nút **Kiểm Tra** thì sẽ xuất hiện thông báo lỗi tương ứng như hình trên.

BÀI TẬP CHƯƠNG 5

Bài 1: Thực hành lại các thí dụ trong bài học để nắm vững nội dung vừa học.

Bài 2: Tạo một ứng dụng ASP.NET thực hành thao tác thiết kế trang bằng mã lệnh.

Tạo trang RegularExpressionValidatorByCodePage.cs có nội dung sau:

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

<!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>
            Valid format for phone number: xxx-xxx-xxxx<br />
        </div>
    </form>
</body>
</html>
```

Phần mã lệnh trang RegularExpressionValidatorByCodePage.aspx.cs

```
public partial class RegularExpressionValidatorByCodePage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Label label = new Label();
        label.Text = "Your phone:";
        this.form1.Controls.Add(label);
        TextBox textBox = new TextBox();
        textBox.ID = "Phone";
```

```

this.form1.Controls.Add(textBox);
Button button = new Button();
button.Text = "OK";
this.form1.Controls.Add(button);

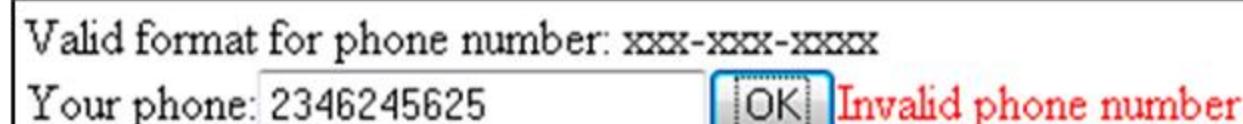
RegularExpressionValidator regularExpressionValidator = new
RegularExpressionValidator();
regularExpressionValidator.ControlToValidate = textBox.ID;
regularExpressionValidator.ValidationExpression = @"(^(\\d{3})\\)?(\\d{3}-
))\\?\\d{3}-\\d{4}";
regularExpressionValidator.Text="Invalid phone number";
this.form1.Controls.Add(regularExpressionValidator);
}
}

```

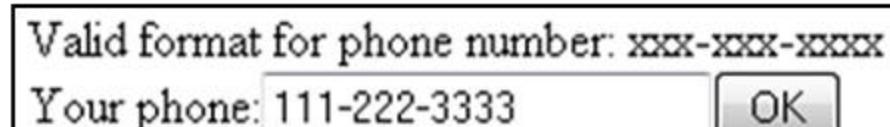
Trong phần này thực hiện tạo các Web User Controls như Label, TextBox, Button và RegularExpressionValidator và gắn lên form.

Kết quả thực thi trang RegularExpressionValidatorByCodePage.aspx:

Nhập dữ liệu cho khung nhập phone sai định dạng, xuất hiện thông báo lỗi.



Nhập dữ liệu theo mẫu xxx-xxx-xxx là dữ liệu hợp lệ.



Bài 3: Tạo một ứng dụng ASP.NET thực hành phần Validation Group

Tạo trang ValidationGroupsPage.aspx

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

```
<!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>
            <div>
                Please enter your email to receive our letter:<br />
                Email:
                <asp:TextBox ID="txtEmail" runat="server"
ValidationGroup="letter"></asp:TextBox>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
ErrorMessage="RequiredFieldValidator" ControlToValidate="txtEmail"
ValidationGroup="letter">Please enter your email</asp:RequiredFieldValidator><br
/>
                &nbsp;<asp:Button ID="Button1" runat="server"
PostBackUrl="~/ValidationGroups.aspx"
                    Text="GO" ValidationGroup="letter" Width="50px" /><br />
                -----
                <br />
                Search:<br />
                Keyword:
                <asp:TextBox ID="txtKeyword" runat="server"></asp:TextBox>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"
ControlToValidate="txtKeyword"
                    ErrorMessage="RequiredFieldValidator" SetFocusOnError="True">Please
enter your email</asp:RequiredFieldValidator><br />
                Certificate:
                <asp:DropDownList ID="ddlCertificate" runat="server" Width="115px">
                    <asp:ListItem></asp:ListItem>
                    <asp:ListItem>B.A</asp:ListItem>
                    <asp:ListItem>B.S</asp:ListItem>
                    <asp:ListItem>M.B.A</asp:ListItem>
                    <asp:ListItem>M.B.S</asp:ListItem>
                </asp:DropDownList>&nbsp;<asp:RequiredFieldValidator
ID="RequiredFieldValidator2"

```

```

    runat="server" ControlToValidate="ddlCertificate"
ErrorMessage="RequiredFieldValidator">Please choose your  

certificate</asp:RequiredFieldValidator><br />
<asp:Button ID="Button2" runat="server" Text="GO" Width="50px" /></div>

</div>
</form>
</body>
</html>

```

Phần giao diện thiết kế trang ValidationGroupsPage.aspx

Please enter your email to receive our letter:

Email: Please enter your email

GO

Search:

Keyword: Please enter your email

Certificate: ▾ Please choose your certificate

GO

Phần mã lệnh trang ValidationGroupsPage.aspx.cs

```

public partial class ValidationGroups : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e) { }
}

```

Kết quả thực thi:

Trường hợp 1:

Khi không nhập dữ liệu cho khung nhập Email, và click nút [GO] ở trên.

Please enter your email to receive our letter:

Email: Please enter your email

[GO]

Search:

Keyword:

Certificate: ▼

[GO]

Nếu nhập dữ liệu, sẽ không có thông báo lỗi.

Trường hợp 2:

Khi không nhập dữ liệu cho khung nhập Keyword hoặc Certificate, và click nút [GO] ở dưới.

Please enter your email to receive our letter:

Email:

[GO]

Search:

Keyword: Please enter your email

Certificate: ▼ Please choose your certificate

[GO]

Nếu nhập dữ liệu, sẽ không có thông báo lỗi.

Chương 6

CÁC ĐÓI TƯỢNG ĐIỀU KHIỂN DỮ LIỆU KHÁC (RICH CONTROLS – LOGIN CONTROLS)

Các vấn đề chính sẽ được đề cập:

- ✓ Các điều khiển hiển thị các trang khác nhau MultiView và Wizard
- ✓ Các điều khiển trong nhóm Login

Kết thúc bài này các bạn có thể:

- *Vận dụng các điều khiển nâng cao trên để thiết kế trang ASP.NET theo yêu cầu của ứng dụng*

6.1. ĐIỀU KHIỂN HIỂN THỊ CÁC TRANG KHÁC NHAU MULTIVIEW

Điều khiển MultiView cho phép bạn ẩn hoặc hiện các phần khác nhau của trang Web, điều khiển này tiện ích khi bạn tạo một TabPage. Nó thực sự tiện ích khi bạn muốn chia một trang web có độ dài lớn thành các phần để hiển thị.

Điều khiển MultiView chứa đựng một hoặc nhiều điều khiển View, bạn sử dụng Multiview để lựa chọn các điều khiển View để trình bày.

Các thuộc tính:

- **ActiveViewIndex:** lựa chọn điều khiển View được đưa ra hiển thị bằng chỉ số Index.
- **Views:** cho phép bạn lấy về tập hợp các điều khiển View chứa đựng trong điều khiển MultiView.

Các phương thức:

- **GetActiveView:** cho phép lấy về thông tin của điều khiển View được lựa chọn.
- **SetActiveView:** cho phép bạn thiết lập điều khiển View được hiển thị.

Và MultiView hỗ trợ sự kiện sau:

- **ActiveViewChanged:** xảy ra khi điều khiển View được lựa chọn.

Cách sử dụng

1. Hiển thị như một TabPage

Thí dụ 1: Thí dụ sau sẽ hướng dẫn bạn tạo một **MultiView** dạng **TabPage**

Tạo trang **MultiView1.aspx**, gồm có ba **Button** và một **MultiView**. Sau đó, gắn ba điều khiển **View** vào trong **MultiView**, nội dung thiết kế trang như sau:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="MultiView1.aspx.cs"
Inherits="MultiView1" %>

<!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>
    <style type="text/css">
        .style1
        {
            font-size: x-large;
        }
        .style2
        {
            font-size: x-large;
            font-weight: bold;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server">
        <div>

            <asp:Button ID="Button1" runat="server" Font-Bold="True"
                onclick="Button1_Click" Text="Panel One" />
            <asp:Button ID="Button2" runat="server" Font-Bold="True"
                onclick="Button2_Click" Text="Panel Two" />

        </div>
    </form>
</body>
</html>
```

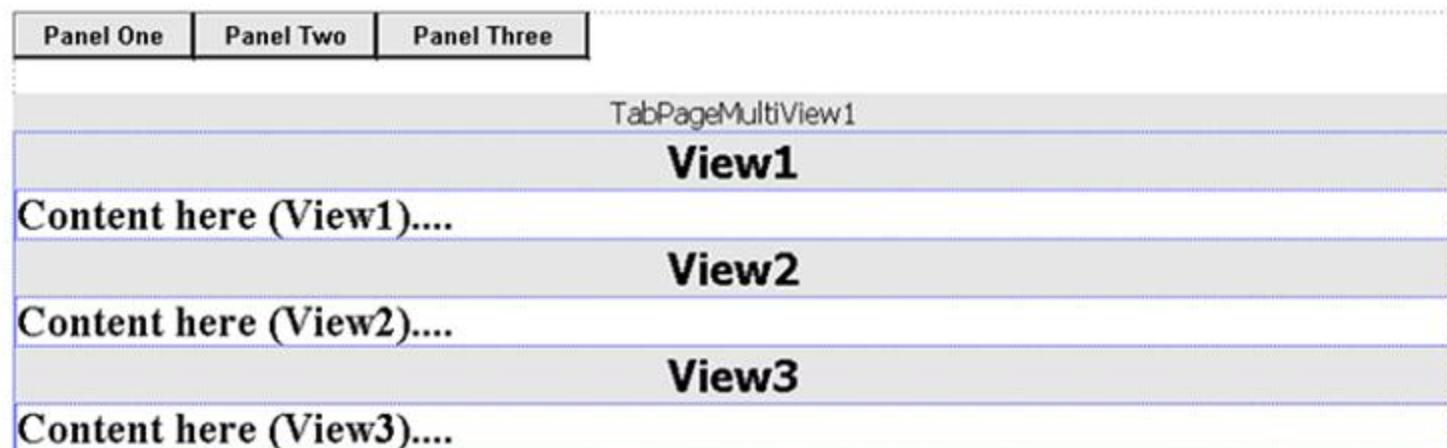
```

<asp:Button ID="Button3" runat="server" Font-Bold="True"
    onclick="Button3_Click" Text="Panel Three" />
<br />
<br />
<asp:MultiView ID="TabPageMultiView1" runat="server">
    <asp:View ID="View1" runat="server">
        <b><span class="style1">Content here (View1)....</span></b>
    </asp:View>
    <asp:View ID="View2" runat="server">
        <b><span class="style1">Content here (View2)....</span></b>
    </asp:View>
    <asp:View ID="View3" runat="server">
        <b><span class="style1">Content here (View3)....</span></b>
    </asp:View>
</asp:MultiView>

</div>
</form>
</body>
</html>

```

Phần giao diện thiết kế trang như sau:



Phần mã lệnh thực thi trang **MultiView1.aspx.cs**:

```

public partial class MultiView1 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        TabPageMultiView1.SetActiveView(View1);
    }
}

```

```
}

protected void Button1_Click(object sender, EventArgs e)
{
    TabPageMultiView1.SetActiveView(View1);
}

protected void Button2_Click(object sender, EventArgs e)
{
    TabPageMultiView1.SetActiveView(View2);
}

protected void Button3_Click(object sender, EventArgs e)
{
    TabPageMultiView1.SetActiveView(View3);
}
```

Kết quả thực thi chương trình:

Trang hiển thị lần đầu:



Content here (View1)....

Click nút [Panel Two], chuyển sang View2.



Content here (View2)....

Click nút [Panel Three], chuyển sang View3.



Content here (View3)....

2. Hiển thị nhiều phần trên trang

Bạn có thể chia một Form có độ dài lớn thành các thành phần nhỏ hơn và hiển thị từng phần, bạn có thể sử dụng các điều khiển Button nằm trong điều khiển MultiView và khi Button được nhấn thì Multiview sẽ xử lý thay đổi hiển thị View khác.

Điều khiển MultiView hỗ trợ các điều khiển lệnh sau:

- **NextView**: MultiView sẽ kích hoạt điều khiển View tiếp theo.
- **PrevView**: MultiView sẽ kích hoạt điều khiển View trước đó.
- **SwitchViewByID**: MultiView sẽ kích hoạt View chỉ định bởi đối số của điều khiển Button.
- **SwitchViewByIndex**: MultiView sẽ kích hoạt View chỉ định bởi đối số của điều khiển Button.

Thí dụ 2: Thí dụ sau sẽ hướng dẫn bạn tạo một Form có nhiều phần với việc sử dụng điều khiển lệnh NextView.

Phần nội dung trang MultiView2.aspx, thiết kế một trong MultiView có bốn điều khiển View, và trong mỗi View sẽ có các Button tương ứng thực hiện việc đi tới OnClick="NextButton_Command" hoặc quay lui OnClick="BackButton_Command" qua lại các View.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="MultiView2.aspx.cs"
Inherits="MultiView2" %>

<!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>
            <h3>      Thí dụ MultiView</h3>
            <asp:Panel id="Page_1ViewPanel"
Width="288px"
Height="60px"
HorizontalAlign =Left
Font-size="12"
BackColor="#C0C0FF"
BorderColor="#404040"
BorderStyle="Double"
runat="Server">
                <asp:MultiView id="MultiView"

```

```
ActiveViewIndex=0
runat="Server">
<asp:View id="Page_1"
    runat="Server">
        <asp:Label id="Page_1Label"
            Font-bold="True"
            Text="Loại ứng dụng mà bạn đang phát triển là gì?"><br><br><br />
        runat="Server"></asp:Label><br><br><br />
        <asp:RadioButton id="Page_1Radio1"
            Text="Winform"
            Checked="False"
            GroupName="RadioGroup1"
            runat="server" >
        </asp:RadioButton><br>
        <asp:RadioButton id="Page_1Radio2"
            Text="WebForm"
            Checked="False"
            GroupName="RadioGroup1"
            runat="server" >
        </asp:RadioButton><br><br>
        <asp:Button id="Page_1Next"
            Text = "Tiếp tục"
            OnClick="NextButton_Command"
            Height="25"
            Width="70"
            runat= "Server">
        </asp:Button>
    </asp:View>

    <asp:View id="Page2"
        runat="Server">
        <asp:Label id="Page2Label"
            Font-bold="True"
            Text="Kinh nghiệm bao lâu rồi?"><br /><br /><br /><br />
```

```

<asp:RadioButton id="Page2Radio1"
    Text="Gần 5 Năm"
    Checked="False"
    GroupName="RadioGroup1"
    runat="Server">
</asp:RadioButton><br>
<asp:RadioButton id="Page2Radio2"
    Text="Hơn 5 năm"
    Checked="False"
    GroupName="RadioGroup1"
    runat="Server">
</asp:RadioButton><br><br>
<asp:Button id="Page2Back"
    Text = "Lùi lại"
    OnClick="BackButton_Command"
    Height="25"
    Width="70"
    runat= "Server">
</asp:Button>
<asp:Button id="Page2Next"
    Text = "Tiếp tục"
    OnClick="NextButton_Command"
    Height="25"
    Width="70"
    runat="Server">
</asp:Button>
</asp:View>

<asp:View id="Page3"
    runat="Server">
<asp:Label id="Page3Label1"
    Font-bold="True"
    Text= "Ngôn ngữ sử dụng chính là gì ?"
    runat="Server"></asp:Label><br><br><br /><br />
<asp:RadioButton id="Page3Radio1"

```

```
Text="Visual Basic.NET"
    Checked="False"
    GroupName="RadioGroup1"
    runat="Server">>
</asp:RadioButton><br>
<asp:RadioButton id="Page3Radio2"
    Text="C#"
    Checked="False"
    GroupName="RadioGroup1"
    runat="Server">>
</asp:RadioButton><br>
<asp:RadioButton id="Page3Radio3"
    Text="C++"
    Checked="False"
    GroupName="RadioGroup1"
```

```
runat="Server">>
</asp:RadioButton><br><br>
<asp:Button id="Page3Back" Text = "Lùi lại"
    OnClick="BackButton_Command"
    Height="25"
    Width="70"
    runat="Server">>
</asp:Button>
<asp:Button id="Page3Next" Text = "Tiếp theo"
    OnClick="NextButton_Command"
    Height="25"
    Width="70"
    runat="Server">>
</asp:Button></asp:View>
```

```
<asp:View id="Page4"
    runat="Server">
    <asp:Label id="Label1"
```

```

Font-bold="True"
Text = "Xin chân thành cảm ơn bạn."
runat="Server">></asp:Label><br />
<br /> <br><br>
<asp:Button id="Page4Save" Text = "Lưu lại"
OnClick="NextButton_Command"
Height="25"
Width="110"
runat="Server">
</asp:Button>
<asp:Button id="Page4Restart"
Text = "Làm lại"
OnClick="BackButton_Command"
Height="25"
Width="110"
runat= "Server">
</asp:Button>
</asp:View>
</asp:MultiView>
</asp:Panel>
</div>
</form>
</body>
</html>
```

Phần giao diện thiết kế trang MultiView2.aspx:

Thí dụ MultiView

MultiView

Page_1

Loại ứng dụng mà bạn đang phát triển là gì?

Winform
 WebForm
 Tiếp tục

Page2

Kinh nghiệm bao lâu rồi?

Gần 5 Năm
 Hơn 5 năm
 Lùi lại Tiếp tục

Page3

Ngôn ngữ sử dụng chính là gì ?

Visual Basic .NET
 C#
 C++
 Lùi lại Tiếp theo

Page4

Xin chân thành cảm ơn bạn.

Lưu lại Làm lại

Phần mã lệnh thực thi trang MultiView2.aspx.cs:

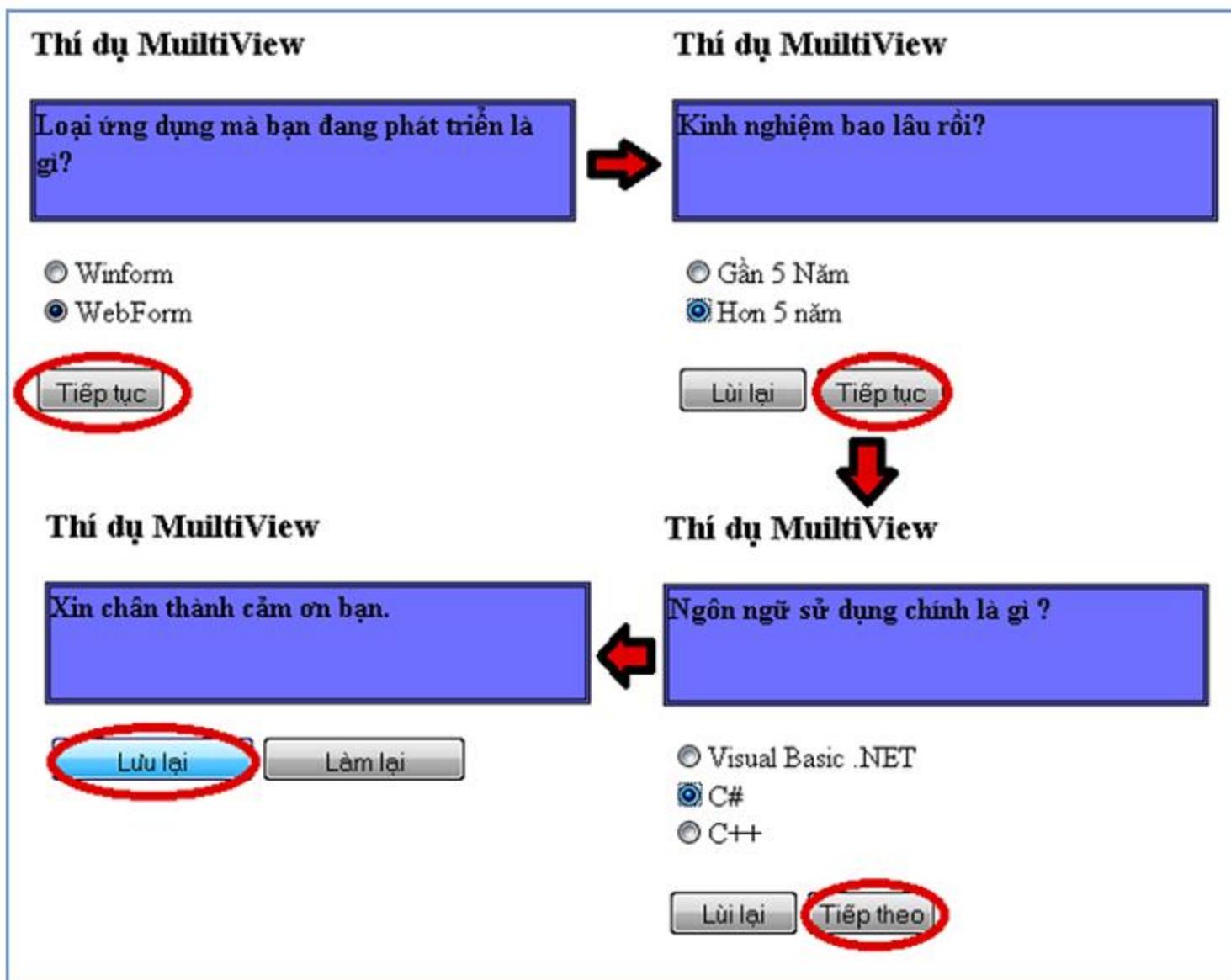
```
public partial class MultiView2: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e) { }

    protected void NextButton_Command(object sender, EventArgs e)
    {
        if(MultiView.ActiveViewIndex > -1 & MultiView.ActiveViewIndex < 3)
        {
            MultiView.ActiveViewIndex += 1;
        }
        else if(MultiView.ActiveViewIndex == 3)
        {
            Page4Save.Enabled = false;
        }
    }
}
```

```
    Page4Restart.Enabled = false;
}
else
{
    throw new Exception("An error occurred.");
}
}

protected void BackButton_Command(object sender, EventArgs e)
{
    if(MultiView.ActiveViewIndex > 0 & MultiView.ActiveViewIndex <= 2)
    {
        MultiView.ActiveViewIndex -= 1;
    }
    else if(MultiView.ActiveViewIndex == 3)
    {
        MultiView.ActiveViewIndex = 0;
    }
    else
    {
        throw new Exception("An error occurred.");
    }
}
}
```

Kết quả thực hiện:



6.2. ĐIỀU KHIỂN WIZARD

Điều khiển Wizard giống với điều khiển MultiView có thể dùng để chia một Form lớn thành nhiều phần nhỏ. Tuy nhiên, nó sẽ có thêm một số thuộc tính mà MultiView không hỗ trợ. Điều khiển Wizard có thể chứa nhiều điều khiển WizardStep con, nhưng chỉ một WizardStep được hiển thị tại một thời điểm.

Các thuộc tính

- **ActiveStep:** cho phép bạn lấy thông tin của WizardStep đang kích hoạt.
- **ActiveStepIndex:** cho phép bạn gán hoặc lấy về chỉ số Index của WizardStep đang kích hoạt.
- **CancelDestinationPageUrl:** cho phép bạn chỉ rõ địa chỉ URL được gửi tới khi người sử dụng nhấn nút Cancel.
- **DisplayCancelButton:** cho phép ẩn hoặc hiện Cancel Button.
- **DisplaySlideBar:** cho phép ẩn hoặc hiện SlideBar (hiển thị tất cả các WizardStep).

- **FinishDestinationPageUrl**: cho phép bạn chỉ định địa chỉ URL được gửi tới khi người dùng nhấn nút Finish.
- **HeaderText**: cho phép bạn chỉ định tiêu đề hiển thị trên đỉnh của điều khiển Wizard.
- **WizardSteps**: cho phép bạn lấy thông tin của các điều khiển WizardStep trong điều khiển Wizard.

Điều khiển Wizard hỗ trợ các Template.

- **FinishNavigationTemplate**: cho phép hiển thị Navigation ở bước kết thúc.
- **HeaderTemplate**: hiển thị thanh tiêu đề ở đầu của điều khiển Wizard.
- **SlideBarTemplate**: cho phép hiển thị SlideBar trong điều khiển Wizard.
- **StartNavigationTemplate**: cho phép hiển thị Navigation ở bước bắt đầu.
- **StepNavigationTemplate**: cho phép hiển thị Navigation ở các bước mà không phải bước bắt đầu và kết thúc.

Điều khiển Wizard hỗ trợ các phương thức:

- **GetHistory()**: cho phép lấy thông tin của các điều khiển Wizard mà đã truy cập.
- **GetStepType()**: cho phép bạn trả về kiểu của mỗi WizardStep riêng theo chỉ số, nó có thể là các thuộc tính sau: Auto, Start, Finish hay Step.
- **MoveTo()**: cho phép bạn di chuyển đến một WizardStep.

Điều khiển Wizard hỗ trợ các sự kiện:

- **ActiveStepChanged**: xảy ra khi một WizardStep trở thành Step được kích hoạt.
- **CancelButtonClick**: xảy ra khi Cancel Button được nhấn.
- **FinishButtonClick**: xảy ra khi Finish Button được nhấn.
- **NextButtonClick**: xảy ra khi Next button được nhấn.
- **PreviousButtonClick**: xảy ra khi Previous button được nhấn.
- **SlideBarButtonClick**: xảy ra khi SlideBar button được nhấn.

Một điều khiển Wizard chứa đựng một hoặc nhiều WizardStep để diễn tả các bước trong quá trình Wizard.

Các WizardStep hỗ trợ các thuộc tính

- **AllowReturn:** ngăn cản hay cho phép người sử dụng trả về bước này từ một bước khác.
- **Name:** tên của điều khiển WizardStep.
- **StepType:** cho phép bạn gán hay lấy về kiểu của WizardStep nó có thể là các giá trị sau: Auto, Finish, Start, Complete và Step.
- **Title:** lấy về hoặc gán tiêu đề của điều khiển WizardStep tiêu đề này được hiển thị ở Wizard Sidebar.
- **Wizard:** cho phép bạn lấy thông tin điều khiển Wizard chứa trong WizardStep.

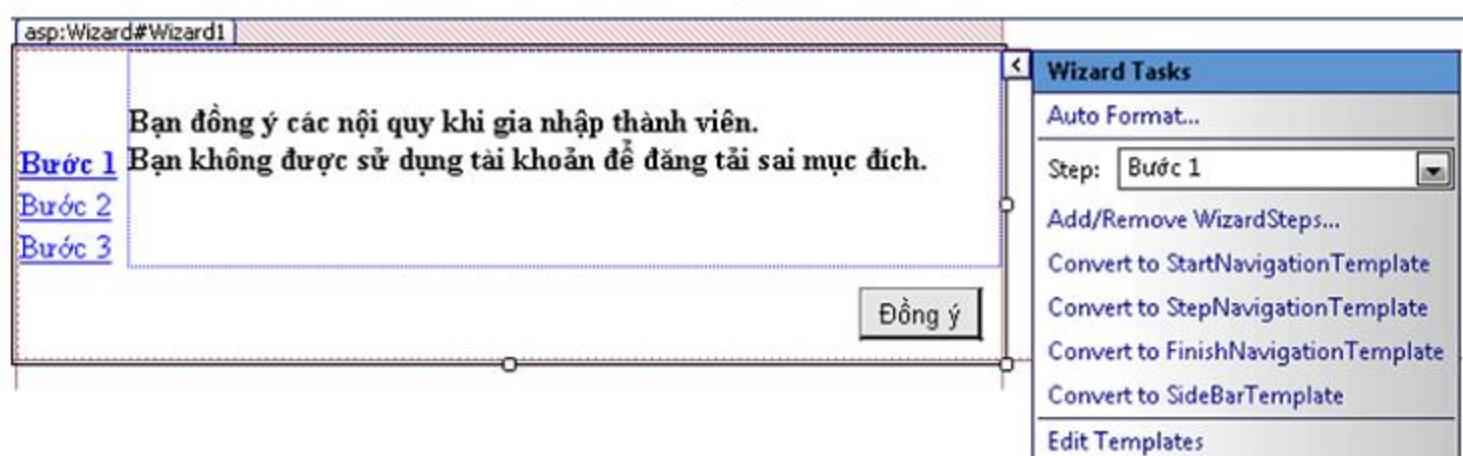
Các sự kiện trong WizardStep

- **Activate:** xảy ra khi một WizardStep được kích hoạt.
- **DeActivate:** xảy ra khi WizardStep khác được kích hoạt.

StepType là thuộc tính quan trọng nhất của Wizard, thuộc tính nào xác định WizardStep được đưa ra như thế nào, mặc định là **Auto**

Thí dụ 3:

Tạo trang ASP.NET có sử dụng điều khiển Wizard và có ba WizardStep bên trong được xây dựng như sau:



Phần mã lệnh Wizard ban đầu chứa các WizardStep:

```
<asp:Wizard ID="Wizard1" runat="server" ActiveStepIndex="0"
FinishCompleteButtonText="Kết thúc"
    FinishPreviousButtonText="Trở lại" Height="155px"
    StartNextButtonText="Đồng ý"
    StepNextButtonText="Tiếp theo" StepPreviousButtonText="Về trước"
    Width="491px">
```

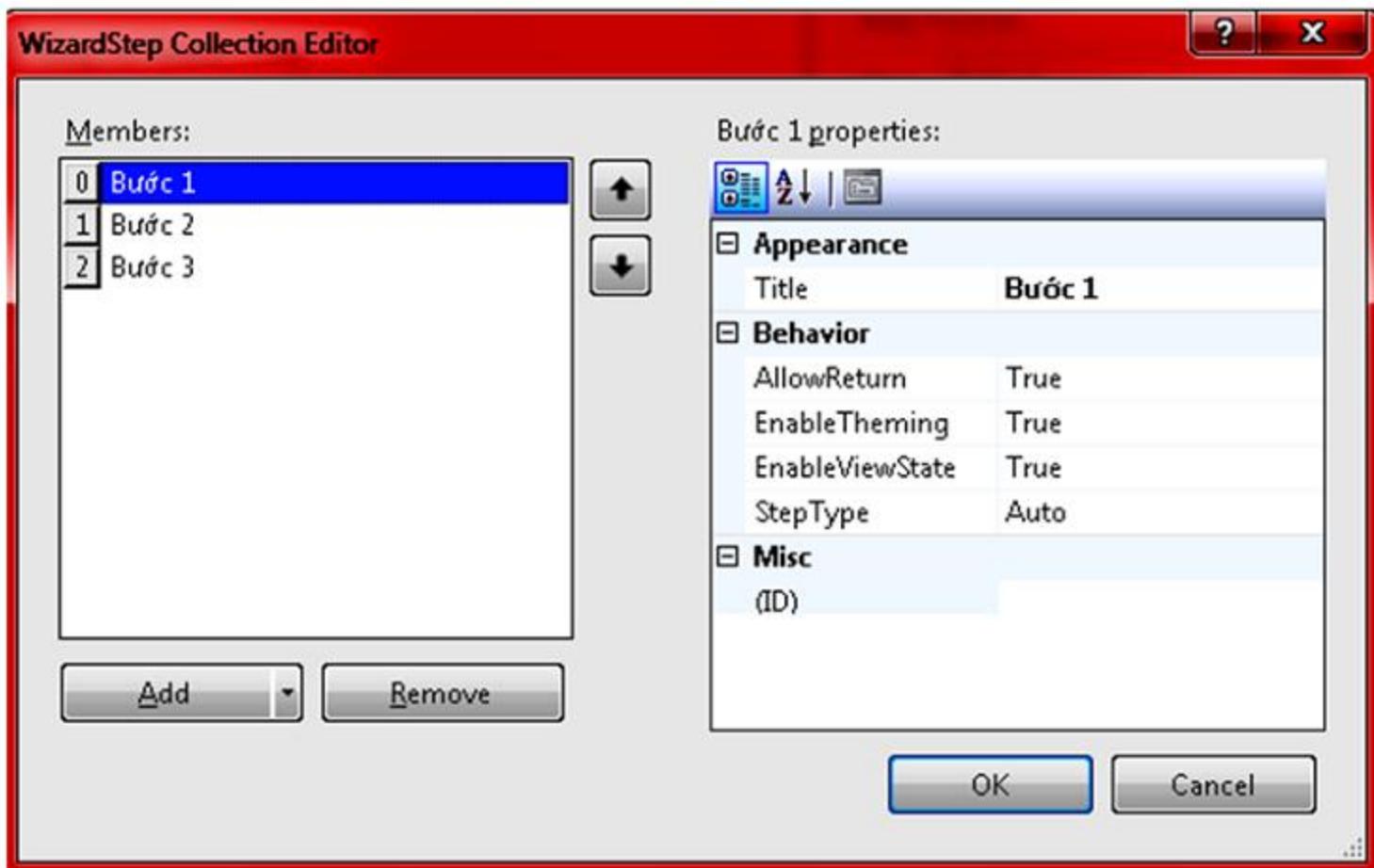
```
<WizardSteps>
```

```
.....
```

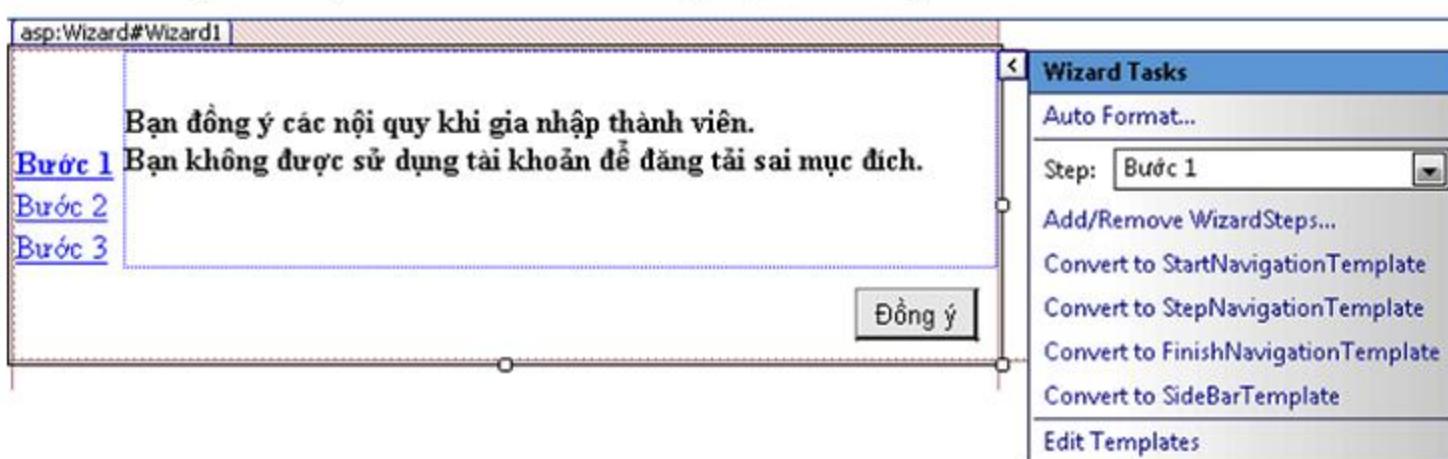
```
</WizardSteps>
```

```
</asp:Wizard>
```

Thực hiện việc thêm các WizardStep, chọn **Add/Remove WizardSteps...**



Thiết kế giao diện cho WizardStep- [Bước 1]



Mã lệnh Thiết kế giao diện WizardStep- [Bước 1]

```
<asp:WizardStep runat="server" Title="Bước 1">
```

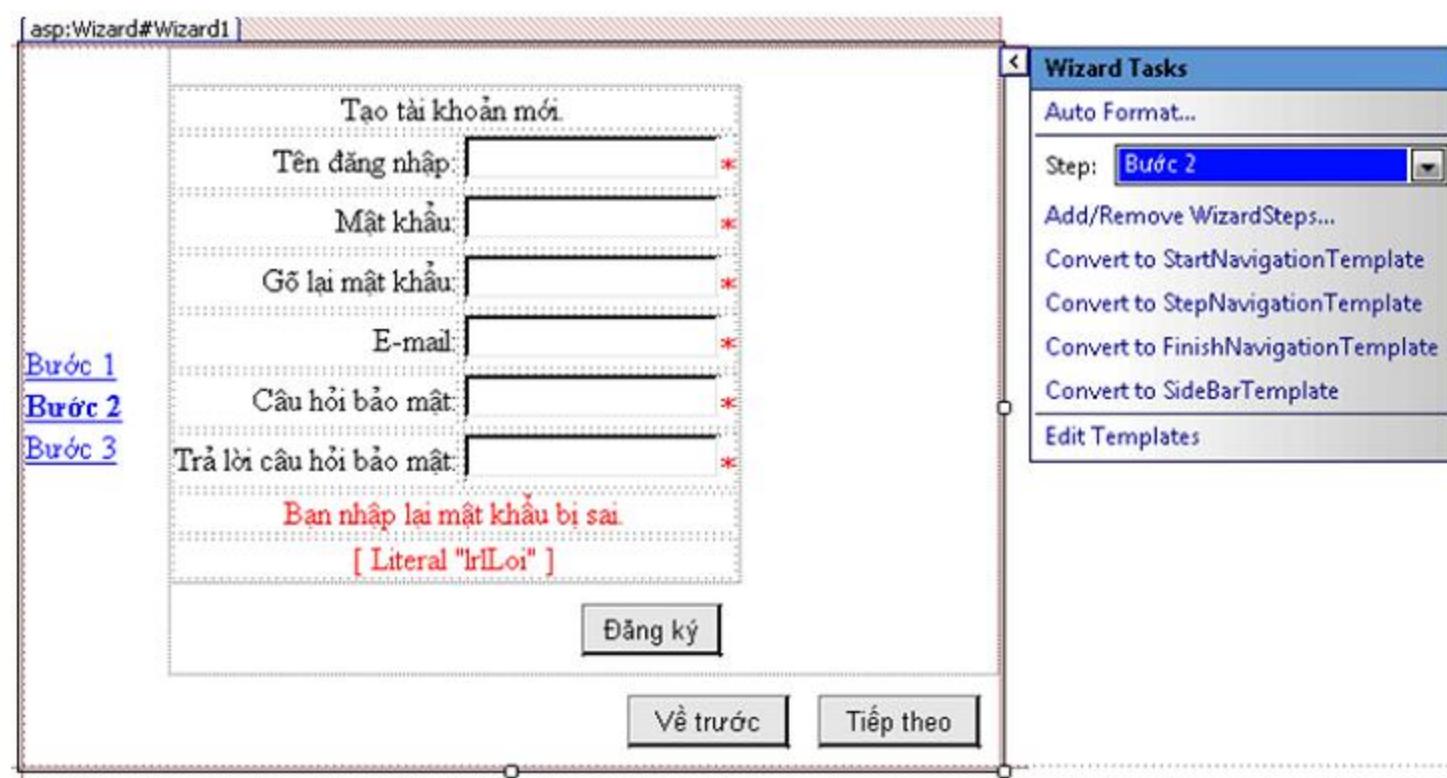
```
    <strong>Bạn đồng ý các nội quy khi gia nhập thành viên.<br />
```

```
        Bạn không được sử dụng tài khoản để đăng tải sai mục đích.<br />
```

```
    </strong>
```

```
</asp:WizardStep>
```

Thiết kế giao diện WizardStep- [Bước 2]



Mã lệnh Thiết kế giao diện WizardStep- [Bước 2]

```
<asp:WizardStep runat="server" Title="Bước 2">
    &nbsp;<asp:CreateUserWizard ID="CreateUserWizard1" runat="server"
CreateUserButtonText="Đăng k&#253;">
    <WizardSteps>
        <asp:CreateUserWizardStep runat="server">
            <ContentTemplate>
                <table border="0">
                    <tr>
                        <td align="center" colspan="2" style="height: 20px">
                            Tạo tài khoản mới.
                        </td>
                    </tr>
                    <tr>
                        <td align="right">
                            <asp:Label ID="UserNameLabel" runat="server"
AssociatedControlID="UserName">Tên đăng nhập:</asp:Label></td>
                            <td>
                                <asp:TextBox ID="UserName"
runat="server"></asp:TextBox>
                                <asp:RequiredFieldValidator ID="UserNameRequired"
runat="server" ControlToValidate="UserName">
                            
```

```

        ErrorMessage="User Name is required."
        ToolTip="User Name is required."
        ValidationGroup="CreateUserWizard1">*</asp:RequiredFieldValidator>

    </td>
</tr>
<tr>
    <td align="right">
        <asp:Label ID="PasswordLabel" runat="server"
AssociatedControlID="Password">Mật khẩu:</asp:Label></td>
        <td>
            <asp:TextBox ID="Password" runat="server"
TextMode="Password"></asp:TextBox>
            <asp:RequiredFieldValidator ID="PasswordRequired"
runat="server" ControlToValidate="Password"
                ErrorMessage="Password is required."
                ToolTip="Password is required."
                ValidationGroup="CreateUserWizard1">*</asp:RequiredFieldValidator>

        </td>
</tr>
<tr>
    <td align="right">
        <asp:Label ID="ConfirmPasswordLabel" runat="server"
AssociatedControlID="ConfirmPassword">Gõ lại mật khẩu:</asp:Label></td>
        <td>
            <asp:TextBox ID="ConfirmPassword" runat="server"
TextMode="Password"></asp:TextBox>
            <asp:RequiredFieldValidator
                ID="ConfirmPasswordRequired" runat="server"
                ControlToValidate="ConfirmPassword"
                    ErrorMessage="Confirm Password is required."
                    ToolTip="Confirm Password is required."
                    ValidationGroup="CreateUserWizard1">*</asp:RequiredFieldValidator>

    <td align="right">

```

```

        <asp:Label ID="EmailLabel" runat="server"
AssociatedControlID="Email">E-mail:</asp:Label></td>

        <td>
            <asp:TextBox ID="Email"
runat="server"></asp:TextBox>
                <asp:RequiredFieldValidator ID="EmailRequired"
runat="server" ControlToValidate="Email"
                    ErrorMessage="E-mail is required." ToolTip="E-mail
is required." ValidationGroup="CreateUserWizard1">*</asp:RequiredFieldValidator>
            </td>
        </tr>
        <tr>
            <td align="right">
                <asp:Label ID="QuestionLabel" runat="server"
AssociatedControlID="Question">Câu hỏi bảo mật:</asp:Label></td>
            <td>
                <asp:TextBox ID="Question"
runat="server"></asp:TextBox>
                <asp:RequiredFieldValidator ID="QuestionRequired"
runat="server" ControlToValidate="Question"
                    ErrorMessage="Security question is required."
                    ToolTip="Security question is required."
ValidationGroup="CreateUserWizard1">*</asp:RequiredFieldValidator>
            </td>
        </tr>
        <tr>
            <td align="right">
                <asp:Label ID="AnswerLabel" runat="server"
AssociatedControlID="Answer">Trả lời câu hỏi bảo mật:</asp:Label></td>
            <td>
                <asp:TextBox ID="Answer"
runat="server"></asp:TextBox>
                <asp:RequiredFieldValidator ID="AnswerRequired"
runat="server" ControlToValidate="Answer"
                    ErrorMessage="Security answer is required."
                    ToolTip="Security answer is required."

```

```

ValidationGroup="CreateUserWizard1">*</asp:RequiredFieldValidator>
    </td>
</tr>
<tr>
    <td align="center" colspan="2" style="height: 21px">
        <asp:CompareValidator ID="PasswordCompare"
runat="server" ControlToCompare="Password"
            ControlToValidate="ConfirmPassword"
Display="Dynamic" ErrorMessage="Bạn nhập lại mật khẩu bị sai. "

```

```

ValidationGroup="CreateUserWizard1"></asp:CompareValidator>
    </td>
</tr>
<tr>
    <td align="center" colspan="2" style="color: red">
        <asp:Literal ID="lrlLoi" runat="server"
EnableViewState="False"></asp:Literal>
    </td>
</tr>
</table>
</ContentTemplate>
</asp:CreateUserWizardStep>
<asp:CompleteWizardStep runat="server">
    </asp:CompleteWizardStep>
</WizardSteps>
</asp:CreateUserWizard>
</asp:WizardStep>

```

Thiết kế giao diện WizardStep- [Bước 3]



Mã lệnh Thiết kế giao diện WizardStep- [Bước 3]

```
<asp:WizardStep runat="server" Title="Bước 3">
```

Bạn đã đăng ký thành công.

Xin chân thành cảm ơn.</asp:WizardStep>

Phần mã lệnh trang:

```
public partial class _Default: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e) { }
}
```

Kết quả thực thi:

Bước 1:

[Bước 1](#) Bạn đồng ý các nội quy khi gia nhập thành viên.
[Bước 2](#) Bạn không được sử dụng tài khoản để đăng tải sai mục đích.
[Bước 3](#)

[Đồng ý](#)

Click [**Đồng ý**] chuyển sang [**Bước 2**],

Tạo tài khoản mới.

Tên đăng nhập:	daonguyen
Mật khẩu:	*****
Gõ lại mật khẩu:	*****
E-mail:	laonguyen@gmail.com
Câu hỏi bảo mật:	ai vay?
Trả lời câu hỏi bảo mật:	toi day

Bước 1

Bước 2

Bước 3

Đăng ký

Về trước **Tiếp theo**

Click **[Đăng ký]** sau đó Click **[Tiếp theo]** chuyển sang **[Bước 3]**

Bước 1 Bạn đã đăng ký thành công.
Bước 2 Xin chân thành cảm ơn.

Bước 3

Trở lại **Kết thúc**

Click nút **[Kết thúc]** để kết thúc thực thi ứng dụng.

Định dạng hiển thị Wizard theo **AutoFormat..**



6.3. NHÓM ĐIỀU KHIỂN LOGIN

Bạn có thể sử dụng các điều khiển Login của ASP.NET để xây dựng các hệ thống đăng ký người sử dụng cho website của mình, Bạn có thể sử dụng các Login Control để tạo form đăng nhập, đăng ký, thay đổi mật khẩu hay ghi nhớ mật khẩu trên Form.

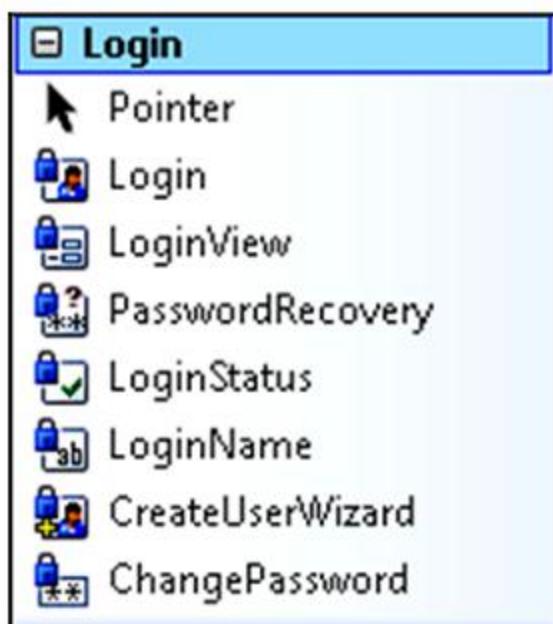
Bao gồm các điều khiển:

- **Login**: cho phép hiển thị Form đăng nhập người sử dụng.
- **CreateUserWizard**: cho phép hiển thị Form đăng ký người sử dụng.
- **LoginStatus**: hiển thị trạng thái Login hay Logout phụ thuộc vào trạng thái kiểm chứng người sử dụng.
- **LoginName**: hiển thị tên người đăng ký hiện tại.
- **ChangePassword**: hiển thị Form cho phép người sử dụng thay đổi mật khẩu.
- **PasswordRecovery**: cho phép người sử dụng khôi phục password, password này sẽ được gửi vào mail cho người sử dụng.
- **LoginView**: hiển thị các nội dung khác nhau tới mỗi người sử dụng phụ thuộc vào authentication hoặc role.

Trong phần này, chúng ta sẽ học chi tiết các điều khiển:

- **Login**: cho phép hiển thị Form đăng nhập người sử dụng.
- **CreateUserWizard**: cho phép hiển thị Form đăng ký người sử dụng.

Các điều khiển còn lại sẽ đề cập trong chương Bảo mật ASP.NET



Hình 6.1: Nhóm điều khiển Login

Điều khiển Login

Điều khiển **Login** đưa ra một form đăng nhập tiêu chuẩn. Mặc định điều khiển **Login** sử dụng ASP.NET **Membership** để kiểm chứng người sử dụng. Tuy nhiên, bạn có thể tùy chỉnh kiểu kiểm chứng người sử dụng với điều khiển **Login**. Điều khiển **Login** hỗ trợ rất nhiều thuộc tính cho phép bạn tùy chỉnh cách hiển thị và ứng xử của điều khiển.

Các thuộc tính của điều khiển <asp:Login>

```
<asp:Login  
    ID="Login1" // tên đối tượng  
    runat="server"  
    // Phần nội dung Text  
    InstructionText="Please enter your own username and password"  
    TitleText="Login"  
    ToolTip="Login"  
    TextLayout="TextOnTop"  
    LoginButtonText="Login"  
    RememberMeText="Remember me for next time."  
    // Phần tạo user mới  
    CreateUserUrl="~/CreateNewUserPage.aspx"  
    CreateUserIconUrl="~/Images/NEW.bmp"  
    CreateUserText="Signup? "  
    // Phần phục hồi/tạo mới lại password  
    PasswordRecoveryUrl="~/ForgotPasswordPage.aspx"  
    PasswordRecoveryIconUrl="~/Images/VIEW.BMP"  
    PasswordRecoveryText="Forget your ID or password?"  
    // Phần trợ giúp  
    HelpPageUrl="~/Help.aspx"  
    HelpPageIconUrl="~/Images/HELP.BMP"  
    HelpPageText="Help"  
    // Các sự kiện chính  
    OnAuthenticate="Login1_Authenticate"  
    OnLoad="Login1_Load"  
    .....  
</asp:Login>
```

Phần giao diện thiết kế trang **LoginControl.aspx**:

Login

Please enter your own username and password

User Name:	<input type="text"/>	*
Password:	<input type="password"/>	*

Remember me for next time.

[Signup?](#)
 [Forget your ID or password?](#)
 [Help](#)

Phần mã lệnh thiết kế trang **LoginControl.aspx**:

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

<!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>

<asp:Login ID="Login1" runat="server"
CreateUserUrl="~/CreateNewUserPage.aspx"
HelpPageUrl="~/Help.aspx"
InstructionText="Please enter your own username and password"
LoginButtonText="Login"
PasswordRecoveryUrl="~/ForgotPasswordPage.aspx"
RememberMeText="Remember me for next time." Width="431px"
CreateUserIconUrl="~/Images/NEW.bmp"
CreateUserText="Signup? "
OnAuthenticate="Login1_Authenticate"
PasswordRecoveryIconUrl="~/Images/VIEW.BMP"
PasswordRecoveryText="Forget your ID or password?" TitleText="Login"
ToolTip="Login" HelpPageIconUrl="~/Images/HELP.BMP"
HelpPageText="Help"
OnLoad="Login1_Load" EnableTheming="True" BackColor="#E3EAEB"
BorderColor="#E6E2D8" BorderPadding="4" BorderStyle="Solid"
BorderWidth="1px"

```

```

Font-Names="Verdana" Font-Size="0.8em" ForeColor="#333333"
TextLayout="TextOnTop" Height="238px"
style="font-weight: 700; font-size: large">
<TitleTextStyle Font-Bold="True" Font-Underline="True"
HorizontalAlign="Left" BackColor="#1C5E55" Font-Size="0.9em"
ForeColor="White" />
<InstructionTextStyle HorizontalAlign="Left" Font-Italic="True"
ForeColor="Black" />
<TextBoxStyle Font-Size="0.8em" />
<LoginButtonStyle BackColor="White" BorderColor="#C5BBAF"
BorderStyle="Solid" BorderWidth="1px"
Font-Names="Verdana" Font-Size="0.8em" ForeColor="#1C5E55" />
</asp:Login>
</div>
</form>
</body>
</html>

```

Phần mã lệnh xử lý trang LoginControl.aspx.cs:

```

public partial class LoginPage: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        this.Login1.UserName="annguyen@gmail.com";
    }

    protected void Login1_Authenticate(object sender, AuthenticateEventArgs e)
    {
        if(Login1.UserName==" annguyen@gmail.com ")
        {
            Response.Redirect("Default.aspx");
        }
        else
            Login1.FailureText="Oh ! Sai roi a nha.";
    }
}

```

```

protected void Login1_Load(object sender, EventArgs e)
{
    this.Login1.DisplayRememberMe= false;
}

```

Kết quả thực thi:

Chuyển sang trang Default.aspx khi click nút [Login]



Nhập sai địa chỉ email (không phải chuỗi “annguyen@gmail.com”) sẽ in thông báo lỗi:

Điều khiển CreateUserWizard

Khi click link **Signup**, sẽ chuyển sang trang CreateNewUserPage.aspx do CreateUserUrl="~/CreateNewUserPage.aspx".

Trang này cho phép bạn tạo mới một user với một số thông tin yêu cầu bắt buộc phải nhập.

Đây là điều khiển dạng **Wizard** nên cũng sẽ có các bước **WizardStep** và một số các thuộc tính sau:

```
<asp:CreateUserWizard ID="CreateUserWizard1" runat="server"
    AutoGeneratePassword="True"
    DisableCreatedUser="True"
    \\ Hiện nút Cancel
    DisplayCancelButton="True"
<!-- sự kiện chính -->
    OnCreatingUser="CreateUserWizard1_CreatingUser"
    OnLoad="CreateUserWizard1_Load"
<!-- có hai bước WizardStep -->
<WizardSteps>
    <asp:CreateUserWizardStep runat="server">
        </asp:CreateUserWizardStep>
    <asp:CompleteWizardStep runat="server">
        </asp:CompleteWizardStep>
</WizardSteps>
<!-- quy định các Style -->
    <SideBarStyle BackColor="#5D7B9D" BorderWidth="0px" Font-Size="0.9em"
    VerticalAlign="Top" />
    <SideBarButtonStyle BorderWidth="0px" Font-Names="Verdana"
    ForeColor="White" />
    <ContinueButtonStyle BackColor="#FFFFBF" BorderColor="#CCCCCC"
    BorderStyle="Solid"
        BorderWidth="1px" Font-Names="Verdana" ForeColor="#284775" />
    <NavigationButtonStyle BackColor="#FFFFBF" BorderColor="#CCCCCC"
    BorderStyle="Solid"
        BorderWidth="1px" Font-Names="Verdana" ForeColor="#284775" />
    <HeaderStyle BackColor="#5D7B9D" BorderStyle="Solid" Font-Bold="True"
    Font-Size="0.9em"
```

```

        ForeColor="White" HorizontalAlign="Center" />

    <CreateUserButtonStyle BackColor="#FFFBFF" BorderColor="#CCCCCC"
BorderStyle="Solid"

        BorderWidth="1px" Font-Names="Verdana" ForeColor="#284775" />

    <TitleTextStyle BackColor="#5D7B9D" Font-Bold="True" ForeColor="White" />
    <StepStyle BorderWidth="0px" />
</asp:CreateUserWizard>

```

Phần giao diện thiết kế trang CreateNewUserPage.aspx:



Phần mã lệnh thực thi trang CreateNewUserPage.aspx.cs

```

public partial class CreateNewUserPage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e) { }

    protected void CreateUserWizard1_CreatingUser(object sender, LoginCancelEventArgs e)
    {
        Response.Redirect("LoginPage.aspx");
    }

    protected void CreateUserWizard1_Load(object sender, EventArgs e)
    {
        this.CreateUserWizard1.UserName = "annguyen";
        this.CreateUserWizard1.Email = "annguyen@gmail.com";
    }
}

```

Kết quả thực thi:

Sign Up for Your New Account

User Name:

E-mail:

Security Question:

Security Answer:

Click nút **[Create User]** sẽ quay lại trang **LoginPage.aspx**.

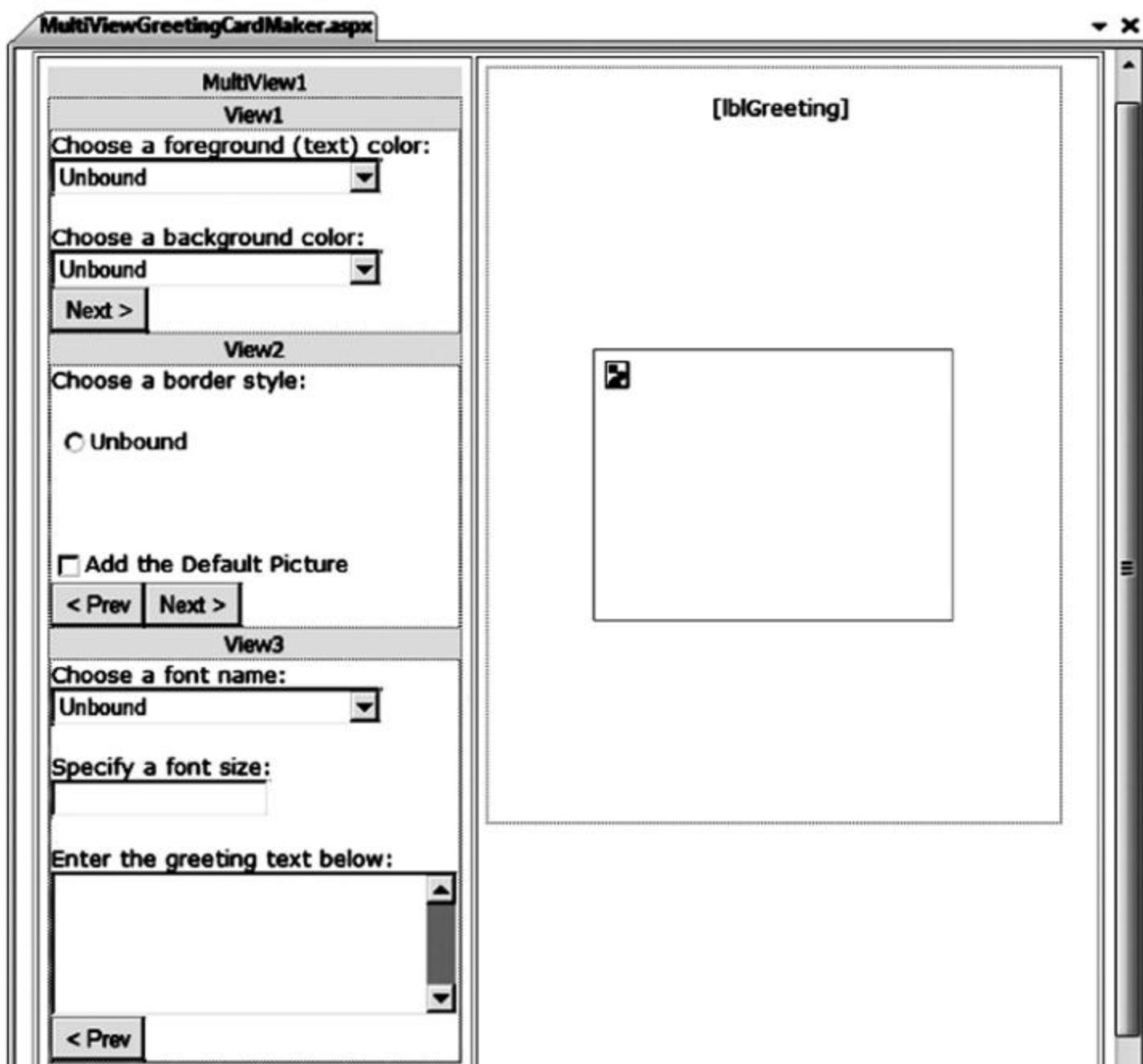
```
protected void CreateUserWizard1_CreatingUser(object sender, LoginCancelEventArgs e)
{
    Response.Redirect("LoginPage.aspx");
}
```

BÀI TẬP CHƯƠNG 6

Bài 1: Thực hành lại các thí dụ trong bài học để nắm vững nội dung vừa học.

Bài 2: Tạo một ứng dụng ASP.NET thực hành với MultiView tạo một thiệp chúc mừng.

Tạo trang MultiViewGreetingCardMaker.aspx có dạng sau:



Phần mã lệnh thiết kế trang MultiViewGreetingCardMaker.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true"  
CodeFile="MultiViewGreetingCardMaker.aspx.cs"  
Inherits="MultiViewGreetingCardMaker" %>  
  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"  
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">  
  
<html xmlns="http://www.w3.org/1999/xhtml" >
```

```

<head runat="server">
    <title>Untitled Page</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <table border="1" cellpadding="5">
                <tr valign="top">
                    <td style="width: 281px">
                        <asp:MultiView id="MultiView1" runat="server" ActiveViewIndex="0">
                            <asp:View ID="View1" runat="server">
                                Choose a foreground (text) color:<br />
                                <asp:DropDownList ID="lstForeColor" runat="server"
AutoPostBack="True" Height="22px"
OnSelectedIndexChanged="ControlChanged" Width="194px">
                                    </asp:DropDownList><br />
                                    <br />
                                Choose a background color:<br />
                                <asp:DropDownList ID="lstBackColor" runat="server"
AutoPostBack="True" Height="22px"
OnSelectedIndexChanged="ControlChanged" Width="194px">
                                    </asp:DropDownList>
                                    <br />
                                    <asp:Button ID="cmdNext" runat="server" Text="Next >" CommandName="NextView" />
                                </asp:View>
                                <asp:View ID="View2" runat="server">
                                    Choose a border style:<br />
                                    <asp:RadioButtonList ID="lstBorder" runat="server"
AutoPostBack="True"
Height="59px" OnSelectedIndexChanged="ControlChanged"
RepeatColumns="2" Width="177px">
                                    </asp:RadioButtonList><br />
                                    <br />
                            </asp:View>
                        </td>
                    </tr>
                </table>
            </div>
        </form>
    </body>

```

```
<asp:CheckBox ID="chkPicture" runat="server"
AutoPostBack="True" OnCheckedChanged="ControlChanged"
    Text="Add the Default Picture" />

<br />
<asp:Button ID="Button3" runat="server" Text="< Prev"
CommandName="PrevView" />
<asp:Button ID="Button4" runat="server" Text="Next >"
CommandName="NextView" />

</asp:View>
<asp:View ID="View3" runat="server">
    Choose a font name:<br />
    <asp:DropDownList ID="lstFontName" runat="server"
AutoPostBack="True" Height="22px"
        OnSelectedIndexChanged="ControlChanged" Width="194px">
        </asp:DropDownList><br />
        <br />
    Specify a font size:<br />
    <asp:TextBox ID="txtFontSize" runat="server" AutoPostBack="True"
OnTextChanged="ControlChanged"></asp:TextBox><br />
    <br />
    Enter the greeting text below:<br />
    <asp:TextBox ID="txtGreeting" runat="server" Height="85px"
OnTextChanged="ControlChanged"
        TextMode="MultiLine" Width="240px"
AutoPostBack="True"></asp:TextBox>

<br />
<asp:Button ID="Button1" runat="server" Text="< Prev"
CommandName="PrevView" />
<asp:Button ID="Button2" runat="server" Text="Apply" />

</asp:View>
</asp:MultiView>
</td>
```

```

<td >
    <asp:Panel ID="pnlCard" runat="server" Height="445px"
HorizontalAlign="Center" Style="z-index: 101;
    " Width="339px">
        <br />
        &nbsp;
        <asp:Label ID="lblGreeting" runat="server" Height="150px"
Width="272px"></asp:Label>
        <asp:Image ID="imgDefault" runat="server" Height="160px"
Visible="False" Width="212px" />
    </asp:Panel>
</td>
</tr>
</table>
</div>

</form>
</body>
</html>

```

Phần mã lệnh thực thi trang MultiViewGreetingCardMaker.aspx.cs:

```

public partial class MultiViewGreetingCardMaker: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            // Get the list of colors.
            string[] colorArray =
Enum.GetNames(typeof(System.Drawing.KnownColor));
            lstBackColor.DataSource = colorArray;
            lstBackColor.DataBind();
            lstForeColor.DataSource = colorArray;
        }
    }
}

```

```
IstForeColor.DataBind();
IstForeColor.SelectedIndex = 34;
IstBackColor.SelectedIndex = 163;

// Get the list of available fonts and add them to the font list.
System.Drawing.Text.InstalledFontCollection fonts;
fonts = new System.Drawing.Text.InstalledFontCollection();
foreach (FontFamily family in fonts.Families)
{
    IstFontName.Items.Add(family.Name);
}

// Set border style options.
string[] borderStyleArray = Enum.GetNames(typeof(BorderStyle));
IstBorder.DataSource = borderStyleArray;
IstBorder.DataBind();

// Select the first border option.
IstBorder.SelectedIndex = 0;

// Set the picture.
imgDefault.ImageUrl = "defaultpic.png";
}

}

private void Update()
{
    // Update the color.
    pnlCard.BackColor = Color.FromName(IstBackColor.SelectedItem.Text);
    lblGreeting.ForeColor = Color.FromName(IstForeColor.SelectedItem.Text);

    // Update the font.
    lblGreeting.Font.Name = IstFontName.SelectedItem.Text;
```

```
try
{
    if (Int32.Parse(txtFontSize.Text) > 0)
    {
        lblGreeting.Font.Size = FontUnit.Point(Int32.Parse(txtFontSize.Text));
    }
}
catch
{
    // Ignore invalid value.
}

try
{
    if (Int32.Parse(txtFontSize.Text) > 0)
    {
        lblGreeting.Font.Size =
            FontUnit.Point(Int32.Parse(txtFontSize.Text));
    }
}
catch
{
    // Ignore invalid value.
}

// Find the appropriate TypeConverter for the BorderStyle enumeration.
TypeConverter cnvrt = TypeDescriptor.GetConverter(typeof(BorderStyle));

// Update the border style using the value from the converter.
pnlCard.BorderStyle = (BorderStyle)cnvrt.ConvertFromString(
    IstBorder.SelectedItem.Text);

// Update the picture.
```

```

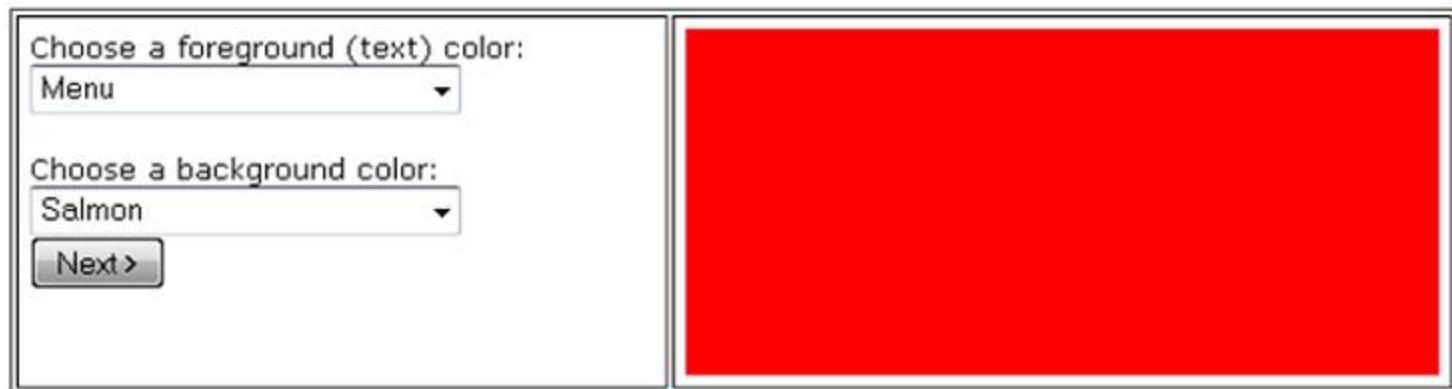
if (chkPicture.Checked == true)
{
    imgDefault.Visible = true;
}
else
{
    imgDefault.Visible = false;
}

// Set the text.
lblGreeting.Text = txtGreeting.Text;
}

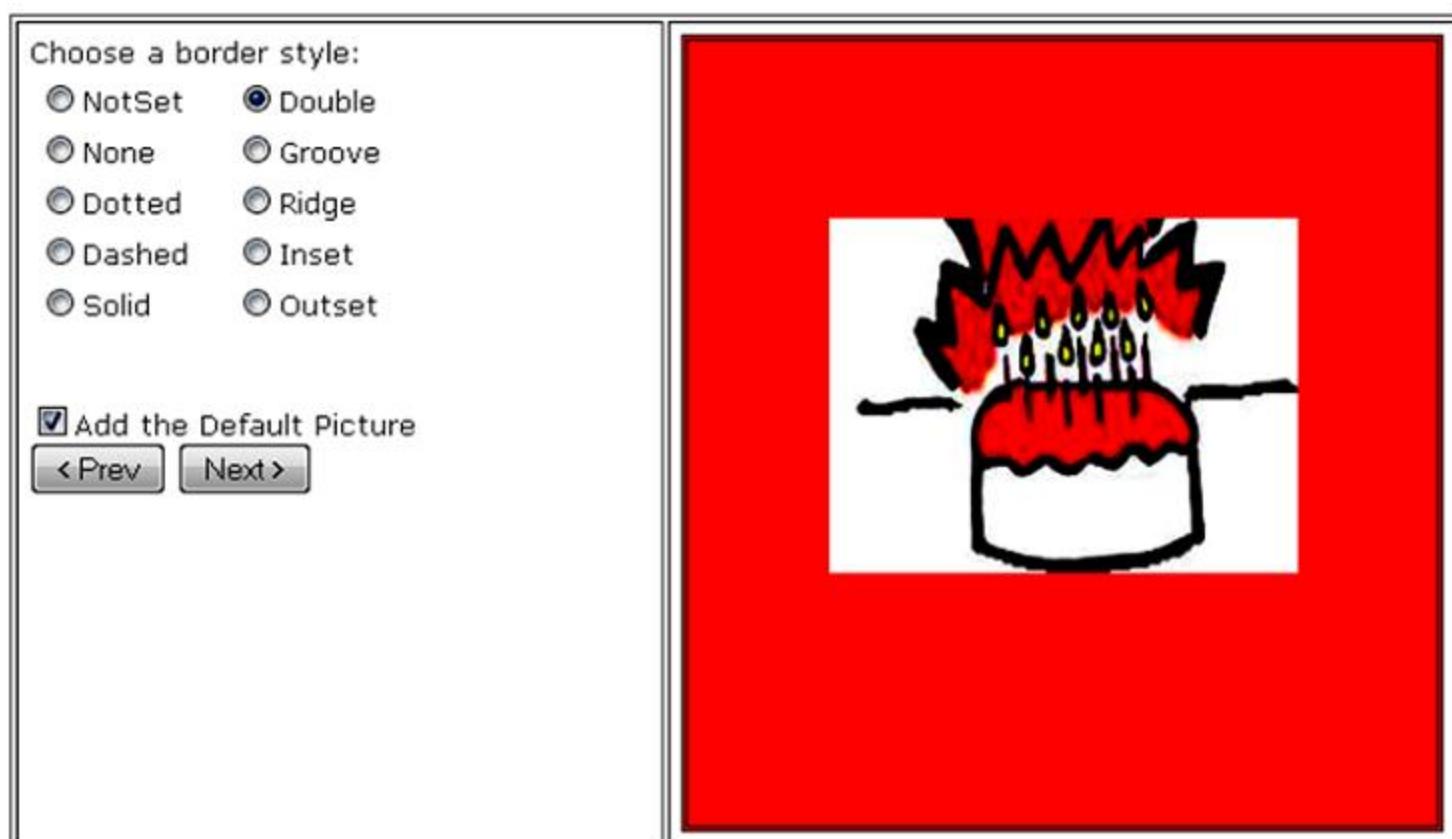
protected void ControlChanged(Object sender, EventArgs e)
{
    // Refresh the greeting card.
    Update();
}

```

Kết quả thực thi:



Chọn giá trị xong, click [Next]



Chọn giá trị xong, click nút [Next].



Chọn giá trị xong, click nút [Apply]

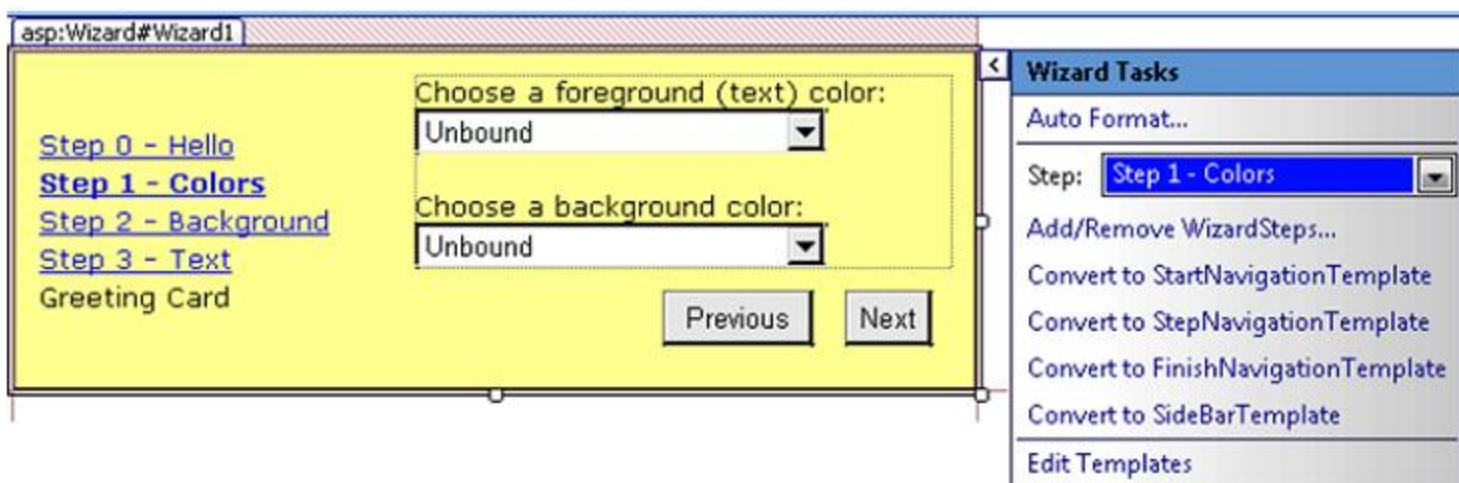
Bài 3: Tạo một ứng dụng ASP.NET thực hành với Wizard tạo một thiệp chúc mừng như trên.

Tạo trang có sử dụng điều khiển **Wizard** có các **WizardStep** được thiết kế như sau:

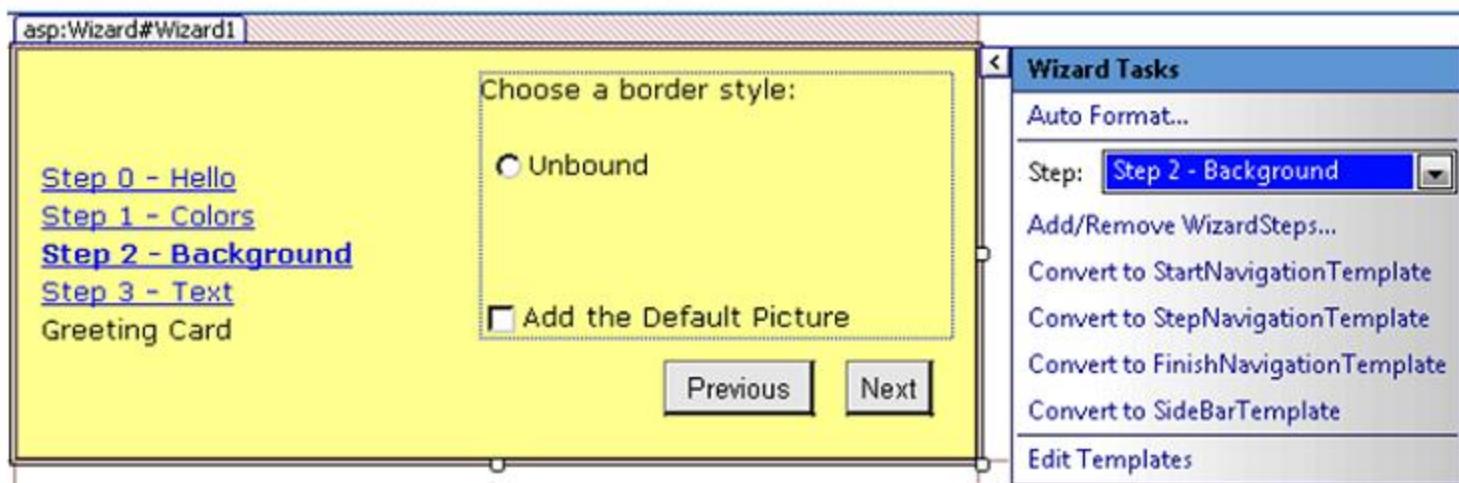
[Step – 0 Hello]



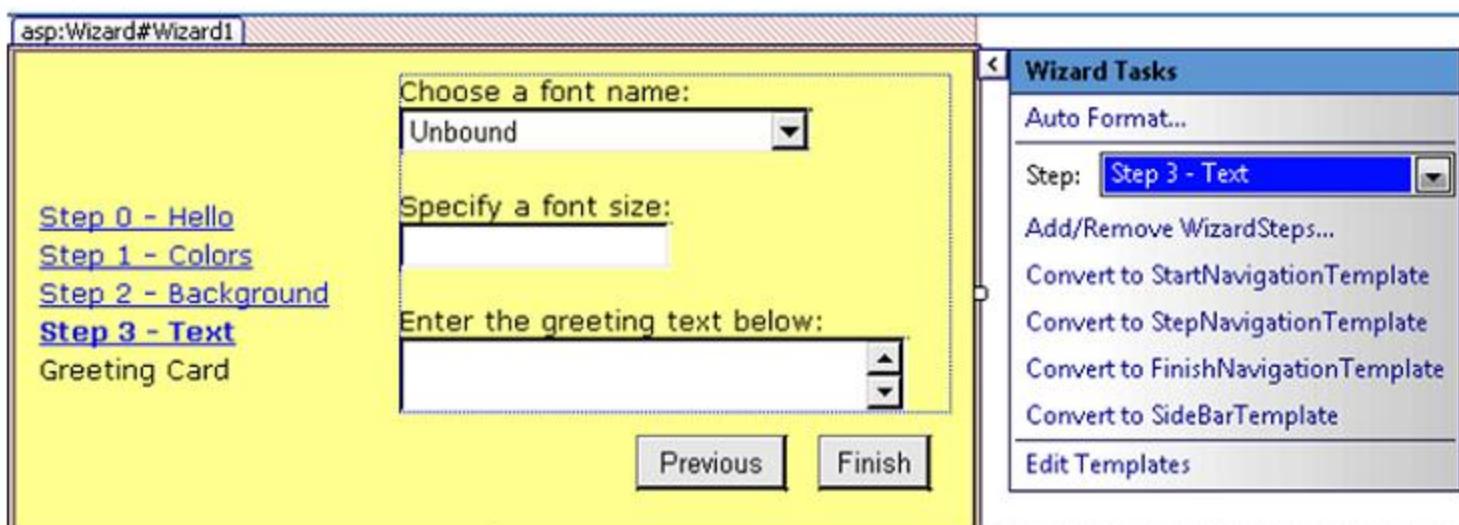
[Step 1 – Colors]



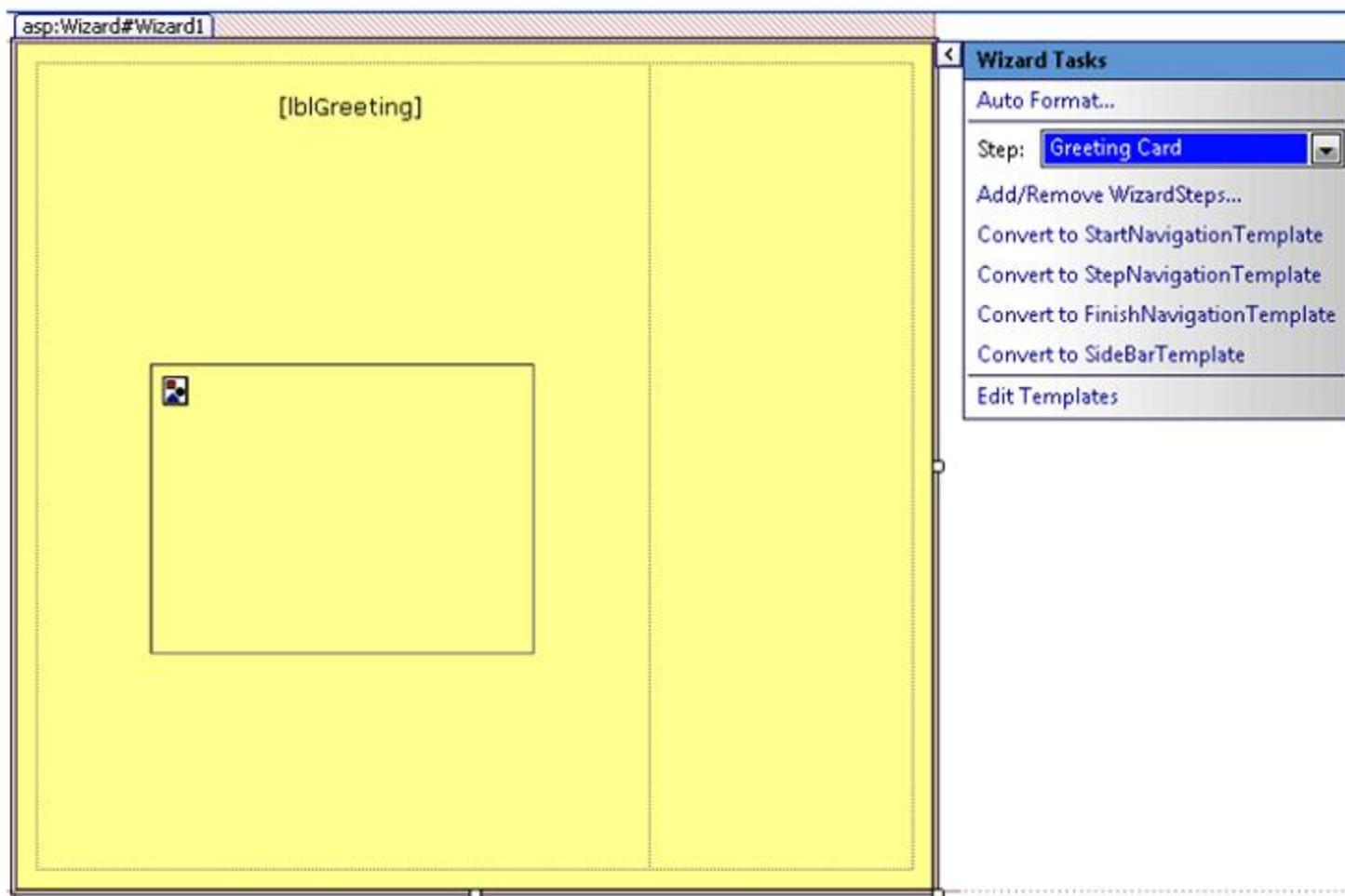
[Step 2 - Background]



[Step 3 - Text]



[Greeting Card]



Phần mã lệnh thiết kế trang WizardGreetingCardMaker.aspx:

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Greeting Card Wizard</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Wizard ID="Wizard1" runat="server" ActiveStepIndex="4"
                BackColor="LemonChiffon" BorderStyle="Groove" BorderWidth="2px"
                CellPadding="10" OnFinishButtonClick="Wizard1_FinishButtonClick"
                Width="456px">
                <WizardSteps>
```

```
<asp:WizardStep runat="server" Title="Step 0 - Hello">
    Hello ASP.NET</asp:WizardStep>

<asp:WizardStep runat="server" Title="Step 1 - Colors">

    Choose a foreground (text) color:<br />
    <asp:DropDownList ID="lstForeColor" runat="server" Width="194px">
        </asp:DropDownList><br />
    <br />

    Choose a background color:<br />
    <asp:DropDownList ID="lstBackColor" runat="server" Width="194px">
        </asp:DropDownList>

    </asp:WizardStep>

<asp:WizardStep runat="server" Title="Step 2 - Background">

    Choose a border style:<br />
    <asp:RadioButtonList ID="lstBorder" runat="server"
        Height="59px" RepeatColumns="2" Width="177px">
        </asp:RadioButtonList><br />
    <br />

    <asp:CheckBox ID="chkPicture" runat="server"
        Text="Add the Default Picture" />

    </asp:WizardStep>

<asp:WizardStep runat="server" Title="Step 3 - Text">

    Choose a font name:<br />
    <asp:DropDownList ID="lstFontName" runat="server" Width="194px">
        </asp:DropDownList><br />
    <br />

    Specify a font size:<br />
    <asp:TextBox ID="txtFontSize" runat="server" ></asp:TextBox><br />
```

```

<br />
Enter the greeting text below:<br />
<asp:TextBox ID="txtGreeting" runat="server"
    TextMode="MultiLine" Width="240px" ></asp:TextBox>

</asp:WizardStep>
<asp:WizardStep runat="server" StepType="Complete" Title="Greeting Card">
    <asp:Panel ID=" pnlCard" runat="server" Height="445px"
HorizontalAlign="Center" Style="z-index: 101;
" Width="339px">
    <br />
    &nbsp;
    <asp:Label ID="lblGreeting" runat="server" Height="150px"
Width="272px"></asp:Label>
    <asp:Image ID="imgDefault" runat="server" Height="160px" Visible="False"
Width="212px" /></asp:Panel>
</asp:WizardStep>
</WizardSteps>
</asp:Wizard>

</div>
</form>
</body>
</html>

```

Phần mã lệnh thực thi trang WizardGreetingCardMaker.aspx.cs:

```

public partial class WizardGreetingCardMaker: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            // Get the list of colors.
            string[] colorArray =
Enum.GetNames(typeof(System.Drawing.KnownColor));

```

```
lstBackColor.DataSource = colorArray;  
lstBackColor.DataBind();  
  
lstForeColor.DataSource = colorArray;  
lstForeColor.DataBind();  
lstForeColor.SelectedIndex = 34;  
lstBackColor.SelectedIndex = 163;  
  
// Get the list of available fonts and add them to the font list.  
System.Drawing.Text.InstalledFontCollection fonts;  
fonts = new System.Drawing.Text.InstalledFontCollection();  
foreach (FontFamily family in fonts.Families)  
{  
    lstFontName.Items.Add(family.Name);  
}  
  
// Set border style options.  
string[] borderStyleArray = Enum.GetNames(typeof(BorderStyle));  
lstBorder.DataSource = borderStyleArray;  
lstBorder.DataBind();  
  
// Select the first border option.  
lstBorder.SelectedIndex = 0;  
  
// Set the picture.  
imgDefault.ImageUrl = "defaultpic.png";  
}  
}  
  
protected void Wizard1_FinishButtonClick(object sender,  
WizardNavigationEventArgs e)  
{  
    Update();  
}
```

```
private void Update()
{
    // Update the color.

    pnlCard.BackColor = Color.FromName(lstBackColor.SelectedItem.Text);
    lblGreeting.ForeColor = Color.FromName(lstForeColor.SelectedItem.Text);

    // Update the font.

    lblGreeting.Font.Name = lstFontName.SelectedItem.Text;
    try
    {
        if (Int32.Parse(txtFontSize.Text) > 0)
        {
            lblGreeting.Font.Size = FontUnit.Point(Int32.Parse(txtFontSize.Text));
        }
    }
    catch
    {
        // Ignore invalid value.
    }

    try
    {
        if (Int32.Parse(txtFontSize.Text) > 0)
        {
            lblGreeting.Font.Size =
                FontUnit.Point(Int32.Parse(txtFontSize.Text));
        }
    }
    catch
    {
        // Ignore invalid value.
    }
}
```

```

// Find the appropriate TypeConverter for the BorderStyle enumeration.
TypeConverter cnvrt = TypeDescriptor.GetConverter(typeof(BorderStyle));

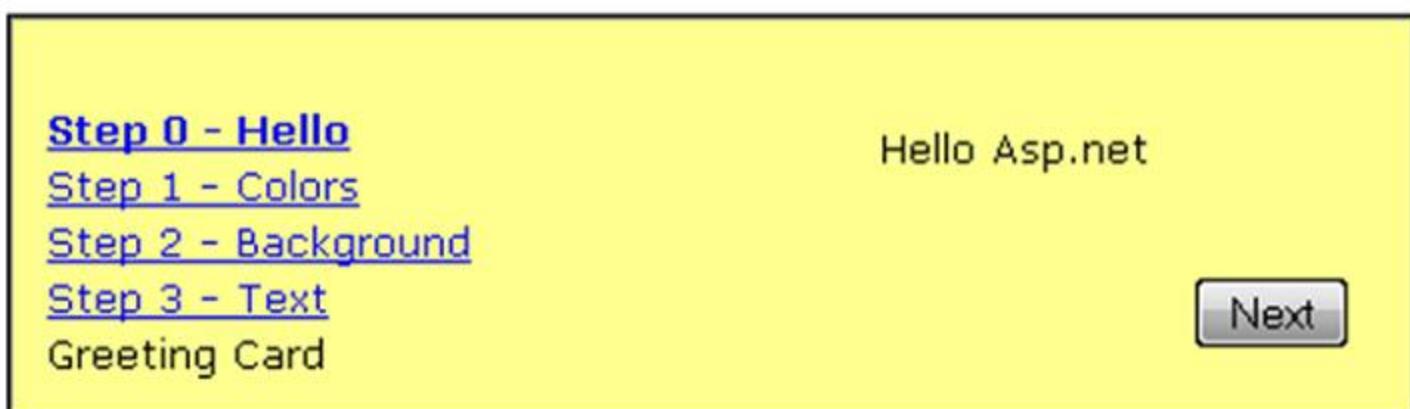
// Update the border style using the value from the converter.
pnlCard.BorderStyle = (BorderStyle)cnvrt.ConvertFromString(
    lstBorder.SelectedItem.Text);

// Update the picture.
if (chkPicture.Checked == true)
{
    imgDefault.Visible = true;
}
else
{
    imgDefault.Visible = false;
}

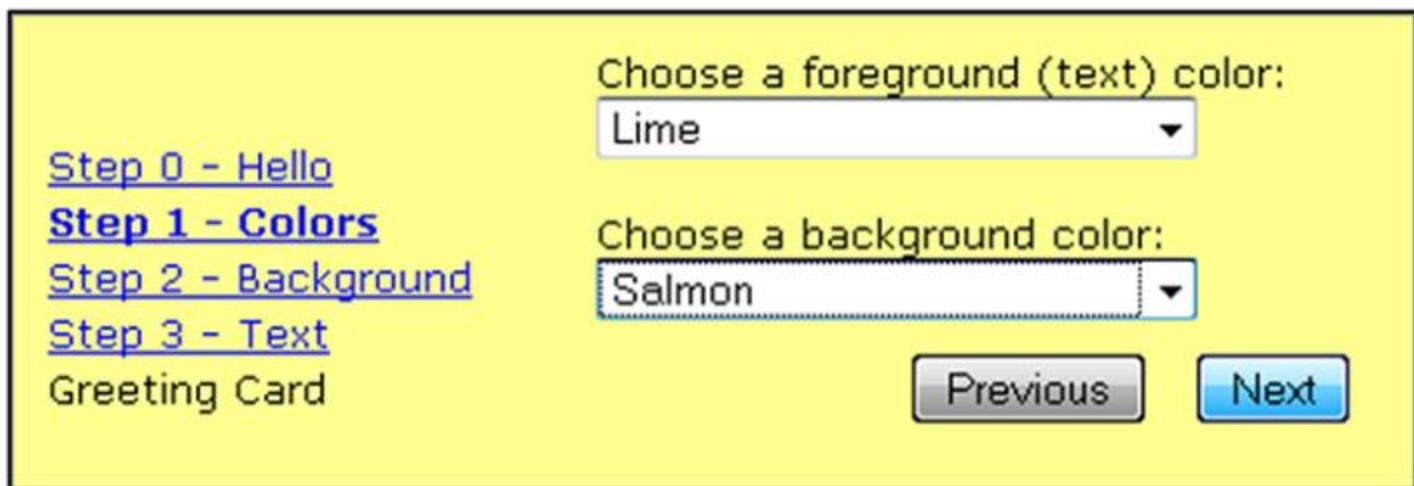
// Set the text.
lblGreeting.Text = txtGreeting.Text;
}
}

```

Kết quả thực thi trang WizardGreetingCardMaker.aspx:



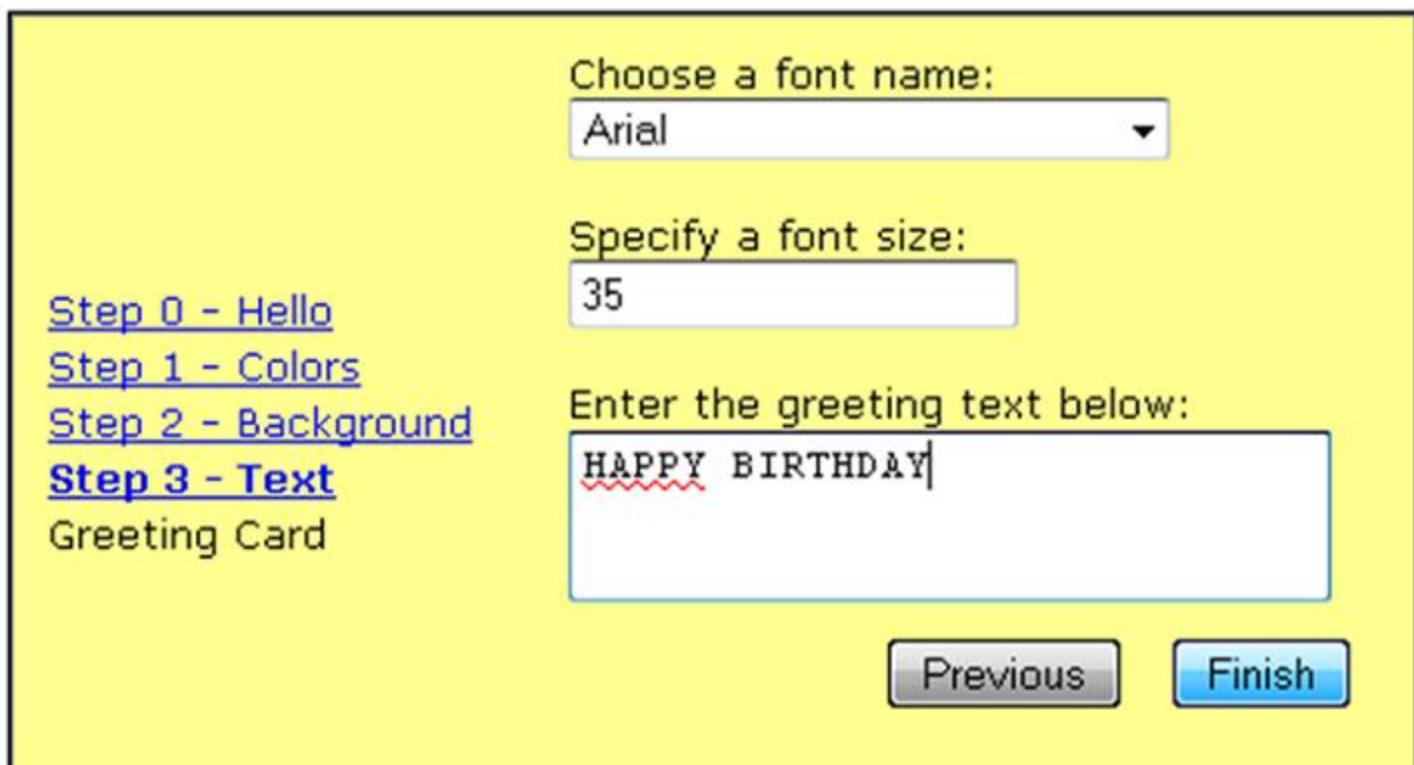
Chọn giá trị xong, click nút [Next].



Chọn giá trị xong, click nút [Next].



Chọn giá trị xong, click nút [Next].



Chọn giá trị xong, click nút [Finish].

HAPPY
BIRTHDAY



Chương 7

GIỚI THIỆU ADO.NET

Các vấn đề chính sẽ được đề cập:

- ✓ *Kiến trúc ADO.NET*
- ✓ *Các đối tượng trong ADO.NET: Connected Layer (Connection, Command, DataReader, DataAdapter) và Disconnected Layer (DataSet, DataTable, DataRow, DataColumn, DataRelation)*
- ✓ *Đối tượng SqlDataSource*

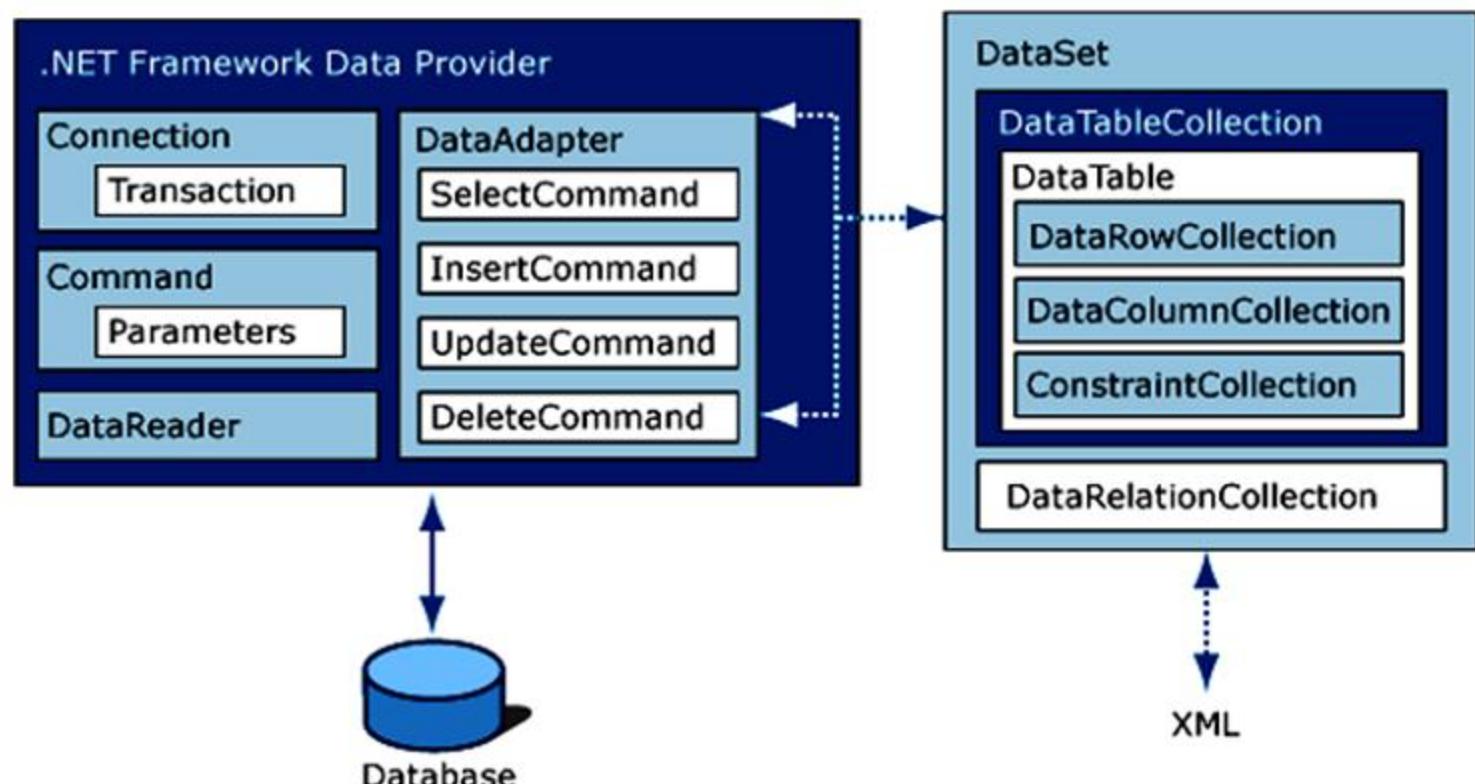
Kết thúc chương này các bạn có thể:

- *Trình bày được kiến trúc ADO.NET*
- *Mô tả và sử dụng đối tượng trong ADO.NET trong ứng dụng ASP.NET*
- *Sử dụng đối tượng SqlDataSource*

7.1. KIẾN TRÚC ADO.NET

ADO.NET là một phần của .NET Framework, được xem là “bộ thư viện lớp” chịu trách nhiệm xử lý dữ liệu trong ngôn ngữ MS.NET.

ADO.NET gồm hai thành phần chính cho việc truy xuất và điều khiển dữ liệu đó là các trình cung cấp dữ liệu .NET Framework (.NET Framework Data Provider) và DataSet. Mô hình dưới đây minh họa mối quan hệ giữa trình cung cấp dữ liệu của .NET Framework và DataSet.

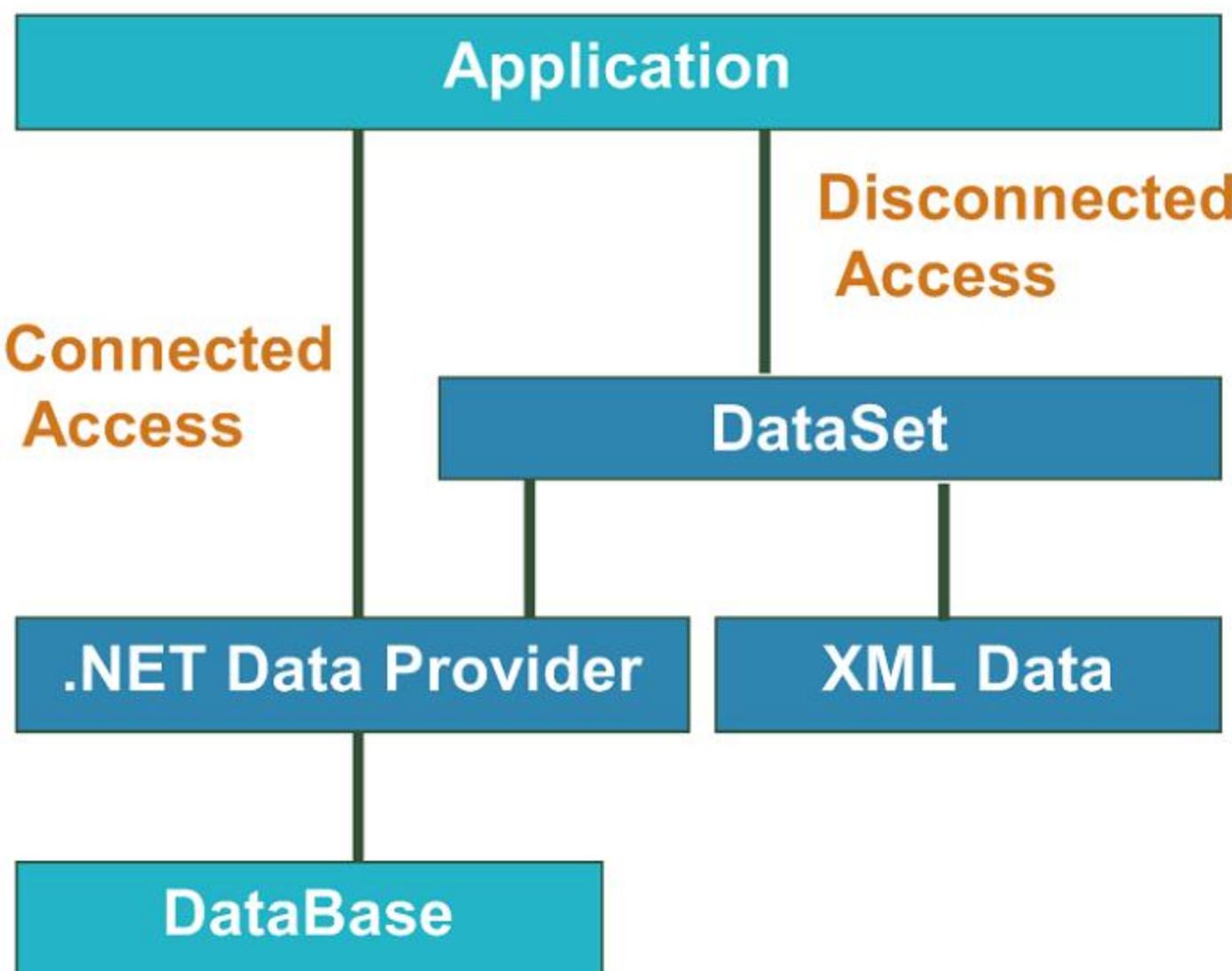


Hình 7.1: Kiến trúc ADO.NET

Nhìn chung việc kết nối giữa ứng dụng và cơ sở dữ liệu (CSDL) thông qua ADO.NET theo hai hướng:

- **Connected Layer:** cơ chế này yêu cầu phải thực hiện kết nối với Database trong khi đang thực hiện các thao tác với dữ liệu. Các đối tượng của cơ chế này là:
 - **Connection:** Đối tượng quản lý đóng/mở kết nối tới Database.
 - **Command:** Đối tượng thực hiện các câu lệnh tương tác truy vấn, rút trích dữ liệu từ database sau khi đã thiết lập kết nối tới dữ liệu và trả về kết quả.
 - **DataReader:** Đối tượng xử lý đọc dữ liệu từ CSDL.
 - **DataAdapter:** Đây là đối tượng rất quan trọng của ADO.NET, là cầu nối của Database và Dataset (Dataset là đối tượng ngắt kết nối), bởi vì đối tượng “ngắt kết nối” Dataset không thể tương tác trực tiếp với Database nên cần một đối tượng trung gian lấy dữ liệu từ Database. Và đó chính là DataAdapter. Vì DataAdapter khi thao tác với Database vẫn phải duy trì kết nối nên nó được liệt kê vào dạng “kết nối”, nhưng bản chất phục vụ cho việc “ngắt kết nối”.
- **Disconnected Layer:** chỉ có một đối tượng chịu trách nhiệm ngắt kết nối đó chính là **DataSet**. Nhiệm vụ của DataSet là nhận dữ liệu về từ DataAdapter và xử lý nó. DataSet có thể được xem như một Database trong bộ nhớ gồm tất cả các bảng, quan hệ..... DataSet có nhiều đối tượng được xem là “con” như: **DataTable**, cấp thấp hơn của DataTable có các đối tượng **DataRow**, **DataColumn**, **DataRelation**. Ngoài ra còn có các đối tượng nhóm: ví dụ: **DataTableCollection**, **DataRowCollection**, **DataColumnCollection**. Việc sử dụng DataSet là một tiền bối lớn của kiến trúc ADO.NET.

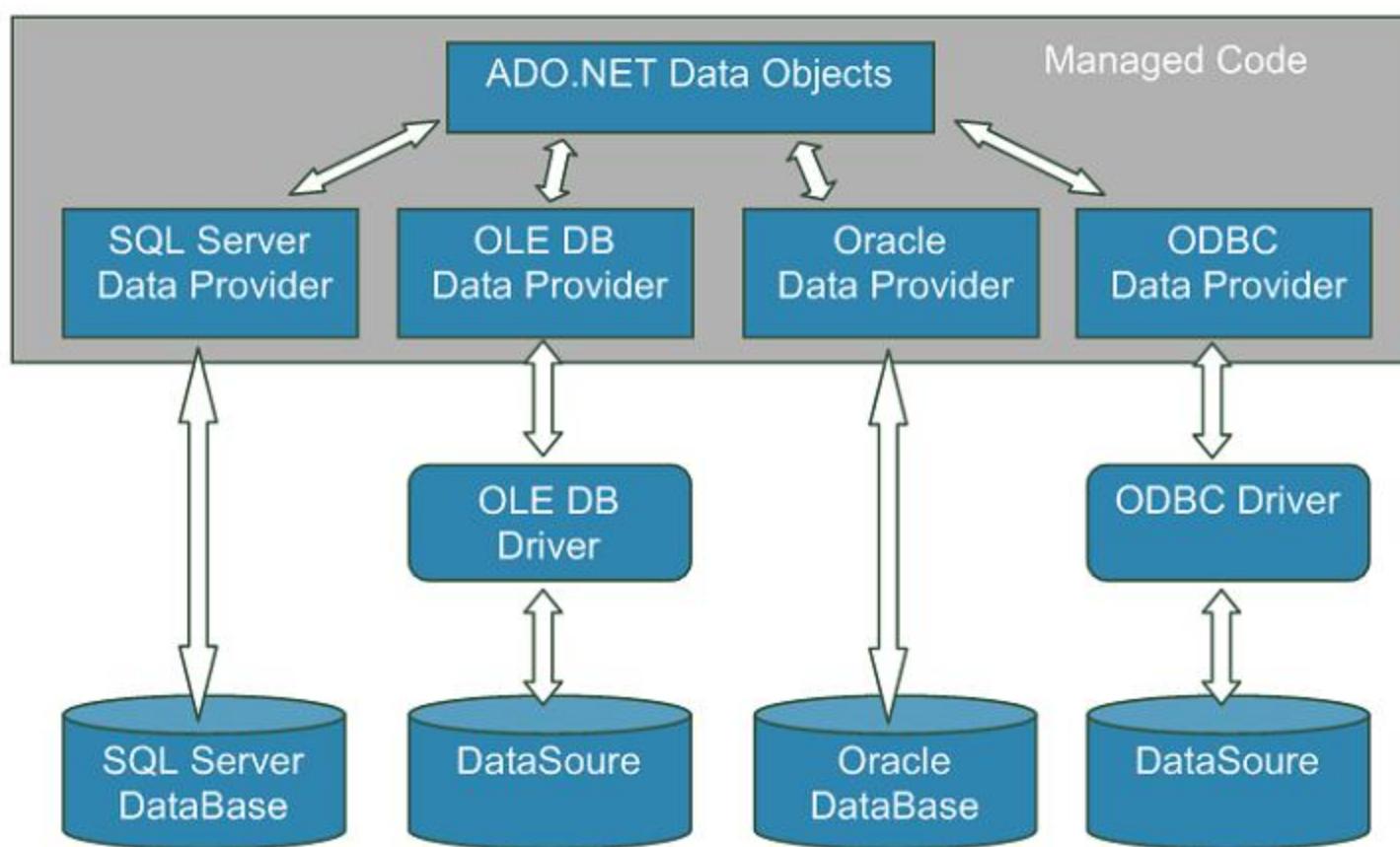
Như vậy có thể nhìn một cách tổng quát như sau:



Hình 7.2: Cơ chế kết nối ADO.NET

7.2. TÌM HIỂU TRÌNH CUNG CẤP DỮ LIỆU CỦA ADO.NET

ADO.NET hỗ trợ nhiều trình cung cấp dữ liệu, mỗi trình cung cấp tương tác với một DBMS cụ thể. Lợi ích đầu tiên của phương pháp này là một trình cung cấp dữ liệu cụ thể có thể được lập trình để truy xuất đến bất cứ thuộc tính nào của một DBMS mà được chỉ định bởi trình cung cấp đó. Lợi ích khác đó là trình cung cấp dữ liệu có thể kết nối ngay lập tức đến CSDL mà không thông qua tầng ánh xạ trung gian (trình điều khiển) giữa các lớp (hình 7.3).



Hình 7.3: Các lớp giữa code và nguồn dữ liệu

Một trình cung cấp dữ liệu là tập hợp các kiểu được định nghĩa trong namespace System.Data, cho biết cách để giao tiếp với một nguồn dữ liệu cụ thể. Một trình cung cấp định nghĩa một tập hợp các kiểu lớp mà cung cấp các chức năng cơ bản.

Bảng 7.1: Các đối tượng cơ bản của một trình cung cấp dữ liệu ADO.NET.

Object	Base Class	Implemented Interfaces	Ý nghĩa
Connection	DbConnection	IDbConnection	Cung cấp khả năng kết nối và ngắt kết nối với nguồn dữ liệu
Command	DbCommand	IDbCommand	Trình bày một truy vấn SQL hoặc stored procedure
DataReader	DbDataReader	IDataReader, IDataRecord	Đọc nguồn dữ liệu theo một chiều
DataAdapter	DbDataAdapter	IDataAdapter, IDbDataAdapter	Chuyển DataSets giữa các đối tượng gọi (caller) và kho dữ liệu
Parameter	DbParameter	IDataParameter, IDbDataParameter	Miêu tả tên gọi tham số bên trong truy vấn có tham số (parameterized query)

Transaction	DbTransaction	IDbTransaction	Đóng gói (Encapsulates) một database transaction
--------------------	----------------------	-----------------------	--

Mặc dù tên cụ thể của các đối tượng cơ bản này sẽ khác nhau tùy thuộc trình cung cấp dữ liệu (ví dụ: SqlConnection khác với OracleConnection khác với OdbcConnection khác với MySqlConnection), nhưng mỗi đối tượng đều kế thừa từ một lớp cơ sở (base class) (DbConnection trong trường hợp các đối tượng kết nối) và thi hành interface (như IDbConnection).

- **Các trình cung cấp dữ liệu ADO.NET được cung cấp bởi Microsoft**

.NET của Microsoft phân phối số lượng lớn các trình cung cấp dữ liệu, bao gồm một trình cung cấp cho Oracle, SQL Server, và khả năng liên kết OLE DB/ODBC. Bảng 7.2 trình bày namespace và assembly cho mỗi trình cung cấp dữ liệu của Microsoft

Bảng 7.2: Các trình cung cấp dữ liệu ADO.NET của Microsoft

Data Provider	Namespace	Assembly
OLE DB	System.Data.OleDb	System.Data.dll
Microsoft SQL Server	System.Data.SqlClient	System.Data.dll
Microsoft SQL Server Mobile	System.Data.SqlServerCe	System.Data.SqlServerCe.dll
ODBC	System.Data.Odbc	System.Data.dll
Oracle	System.Data.OracleClient	System.Data.OracleClient.dll

Trình cung cấp dữ liệu OLE DB và ODBC chỉ hữu ích nếu tương tác với một DBMS mà không định nghĩa một trình cung cấp dữ liệu .NET cụ thể.

7.3. CÁC NAMESPACE CỦA ADO.NET

.NET cung cấp một số namespace cho ADO.NET, một vài namespace trong đó được thể hiện trong bảng 7.3.

Bảng 7.3: Các namespace của ADO.NET

Namespace	Ý nghĩa
Microsoft.SqlServer.Server	Namespace này cung cấp các loại phục vụ việc tích hợp CLR và SQL Server 2005 được dễ dàng
System.Data	Namespace này định nghĩa các loại ADO.NET cơ sở được sử dụng bởi tất cả trình cung cấp dữ liệu, bao gồm các Interface phổ biến và các loại thể hiện lớp ngắt kết nối (DataSet, DataTable, etc.)
System.Data.Common	Namespace này chứa các loại được dùng chung trong tất cả các trình cung cấp của ADO.NET, bao gồm các lớp cơ sở ảo (abstract base class) phổ biến
System.Data.SqlClient	Namespace này chứa các loại mà cho phép nhận ra các thể hiện Microsoft SQL Server được cài đặt trên mạng cục bộ hiện hành
System.Data.SqlTypes	Namespace này chứa các loại dữ liệu tự nhiên (native data types) được dùng bởi Microsoft SQL Server

Namespace System.Data

Namespace này chứa các loại được dùng chung trong tất cả các trình cung cấp dữ liệu của ADO.NET. Bảng 7.4 liệt kê một vài loại chính.

Bảng 7.4: Các thành viên chính của **System.Data** Namespace

Type	Ý nghĩa
Constraint	Trình bày (Represent) một ràng buộc cho một đối tượng DataColumn đã cho
DataTable	Trình bày một column bên trong một đối tượng DataTable
DataRelation	Trình bày mối quan hệ parent/child giữa hai đối tượng DataTable
DataRow	Trình bày một dòng bên trong một đối tượng DataTable
DataSet	Trình bày một vùng trong bộ nhớ của dữ liệu bao gồm mọi thành phần của đối tượng DataTable có quan hệ với nhau

DataTable	Trình bày bảng trong bộ nhớ
DataTableReader	Cho phép xem xét một DataTable
DataView	Trình bày một view của một DataTable cho việc sắp xếp, việc lọc, việc tìm kiếm, chỉnh sửa và điều hướng
IDataAdapter	Định nghĩa cách hành xử chính (core behavior) của một đối tượng DataAdapter
IDataParameter	Định nghĩa cách hành xử chính (core behavior) của một đối tượng parameter
IDataReader	Định nghĩa cách hành xử chính (core behavior) của một đối tượng DataReader
IDbCommand	Định nghĩa cách hành xử chính (core behavior) của một đối tượng command
IDbDataAdapter	Kế thừa IDataAdapter để cung cấp thêm các chức năng của một đối tượng DataAdapter
IDbTransaction	Định nghĩa cách hành xử cốt lõi (core behavior) của một đối tượng Transaction

- **Vai trò của IDbConnection Interface**

Được thi hành bởi đối tượng connection của trình cung cấp dữ liệu. Interface này định nghĩa một tập hợp các thành phần được dùng để tạo một kết nối đến kho dữ liệu, và nó cũng cho phép lấy được đối tượng Transaction của trình cung cấp dữ liệu. Dưới đây định nghĩa của IDbConnection:

```
public interface IDbConnection: IDisposable
{
    string ConnectionString { get; set; }
    int ConnectionTimeout { get; }
    string Database { get; }
    ConnectionState State { get; }
    IDbTransaction BeginTransaction();
    IDbTransaction BeginTransaction(IsolationLevel il);
    void ChangeDatabase(string databaseName);
    void Close();
    IDbCommand CreateCommand();
    void Open();
}
```

Chú ý: phương thức Close() là tương đương về mặt chức năng với việc gọi phương thức Dispose() trực tiếp hoặc gián tiếp bên trong phạm vi using của C#.

- **Vai trò của IDbTransaction Interface**

Việc nạp chồng phương thức BeginTransaction() (overloaded BeginTransaction() method) được định nghĩa bởi IDbConnection cung cấp truy xuất đến đối tượng Transaction của trình cung cấp. Việc sử dụng các thành phần được định nghĩa bởi IDbTransaction, chúng ta có thể tương tác với một phiên chuyển giao (transactional session) và kho dữ liệu ở dưới.

```
public interface IDbTransaction: IDisposable
{
    IDbConnection Connection { get; }
    IsolationLevel IsolationLevel { get; }
    void Commit();
    void Rollback();
}
```

- **Vai trò của IDbCommand Interface**

IDbCommand Interface được thi hành bởi đối tượng command của trình cung cấp dữ liệu. Đối tượng command cho phép thao tác với các câu lệnh SQL, Stored Procedures, và các truy vấn có tham số (parameterized queries). Trong phần bổ sung, các đối tượng command truy xuất đến DataReader của trình cung cấp dữ liệu thông qua việc nạp chồng phương thức ExecuteReader() (overloaded ExecuteReader() method).

```
public interface IDbCommand: IDisposable
{
    string CommandText { get; set; }
    int CommandTimeout { get; set; }
    CommandType CommandType { get; set; }
    IDbConnection Connection { get; set; }
    IDataParameterCollection Parameters { get; }
    IDbTransaction Transaction { get; set; }
    UpdateRowSource UpdatedRowSource { get; set; }
    void Cancel();
    IDbDataParameter CreateParameter();
    int ExecuteNonQuery();
```

```
    IDataReader ExecuteReader();
    IDataReader ExecuteReader(CommandBehavior behavior);
    object ExecuteScalar();
    void Prepare();
}
```

- **Vai trò của Interface IDbDataParameter và IDataParameter**

Interface này cung cấp truy xuất đến một tập hợp các loại lớp tùy ý (compliant) IDbDataParameter (ví dụ: các đối tượng tham số):

```
public interface IDbDataParameter: IDataParameter
{
    byte Precision { get; set; }
    byte Scale { get; set; }
    int Size { get; set; }
}
```

IDbDataParameter kế thừa (extend) interface IDataParameter để có được các hành xử (behaviors) bổ sung sau đây:

```
public interface IDataParameter
{
    DbType DbType { get; set; }
    ParameterDirection Direction { get; set; }
    bool IsNullable { get; }
    string ParameterName { get; set; }
    string SourceColumn { get; set; }
    DataRowVersion SourceVersion { get; set; }
    object Value { get; set; }
}
```

Chức năng của các Interface IDbDataParameter và IDataParameter cho phép trình bày các tham số bên trong một lệnh SQL (bao gồm Stored Procedures) qua các đối tượng parameter của ADO.NET cụ thể hơn là những ký tự chuỗi hard-coded.

- **Vai trò của IDbDataAdapter and IDataAdapter Interfaces.**

IDbDataAdapter Interface định nghĩa một tập hợp các thuộc tính mà được dùng để duy trì các câu lệnh SQL cho các hoạt động có liên quan đến select, insert, update và delete.

```
public interface IDbDataAdapter: IDataAdapter
{
    IDbCommand DeleteCommand { get; set; }
    IDbCommand InsertCommand { get; set; }
    IDbCommand SelectCommand { get; set; }
    IDbCommand UpdateCommand { get; set; }
}
```

Có bốn thuộc tính được thêm vào, DataAdapter cũng được định nghĩa trong Interface cơ sở, IDataAdapter. Interface này định nghĩa chức năng chính của một loại DataAdapter: khả năng chuyển các DataSet giữa đối tượng gọi và kho dữ liệu bên dưới sử dụng các phương thức Fill() và Update(). IDataAdapter Interface cho phép ánh xạ các tên cột trong bảng dữ liệu thành các tên hiển thị thân thiện với người dùng hơn thông qua thuộc tính TableMappings.

```
public interface IDataAdapter
{
    MissingMappingAction MissingMappingAction { get; set; }
    MissingSchemaAction MissingSchemaAction { get; set; }
    ITableMappingCollection TableMappings { get; }
    int Fill(System.Data.DataSet dataSet);
    DataTable[] FillSchema(DataSet dataSet, SchemaType schemaType);
    IDataParameter[] GetFillParameters();
    int Update(DataSet dataSet);
}
```

- **Vai trò của IDataReader and IDataRecord**

Interface IDataReader trình bày các hành xử (behaviors) phổ biến được hỗ trợ bởi một đối tượng DataReader cho trước.

```
public interface IDataReader: IDisposable, IDataRecord
{
    int Depth { get; }
    bool IsClosed { get; }
    int RecordsAffected { get; }
    void Close();
    DataTable GetSchemaTable();
    bool NextResult();
```

```
    bool Read();  
}
```

IDataReader kế thừa IDataRecord, định nghĩa một số thành phần cho phép trích một giá trị được phân loại một cách rõ ràng. Dưới đây là danh sách không đầy đủ của các phương thức GetXXX() khác nhau được định nghĩa bởi IDataRecord.

```
public interface IDataRecord  
{  
    int FieldCount { get; }  
  
    object this[ string name ] { get; }  
    object this[ int i ] { get; }  
  
    bool GetBoolean(int i);  
    byte GetByte(int i);  
    char GetChar(int i);  
    DateTime GetDateTime(int i);  
    Decimal GetDecimal(int i);  
    float GetFloat(int i);  
    short GetInt16(int i);  
    int GetInt32(int i);  
    long GetInt64(int i);  
  
    ...  
    bool IsDBNull(int i);  
}
```

Chú ý: phương thức `IDataReader.IsDBNull()` thiết lập giá trị cho một trường là null.

7.4. TÌM HIỂU CƠ CHẾ KẾT NỐI CỦA ADO.NET QUA CONNECTED LAYER

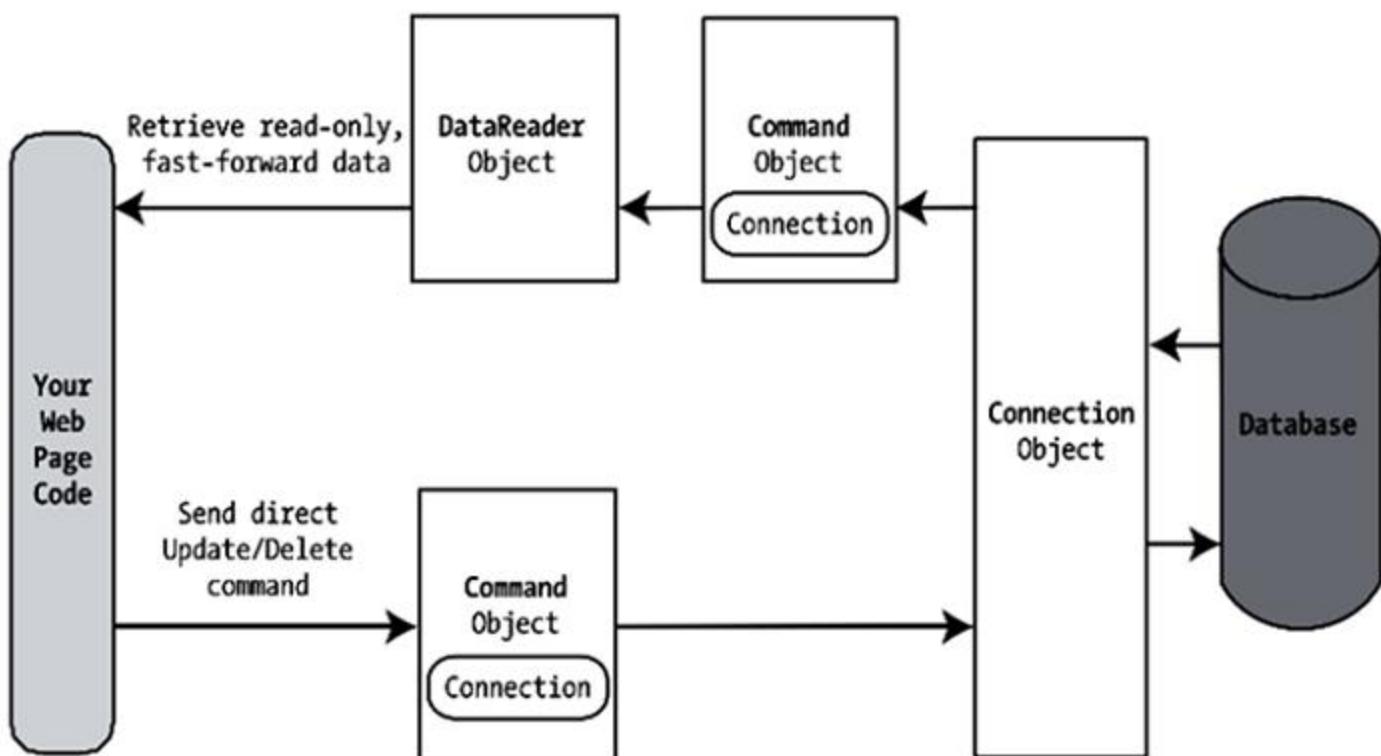
Để kết nối với CSDL và đọc các mẫu tin sử dụng đối tượng DataReader, thực hiện các bước sau:

B1. Chỉ định, cấu hình và mở đối tượng kết nối.

B2. Chỉ định và cấu hình đối tượng Command, chỉ rõ đối tượng Connection như một Constructor có đối số hoặc qua thuộc tính Connection.

B3. Gọi `ExecuteReader()` trên đối tượng Command đã được cấu hình.

B4. Xử lý mỗi mẫu tin sử dụng phương thức `Read()` của DataReader.



Hình 7.4: Kết nối đến CSDL và đọc các mẫu tin theo hai cách

- **Làm việc với các đối tượng Connection**

Các dạng khai báo chuỗi kết nối tùy theo loại Database Provider

Các khai báo Connection String:

[OleDb Provider]

```
OleDbConnection myConnection = new OleDbConnection();
```

- // Windows Authentication

```
myConnection.ConnectionString =
"Provider=SQLOLEDB.1;Data Source=localhost;" + "Initial
Catalog=Pubs;Integrated Security=SSPI";
```

- // SQL Authentication

```
myConnection.ConnectionString =
"Provider=SQLOLEDB.1;Data Source=localhost;" + "Initial
Catalog=Pubs;User ID=sa;Password=123456";
```

[SQLServer Provider]

```
SqlConnection myConnection = new SqlConnection();
```

- myConnection.ConnectionString = "Data Source=localhost;" +
"Initial Catalog=Pubs;Integrated Security=SSPI";

[AttachDBFile] với tập tin CSDL

```
myConnection.ConnectionString = @"Data
Source=localhost\SQLEXPRESS;" +
```

```
@"AttachDBFilename=|DataDirectory|\Pubs.mdf;Integrated
Security=True";
```

Bước thứ nhất khi làm việc với một trình cung cấp dữ liệu là thiết lập một session với nguồn dữ liệu sử dụng đối tượng connection (kế thừa từ **DbConnection**). Các đối tượng **Connection** của .NET được cung cấp với một chuỗi kết nối được định dạng, gồm một số cặp name/value được phân cách bởi dấu chấm phẩy (;). Thông tin này được dùng để chỉ định tên máy sẽ kết nối, các thiết lập bảo mật được yêu cầu, tên của CSDL trên máy đó, và thông tin trình cung cấp dữ liệu cụ thể khác.

Tên **Initial Catalog** xác định tên CSDL. Data Source xác định tên máy chứa CSDL. Ở đây, (local) cho phép chúng ta định nghĩa một Token riêng biệt (single) để chỉ rõ máy cục bộ hiện hành, khi token \SQLEXPRESS cho trình cung cấp dữ liệu SQL biết chúng ta đang kết nối đến bản cài đặt SQL Server Express edition (nếu chúng ta đã có Pubs trên SQL Server 2005/2008 hoặc các phiên bản trước đây, chỉ rõ Data Source=(local)\SQLEXPRESS).

Integrated Security thiết lập thông tin bảo mật, nếu chọn Integrated Security là SSPI (tương đương true), để sử dụng tài khoản Windows hiện hành để xác nhận quyền người dùng.

User ID + Password thiết lập thông tin bảo mật, nếu chọn để sử dụng tài khoản SQL Server hiện hành để xác nhận quyền người dùng.

Sau khi chuỗi kết nối được thiết lập, gọi **Open()** mở kết nối với CSDL. Ngoài việc bổ sung các thành phần **ConnectionString**, **Open()**, và **Close()**, một đối tượng kết nối cung cấp một số thành phần mà cho phép thêm vào các thiết lập thuộc về cấu hình kết nối, như là các thiết lập **Timeout**, thông tin thuộc về giao tác (**transactional**). Bảng 7.5 liệt kê một vài các thành phần của lớp cơ sở **DbConnection**.

Bảng 7.5: Các thành phần của **DbConnection**

Member	Ý nghĩa
BeginTransaction()	Phương thức này được sử dụng để bắt đầu một database Transaction
ChangeDatabase()	Phương thức này thay đổi CSDL trên một kết nối mở (open connection)
ConnectionTimeout	Thuộc tính này trả ra một khoảng thời gian chờ trong khi thiết lập một kết nối trước khi kết thúc và tạo ra một lỗi (mặc định là 15 giây)

Database	Thuộc tính này lấy ra tên của CSDL được duy trì bởi đối tượng connection
DataSource	Thuộc tính này lấy vị trí của CSDL được duy trì bởi đối tượng connection
GetSchema()	Phương thức này trả ra một DataSet mà chứa thông tin bảng từ nguồn dữ liệu
State	Thuộc tính này thiết lập trạng thái hiện hành của việc kết nối, được thể hiện bởi việc liệt kê ConnectionState

Thí dụ 1

Tạo trang ConnectionTester.aspx thực hiện việc tạo kết nối đến CSDL MS SQL Server sử dụng đối tượng SqlConnection với chuỗi kết nối có dạng sau:

```
// Thiết lập chuỗi kết nối đến database Pubs.

string connectionString = "Data Source=localhost\\SQLEXPRESS;Initial Catalog=pubs;";

if (optWindows.Checked)
    { // xác thực với Windows Authenticate
        connectionString += "Integrated Security=SSPI";
    }
else
    { // xác thực với SQL Authenticate
        connectionString += "User ID=sa;Password=123456";
    }

// Define the ADO.NET Connection object.

SqlConnection myConnection = new SqlConnection(connectionString);
```

Nội dung thiết kế trang ConnectionTester.aspx

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

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

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

```

<title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:RadioButton id="optSQL" runat="server" Text="Use SQL Authentication
            (with sa account)" GroupName="Authentication" Font-Size="Smaller" Font-
            Names="Verdana"></asp:RadioButton>
            <br />
            <asp:RadioButton id="optWindows" runat="server" Text="Use
            Windows Integrated Authentication" GroupName="Authentication" Font-
            Size="Smaller" Font-Names="Verdana" Checked="True"></asp:RadioButton>
            <br />
            <br />
            <asp:button id="cmdConnect" runat="server" Text="Connect"
            onclick="cmdConnect_Click"></asp:button>
            <br />
            <br />
            <asp:label id="lblInfo" runat="server" Height="128px"
            Width="464px" Font-Size="Small" Font-Names="Verdana"
            ForeColor="Maroon"></asp:label>
        </div>
    </form>
</body>
</html>

```

Phần mã lệnh ConnectionTester.aspx.cs

```

public partial class ConnectionTester : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected void cmdConnect_Click(object sender, EventArgs e)
    {
        // Define the connection string.
        string connectionString = "Data Source=localhost\\SQLEXPRESS;Initial
        Catalog=pubs;";
    }
}

```

```
if (optWindows.Checked)
{
    connectionString += "Integrated Security=SSPI";
}
else
{
    connectionString += "User ID=sa;Password=123456";
}

// Define the ADO.NET Connection object.
SqlConnection myConnection = new SqlConnection(connectionString);

try
{
    // Try to open the connection.
    myConnection.Open();
    lblInfo.Text = "<b>Server Version:</b> " + myConnection.ServerVersion;
    lblInfo.Text += "<br /><b>Connection Is:</b> " +
myConnection.State.ToString();
}
catch (Exception err)
{
    // Handle an error by displaying the information.
    lblInfo.Text = "Error reading the database. ";
    lblInfo.Text += err.Message;
}
finally
{
    // Either way, make sure the connection is properly closed.
    // (Even if the connection wasn't opened successfully,
    // calling Close() won't cause an error.)
    myConnection.Close();
    lblInfo.Text += "<br /><b>Now Connection Is:</b> ";
    lblInfo.Text += myConnection.State.ToString();
}
}
```

Kết quả thực thi trong hai trường hợp kết nối CSDL bị lỗi hay thành công.

<p>● Use SQL Authentication (with sa account) ○ Use Windows Integrated Authentication</p> <p>Connect</p> <p>Error reading the database. Cannot open database "Pubs" requested by the login. The login failed. Login failed for user 'sa'. Now Connection Is: Closed</p>	<p>● Use SQL Authentication (with sa account) ○ Use Windows Integrated Authentication</p> <p>Connect</p> <p>Server Version: 09.00.1399 Connection Is: Open Now Connection Is: Closed</p>
---	--

- **Làm việc với các đối tượng ConnectionStringBuilder**

Làm việc với các chuỗi kết nối có thể hơi cồng kềnh, chúng được trình bày như các ký tự chuỗi, mà thật khó khăn để bảo trì và dễ xảy ra lỗi. Các trình cung cấp dữ liệu ADO.NET được Microsoft đã hỗ trợ các đối tượng trình tạo chuỗi kết nối (*connection string builder*), cho phép thiết lập các cặp name/value sử dụng các thuộc tính được định kiểu rõ ràng. Ở đây, tạo một thể hiện của **SqlConnectionStringBuilder**, thiết lập các thuộc tính phù hợp, và lấy được chuỗi ở trong thông qua thuộc tính **ConnectionString**. Một khi đã kết hợp đối tượng với dữ liệu chuỗi ban đầu, chúng ta có thể thay đổi cặp giá trị name/value sử dụng các thuộc tính có liên quan.

Thí dụ 2

Thêm vào một đoạn mã lệnh sử dụng đối tượng **SqlConnectionStringBuilder** để lấy nội dung trong chuỗi **connectionString** và thiết lập giá trị name/value tương ứng cho các thuộc tính **DataSource**, **UserID**, **Password**,...

```
protected void cmdConnect_Click(object sender, EventArgs e)
{
    // Define the connection string.
    string connectionString = "Data Source=localhost\\SQLEXPRESS;Initial
Catalog=Pubs;";

    if (optWindows.Checked)
    {
        connectionString += "Integrated Security=SSPI";
    }
    else
    {
        connectionString += "User ID=sa;Password=123456";
    }
}
```

```
}

// Define the ADO.NET Connection object.

SqlConnection myConnection = new SqlConnection(connectionString);

try
{
    // Try to open the connection.

    myConnection.Open();

    lblInfo.Text = "<b>Server Version:</b> " + myConnection.ServerVersion;

    lblInfo.Text += "<br /><b>Connection Is:</b> " +
myConnection.State.ToString();

    // Create a new SqlConnectionStringBuilder based on the
    // partial connection string retrieved from the config file.

    SqlConnectionStringBuilder builder =
        new SqlConnectionStringBuilder(connectionString);

    SqlConnectionStringBuilder builder2 =
        new SqlConnectionStringBuilder();

    // Supply the additional values.

    builder2.DataSource = builder.DataSource;
    builder2.UserID = builder.UserID;
    builder2.Password = builder.Password;

    lblInfo.Text += "<br /><b>ConnectionString Is:</b> " +
builder2.ConnectionString;

}

catch (Exception err)
{
    // Handle an error by displaying the information.

    lblInfo.Text = "Error reading the database. ";
    lblInfo.Text += err.Message;

}

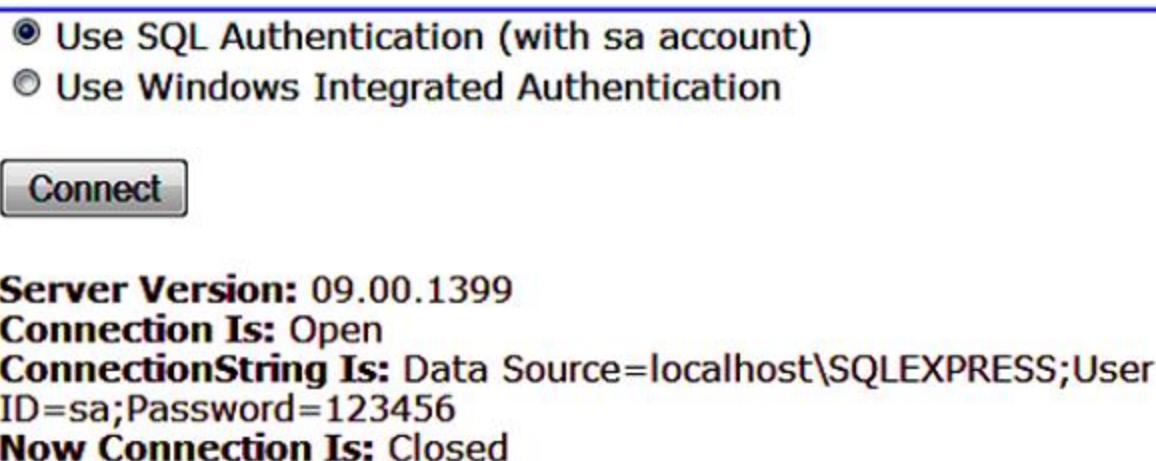
finally
{
    // Either way, make sure the connection is properly closed.
}
```

```

    // (Even if the connection wasn't opened successfully,
    // calling Close() won't cause an error.)
    myConnection.Close();
    lblInfo.Text += "<br /><b>Now Connection Is:</b> ";
    lblInfo.Text += myConnection.State.ToString();
}
}

```

Kết quả thực thi trang ConnectionTester.aspx



Một cách khác, thực hiện việc khai báo cấu hình ConnectionString trong **Web.config**: không cần phải hard-code ở các trang mã lệnh

```

<configuration>
.....
<connectionStrings>
    <add name="Northwind" connectionString="Data
Source=(local)\SQLEXPRESS;Initial Catalog=Northwind;Integrated Security=SSPI"/>
</connectionStrings>
.....
</configuration>

```

Lấy nội dung ConnectionString trong thẻ `<connectionStrings>` trong tập tin Web.config. Chú ý phải khai báo thêm:

using System.Web.Configuration;

```

private string connectionString =
    WebConfigurationManager.ConnectionStrings["Pubs"].ConnectionString;

```

- **Làm việc với các đối tượng Command.**

Sau khi kết nối với CSDL, việc tiếp theo là submit các truy vấn SQL đến CSDL. Kiểu SqlCommand (ké thừa từ DbCommand). Kiểu của Command theo lý thuyết sử dụng thuộc tính CommandType, mà có thể lấy ra bất cứ giá trị nào từ CommandType enum:

```
public enum CommandType
{
    StoredProcedure,
    TableDirect,
    Text // Default value.
}
```

Khi tạo một đối tượng command, cũng có thể kết hợp truy vấn SQL như một Constructor có tham số hoặc trực tiếp thông qua thuộc tính CommandText. Ngoài ra khi đang tạo một đối tượng command, cần chỉ rõ kết nối được sử dụng. Mặt khác, có thể tạo một Constructor có đối số hoặc thông qua thuộc tính Connection:

```
SqlConnection cn = new SqlConnection();
...
// Create command object via ctor args.
string strSQL = "Select * From Customers";
SqlCommand myCommand = new SqlCommand(strSQL, cn);
// Create another command object via properties.
SqlCommand testCommand = new SqlCommand();
testCommand.Connection = cn;
testCommand.CommandText = strSQL;
...
```

Bảng 7.6: Các thành phần của DbCommand

Member	Ý nghĩa
CommandTimeout	Lấy ra hoặc thiết lập thời gian chờ trong khi đang thực thi lệnh trước khi kết thúc
Connection	Lấy ra hoặc thiết lập DbConnection được dùng bởi thẻ hiện này của DbCommand
Parameters	Lấy ra tập hợp các kiểu của SqlParameter được dùng cho một truy vấn có tham số
Cancel()	Hủy bỏ việc thực thi của một lệnh
ExecuteReader()	Thực hiện câu lệnh trong CommandText. Kết quả

	trả về là DataReader của trình cung cấp dữ liệu
ExecuteNonQuery()	Thực hiện câu lệnh trong CommandText và không có kết quả trả về
ExecuteScalar()	Thực hiện câu lệnh trong CommandText, kết quả trả về là một giá trị đơn
ExecuteXmlReader()	Phương thức này trả ra một System.Xml.XmlReader mà cho phép xử lý luồng đầu vào của XML
Prepare()	Tạo một phiên bản được chuẩn bị (hoặc được biên dịch) của lệnh trên nguồn dữ liệu

- **Làm việc với các Data Reader**

Một khi đã thiết lập một kết nối thực sự và câu lệnh SQL, bước tiếp theo là **submit** truy vấn đến nguồn dữ liệu. Kiểu DbDataReader (mà thực thi giao diện **IDataReader**) là cách đơn giản và chắc chắn để lấy ra thông tin từ một kho dữ liệu.

Các đối tượng **DataReader** thu được từ đối tượng **Command** qua lời gọi đến **ExecuteReader()**. Khi làm việc với phương thức này, có thể đóng một cách tự động đối tượng kết nối có liên quan bởi việc chỉ rõ **CommandBehavior.CloseConnection**.

Việc sử dụng phương thức **Read()** của DataReader để duyệt qua các mẫu tin, (trả ra giá trị false nếu duyệt qua mẫu tin cuối cùng). Chú ý rằng, chúng ta gọi **Close()** ngay khi kết thúc việc xử lý các mẫu tin, để giải phóng đối tượng kết nối:

Indexer của một đối tượng DataReader đã được overload để lấy một chuỗi (string) (miêu tả tên của cột) hoặc một int (miêu tả số thứ tự của cột). Do đó, có thể tránh các tên chuỗi hard-code với việc cập nhật dưới đây (chú ý sử dụng thuộc tính FieldCount)

Thí dụ 3

Tạo trang AuthorBrowser.aspx thực hiện đọc nội dung của bảng Authors trong CSDL Pubs và đưa vào một DropDownList. Khi chọn một giá trị trong DropDownList, nội dung mẫu tin lựa chọn sẽ được hiển thị trên trang.

Nội dung thiết kế trang AuthorBrowser.aspx

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

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:label id="Label1" runat="server" Width="120px" Height="20px">Select
Author:</asp:label>&nbsp;
            <asp:dropdownlist id="lstAuthor" runat="server" Width="256px"
Height="22px" AutoPostBack="True"
onselectedindexchanged="lstAuthor_SelectedIndexChanged">
                </asp:dropdownlist>
                <br />      <br />      <br />
                <asp:label id="lblResults" runat="server" Width="384px" Height="168px">
                </asp:label>
            </div>
        </form>
    </body>
</html>

```

Phần mã lệnh trang AuthorBrowser.aspx.cs

```

public partial class AuthorBrowser: System.Web.UI.Page
{
    private string connectionString =
        WebConfigurationManager.ConnectionStrings["Pubs"].ConnectionString;
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!this.IsPostBack)
        {
            FillAuthorList();
        }
    }

    private void FillAuthorList()
    {

```

```
IstAuthor.Items.Clear();

// Define the Select statement.
// Three pieces of information are needed: the unique id,
// and the first and last name.
string selectSQL = "SELECT au_lname, au_fname, au_id FROM Authors";

// Define the ADO.NET objects.
SqlConnection con = new SqlConnection(connectionString);
SqlCommand cmd = new SqlCommand(selectSQL, con);
SqlDataReader reader;

// Try to open database and read information.
try
{
    con.Open();
    reader = cmd.ExecuteReader();

    // For each item, add the author name to the displayed
    // list box text, and store the unique ID in the Value Thuộc tính.
    while (reader.Read())
    {
        ListItem newItem = new ListItem();
        newItem.Text = reader["au_lname"] + ", " + reader["au_fname"];
        newItem.Value = reader["au_id"].ToString();
        lstAuthor.Items.Add(newItem);
    }
    reader.Close();
}
catch (Exception err)
{
    lblResults.Text = "Error reading list of names. ";
    lblResults.Text += err.Message;
}
finally
{
```

```
    con.Close();
}
}

protected void lstAuthor_SelectedIndexChanged(object sender, EventArgs e)
{
    // Create a Select statement that searches for a record
    // matching the specific author id from the Value Thuộc tính.
    string selectSQL;
    selectSQL = "SELECT * FROM Authors ";
    selectSQL += "WHERE au_id=" + lstAuthor.SelectedItem.Value + "";

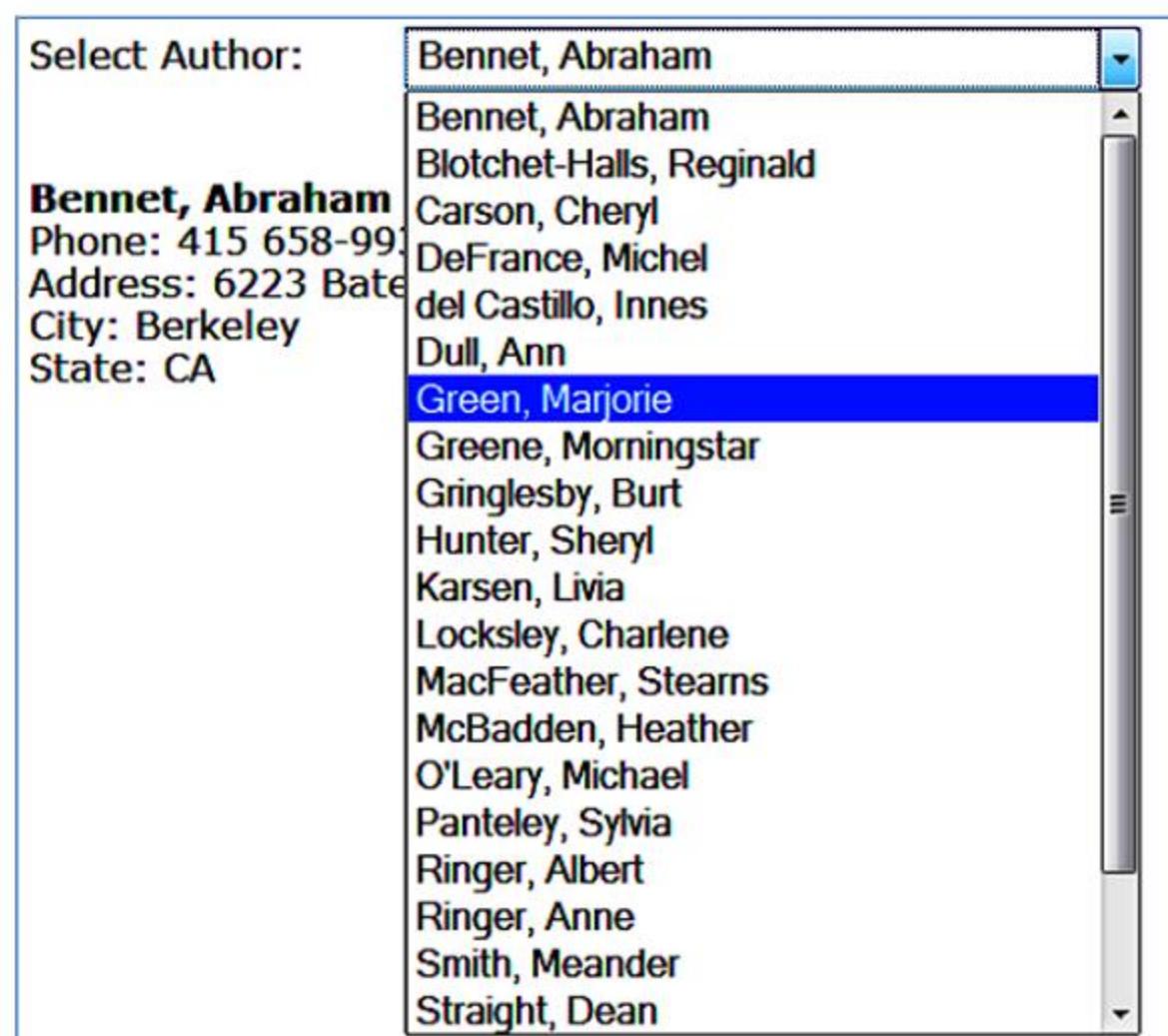
    // Define the ADO.NET objects.
    SqlConnection con = new SqlConnection(connectionString);
    SqlCommand cmd = new SqlCommand(selectSQL, con);
    SqlDataReader reader;
    // Try to open database and read information.
    try
    {
        con.Open();
        reader = cmd.ExecuteReader();
        reader.Read();
        StringBuilder sb = new StringBuilder();
        sb.Append("<b>");
        sb.Append(reader["au_lname"]);
        sb.Append(", ");
        sb.Append(reader["au_fname"]);
        sb.Append("</b><br />");
        sb.Append("Phone: ");
        sb.Append(reader["phone"]);
        sb.Append("<br />");
        sb.Append("Address: ");
        sb.Append(reader["address"]);
        sb.Append("<br />");
        sb.Append("City: ");
        sb.Append(reader["city"]);
        sb.Append("<br />");
```

```

        sb.Append("State: ");
        sb.Append(reader["state"]);
        sb.Append("<br />");
        lblResults.Text = sb.ToString();
        reader.Close();
    }
    catch (Exception err)
    {
        lblResults.Text = "Error getting author. ";
        lblResults.Text += err.Message;
    }
    finally
    {
        con.Close();
    }
}

```

Kết quả thực hiện:



Kết quả thực thi sau khi lựa chọn giá trị trong **DropDownList**, thực thi sự kiện **SelectedIndexChanged** cho ra thông tin giá trị đã chọn:

Select Author: **Green, Marjorie**

Green, Marjorie
Phone: 415 986-7020
Address: 309 63rd St. #411
City: Oakland
State: CA

- **Thu được nhiều bộ kết quả sử dụng DataReader**

Các đối tượng **DataReader** có thể thu được nhiều bộ kết quả sử dụng một đối tượng **Command**. Thí dụ, để thu được tất cả các dòng từ bảng Inventory cũng như từ bảng Customers, chúng ta có thể chỉ định các câu lệnh SQL select sử dụng dấu chấm phẩy (;) cuối dòng:

```
string strSQL = "Select * From Inventory; Select * from Customers";
```

Một khi chúng ta thu được **DataReader**, chúng ta có thể duyệt qua các mẫu tin thông qua phương thức **NextResult()**.

```
do{  
    while (myDataReader.Read())  
    {  
        .....  
        for (int i = 0; i < myDataReader.FieldCount; i++)  
        {  
            .....  
            myDataReader.GetName(i);  
            myDataReader.GetValue(i).ToString().Trim();  
        }  
        .....  
    }  
} while (myDataReader.NextResult());
```

- **Làm việc với các đối tượng Parameterized Command**

Một **parameterized query** được dùng để trình bày các tham số SQL như các đối tượng. Các truy vấn có tham số thực thi nhanh hơn một chuỗi

SQL bình thường, bởi vì chúng được phân tách một cách rõ ràng (hơn là chuỗi SQL được gắn đến thuộc tính **CommandText**).

Để hỗ trợ các truy vấn có tham số, đối tượng Command của ADO.NET duy trì một tập hợp của các đối tượng tham số riêng biệt. Mặc định, tập hợp này empty, nhưng chúng ta dễ dàng để thêm bất cứ giá trị nào của đối tượng tham số mà ánh xạ đến một “tham số thay thế” trong truy vấn SQL. Khi muốn kết hợp một tham số bên trong một truy vấn SQL đến một thành viên trong tập hợp các tham số của đối tượng command, thêm tiền tố @ trước tham số SQL.

• Chỉ rõ các tham số sử dụng **DbParameter**

Để xây dựng một truy vấn có tham số, chúng ra bắt đầu với **DbParameter** (là lớp cở sở cho đối tượng tham số cụ thể của một trình cung cấp). Lớp này chứa một số thuộc tính mà cho phép chúng ta cấu hình tên, kích cỡ và kiểu dữ liệu của tham số, cũng như các đặc điểm khác như là hướng của tham số.

Bảng 7.7: Các thành phần chính của **DbParameter**

Thuộc tính	Ý nghĩa
DbType	Lấy ra hoặc thiết lập kiểu dữ liệu tự nhiên từ nguồn dữ liệu, được trình bày như kiểu dữ liệu CLR
Direction	Lấy ra hoặc thiết lập tham số chỉ nhập, chỉ xuất, cả hai, hoặc trả ra tham số giá trị
IsNullable	Lấy ra hoặc thiết lập tham số chấp nhận các giá trị Null
ParameterName	Lấy ra hoặc thiết lập tên của DbParameter
Size	Lấy ra hoặc thiết lập kích cỡ tham số tối thiểu của dữ liệu (chỉ thật sự hữu ích cho dữ liệu là văn bản)
Value	Lấy ra hoặc thiết lập giá trị cho tham số

Để minh họa tập hợp của các đối tượng command của các đối tượng **DBParameter** quan sát đoạn code sau:

```
// Define ADO.NET objects.  
  
string insertSQL;  
  
insertSQL = "INSERT INTO Authors (";  
insertSQL += "au_id, au_fname, au_lname, ";  
insertSQL += "phone, address, city, state, zip, contract) ";
```

```

insertSQL += "VALUES (";
insertSQL += "@au_id, @au_fname, @au_lname, ";
insertSQL += "@phone, @address, @city, @state, @zip, @contract)";

SqlConnection con = new SqlConnection(connectionString);
SqlCommand cmd = new SqlCommand(insertSQL, con);

```

Sau đó đưa các Parameters với các giá trị lấy từ các điều khiển nhập vào trong đối tượng Command hiện tại **cmd**

```

// Add the parameters.

cmd.Parameters.AddWithValue("@au_id", txtID.Text);
cmd.Parameters.AddWithValue("@au_fname", txtFirstName.Text);
cmd.Parameters.AddWithValue("@au_Lname", txtLastName.Text);
cmd.Parameters.AddWithValue("@phone", txtPhone.Text);
cmd.Parameters.AddWithValue("@address", txtAddress.Text);
cmd.Parameters.AddWithValue("@city", txtCity.Text);
cmd.Parameters.AddWithValue("@state", txtState.Text);
cmd.Parameters.AddWithValue("@zip", txtZip.Text);
cmd.Parameters.AddWithValue("@contract",
Convert.ToInt16(chkContract.Checked));

```

.....

Hoặc khởi tạo đối tượng **SqlParameter** và đặt vào trong phần danh sách tham số của đối tượng **Command**.

```

// Add the parameters.

cmd.Parameters.Add(new SqlParameter("@au_id", txtID.Text));
cmd.Parameters.Add(new SqlParameter("@au_fname", txtFirstName.Text));
cmd.Parameters.Add(new SqlParameter("@au_Lname", txtLastName.Text));
cmd.Parameters.Add(new SqlParameter("@phone", txtPhone.Text));
cmd.Parameters.Add(new SqlParameter("@address", txtAddress.Text));
cmd.Parameters.Add(new SqlParameter("@city", txtCity.Text));
cmd.Parameters.Add(new SqlParameter("@state", txtState.Text));
cmd.Parameters.Add(new SqlParameter("@zip", txtZip.Text));
cmd.Parameters.Add(new SqlParameter("@contract",
Convert.ToInt16(chkContract.Checked)));

```

Thí dụ 4

Tạo trang AuthorManager.aspx thiết kế với giao diện sau:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="AuthorManager.aspx.cs" Inherits="AuthorManager" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <div style="background-color: #FFE0C0; height: 70px; width: 601px; padding: 7px">
                <br />
            <asp:Label id="Label1" runat="server" Width="99px" Height="19px">Select
Author:</asp:Label>
            <asp:DropDownList id="lstAuthor" runat="server" Width="187px" Height="21px"
AutoPostBack="True"
onselectedindexchanged="lstAuthor_SelectedIndexChanged"></asp:DropDownList>&
nbsp;&nbsp;&nbsp;
            <asp:Button id="cmdUpdate" runat="server" Text="Update"
onclick="cmdUpdate_Click"></asp:Button>&nbsp;
            <asp:Button id="cmdDelete" runat="server" Text="Delete"
onclick="cmdDelete_Click"></asp:Button> <br />
            <asp:Label id="Label11" runat="server" Width="99px"
Height="19px">Or:</asp:Label>
            <asp:Button id="cmdNew" runat="server" Width="91px" Height="24px" Text="Create
New" onclick="cmdNew_Click"></asp:Button>&nbsp;
            <asp:Button id="cmdInsert" runat="server" Width="85px" Height="24px" Text="Insert
New" onclick="cmdInsert_Click"></asp:Button>
        </div>
        <br />
        <div style="background-color: #E0E0E0; height: 392px; width: 601px; padding: 7px">
            <asp:Label id="Label7" runat="server" Width="70px">Unique ID:</asp:Label>
            <asp:TextBox id="txtID" runat="server" Width="184px"></asp:TextBox>&nbsp;
```

```

(required:&nbsp;###-##-#### form)<br />

<asp:Label id="Label2" runat="server" Width="70px">First  

Name:</asp:Label>
<asp:TextBox id="txtFirstName" runat="server"
Width="184px"></asp:TextBox><br />

<asp:Label id="Label3" runat="server" Width="70px">Last  

Name:</asp:Label>
<asp:TextBox id="txtLastName" runat="server"
Width="183px"></asp:TextBox><br />

<asp:Label id="Label4" runat="server"
Width="70px">Phone:</asp:Label>
<asp:TextBox id="txtPhone" runat="server"
Width="183px"></asp:TextBox><br />

<asp:Label id="Label5" runat="server"
Width="70px">Address:</asp:Label>
<asp:TextBox id="txtAddress" runat="server"
Width="183px"></asp:TextBox><br />

<asp:Label id="Label6" runat="server" Width="70px">City:</asp:Label>
<asp:TextBox id="txtCity" runat="server" Width="184px"></asp:TextBox><br />
<asp:Label id="Label7" runat="server" Width="70px">State:</asp:Label>
<asp:TextBox id="txtState" runat="server" Width="184px"></asp:TextBox><br />
<asp:Label id="Label9" runat="server" Width="70px">Zip Code:</asp:Label>
<asp:TextBox id="txtZip" runat="server" Width="184px"></asp:TextBox>&nbsp;
(required: any five&nbsp;digits)<br /> <br />

<asp:Label id="Label8" runat="server" Width="93px"
Height="19px">Contract:</asp:Label>&nbsp;
<asp:CheckBox id="chkContract" runat="server"></asp:CheckBox><br /> <br />
<asp:Label id="lblResults" runat="server" Width="575px" Height="121px" Font-
Bold="True"></asp:Label>
</div>
</div>

```

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

Giao diện trang AuthorManager.aspx như hình sau:

The screenshot shows a web page with an orange header bar. In the header, there is a dropdown menu labeled "Select Author" with "Unbound" selected, and buttons for "Update" and "Delete". Below the header, there is a section labeled "Or:" with buttons for "Create New" and "Insert New". The main content area has a grey background and contains the following fields:

Unique ID:	<input type="text"/>	(required: ####-##-##### form)
First Name:	<input type="text"/>	
Last Name:	<input type="text"/>	
Phone:	<input type="text"/>	
Address:	<input type="text"/>	
City:	<input type="text"/>	
State:	<input type="text"/>	
Zip Code:	<input type="text"/>	(required: any five digits)
Contract:	<input type="checkbox"/> [chkContract]	

At the bottom left of the content area, there is a button labeled "[tblResults]".

Phần mã lệnh thực thi trên trang AuthorManager.aspx.cs

```
public partial class AuthorManager: System.Web.UI.Page  
{  
    private string connectionString =  
        WebConfigurationManager.ConnectionStrings["Pubs"].ConnectionString;  
    protected void Page_Load(object sender, EventArgs e)  
    {  
        if (!this.IsPostBack)  
        {  
            FillAuthorList();  
        }  
    }  
    private void FillAuthorList()  
    {  
        lstAuthor.Items.Clear();  
    }
```

```
// Define the Select statement.  
// Three pieces of information are needed: the unique id,  
// and the first and last name.  
string selectSQL = "SELECT au_lname, au_fname, au_id FROM Authors";  
// Define the ADO.NET objects.  
SqlConnection con = new SqlConnection(connectionString);  
SqlCommand cmd = new SqlCommand(selectSQL, con);  
SqlDataReader reader;  
// Try to open database and read information.  
try  
{  
    con.Open();  
    reader = cmd.ExecuteReader();  
    // For each item, add the author name to the displayed  
    // list box text, and store the unique ID in the Value Thuộc tính.  
    while (reader.Read())  
    {  
        ListItem newItem = new ListItem();  
        newItem.Text = reader["au_lname"] + ", " + reader["au_fname"];  
        newItem.Value = reader["au_id"].ToString();  
        lstAuthor.Items.Add(newItem);  
    }  
    reader.Close();  
}  
catch (Exception err)  
{  
    lblResults.Text = "Error reading list of names. ";  
    lblResults.Text += err.Message;  
}  
finally  
{  
    con.Close();  
}
```

```
}

protected void lstAuthor_SelectedIndexChanged(object sender, EventArgs e)
{
    // Define ADO.NET objects.

    string selectSQL;
    selectSQL = "SELECT * FROM Authors ";
    selectSQL += "WHERE au_id=" + lstAuthor.SelectedItem.Value + "";
    SqlConnection con = new SqlConnection(connectionString);
    SqlCommand cmd = new SqlCommand(selectSQL, con);
    SqlDataReader reader;

    // Try to open database and read information.

    try
    {
        con.Open();
        reader = cmd.ExecuteReader();
        reader.Read();

        // Fill the controls.

        txtID.Text = reader["au_id"].ToString();
        txtFirstName.Text = reader["au_fname"].ToString();
        txtLastName.Text = reader["au_lname"].ToString();
        txtPhone.Text = reader["phone"].ToString();
        txtAddress.Text = reader["address"].ToString();
        txtCity.Text = reader["city"].ToString();
        txtState.Text = reader["state"].ToString();
        txtZip.Text = reader["zip"].ToString();
        chkContract.Checked = (bool)reader["contract"];
        reader.Close();
        lblResults.Text = "";
    }
    catch (Exception err)
    {
        lblResults.Text = "Error getting author. ";
        lblResults.Text += err.Message;
    }
}
```

```
        }

        finally
        {
            con.Close();
        }
    }

protected void cmdNew_Click(object sender, EventArgs e)
{
    txtID.Text = "";
    txtFirstName.Text = "";
    txtLastName.Text = "";
    txtPhone.Text = "";
    txtAddress.Text = "";
    txtCity.Text = "";
    txtState.Text = "";
    txtZip.Text = "";
    chkContract.Checked = false;
    lblResults.Text = "Click Insert New to add the completed record.";
}

protected void cmdInsert_Click(object sender, EventArgs e)
{
    // Perform user-defined checks.
    if (txtID.Text == "" || txtFirstName.Text == "" || txtLastName.Text == "")
    {
        lblResults.Text = "Records require an ID, first name, and last name.";
        return;
    }

    // Define ADO.NET objects.
    string insertSQL;
    insertSQL = "INSERT INTO Authors (";

    insertSQL += "au_id, au_fname, au_lname, ";
    insertSQL += "phone, address, city, state, zip, contract) ";
    insertSQL += "VALUES (";
    insertSQL += "@au_id, @au_fname, @au_lname, ";
```

```
insertSQL += "@phone, @address, @city, @state, @zip, @contract)";

SqlConnection con = new SqlConnection(connectionString);
SqlCommand cmd = new SqlCommand(insertSQL, con);
// Add the parameters.
cmd.Parameters.Add(new SqlParameter("@au_id", txtID.Text));
cmd.Parameters.Add(new SqlParameter("@au_fname", txtFirstName.Text));
cmd.Parameters.Add(new SqlParameter("@au_Lname", txtLastName.Text));
cmd.Parameters.Add(new SqlParameter("@phone", txtPhone.Text));
cmd.Parameters.Add(new SqlParameter("@address", txtAddress.Text));
cmd.Parameters.Add(new SqlParameter("@city", txtCity.Text));
cmd.Parameters.Add(new SqlParameter("@state", txtState.Text));
cmd.Parameters.Add(new SqlParameter("@zip", txtZip.Text));
cmd.Parameters.Add(new SqlParameter("@contract",
Convert.ToInt16(chkContract.Checked)));

// Try to open the database and execute the update.
int added = 0;
try
{
    con.Open();
    added = cmd.ExecuteNonQuery();
    lblResults.Text = added.ToString() + " record inserted.";
}
catch (Exception err)
{
    lblResults.Text = "Error inserting record. ";
    lblResults.Text += err.Message;
}
finally
{
    con.Close();
}

// If the insert succeeded, refresh the author list.
```

```
if (added > 0)
{
    FillAuthorList();
}
}

protected void cmdUpdate_Click(object sender, EventArgs e)
{
    // Define ADO.NET objects.
    string updateSQL;
    updateSQL = "UPDATE Authors SET ";
    updateSQL += "au_fname=@au_fname, au_lname=@au_lname, ";
    updateSQL += "phone=@phone, address=@address, city=@city, state=@state, ";
    updateSQL += "zip=@zip, contract=@contract ";
    updateSQL += "WHERE au_id=@au_id_original";

    SqlConnection con = new SqlConnection(connectionString);
    SqlCommand cmd = new SqlCommand(updateSQL, con);

    // Add the parameters.
    cmd.Parameters.AddWithValue("@au_fname", txtFirstName.Text);
    cmd.Parameters.AddWithValue("@au_Lname", txtLastName.Text);
    cmd.Parameters.AddWithValue("@phone", txtPhone.Text);
    cmd.Parameters.AddWithValue("@address", txtAddress.Text);
    cmd.Parameters.AddWithValue("@city", txtCity.Text);
    cmd.Parameters.AddWithValue("@state", txtState.Text);
    cmd.Parameters.AddWithValue("@zip", txtZip.Text);
    cmd.Parameters.AddWithValue("@contract",
        Convert.ToInt16(chkContract.Checked));
    cmd.Parameters.AddWithValue("@au_id_original",
        lstAuthor.SelectedItem.Value);

    // Try to open database and execute the update.
    int updated = 0;
    try
    {
        con.Open();
```

```
updated = cmd.ExecuteNonQuery();

lblResults.Text = updated.ToString() + " record updated.";

}

catch (Exception err)
{
    lblResults.Text = "Error updating author. ";
    lblResults.Text += err.Message;
}

finally
{
    con.Close();
}

// If the updated succeeded, refresh the author list.
if (updated > 0)
{
    FillAuthorList();
}

}

protected void cmdDelete_Click(object sender, EventArgs e)
{
    // Define ADO.NET objects.

    string deleteSQL;

    deleteSQL = "DELETE FROM Authors ";
    deleteSQL += "WHERE au_id=@au_id";

    SqlConnection con = new SqlConnection(connectionString);
    SqlCommand cmd = new SqlCommand(deleteSQL, con);
    cmd.Parameters.AddWithValue("@au_id ", lstAuthor.SelectedItem.Value);

    // Try to open the database and delete the record.

    int deleted = 0;
    try
    {
        con.Open();
        deleted = cmd.ExecuteNonQuery();
```

```

lblResults.Text = "Record deleted.";

}

catch (Exception err)
{
    lblResults.Text = "Error deleting author. ";
    lblResults.Text += err.Message;
}

finally
{
    con.Close();
}

// If the delete succeeded, refresh the author list.
if (deleted > 0)
{
    FillAuthorList();
}
}
}
}

```

Kết quả thực thi:

Select Author: **Bennet, Abraham**

Or:

Unique ID:	<input type="text"/>	(required: ####-##-##### form)
First Name:	<input type="text"/>	
Last Name:	<input type="text"/>	
Phone:	<input type="text"/>	
Address:	<input type="text"/>	
City:	<input type="text"/>	
State:	<input type="text"/>	
Zip Code:	<input type="text"/>	(required: any five digits)
Contract:	<input type="checkbox"/>	

Khi click chọn một tác giả, thông tin tác giả sẽ điền vào các điều khiển dữ liệu nhập phía dưới:



Unique ID:	238-95-7766	(required: ####-##-##### form)
First Name:	Cheryl	
Last Name:	Carson	
Phone:	415 548-7723	
Address:	589 Darwin Ln.	
City:	Berkeley	
State:	CA	
Zip Code:	94705	(required: any five digits)
Contract:	<input checked="" type="checkbox"/>	

Thực thi việc tạo mới một mẫu tin tác giả, click nút [Create New], hệ thống sẽ đặt các giá trị điều khiển nhập dữ liệu về ban đầu (rỗng) cho phép việc nhập thông tin mới.

```
protected void cmdNew_Click(object sender, EventArgs e) {  
    txtID.Text = "";    txtFirstName.Text = "";  
    txtLastName.Text = "";  
    txtPhone.Text = "";  
    txtAddress.Text = "";  
    txtCity.Text = "";  
    txtState.Text = "";  
    txtZip.Text = "";  
    chkContract.Checked = false;  
    lblResults.Text = "Click Insert New to add the completed record.";  
}
```

Sau khi nhập xong click nút [Insert New], triệu gọi phương thức **cmdInsert_Click(object sender, EventArgs e)**

```
protected void cmdInsert_Click(object sender, EventArgs e)  
{  
    // Perform user-defined checks.  
    if (txtID.Text == "" || txtFirstName.Text == "" || txtLastName.Text == "")  
    {  
        lblResults.Text = "Records require an ID, first name, and last name.";  
    }  
}
```

```

    return;
}

// Define ADO.NET objects.

string insertSQL;
insertSQL = "INSERT INTO Authors (";

insertSQL += "au_id, au_fname, au_lname, ";
insertSQL += "phone, address, city, state, zip, contract) ";
insertSQL += "VALUES (";

insertSQL += "@au_id, @au_fname, @au_lname, ";
insertSQL += "@phone, @address, @city, @state, @zip, @contract)";

SqlConnection con = new SqlConnection(connectionString);
SqlCommand cmd = new SqlCommand(insertSQL, con);

// Add the parameters.

cmd.Parameters.Add(new SqlParameter("@au_id", txtID.Text));
cmd.Parameters.Add(new SqlParameter("@au_fname", txtFirstName.Text));
cmd.Parameters.Add(new SqlParameter("@au_Lname", txtLastName.Text));
cmd.Parameters.Add(new SqlParameter("@phone", txtPhone.Text));
cmd.Parameters.Add(new SqlParameter("@address", txtAddress.Text));
cmd.Parameters.Add(new SqlParameter("@city", txtCity.Text));
cmd.Parameters.Add(new SqlParameter("@state", txtState.Text));
cmd.Parameters.Add(new SqlParameter("@zip", txtZip.Text));
cmd.Parameters.Add(new SqlParameter("@contract",
Convert.ToInt16(chkContract.Checked)));

// Try to open the database and execute the update.

int added = 0;
try {
    con.Open();
    added = cmd.ExecuteNonQuery();
    lblResults.Text = added.ToString() + " record inserted.";
}
catch (Exception err) {
    lblResults.Text = "Error inserting record. ";
    lblResults.Text += err.Message;
}
finally

```

```

{
    con.Close();
}

// If the insert succeeded, refresh the author list.

if (added > 0)
{
    FillAuthorList();
}
}

```

Sau khi sửa đổi dữ liệu của mẫu tin tác giả hiện tại trong các khung nhập dữ liệu, click nút **[Update]** để cập nhật lại. Muốn xóa mẫu tin tác giả hiện tại, click nút **[Delete]**.

- **Tìm hiểu các Database Transaction**

Để ràng buộc việc kiểm tra của lớp được kết nối của ADO.NET, chúng ta sẽ tìm khái niệm của một Database Transaction. Một cách đơn giản, Transaction (giao tác) là một tập các hành động cùng thực hiện một chức năng, và chúng chỉ có thể cùng nhau thành công, hoặc cùng nhau thất bại.

Các Transaction là rất quan trọng khi một thao tác CSDL cần phải tác động với nhiều bảng hoặc nhiều Stored Procedure (hoặc phối hợp của các phần nhỏ của CSDL).

Thí dụ:

Ngân hàng của Soda muốn tăng lãi suất tiền gửi cho 700 khách hàng. Do đó, Soda phải cập nhật 700 record trong CSDL. Tuy nhiên, trong khi server cập nhật đến record thứ 500 thì bị... cúp điện. Kết quả, chỉ có 500 khách hàng được tăng, 500 khách hàng còn lại ngồi chờ Soda... xin lỗi.

Chờ đến khi có điện, Soda sẽ có hai lựa chọn: một là quay lùi, bỏ việc cập nhật cho 500 record đầu tiên, hai là cập nhật tiếp 500 record còn lại.

- Việc cập nhật 700 record như trên được gọi là một Transaction (giao tác)
- Việc quay lùi, bỏ cập nhật 500 record đầu tiên, được gọi là quá trình Rollback (quay lùi).
- Việc tiếp tục cập nhật 500 record còn lại, được gọi là quá trình Commit (tạm dịch là "cam kết").

Thông thường, ta sử dụng Transaction cho các câu lệnh SQL insert, update, delete trong CSDL. Nếu đang thực hiện mà bị mất điện hay treo máy, thì khi hệ thống hoạt động lại, bộ quản lý Transaction sẽ tự động thực hiện điều này mà không cần người dùng phải nhúng tay vào.

Chúng ta có thể lập trình với Transaction bên trong hệ quản trị CSDL (ví dụ, viết bên trong StoreProcedure của SQL Server). Hoặc chúng ta có thể lập trình ngay trong ứng dụng của mình, thông qua đối tượng Transaction của ADO.NET.

- **Các thành phần chính của một đối tượng giao tác (Transaction) của ADO.NET**

Các đối tượng Transaction được tìm thấy bên trong trình cung cấp dữ liệu ADO.NET, tất cả được kế thừa từ DBTransaction và thực thi interface IDbTransaction.

```
public interface IDbTransaction: IDisposable
{
    IDbConnection Connection { get; }

    IsolationLevel IsolationLevel { get; }

    void Commit();

    void Rollback();
}
```

Chú ý: Thuộc tính Connection, mà sẽ trả ra một tham chiếu đến đối tượng Connection mà đã khởi tạo Transaction hiện hành. Phương thức Commit() được gọi khi mỗi thao tác trên CSDL thành công. Ngược lại, phương thức Rollback() có thể được gọi khi một biệt lệ trong thời gian thực thi xuất hiện, và sẽ thông báo DMBS để bỏ qua bất cứ sự thay đổi nào đang xem xét, để lại không đụng chạm đến dữ liệu ban đầu.

Chú ý: thuộc tính IsolationLevel của một đối tượng Transaction cho phép chúng ta chỉ rõ một Transaction phải được xem xét chống lại các hành động của các Transaction song song khác. Mặc định, các giao tác (transaction) được cô lập một cách hoàn toàn cho đến khi committed.

Ngoài các thành phần được định nghĩa bởi interface IDbTransaction, kiểu SqlTransaction định nghĩa một thành phần thêm vào được đặt tên là Save(), mà cho phép định nghĩa *save points*. Nội dung này cho phép rollback một giao tác (transaction) bị hỏng cho đến một điểm được đặt tên, thay vì roll back toàn bộ giao tác. Khi gọi Save() sử dụng một đối tượng SqlTransaction, chúng ta có thể chỉ rõ một chuỗi thân thiện. Khi gọi Rollback(), chúng ta có thể chỉ định biến này

như một đối số thực sự. Khi gọi Rollback() không có đối số, tất cả các sự thay đổi đang xem xét được trả lại từ đầu.

- **Thí dụ về Database Transaction**

Để minh họa việc sử dụng các giao tác (transaction) của ADO.NET, thêm một bảng mới có tên CreditRisks đến CSDL AutoLot, mà có các cột giống như bảng Customers (CustID [là primary key], FirstName, and LastName).

Hành động di chuyển khách hàng từ Customers sang bảng CreditRisks. Một cách cụ thể, chúng ta cần chắc chắn rằng chúng ta xóa một cách thành công các mẫu tin từ bảng Customers và thêm chúng đến bảng CreditRisks hoặc không có những thao tác nào xảy ra trên CSDL đó.

Để minh họa cách làm việc với giao tác (transaction) của ADO.NET, quan sát phương thức public với tên ProcessCreditRisk() bên dưới:

```
// A new member of the InventoryDAL class.  
public void ProcessCreditRisk(bool throwEx, int custID)  
{  
    // First, look up current name based on customer ID.  
    string fName = string.Empty;  
    string lName = string.Empty;  
    SqlCommand cmdSelect = new SqlCommand(  
        string.Format("Select * from Customers where CustID = {0}", custID), sqlCn);  
    using (SqlDataReader dr = cmdSelect.ExecuteReader())  
    {  
        while (dr.Read())  
        {  
            fName = (string)dr["FirstName"];  
            lName = (string)dr["LastName"];  
        }  
    }  
    // Create command objects that represent each step of the operation.  
    SqlCommand cmdRemove = new SqlCommand(  
        string.Format("Delete from Customers where CustID = {0}", custID), sqlCn);  
    SqlCommand cmdInsert = new SqlCommand(string.Format("Insert Into CreditRisks" +  
        "(CustID, FirstName, LastName) Values" +  
        "({0}, '{1}', '{2}')", custID, fName, lName), sqlCn);
```

```

// We will get this from the connection object.

SqlTransaction tx = null;
try
{
    tx = sqlCn.BeginTransaction();
    // Enlist the commands into this transaction.
    cmdInsert.Transaction = tx;
    cmdRemove.Transaction = tx;
    // Execute the commands.
    cmdInsert.ExecuteNonQuery();
    cmdRemove.ExecuteNonQuery();
    // Simulate error.
    if (throwEx)
    {
        throw new ApplicationException("Sorry! Database error! Tx failed...");
    }

    // Commit it!
    tx.Commit();
}
catch (Exception ex)
{
    // Any error will roll back transaction.
    tx.Rollback();
}
}

```

Ở đây, có một tham số bool đầu vào thể hiện rằng chúng ta sẽ ném ra một biệt lệ khi việc xử lý Customer bị lỗi.

Một khi chúng ta đã thu được first và last name của customer đầu tiên dựa trên tham số custID, chú ý rằng chúng ta đang sử dụng hai đối tượng SqlCommand để trình bày mỗi bước trong giao tác (transaction), và chúng ta lấy được một đối tượng SqlTransaction hợp lệ từ đối tượng kết nối qua BeginTransaction(). Tiếp theo, quan trọng nhất, chúng ta phải tranh thủ mỗi đối tượng command bởi việc gán thuộc tính Transaction đến đối tượng giao tác (transaction) mà chúng ta vừa lấy ra.

Sau khi gọi ExecuteNonQuery() trên mỗi lệnh, sẽ ném ra một biệt lệ nếu (và chỉ nếu) giá trị của tham số bool là đúng. Trong trường hợp này, tất cả các thao tác CSDL sắp xảy ra sẽ được roll back. Nếu chúng ta không ném ra một biệt lệ, các bước sẽ bị committed đến các bảng CSDL một khi chúng ta gọi Commit().

Tổng kết

Trong phần này, chúng ta xem xét Connected Layer và tìm hiểu vai trò của trình cung cấp dữ liệu, việc thực thi cụ thể của một vài lớp cơ sở ảo (trong System.Data.Common) để truy xuất đến CSDL và các kiểu namespace and Interface (trong System.Data namespace). Xây dựng cơ sở mã lệnh sử dụng ADO.NET Data Provider Factory Model.

Việc sử dụng các đối tượng Connected, các đối tượng giao tác (transaction), và các đối tượng DataReader của lớp kết nối, để select, update, insert, and delete các mẫu tin. Cũng như việc gọi các đối tượng command hỗ trợ một tập hợp tham số bên trong, dùng để thực thi các lệnh SQL hoặc Store Procedure.

7.5. DISCONNECTED LAYER

Khi sử dụng khía cạnh này của ADO.NET, chúng ta có thể mô hình dữ liệu của CSDL bên trong tầng gọi sử dụng nhiều thành phần của namespace System.Data (đặc biệt là **DataSet**, **DataTable**, **DataRow**, **DataColumn**, **DataView**, và **DataRelation**).

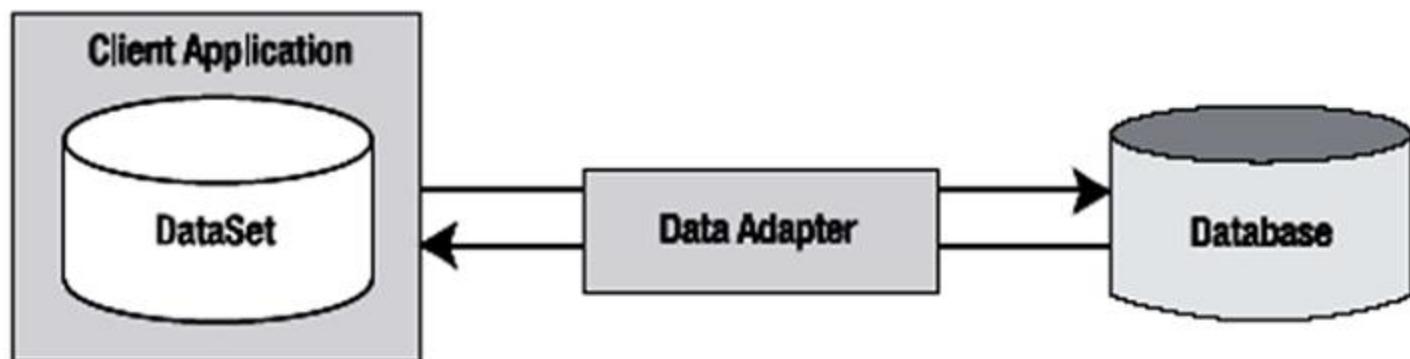
Khi sử dụng khía cạnh "ngắt kết nối" này của ADO.NET chúng ta sẽ thu được những đối tượng DataSet thường trú sử dụng đối tượng DataAdapter của trình cung cấp dữ liệu. Chức năng của đối tượng DataAdapter như một cầu nối giữa tầng client và một CSDL quan hệ.

7.5.1. Tìm hiểu cơ chế ngắt kết nối của ADO.NET

Sử dụng những kiểu ngắt kết nối chúng ta sẽ tác động đến một đối tượng đặc biệt là Data Adapter để load và update dữ liệu. Những đối tượng DataAdapter sử dụng các đối tượng DataSet để chuyển dữ liệu giữa đối tượng gọi và data source. Kiểu DataSet là một kho chứa các đối tượng DataTable, mỗi đối tượng đó chứa một tập hợp các đối tượng DataRow và DataColumn.

Đối tượng DataAdapter của trình cung cấp dữ liệu xử lý việc kết nối CSDL một cách tự động. Khi đối tượng gọi nhận được đối tượng DataSet, tầng gọi bị ngắt kết nối hoàn toàn từ CSDL và rời khỏi với một bản sao cục bộ của dữ liệu. Đối tượng gọi có thể insert, delete, hoặc update những hàng từ một DataTable đã cho, nhưng CSDL vật lý không

được cập nhật cho đến khi đối tượng gọi chuyển DataSet tới DataAdapter để cập nhật.



Hình 7.5: Các đối tượng data adapter chuyển các DataSet đến tầng client

7.5.2. Tìm hiểu vai trò của DataSet

DataSet là một thể hiện của dữ liệu quan hệ. Một DataSet là một lớp chứa ba collection bên trong (xem hình 1.5)



Hình 7.6: Mô hình của DataSet

Thuộc tính Tables của DataSet cho phép bạn truy xuất DataTableCollection mà chứa các DataTable riêng lẻ. Một DataSet có thể được sử dụng để thể hiện những mối quan hệ cha/con giữa các bảng của nó. Chẳng hạn, một quan hệ có thể được tạo ra giữa hai bảng để mô hình một sự ràng buộc khóa ngoại sử dụng kiểu DataRelation. Đối tượng này có thể sau đó được thêm vào DataRelationCollection thông qua thuộc tính Relations.

Thuộc tính ExtendedProperties cho phép truy xuất tới đối tượng Thuộc tínhCollection mà cho phép bạn kết hợp bất kỳ thông tin mở rộng nào tới DataSet như cặp name/value. Chẳng hạn, bạn có thể kết hợp tên công ty của bạn tới một DataSet.

Chú ý: Lớp DataTable cũng hỗ trợ những thuộc tính mở rộng qua thuộc tính ExtendedProperties.

- **Các thuộc tính chính của DataSet.**

Chúng ta hãy xem một số thành phần chính của DataSet. Bên ngoài những thuộc tính Tables, Relations, và ExtendedProperties. Bảng 7.8 mô tả một số thuộc tính bổ sung quan trọng.

Bảng 7.8: Các thuộc tính quan trọng của **DataSet**

Thuộc tính	Ý nghĩa
CaseSensitive	Cho biết có phải những chuỗi trong đối tượng DataTable là phân biệt hoa/thường hay không
DataSetName	Thể hiện tên một cách thân thiện của DataSet này
EnforceConstraints	Đưa ra hay thiết lập một giá trị cho biết có những quy tắc ràng buộc hay không
HasErrors	Đưa ra một giá trị cho biết có những lỗi trong bất kỳ hàng nào trong DataTables của DataSet hay không
RemotingFormat	Cho phép bạn định nghĩa cách DataSet xuất nội dung của nó (nhi phân hay XML)

- **Các phương thức chính của DataSet.**

DataSet cung cấp những phương thức mà cho phép bạn sao chép nội dung DataSet của bạn, điều hướng giữa những bảng bên trong, và thiết lập những điểm bắt đầu và kết thúc của một lô những sự cập nhật. Bảng 7.9 mô tả một số phương thức cốt lõi.

Bảng 7.9: Các phương thức quan trọng của **DataSet**

Methods	Ý nghĩa
AcceptChanges()	Xác nhận mọi sự thay đổi gần nhất đến DataSet từ khi nó được load hay AcceptChanges()
Clear()	Xóa hoàn toàn dữ liệu DataSet bằng việc loại bỏ mỗi hàng trong mỗi DataTable
Clone()	Sao chép cấu trúc của DataSet, bao gồm mọi DataTables, cũng như tất cả quan hệ và bất kỳ sự ràng buộc nào
Copy()	Sao chép cả cấu trúc lẫn dữ liệu cho DataSet
GetChanges()	Trả về một sự sao chép của DataSet chứa mọi sự thay đổi được tạo ra nó khi nó được load gần nhất hay từ khi AcceptChanges() được gọi
HasChanges()	Đưa ra một giá trị cho biết có phải DataSet có những sự thay đổi hay không
Merge()	Merge DataSet này với một DataSet xác định

ReadXml()	Cho phép bạn đọc dữ liệu XML vào trong DataSet
ReadXmlSchema()	
RejectChanges()	Quay trở lại mọi sự thay đổi gần nhất đối với DataSet từ khi nó được khởi tạo hay AcceptChanges()
WriteXml()	Cho phép bạn ghi nội dung của một DataSet ra XML
WriteXmlSchema()	

7.5.3. Làm việc với DataColumns

Kiểu **DataColumn** thể hiện một cột đơn bên trong một **DataTable**. DataColumn thể hiện sự thiết lập thông tin mô hình của bảng. Chẳng hạn, mô hình bảng **Inventory** của CSDL **AutoLot**, sẽ tạo ra bốn **DataColumns**, với mỗi cột (**CarID**, **Make**, **Color**, và **PetName**). Mỗi khi khởi tạo các đối tượng **DataColumn**, chúng được thêm vào trong collection **Columns** của kiểu **DataTable** (qua thuộc tính **Column**).

Bảng 7.10: Các thuộc tính của DataColumn

Properties	Ý nghĩa
AllowDBNull	Thuộc tính này được dùng nếu một hàng có thể có giá trị null trong cột này. Giá trị mặc định là true
AutoIncrement AutoIncrementSeed AutoIncrementStep	Những thuộc tính này được dùng để cấu hình việc tăng tự động (autoincrement behavior) cho một cột nhất định. Điều đó được dùng khi chúng ta muốn bảo đảm những giá trị là duy nhất trong một DataColumn đã. Theo mặc định, một DataColumn không hỗ trợ tăng tự động
Caption	Thuộc tính này dùng để thiết lập caption sẽ được hiển thị cho cột này
ColumnMapping	Thuộc tính này xác định cách một DataColumn được thể hiện khi một DataSet được lưu giữ như một tài liệu XML sử dụng phương thức DataSet.WriteXml()
ColumnName	Thuộc tính này thiết lập tên của cột trong collection Columns. Nếu không đặt ColumnName rõ ràng, thì giá trị mặc định là Column với hậu tố số (n +1) (ví dụ., Column1, Column2, Column3,...)
DataType	Thuộc tính này định nghĩa kiểu dữ liệu (Boolean, string, float,...) được lưu giữ trong cột
DefaultValue	Thuộc tính này thiết lập giá trị mặc định được

	gán đến cột khi chèn những hàng mới
Expression	Thuộc tính này thiết lập biểu thức được sử dụng để lọc các hàng, tính toán một giá trị của cột, hoặc tạo ra một cột kết hợp
Ordinal	Thuộc tính này đưa ra vị trí của cột trong collection Columns
ReadOnly	Thuộc tính này xác định nếu chỉ được sửa đổi một lần khi được thêm vào bảng. Mặc định là false
Table	Thuộc tính này đưa ra DataTable chứa DataColumn
Unique	Thuộc tính này thiết lập một giá trị cho biết có phải những giá trị trong mỗi hàng của cột là duy nhất hay không. Nếu một cột được gán một ràng buộc khóa chính, thuộc tính Unique nên được thiết lập là true

- **Xây dựng một DataColumn**

Tiếp tục với project SimpleDataSet cho cột CarID là khóa chính của bảng, cấu hình đối tượng DataColumn là read-only, duy nhất, và không null (sử dụng thuộc tính ReadOnly, Unique, và AllowDBNull).

```
private void MakeTable()
{
    // Create a DataTable.
    DataTable table = new DataTable("Product");

    // Create a DataColumn and set various properties.
    DataColumn column = new DataColumn();
    column.DataType = System.Type.GetType("System.Decimal");
    column.AllowDBNull = false;
    column.Caption = "Price";
    column.ColumnName = "Price";
    column.DefaultValue = 25;

    // Add the column to the table.
    table.Columns.Add(column);

    // Add 7 rows and set values.
    DataRow row;
    for(int i = 0; i < 7; i++)
    {
        row = table.NewRow();
        row["Price"] = i * 100;
        table.Rows.Add(row);
    }
}
```

```

row = table.NewRow();
row["Price"] = i + 1;
// Be sure to add the new row to the
// DataRowCollection.
table.Rows.Add(row);
}
}

```

• Cho phép các field tăng tự động

Một khía cạnh của DataColumn là chúng ta có thể cấu hình tăng tự động (*autoincrement*). Một cột autoincrementing được sử dụng để bảo đảm rằng khi một hàng mới được thêm vào một bảng đã cho, giá trị của cột này được gán một cách tự động trên cơ sở tăng dần.

Để làm việc này sử dụng những thuộc tính AutoIncrement, AutoIncrementSeed và AutoIncrementStep. Cho code sau cập nhật cấu trúc của carIDColumn DataColumn

```

private void AddAutoIncrementColumn()*
{
    DataColumn column = new DataColumn();
    column.DataType = System.Type.GetType("System.Int32");
    column.AutoIncrement = true;
    column.AutoIncrementSeed = 700;
    column.AutoIncrementStep = 7;

    // Add the column to a new DataTable.
    DataTable table = new DataTable("table");
    table.Columns.Add(column);
}

```

Ở đây, đối tượng carIDColumn đã được cấu hình để bảo đảm rằng những dòng có giá trị tăng là 1. Vì giá trị khởi đầu đã được thiết lập là 0, nên cột này sẽ được đánh số 0, 1, 2, 3,...

7.5.4. Làm việc với DataRows

Một collection của các đối tượng DataColumn thể hiện mô hình của một DataTable. Còn một collection của các kiểu DataRow đại diện cho dữ liệu thực tế trong bảng. Sử dụng những thành phần của lớp DataRow, chúng ta có thể insert, remove, và thao tác những giá trị trong bảng. Bảng 7.11 trình bày một vài thành phần của kiểu DataRow.

Bảng 7.11: Các thành phần chính của kiểu DataRow

Members	Ý nghĩa
HasErrors GetColumnsInError() GetColumnError() ClearErrors() RowError	Thuộc tính HasErrors trả về một giá trị Boolean cho biết nếu có lỗi. Như vậy, phương thức GetColumnsInError() được sử dụng để thu được những thành phần lỗi và GetColumnError được sử dụng để thu được sự mô tả lỗi, trong khi phương thức ClearErrors() loại bỏ lỗi cho hàng. Thuộc tính RowError cho phép cấu hình một sự mô tả lỗi cho một hàng
ItemArray	Thuộc tính này thiết lập tất cả các giá trị cho hàng này sử dụng một mảng các đối tượng
RowState	Thuộc tính này được dùng để xác định " trạng thái " hiện thời của DataRow
GetChildRows	Trả về những dòng con của DataRow
GetParentRows	Trả về những dòng cha của DataRow
Table	Thuộc tính này được dùng để thu được một tham chiếu tới DataTable chứa DataRow
AcceptChanges() RejectChanges()	Những phương thức này xác nhận hay loại bỏ mọi sự thay đổi đến hàng từ lần cuối cùng AcceptChanges() được gọi
BeginEdit() EndEdit() CancelEdit()	Những phương thức này bắt đầu, kết thúc, hoặc hủy bỏ một thao tác hiệu chỉnh trên một đối tượng DataRow
Delete()	Phương thức này đánh dấu hàng sẽ được remove khi phương thức AcceptChanges() được gọi
IsNull()	Phương thức này đưa ra một giá trị cho biết có phải cột được xác định chứa một giá trị null hay không

7.5.5. Làm việc với DataTables

Kiểu DataTable định nghĩa một số lớn các thành phần, nhiều thành phần của nó giống về tên và chức năng của DataSet. Bảng 7.12 mô tả một số thành phần cốt lõi của kiểu DataTable ngoài Rows và Columns.

Bảng 7.12: Các thành phần chính của kiểu DataTable

Member	Ý nghĩa
CaseSensitive	Cho biết chuỗi bên trong bảng có phân biệt hoa/thường hay không. Giá trị mặc định là false
ChildRelations	Trả về tập hợp của những quan hệ con cho DataTable này (nếu có)
Constraints	Đưa ra tập hợp của những ràng buộc của bảng
Copy()	Phương thức sao chép mô hình và dữ liệu của một DataTable đã cho vào trong một thể hiện mới
DataSet	Đưa ra DataSet mà chứa bảng này (nếu có)
DefaultView	Đưa ra một view mặc định của bảng
MinimumCapacity	Đưa ra hay thiết lập số lượng hàng ban đầu trong bảng (mặc định là 5)
ParentRelations	Đưa ra tập hợp của những quan hệ cha cho DataTable này
PrimaryKey	Thiết lập những cột mà có chức năng khóa chính cho bảng dữ liệu
RemotingFormat	Cho phép bạn định nghĩa cách DataSet xuất nội dung của nó (nhị phân hay XML)
TableName	Thiết lập tên của bảng

7.5.6. Làm việc với DataRelation

Dataset lưu trữ những mối quan hệ trong đối tượng **DataRelationCollection** của nó. Một mối quan hệ được biểu diễn bởi một đối tượng **DataRelation**, kết hợp những dòng trong DataTable này với những DataTable khác

Mối quan hệ (Relationship) cho phép điều hướng từ một bảng đến một bảng khác bên trong Dataset. Những thành phần chủ yếu của DataRelation là tên của quan hệ, tên của những bảng quan hệ và những cột quan hệ trong mỗi bảng. Mỗi quan hệ có thể được xây dựng với nhiều hơn một cột cho mỗi bảng bằng cách chỉ ra một mảng các đối tượng của DataColumn như là những cột chính. Khi một quan hệ được thêm vào DataRelationCollection, nó có thể thêm vào một UniqueKeyConstraint và một ForeignKeyConstraint để đảm bảo tính toàn vẹn dữ liệu khi có những thay đổi giá trị trên những cột quan hệ.

Đối tượng DataRelation

Một DataRelation được sử dụng để quan hệ giữa hai đối tượng DataTable với nhau thông qua đối tượng DataColumn. Ví dụ trong mối quan hệ khách hàng và hóa đơn, bảng khách hàng là cha và bảng hóa đơn là con trong mối quan hệ.

Mỗi quan hệ được tạo ra giữa những cột tương thích trong bảng cha và bảng con. Đó là kiểu dữ liệu giữa hai cột là duy nhất.

Những mối quan hệ cũng có thể xếp tầng những thay đổi khác nhau từ những dòng dữ liệu cha đến những dòng dữ liệu con. Để điều khiển cách giá trị bị thay đổi trong những dòng dữ liệu con, ta thêm vào một ForeignKeyConstraint vào ConstraintCollection của đối tượng DataTable. ConstraintCollection hoạt động khi dữ liệu trên bảng cha bị xóa hay được cập nhật.

Khi một DataRelation được tạo, đầu tiên nó kiểm tra xem mối quan hệ được tạo hay không. Sau khi nó được thêm vào DataRelationCollection. Mỗi quan hệ được duy trì bởi việc không cho phép bất kỳ sự thay đổi nào mà vi phạm ràng buộc này.

Các đối tượng DataRelation được chứa trong DataRelationCollection. Bạn có thể truy cập DataRelationCollection thông qua thuộc tính Collections của DataSet và thuộc tính ChildRelation and ParentRelation của DataTable

Bảng 7.13: Mô tả một số thành phần cốt lõi của đối tượng DataRelation

Thuộc tính	Diễn tả
ChildColumns	Trả về các đối tượng DataColumns con của quan hệ này
ChildKeyConstraint	Trả về ForeignKeyConstraint cho quan hệ
ChildTable	Trả về bảng con của quan hệ
DataSet	Trả về Dataset chứa DataRelation này
ExtendedProperties	Trả về tập hợp mà lưu trữ những thuộc tính tùy chọn
Nested	Trả về hoặc thiết lập một giá trị cho biết các đối tượng DataRelation là ẩn
ParentColumns	Trả về một mảng đối tượng DataColumn mà là những cột cha của DataRelation này

ParentKeyConstraint	Trả về UniqueConstraint mà chắc rằng những giá trị trong cột cha của DataRelation là duy nhất
RelationName	Trả về hoặc thiết lập tên dùng để truy xuất một DataRelation từ DataRelationCollection

Thí dụ

Tạo trang TableRelationships.aspx cho phép thiết lập các mối liên kết.

- Mỗi liên kết của hai bảng Authors (Tác giả) và TitleAuthor (Tựa sách của Tác giả) theo vùng khóa liên kết là au_id (Mã Tác giả)

```
DataRelation Authors_TitleAuthor = new DataRelation("Authors_TitleAuthor",
    dsPubs.Tables["Authors"].Columns["au_id"],
    dsPubs.Tables["TitleAuthor"].Columns["au_id"]);
```

- Mỗi liên kết của hai bảng Titles (Tựa sách) và TitleAuthor (Tựa sách của Tác giả) theo vùng khóa liên kết là title_id (Mã sách)

```
DataRelation Titles_TitleAuthor = new DataRelation("Titles_TitleAuthor",
    dsPubs.Tables["Titles"].Columns["title_id"],
    dsPubs.Tables["TitleAuthor"].Columns["title_id"]);
```

Nội dung thiết kế trang TableRelationships.aspx có điều khiển nhãn chứa danh sách từng tác giả và các tựa sách của tác giả nếu có.

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="TableRelationships.aspx.cs" Inherits="TableRelationships" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="lblList" runat="server" EnableViewState="False"></asp:Label>
        </div>
    </form>
</body>
```

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

Phần mã lệnh:

```
public partial class TableRelationships: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!this.IsPostBack){
            CreateList();
        }
    }

    private string connectionString =
        WebConfigurationManager.ConnectionStrings["Pubs"].ConnectionString;

    private void CreateList() {
        // Define ADO.NET objects.

        string selectSQL = "SELECT au_lname, au_fname, au_id FROM Authors";
        SqlConnection con = new SqlConnection(connectionString);
        SqlCommand cmd = new SqlCommand(selectSQL, con);
        SqlDataAdapter adapter = new SqlDataAdapter(cmd);

        DataSet dsPubs = new DataSet();
        try
        {
            con.Open();
            adapter.Fill(dsPubs, "Authors");
            // This command is still linked to the data adapter.
            cmd.CommandText = "SELECT au_id, title_id FROM TitleAuthor";
            adapter.Fill(dsPubs, "TitleAuthor");
            // This command is still linked to the data adapter.
            cmd.CommandText = "SELECT title_id, title FROM Titles";
            adapter.Fill(dsPubs, "Titles");
        }
        catch (Exception err)
```

```

{
    lblList.Text = "Error reading list of names. ";
    lblList.Text += err.Message;
}

finally
{
    con.Close();
}

DataRelation Titles_TitleAuthor = new DataRelation("Titles_TitleAuthor",
    dsPubs.Tables["Titles"].Columns["title_id"],
    dsPubs.Tables["TitleAuthor"].Columns["title_id"]);

DataRelation Authors_TitleAuthor = new DataRelation("Authors_TitleAuthor",
    dsPubs.Tables["Authors"].Columns["au_id"],
    dsPubs.Tables["TitleAuthor"].Columns["au_id"]);

dsPubs.Relations.Add(Titles_TitleAuthor);
dsPubs.Relations.Add(Authors_TitleAuthor);

foreach (DataRow rowAuthor in dsPubs.Tables["Authors"].Rows)
{
    lblList.Text += "<br /><b>" + rowAuthor["au_fname"];
    lblList.Text += " " + rowAuthor["au_lname"] + "</b><br />";
    foreach (DataRow rowTitleAuthor in
        rowAuthor.GetChildRows(Authors_TitleAuthor))
    {
        foreach (DataRow rowTitle in
            rowTitleAuthor.GetParentRows(Titles_TitleAuthor))
        {
            lblList.Text += "&nbsp;&nbsp;";
            lblList.Text += rowTitle["title"] + "<br />";
        }
    }
}

```

```
    }
}
}
}
```

Kết quả thực thi liệt kê danh sách các tác giả và các tựa sách của tác giả có trong CSDL:

Abraham Bennet

The Busy Executive's Database Guide

Reginald Blotchet-Halls

Fifty Years in Buckingham Palace Kitchens

Cheryl Carson

But Is It User Friendly?

Michel DeFrance

The Gourmet Microwave

Innes del Castillo

Silicon Valley Gastronomic Treats

Ann Dull

Secrets of Silicon Valley

Marjorie Green

The Busy Executive's Database Guide

You Can Combat Computer Stress!

Morningstar Greene

Burt Gringlesby

Sushi, Anyone?

Sheryl Hunter

Secrets of Silicon Valley

Livia Karsen

Computer Phobic AND Non-Phobic Individuals: Behavior Variations

Charlene Locksley

Net Etiquette

Emotional Security: A New Algorithm

Stearns MacFeather

Cooking with Computers: Surreptitious Balance Sheets

Computer Phobic AND Non-Phobic Individuals: Behavior Variations

7.5.7. Xuất các đối tượng DataTable/ DataSet ra XML

Cả DataSet lẫn DataTables hỗ trợ các phương thức WriteXml() Và ReadXml(). WriteXml() cho phép bạn ghi nội dung của đối tượng ra một file XML. ReadXml() Cho phép bạn đọc nội dung một DataSet (hay DataTable) từ một tài liệu XML đã cho. Ngoài ra, cả DataSets và DataTables hỗ trợ WriteXmlSchema() và ReadXmlSchema() để lưu trữ hoặc load một file *.xsd.

```
static void DataSetAsXml(DataSet carsInventoryDS)
```

```
{
```

```
    // Save this DataSet as XML.
```

```
    carsInventoryDS.WriteXml("carsDataSet.xml");
```

```
carsInventoryDS.WriteXmlSchema("carsDataSet.xsd");  
// Clear out DataSet.  
carsInventoryDS.Clear();  
// Load DataSet from XML file.  
carsInventoryDS.ReadXml("carsDataSet.xml");  
}
```

Nếu mở file carsDataSet.xml (nằm trong folder bin\Debug của project), chúng ta sẽ nhìn thấy mỗi cột trong bảng đã được mã hóa như một thẻ của XML:

```
<?xml version="1.0" standalone="yes"?>  
<Car_x0020_Inventory>  
  <Inventory>  
    <CarID>0</CarID>  
    <Make>BMW</Make>  
    <Color>Black</Color>  
    <PetName>Hamlet</PetName>  
  </Inventory>  
  <Inventory>  
    <CarID>1</CarID>  
    <Make>Saab</Make>  
    <Color>Red</Color>  
    <PetName>Sea Breeze</PetName>  
  </Inventory>  
</Car_x0020_Inventory>
```

7.6. ĐỐI TƯỢNG DỮ LIỆU SQLDATASOURCE

SqlDataSource Control cho phép kết nối đến cơ sở dữ liệu quan hệ MS SQL Server. SQLDataSource tự động phát sinh các lệnh: Insert, Update, Delete, Select và lọc dữ liệu, sắp xếp, phân trang,...

Chúng ta sẽ sử dụng control này với các Data Controls trong các chương sau đây.

Thí dụ

Tạo SqlDataSource sử dụng Wizard.

Thiết kế giao diện trang gồm đối tượng dữ liệu SqlDataSource và GridView (trong thẻ Data của hộp công cụ).

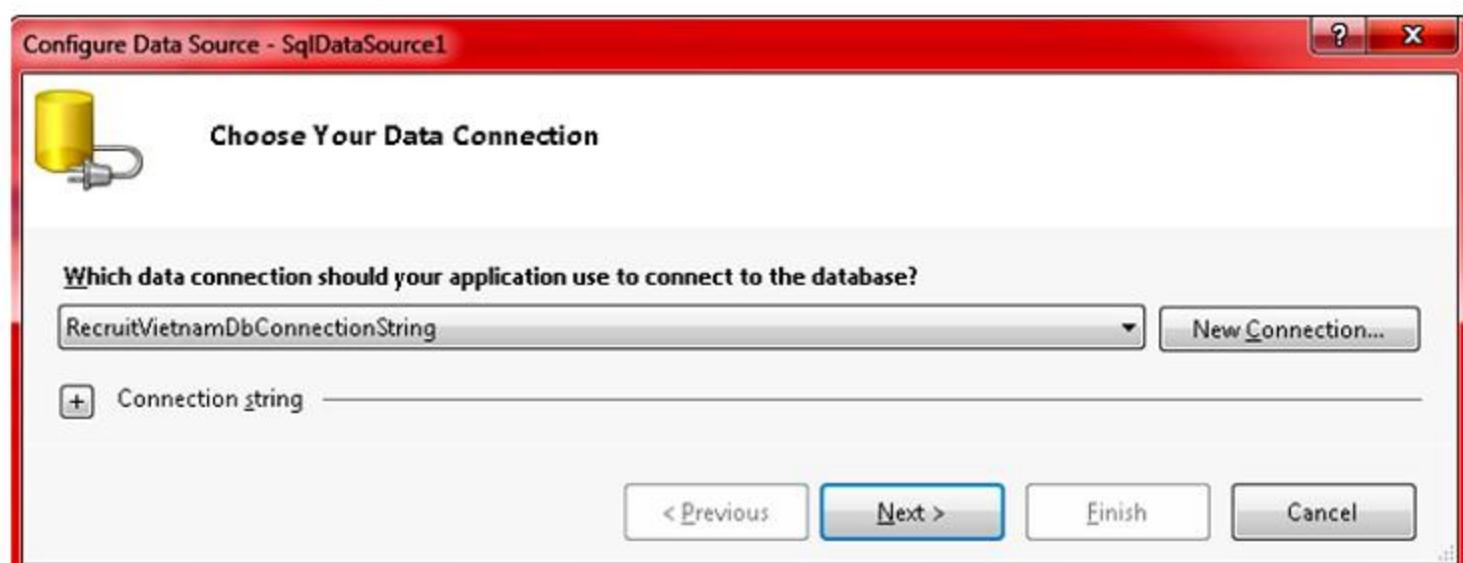
UserID	Email	FirstName	LastName
0	abc	abc	abc
1	abc	abc	abc
2	abc	abc	abc
3	abc	abc	abc
4	abc	abc	abc

Cáu hình làm việc cho SqlDataSource

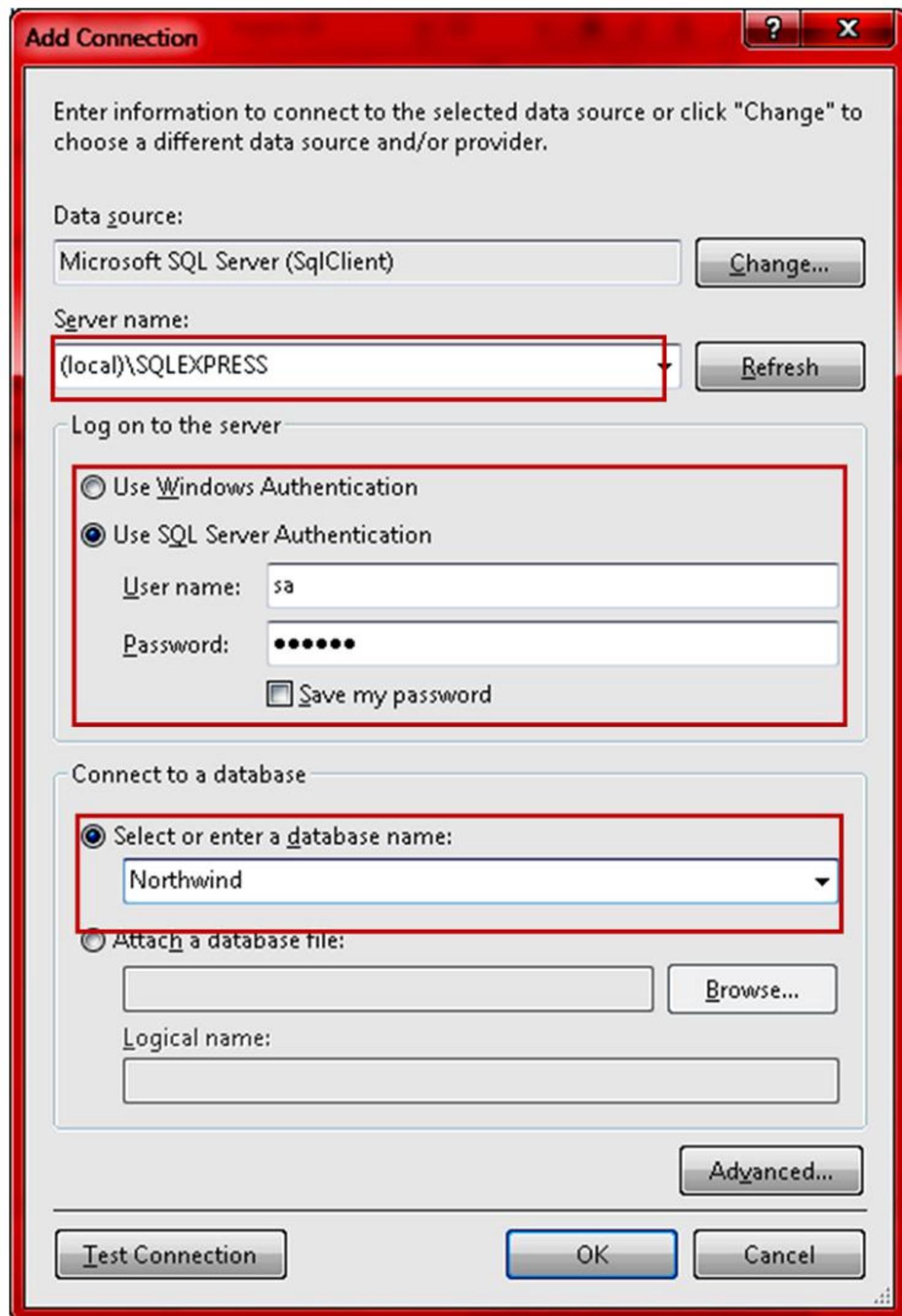
SqlDataSource Tasks

- Configure Data Source...
- Refresh Schema

Xuất hiện hộp thoại Configure Data Source, click nút New Connection... để tạo kết nối mới:

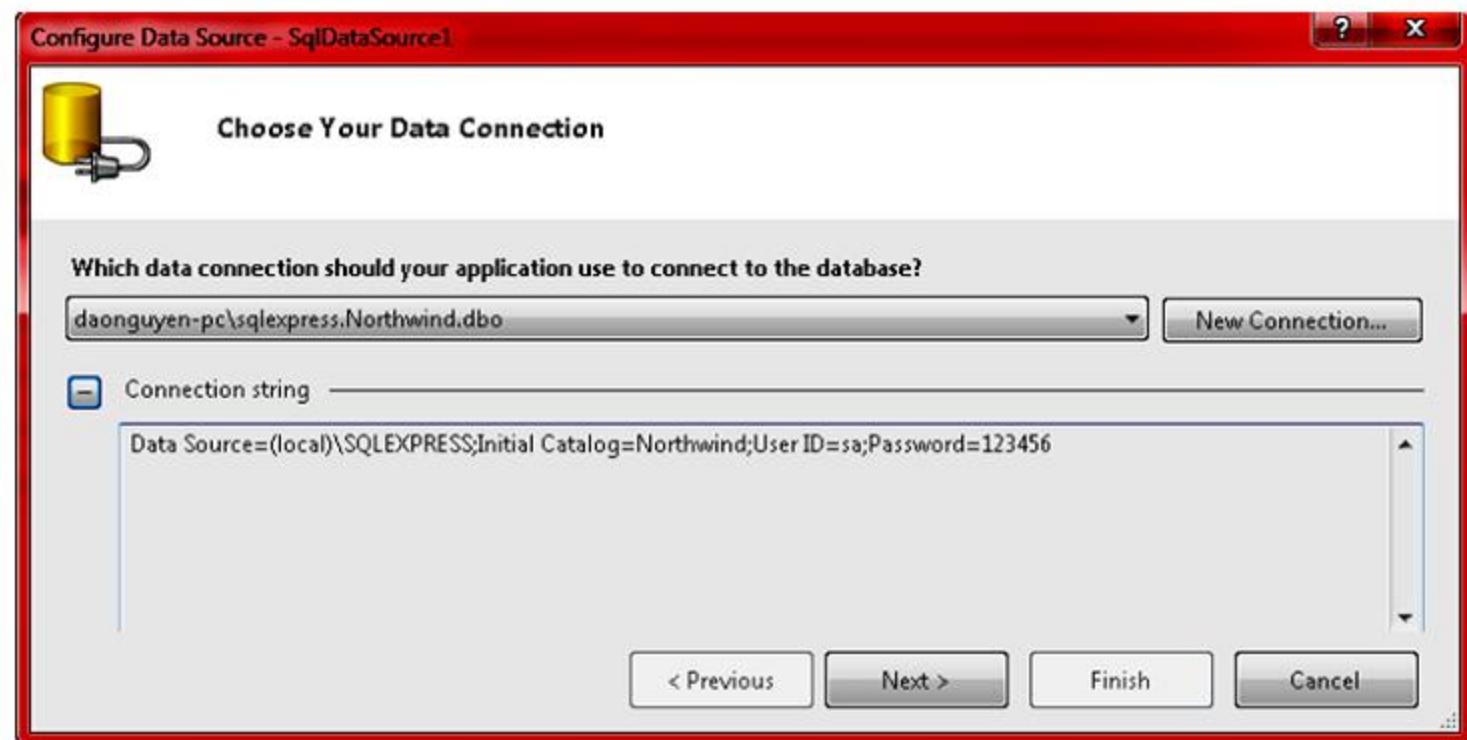


Xuất hiện hộp thoại



Nhập tên SQL Server trong khung Server Name. Chọn đăng nhập vào SQL theo chứng thực Windows Authentication hoặc SQL Authentication (nhập User ID và Password). Nếu kết nối SQL Server được, tiếp đến sẽ chọn CSLD trong ComboBox [Select or Enter database name ;]

Click nút OK để hoàn tất bước tạo kết nối. Màn hình quay lại hộp thoại Configure Data Source. Click [+Connection String] sẽ thấy nội dung chuỗi kết nối vừa tạo như hình dưới đây:



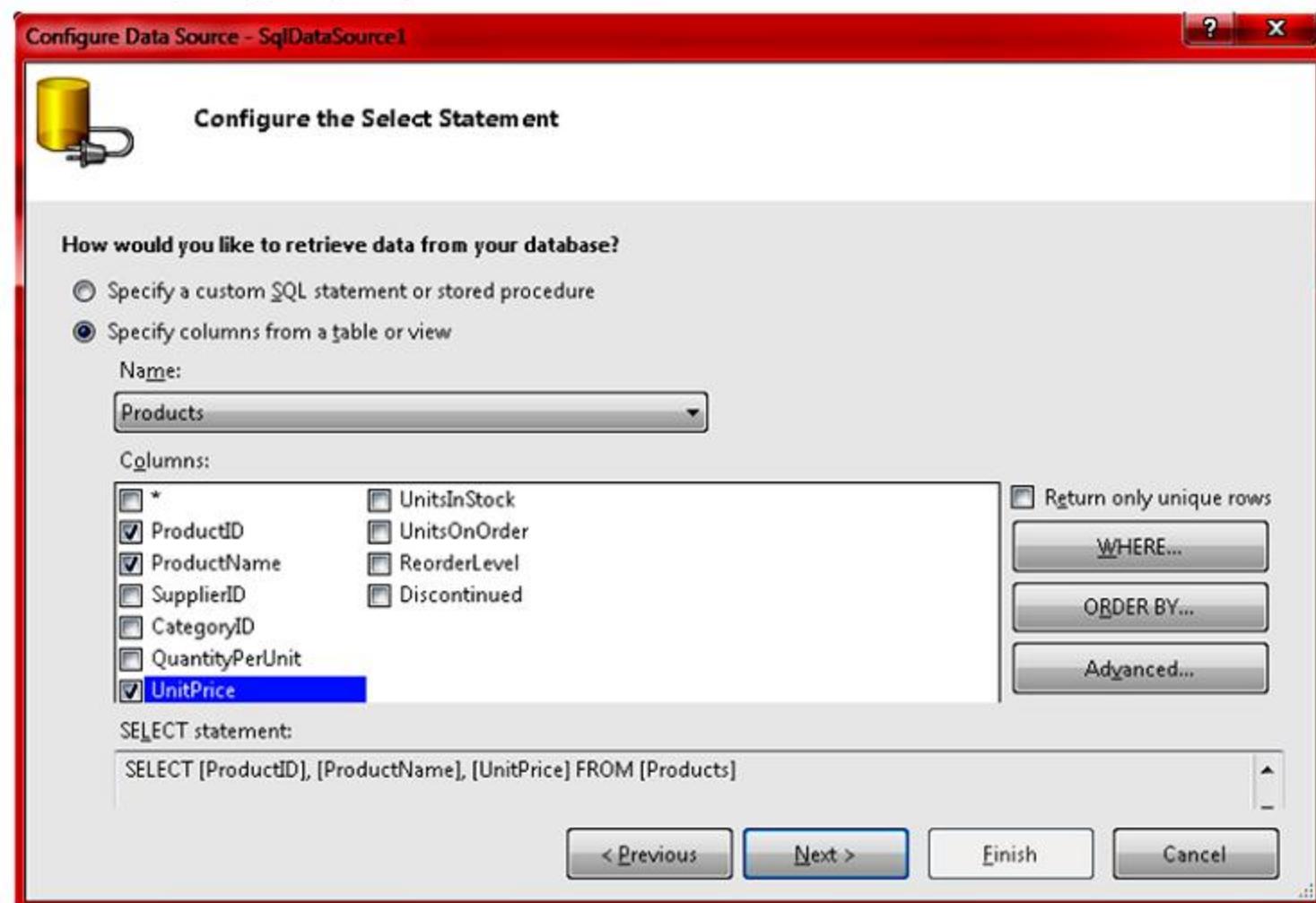
Click nút [Next], chuyển sang hộp thoại kế tiếp cho phép chọn hay nhập lại tên chuỗi kết nối. Chuỗi ConnectionString này sẽ được ghi vào tập tin Web.config.

```
<connectionStrings>
<add name="NorthwindConnectionString" connectionString="Data
Source=(local)\SQLEXPRESS;Initial Catalog=Northwind;User
ID=sa;Password=123456"
providerName="System.Data.SqlClient" />
</connectionStrings>
```



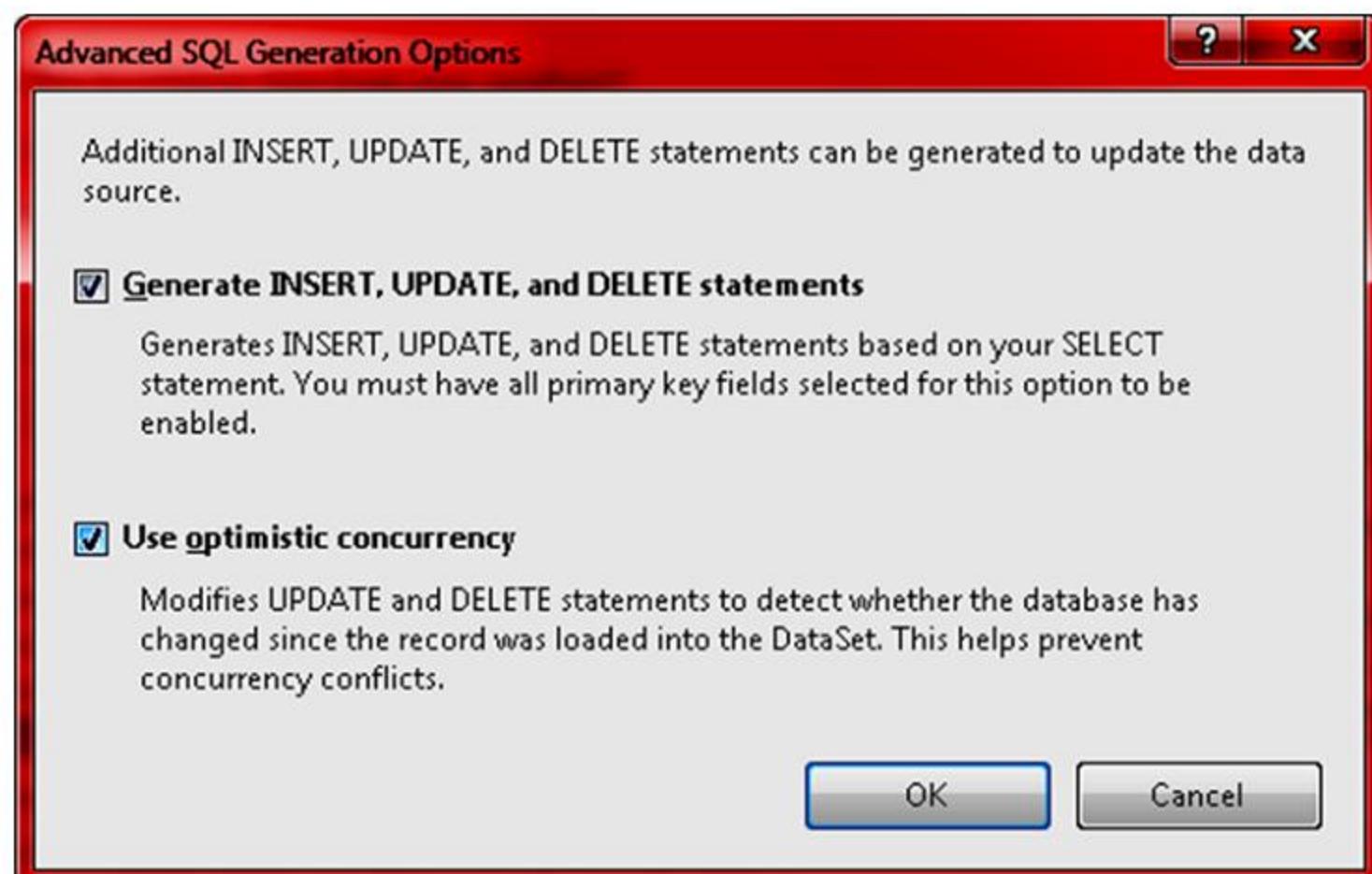
Click nút [Next] chuyển sang màn hình chọn bảng dữ liệu hoặc thủ tục lưu trữ (store procedure). Ta chọn bảng **Products**, và chọn ba vùng

ProductId, ProductName, UnitPrice cho SqlDataSource. Câu lệnh truy vấn tương ứng được tạo ra:

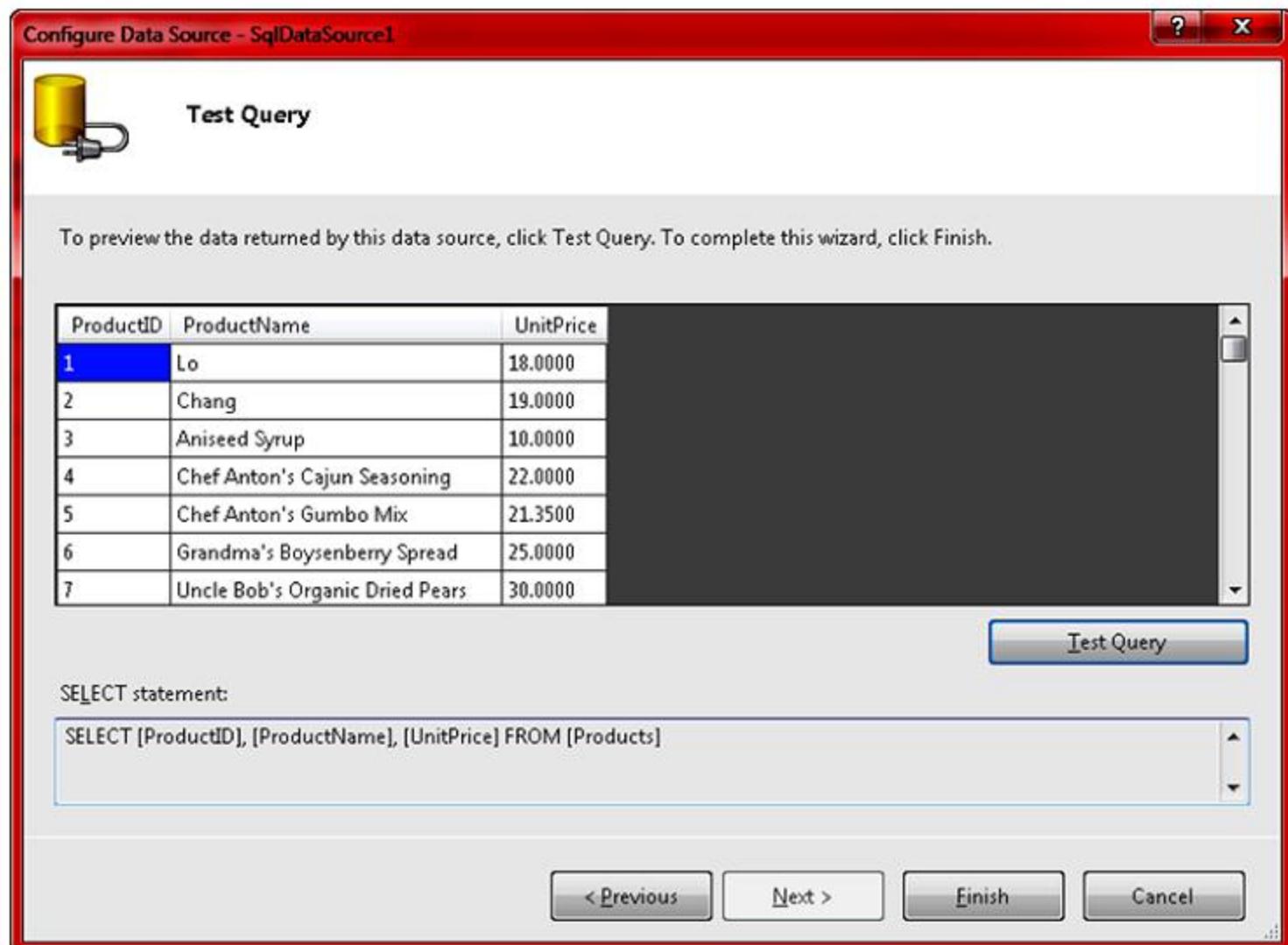


Click nút **[WHERE...]** để tạo bộ lọc cho câu truy vấn **SELECT**.

Click nút **[Advanced..]** để tạo thêm các câu lệnh truy vấn **Insert**, **Delete**, **Update** cho cấu trúc SqlDataSource.



Click nút [Next] chuyển sang phần Test Query. Click nút [Test Query] sẽ có kết quả truy vấn hiển thị trong khung hộp thoại tương ứng với lệnh Select.



Click nút [Finish] để kết thúc. Phần mã lệnh thiết kế SqlDataSource có nội dung sau:

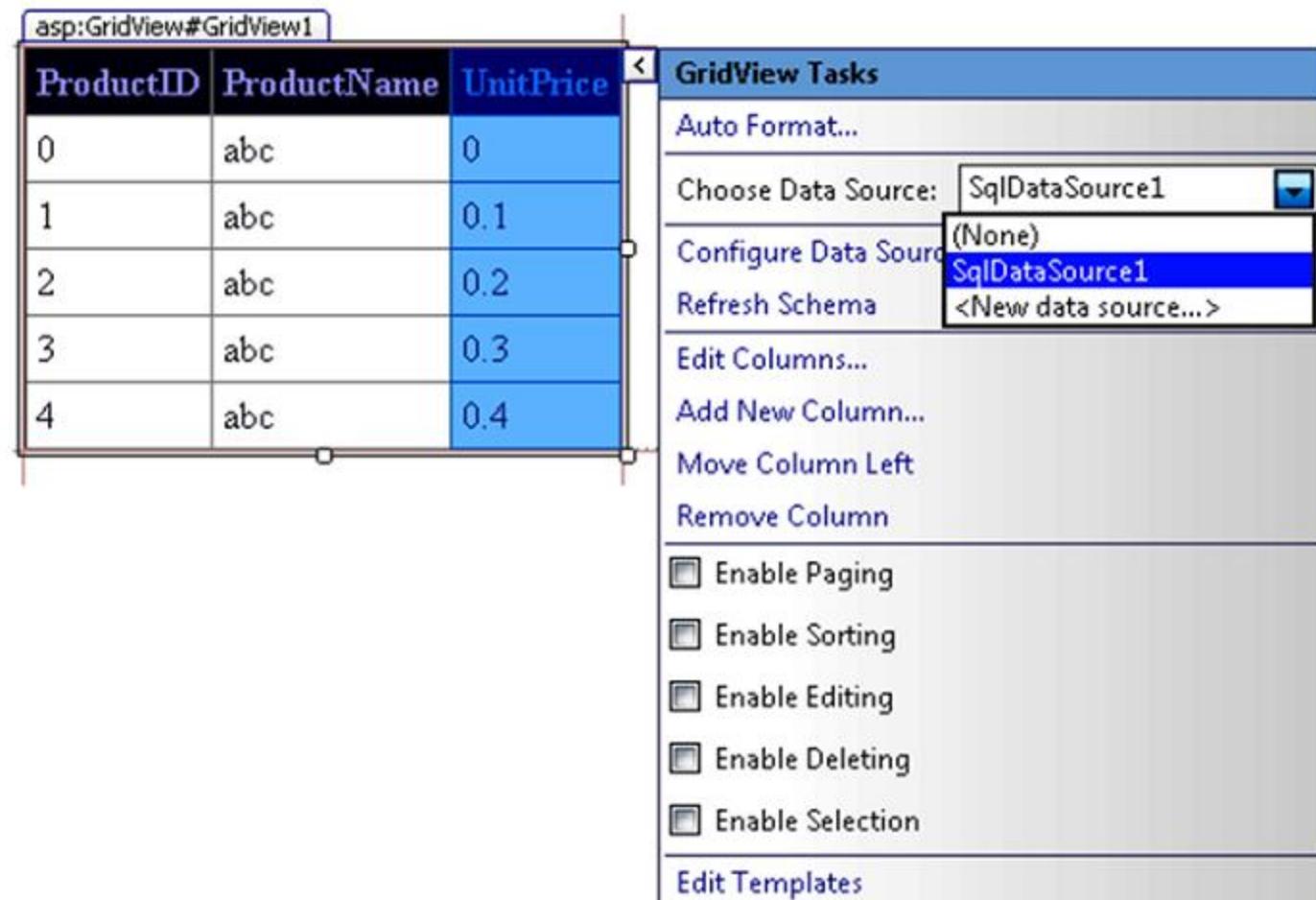
```
<asp:SqlDataSource ID="SqlDataSource1" runat="server">
    ConnectionString="<%$ ConnectionStrings:NorthwindConnectionString %>"
    OldValuesParameterFormatString="original_{0}"
    SelectCommand="SELECT [ProductID], [ProductName], [UnitPrice] FROM [Products]"
    ConflictDetection="CompareAllValues"
    DeleteCommand="DELETE FROM [Products] WHERE [ProductID] =
        @original_ProductID AND [ProductName] = @original_ProductName AND
        ([UnitPrice] = @original_UnitPrice) OR ([UnitPrice] IS NULL AND
        @original_UnitPrice IS NULL)"
    InsertCommand="INSERT INTO [Products] ([ProductName], [UnitPrice])
        VALUES (@ProductName, @UnitPrice)"
    UpdateCommand="UPDATE [Products] SET [ProductName] =
        @ProductName, [UnitPrice] = @UnitPrice WHERE [ProductID] =
        @original_ProductID AND [ProductName] = @original_ProductName AND
        ([UnitPrice] = @original_UnitPrice) OR ([UnitPrice] IS NULL AND
        @original_UnitPrice IS NULL))">
```

```

<DeleteParameters>
    <asp:Parameter Name="original_ProductID" Type="Int32" />
    <asp:Parameter Name="original_ProductName" Type="String" />
    <asp:Parameter Name="original_UnitPrice" Type="Decimal" />
</DeleteParameters>
<UpdateParameters>
    <asp:Parameter Name="ProductName" Type="String" />
    <asp:Parameter Name="UnitPrice" Type="Decimal" />
    <asp:Parameter Name="original_ProductID" Type="Int32" />
    <asp:Parameter Name="original_ProductName" Type="String" />
    <asp:Parameter Name="original_UnitPrice" Type="Decimal" />
</UpdateParameters>
<InsertParameters>
    <asp:Parameter Name="ProductName" Type="String" />
    <asp:Parameter Name="UnitPrice" Type="Decimal" />
</InsertParameters>
</asp:SqlDataSource>

```

Sau đó thực hiện việc gắn thuộc tính DataSoure của GridView với đối tượng SqlDataSource như hình sau. Các ô kiểm tra [X] Enable Paging/Sorting/Editing/Deleting/Selection cho phép GridView thực hiện phân trang/sắp xếp/cập nhật/xóa/chọn dữ liệu trong bảng **Products**. Nội dung này sẽ được đề cập rõ trong Chương 9.



Phần mã lệnh thực thi trang:

```
public partial class SqlDataSourcePage: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        this.SqlDataSource1.ConnectionString =
            WebConfigurationManager.ConnectionStrings[
                "NorthwindConnectionString"].ConnectionString;
    }
}
```

Kết quả thực thi:

ProductID	ProductName	UnitPrice
1	Lo	18.0000
2	Chang	19.0000
3	Aniseed Syrup	10.0000
4	Chef Anton's Cajun Seasoning	22.0000
5	Chef Anton's Gumbo Mix	21.3500
6	Grandma's Boysenberry Spread	25.0000
7	Uncle Bob's Organic Dried Pears	30.0000

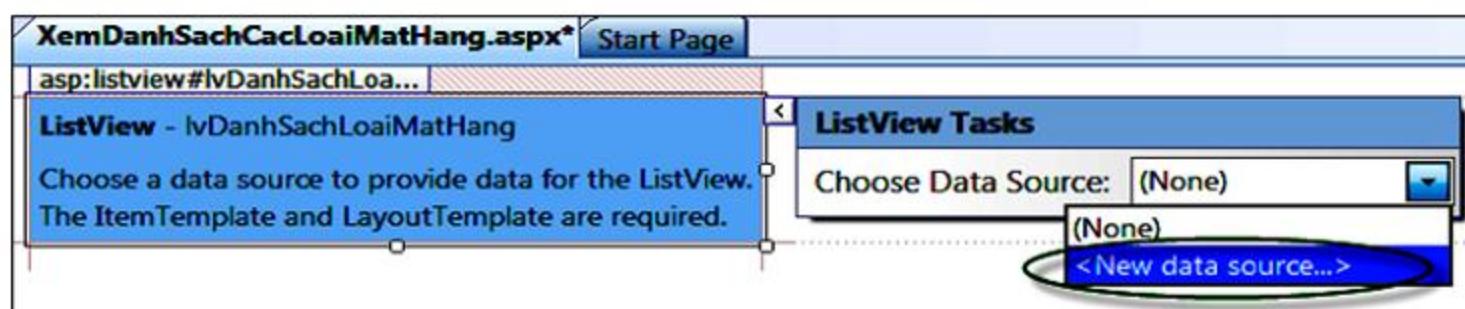
Tổng kết

Chương này trình bày chi tiết lớp ngắt kết nối của ADO.NET. Trung tâm của lớp ngắt kết nối là DataSet. Kiểu này thể hiện những bảng, những quan hệ, những ràng buộc, nó có thể được xem như một Database trong bộ nhớ.

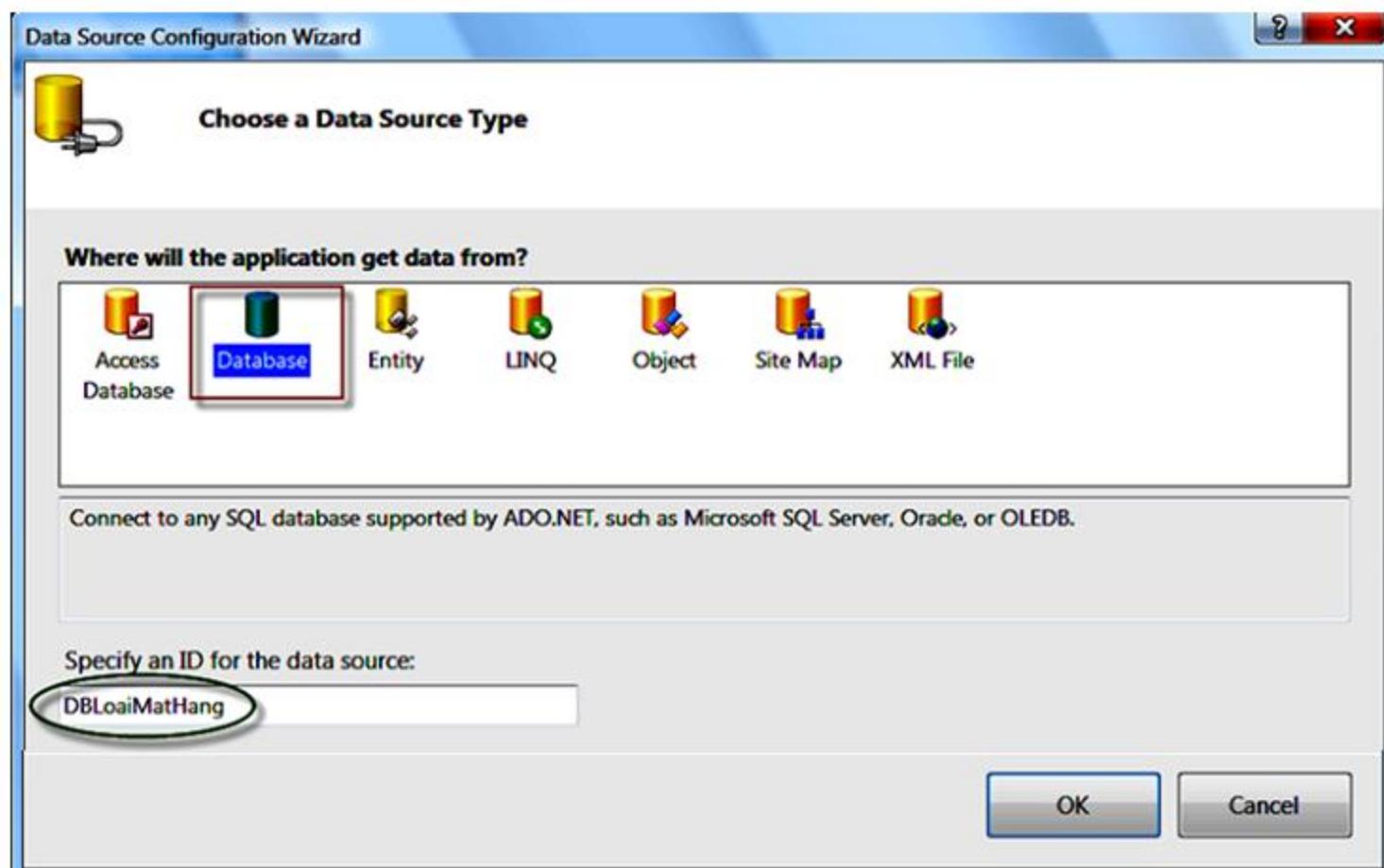
BÀI TẬP CHƯƠNG 7

Bài 1: Thực hành lại các thí dụ trong bài học để ôn lại nội dung vừa học.

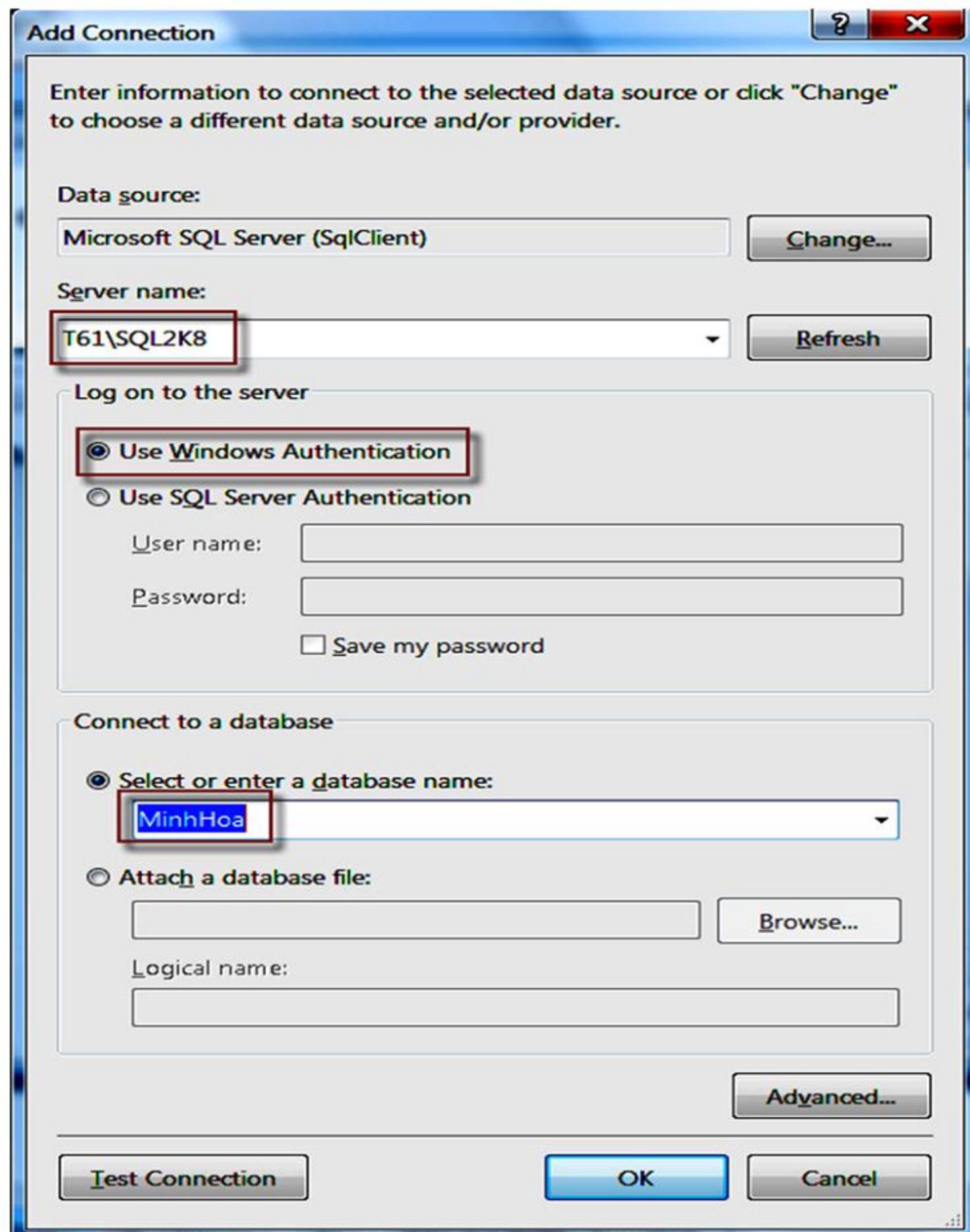
Bài 2: Tạo một trang ASP.NET tên XemDanhSachLoaiMatHang.aspx, gồm một ListView control đặt tên lvDanhSachLoaiMatHang.



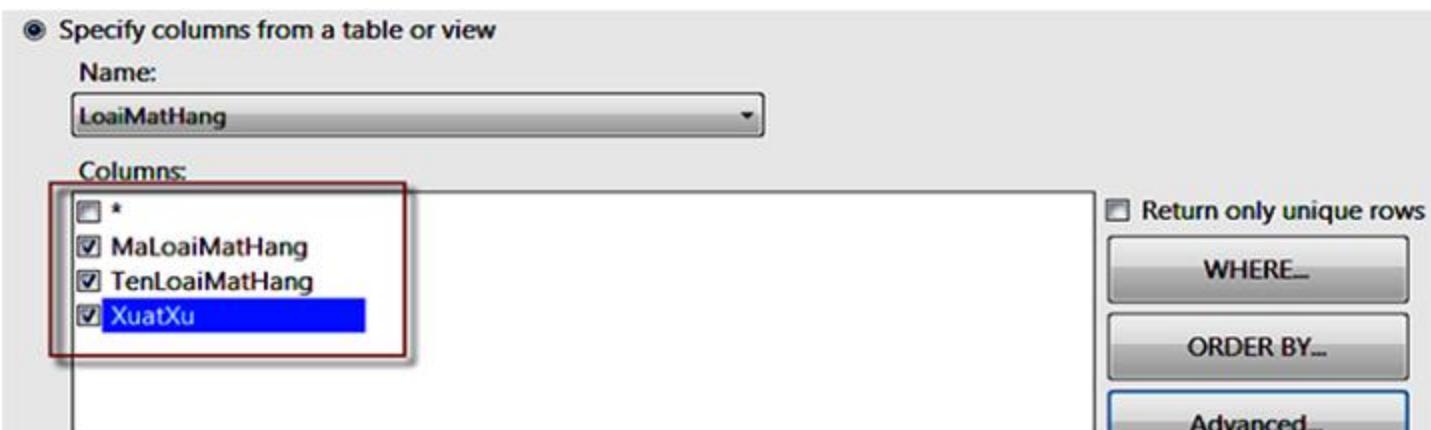
Bước 1. Nhấp vào biểu tượng trên ListView control chọn *New data source* cửa sổ *DataSource Configuration Wizard* xuất hiện. Sau đó chọn *Database* và đặt tên *ID* là *DBLoaiMatHang* như hình bên dưới, nhấp OK.



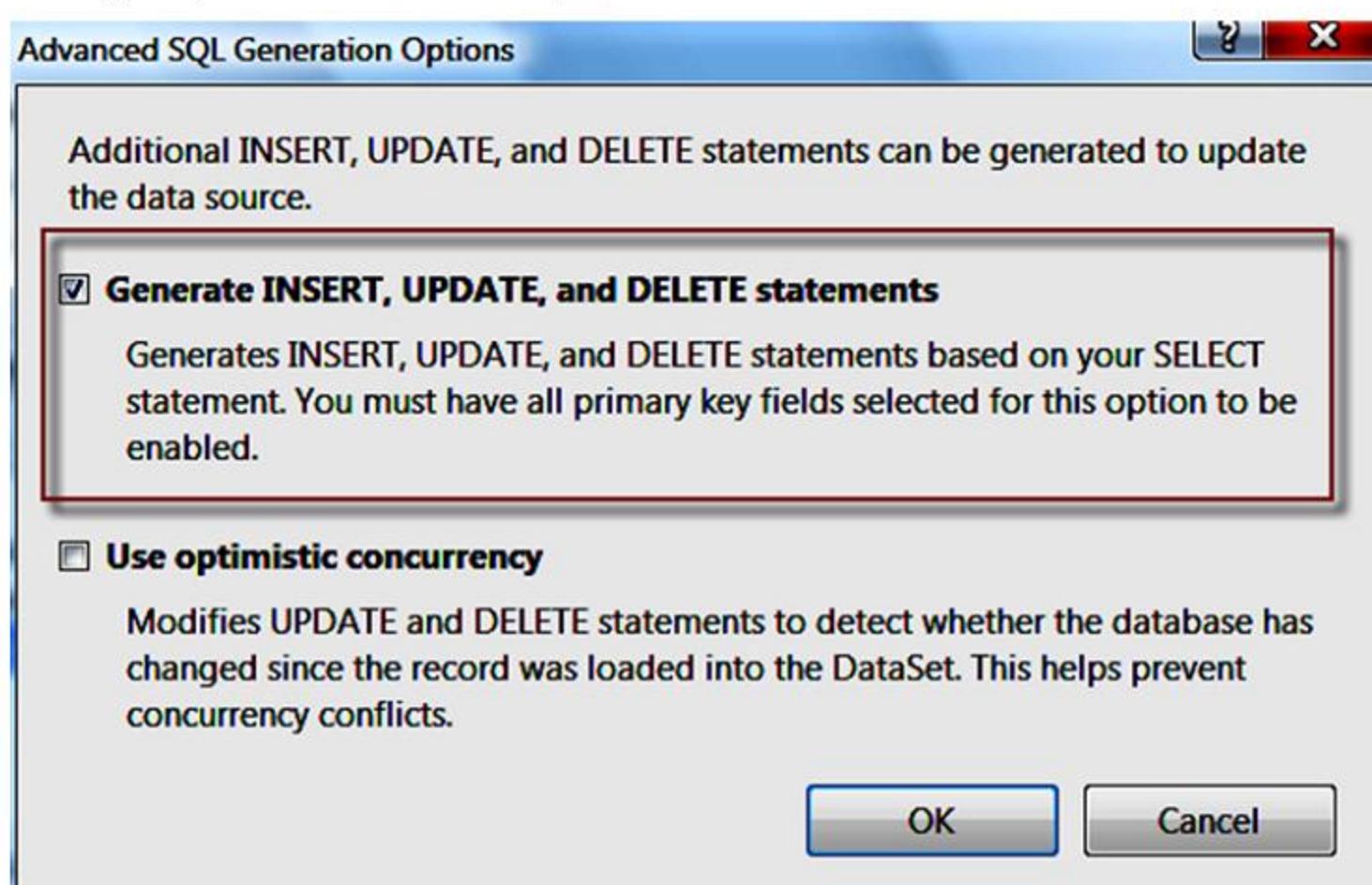
Bước 2. Trên cửa sổ *Configure Data Source | New Connection* và thiết lập các thông số trong cửa sổ *Add Connection* như hình sau.

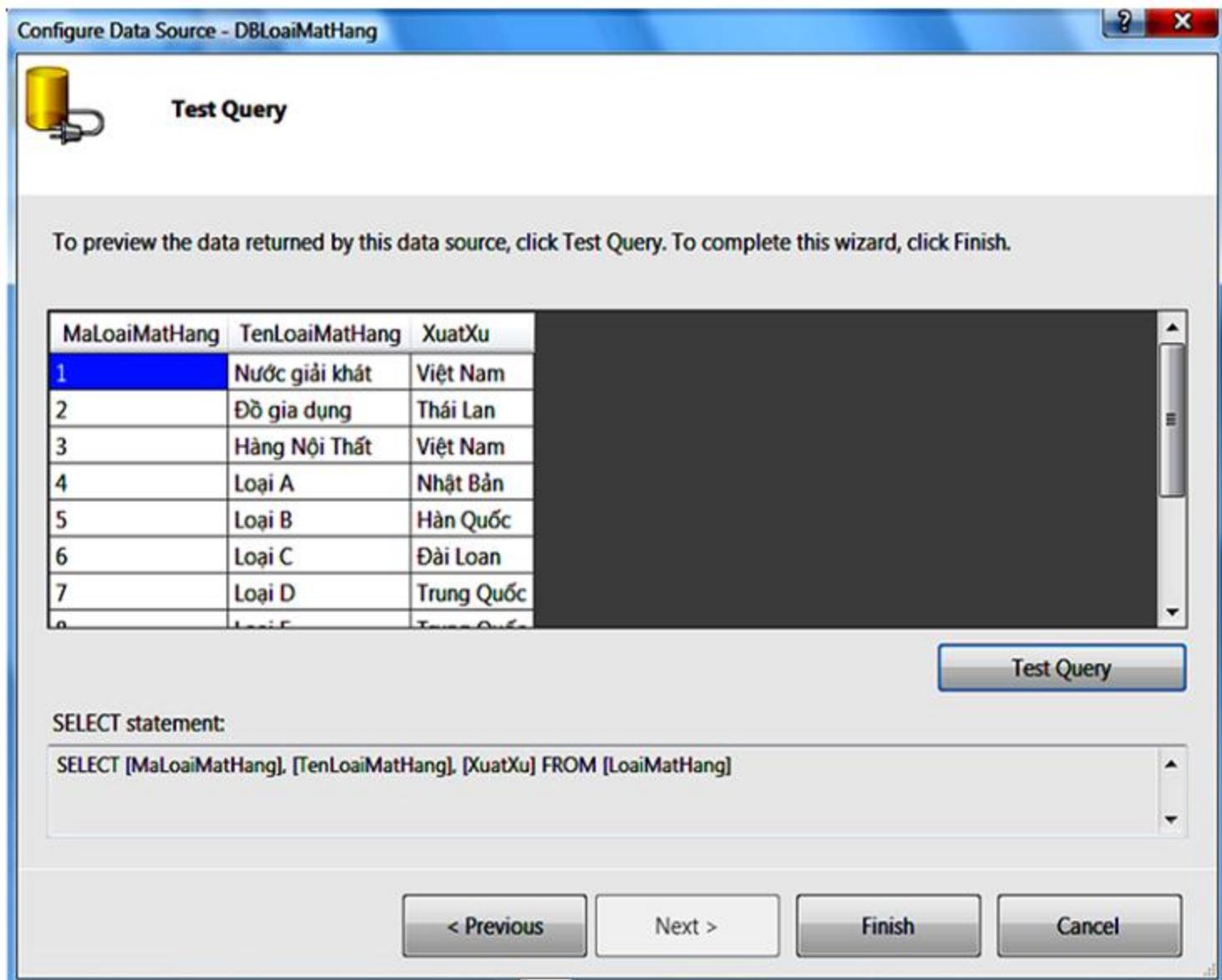


→ nhấp OK để trở về cửa sổ Configure Data Source | nhấp Next, xuất hiện thông báo lưu thông tin kết nối cơ sở dữ liệu vào trong tập tin cấu hình Web.config | nhấp Next | chọn các cột | nhấn Advanced...

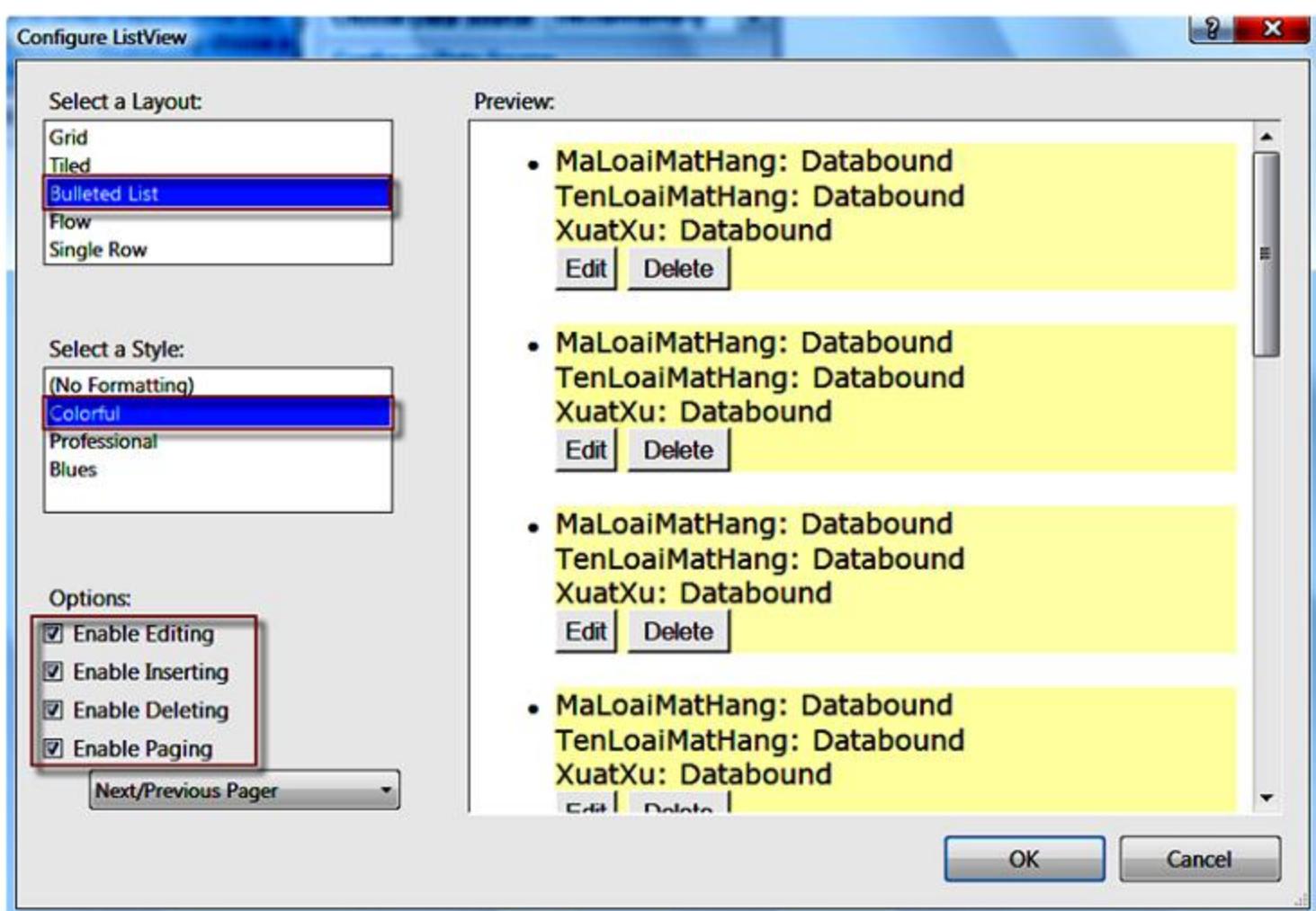


Bước 3. Trên cửa sổ *Advanced SQL Generation Options | Generate INSERT, UPDATE, and DELETE statements* | *OK*. Tiếp tục nhấn *Next | Test Query* để kiểm tra dữ liệu | nhấn *Finish* để hoàn tất.





Bước 4. Nhấp vào biểu tượng trên ListView control | *Configure ListView*, cửa sổ *Configure ListView* xuất hiện và thiết lập như hình dưới, nhấn OK để hoàn tất.



Bước 5. Nhấn Ctrl+F5 để thi hành ứng dụng, kết quả như hình bên.

Xem danh sách các loại mặt hàng - Windows Internet Explorer
http://localhost:51592/XemDanhSachCacLoaiMatHang.aspx

Xem danh sách các loại mặt hàng

- MaLoaiMatHang: 1
TenLoaiMatHang: Nước giải khát
XuatXu: Việt Nam
Update Cancel
- MaLoaiMatHang: 2
TenLoaiMatHang: Đồ gia dụng
XuatXu: Thái Lan
Edit Delete
- MaLoaiMatHang: 3
TenLoaiMatHang: Hàng Nội Thất
XuatXu: Việt Nam
Edit Delete
- MaLoaiMatHang: 4
TenLoaiMatHang: Loại A
XuatXu: Nhật Bản
Edit Delete
- MaLoaiMatHang: 5
TenLoaiMatHang: Loại B
XuatXu: Hàn Quốc

Chương 8

TÌM HIỂU VÀ ỨNG DỤNG CƠ CHẾ DATA BINDING

Các vấn đề chính sẽ được đề cập:

- ✓ *Giới thiệu Data Binding*
- ✓ *Các dạng Data Binding*

Kết thúc chương này các bạn có thể:

- *Sử dụng kỹ thuật DataBinding với các đối tượng dữ liệu (data controls) trong ứng dụng ASP.NET*

8.1. GIỚI THIỆU DATABINDING

ASP.NET cung cấp cho chúng ta rất nhiều điều khiển để cho phép hiển thị cũng như tiếp nhận thông tin từ người dùng. Có những điều khiển cho phép chúng ta hiển thị những thông tin tĩnh (Static – tức là giá trị xác định được ngay khi lập trình), một số hiển thị được cả những thông tin động (Dynamic - tức là được tính toán khi chạy chương trình). Để việc hiển thị thông tin động này một cách đơn giản và nhanh chóng, ASP.NET cung cấp cho chúng ta một đặc tính gọi là "Data Binding" (kết nối dữ liệu).

Data Binding là một kỹ thuật kết nối dữ liệu với những đối tượng. Sử dụng data binding, bạn có thể nối dữ liệu trong một nguồn dữ liệu đến một đối tượng người dùng. Mọi việc thay đổi trên các đối tượng giao diện người dùng có thể trực tiếp được cập nhật vào nguồn dữ liệu.

Từ "data" cũng cần phải được hiểu theo nghĩa rộng, nó không nhất thiết phải là cái gì đó liên quan đến Cơ sở dữ liệu như ta thường nghĩ mà có thể là một thuộc tính, một mảng, một tập hợp, một danh sách, một trường dữ liệu trong bảng CSDL... hay tổng quát là một biểu thức trả về giá trị.

Có hai kiểu binding dữ liệu đó là **Simple Data Binding** và **Repeated Data Binding**. Chúng ta quan tâm nhiều đến kiểu thứ hai – Repeated Data Binding - với nguồn dữ liệu được truy xuất từ cơ sở dữ liệu.

8.2. CÁC DẠNG DATA BINDING

8.2.1. Dạng kết nối dữ liệu đơn (Single Data Binding)

Trong ASP.NET, có thể gắn một giá trị đơn lẻ vào trang được gọi là Single Data Binding. Cú pháp để gắn dữ liệu đơn vào trang như sau:
`<%# biểu_thức %>`

Trong đó: **biểu_thức** có thể là một hằng, một biến, một hàm, một biểu thức hoặc có thể là một thuộc tính khác.

Một số cách dùng dạng kết nối dữ liệu đơn:

Hằng số: `<%# 20 %>
`

Hằng xâu: `<%# "Xin chào" %>
` Biểu thức: `<%# 10+5 %>
`

Hàm: `<%# "Sin(3.14/2) = " + Math.Sin(3.14/2) %>
`

Thuộc tính khác: `<%# "Tiêu đề của trang là " + this.Title %>`

Có thể gắn kết tới một biểu thức, một biến, thuộc tính ... bất kỳ.

Chú ý: Trong thủ tục Page_Load cần thêm lệnh **this.DataBind()** để thực sự gắn kết.

Thí dụ 1:

Tạo trang SimpleDataBinding.aspx minh họa dạng kết nối dữ liệu đơn giản.

Phần mã lệnh thiết kế trang SimpleDataBinding.aspx trong đó có sử dụng dạng kết nối dữ liệu đơn giản với biến **TransactionCount**.

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Simple Data Binding</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
```

```

<div>
    <asp:Label id="lblDynamic" runat="server" Font-Size="X-Large" >
        There were <%# TransactionCount %> transactions today.
        I see that you are using <%# Request.Browser.Browser %>.
    </asp:Label>
</div>
</form>
</body>
</html>

```

Phần mã lệnh thực thi trang SimpleDataBinding.aspx.cs:

```

public partial class SimpleDataBinding : System.Web.UI.Page
{
    protected int TransactionCount;

    protected void Page_Load(object sender, EventArgs e)
    {
        // (You could use database code here
        // to look up a value for TransactionCount.)
        TransactionCount = 10;

        // Now convert all the data binding expressions on the page.
        this.DataBind();
    }
}

```

Kết quả thực thi trang SimpleDataBinding.aspx:

There were 10 transactions today. I see that you are using Firefox.

8.2.2. Dạng kết nối dữ liệu có sự lặp lại (Repeated Data Binding)

Có rất nhiều trường hợp dữ liệu cần hiển thị là một danh sách (như: mảng, bảng, DataReader,...) hay tổng quát là một tập hợp các mục (Collection Items). Trong trường hợp như vậy, hoàn toàn có thể dùng cơ chế DataBinding trong ASP.NET để gắn kết quả đó vào một điều khiển dạng danh sách (như ListBox, DropDownList, CheckBoxList,...) để hiển thị mà không cần phải viết nhiều dòng code.

Các điều khiển cho phép gắn kết dữ liệu thường có ba thuộc tính với các ý nghĩa như sau:

DataSource: là thuộc tính để chỉ đến nguồn dữ liệu cần gắn kết. Nguồn dữ liệu này phải là một tập hợp. Ví dụ: DataTabe, Array,...

DataSourceID: chỉ đến một đối tượng cung cấp nguồn dữ liệu. Có thể sử dụng hoặc thuộc tính DataSourceID hoặc DataSource nhưng không được cả hai.

DataTextField: cho biết là gắn kết với trường nào của mỗi mục dữ liệu.

Thí dụ 2

Tạo trang ListDataBinding.aspx minh họa kết nối dữ liệu dạng **Repeated Data Binding**

Phần mã lệnh thiết kế trang ListDataBinding.aspx:

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:ListBox id="MyListBox" runat="server" Width="197px"
Height="193px"></asp:ListBox>
            <br /><br />
            <select id="MyHTMLSelect" size="1" runat="server"/>
            <br /><br />
            <asp:DropDownList id="MyDropDownListBox"
runat="server" Width="248px" Height="22px"></asp:DropDownList>
            <br /><br />
```

```

<asp:CheckBoxList id="MyCheckBoxList" runat="server"
Width="201px" Height="157px"></asp:CheckBoxList>
<br /><br />
<asp:RadioButtonList id="MyRadioButtonList"
runat="server" Width="249px" Height="158px"></asp:RadioButtonList>
</div>
</form>
</body>
</html>

```

Phần mã lệnh thực thi trang ListDataBinding.aspx.cs:

```

public partial class ListDataBinding : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e) {
        // Create and fill the collection.
        List<string> fruit = new List<string>();
        fruit.Add("Kiwi");
        fruit.Add("Pear");
        fruit.Add("Mango");
        fruit.Add("Blueberry");
        fruit.Add("Apricot");
        fruit.Add("Banana");
        fruit.Add("Peach");
        fruit.Add("Plum");
        // Define the binding for the list controls.
        MyListBox.DataSource = fruit;
        MyDropDownListBox.DataSource = fruit;
        MyHTMLSelect.DataSource = fruit;
        MyCheckBoxList.DataSource = fruit;
        MyRadioButtonList.DataSource = fruit;
        // Activate the binding.
        this.DataBind();
    }
}

```

Kết quả thực thi cho thấy rõ sự kết nối dữ liệu từ tập dữ liệu List<string> đến các điều khiển dữ liệu của ASP.NET:

The screenshot shows a Windows application window with the following elements:

- A **ListBox** containing the following items: Kiwi, Pear, Mango, Blueberry, Apricot, **Banana**, Peach, Plum. The item "Banana" is highlighted with a blue selection bar.
- An **ComboBox** with the value "Mango" and a dropdown arrow.
- A **CheckedListBox** with the following checked items: Mango, Banana.
- A **RadioButtonList** with the following selected items: Pear, Mango.

Chú ý: Nếu không có phát biểu **this.DataBind()** thì sẽ không có kết quả như trên.

Thí dụ 3:

Tạo trang DataSetBinding.aspx liên kết dữ liệu ListBox với đối tượng ADO.NET là DataSet

Phần mã lệnh thiết kế trang DataSetBinding.aspx:

```

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:ListBox ID="lstUser" runat="server" Height="152px"
Width="192px"></asp:ListBox></div>
        </form>
    </body>
</html>

```

Phần mã lệnh thực thi trang DataSetBinding.aspx.cs:

```

public partial class DataSetBinding : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // Define a DataSet with a single DataTable.
        DataSet dsInternal = new DataSet();
        dsInternal.Tables.Add("Users");

        // Define two columns for this table.
        dsInternal.Tables["Users"].Columns.Add("Name");
        dsInternal.Tables["Users"].Columns.Add("Country");

        // Add some actual information into the table.
        DataRow rowNew = dsInternal.Tables["Users"].NewRow();
        rowNew["Name"] = "John";
    }
}

```

```

rowNew["Country"] = "Uganda";
dsInternal.Tables["Users"].Rows.Add(rowNew);

rowNew = dsInternal.Tables["Users"].NewRow();
rowNew["Name"] = "Samantha";
rowNew["Country"] = "Belgium";
dsInternal.Tables["Users"].Rows.Add(rowNew);

rowNew = dsInternal.Tables["Users"].NewRow();
rowNew["Name"] = "Rico";
rowNew["Country"] = "Japan";
dsInternal.Tables["Users"].Rows.Add(rowNew);

// Define the binding.
lstUser.DataSource = dsInternal.Tables["Users"];
lstUser.DataTextField = "Name";

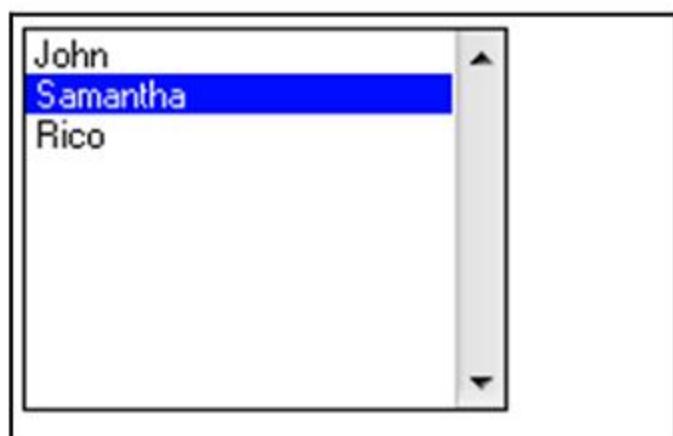
// Define the binding.
lstUser.DataSource = dsInternal;
lstUser.DataMember = "Users";
lstUser.DataTextField = "Name";

this.DataBind(); // Could also use lstItems.DataBind() to bind just the list box.

}
}

```

Kết quả thực thi:



BÀI TẬP CHƯƠNG 8

Bài 1: Thực hiện các bài tập thí dụ trong bài học để nắm vững nội dung của bài.

Bài 2: Tạo trang ASP.NET kết nối dữ liệu với đối tượng ảnh Image:

```
<asp:Image id="imgDynamic" ImageUrl="<%# URL %>" runat="server" />
```

Đối tượng Label:

```
<asp:Label id="lblDynamic" runat="server"><%# URL %></asp:Label>
```

Đối tượng CheckBox:

```
<asp:CheckBox id="chkDynamic" Text="<%# URL %>" runat="server" />
```

Đối tượng HyperLink:

```
<asp:Hyperlink id="lnkDynamic" Text="Click here!" NavigateUrl="<%# URL %>" runat="server" />
```

Phần mã lệnh thiết kế trang DataBindingUrl.aspx:

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

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

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

```
<head runat="server">
```

```
    <title>Untitled Page</title>
```

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

```
</head>
```

```
<body>
```

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

```
        <div>
```

```
            <asp:Label id="lblDynamic" runat="server"><%# URL %></asp:Label>
```

```
            <br /><br />
```

```
            <asp:CheckBox id="chkDynamic" Text="<%# URL %>" runat="server" />
```

```
            <br /><br />
```

```

<asp:Hyperlink id="lnkDynamic" Text="Click here!" NavigateUrl="<%# URL %>" runat="server" />
<br /><br />
<asp:Image id="imgDynamic" ImageUrl="<%# URL %>" runat="server" />

</div>
</form>
</body>
</html>

```

Phần mã lệnh thực thi trang:

```

public partial class DataBindingUrl: System.Web.UI.Page
{
    public string URL;
    protected void Page_Load(Object sender, EventArgs e)
    {
        URL = "Images/picture.jpg";
        this.DataBind();
    }
}

```

Kết quả thực thi:



Bài 3: Tạo trang ASP.NET RecordEditor.aspx gắn kết dữ liệu của bảng **Product** trong CSDL Northwind với các điều khiển dữ liệu **Label** và **ListBox** trên trang.

Phần mã lệnh thiết kế trang RecordEditor.aspx:

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Record Editor</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:DropDownList ID="lstProduct" runat="server" AutoPostBack="True"
Width="280px" OnSelectedIndexChanged="lstProduct_SelectedIndexChanged">
                </asp:DropDownList>
            <br />
            <br />
            <hr />
            <br />
            <table>
                <tr>
                    <td style="width: 234px" valign="top">
                        <asp:Label ID="lblRecordInfo" runat="server"></asp:Label></td>
                    <td style="width: 190px">
                        <asp:Panel ID=" pnlCategory" runat="server" Visible="False">
                            <asp:ListBox ID="lstCategory" runat="server" Height="120px"
Width="152px">
                                </asp:ListBox><br />
                            <br />
                            <asp:Button ID="cmdUpdate" runat="server" Text="Update"
OnClick="cmdUpdate_Click" />
                        </asp:Panel>
                    </td>
                </tr>
            </table>
        </div>
    </form>
</body>
```

```
</div>
</form>
</body>
</html>
```

Phần mã lệnh thực thi trang RecordEditor.aspx.cs:

```
public partial class RecordEditor: System.Web.UI.Page
{
    private string connectionString =
WebConfigurationManager.ConnectionStrings["Northwind"].ConnectionString;
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!this.IsPostBack)
        {
            // Define the ADO.NET objects for selecting Products.
            string selectSQL = "SELECT ProductName, ProductID FROM Products";
            SqlConnection con = new SqlConnection(connectionString);
            SqlCommand cmd = new SqlCommand(selectSQL, con);

            // Open the connection.
            con.Open();

            // Define the binding.
            IstProduct.DataSource = cmd.ExecuteReader();
            IstProduct.DataTextField = "ProductName";
            IstProduct.DataValueField = "ProductID";

            // Activate the binding.
            IstProduct.DataBind();

            con.Close();

            // Make sure nothing is currently selected.
            IstProduct.SelectedIndex = -1;
        }
    }

    protected void IstProduct_SelectedIndexChanged(object sender, EventArgs e)
```

```

{
    // Create a command for selecting the matching product record.
    string selectProduct = "SELECT ProductName, QuantityPerUnit, " +
        "CategoryName FROM Products INNER JOIN Categories ON " +
        "Categories.CategoryID=Products.CategoryID " +
        "WHERE ProductID=@ProductID";

    // Create the Connection and Command objects.
    SqlConnection con = new SqlConnection(connectionString);
    SqlCommand cmdProducts = new SqlCommand(selectProduct, con);
    cmdProducts.Parameters.AddWithValue("@ProductID",
        lstProduct.SelectedItem.Value);

    // Retrieve the information for the selected product.
    using (con)
    {
        con.Open();
        SqlDataReader reader = cmdProducts.ExecuteReader();
        reader.Read();

        // Update the display.
        lblRecordInfo.Text = "<b>Product:</b> " + reader["ProductName"] + "<br />";
        lblRecordInfo.Text += "<b>Quantity:</b> " + reader["QuantityPerUnit"] + "<br />";
        lblRecordInfo.Text += "<b>Category:</b> " + reader["CategoryName"];

        // Store the corresponding CategoryName for future reference.
        string matchCategory = reader["CategoryName"].ToString();
        // Close the reader.
        reader.Close();

        // Create a new Command for selecting categories.
        string selectCategory = "SELECT CategoryName, CategoryID FROM
Categories";
        SqlCommand cmdCategories = new SqlCommand(selectCategory, con);

        // Retrieve the category information, and bind it.
        lstCategory.DataSource = cmdCategories.ExecuteReader();
        lstCategory.DataTextField = "CategoryName";
        lstCategory.DataValueField = "CategoryID";
        lstCategory.DataBind();
    }
}

```

```

// Highlight the matching category in the list.
    lstCategory.Items.FindByText(matchCategory).Selected = true;
}
pnlCategory.Visible = true;
}

protected void cmdUpdate_Click(object sender, EventArgs e)
{
    // Define the Command.
    string updateCommand = "UPDATE Products " +
        "SET CategoryID=@CategoryID WHERE ProductID=@ProductID";

    SqlConnection con = new SqlConnection(connectionString);
    SqlCommand cmd = new SqlCommand(updateCommand, con);

    cmd.Parameters.AddWithValue("@CategoryID",
        lstCategory.SelectedItem.Value);
    cmd.Parameters.AddWithValue("@ProductID",
        lstProduct.SelectedItem.Value);

    // Perform the update.
    using (con)
    {
        con.Open();
        cmd.ExecuteNonQuery();
    }
}

```

Kết quả thực thi:

Đầu tiên hiển thị nội dung một DropDownList:



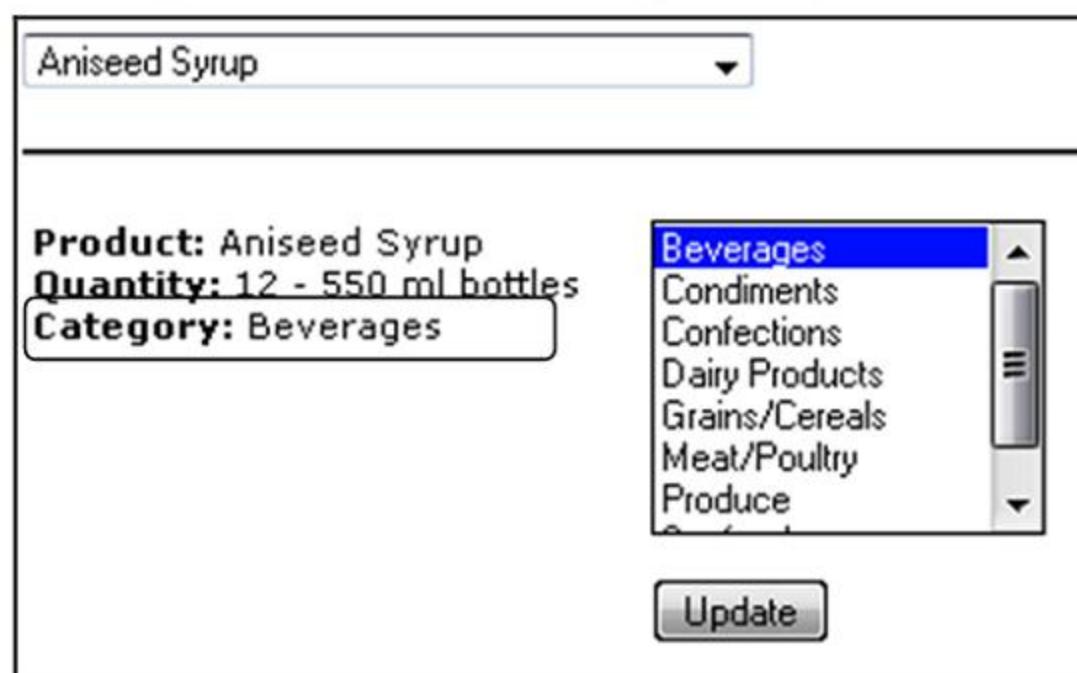
Khi chọn giá trị trong DropDownList trên sẽ gây ra sự kiện **lstProduct_SelectedIndexChanged(object sender, EventArgs e)** và sẽ hiển thị kết quả lựa chọn:



Để ý thấy giá trị trong **ListBox** đang chọn chưa thông tin **Category** của mẫu tin **Product** vừa được chọn. Nếu thay đổi giá trị khác và click nút Update dữ liệu mới thay đổi của vùng Category trên mẫu tin hiện tại sẽ được cập nhật lại và hiện ra ở lần chọn sau.



Kết quả hiện ra ngay sau khi chọn lại sản phẩm này.



Chương 9

CÁC ĐỐI TƯỢNG DỮ LIỆU (DATA CONTROLS)

Các bạn cần trình bày được các vấn đề sau:

- Các đặc điểm của ADO.NET
- Kết nối cơ sở dữ liệu sử dụng đối tượng `SQLDataSource Control`
- Thực thi được ràng buộc dữ liệu với các `DataControls`: `GridView`, `DataList`, `ListView`, ...

9.1. ĐỐI TƯỢNG DỮ LIỆU GRIDVIEW

Trong bài học này, chúng ta sẽ đi tìm hiểu và sử dụng một số tính năng nổi bật của GridView, từ đó có thể áp dụng làm các trang liệt kê hàng hóa cho ứng dụng web.

9.1.1. Tìm hiểu lớp GridView

GridView có lẽ là một điều khiển trình diễn dữ liệu quan trọng nhất của ASP.NET. Nó cho phép gắn và hiển thị dữ liệu ở dạng bảng, trong đó mỗi hàng là một bản ghi, mỗi cột ứng với một trường dữ liệu. Ngoài việc hiển thị, GridView còn có rất nhiều tính năng khác mà trước đây người ta phải viết rất nhiều dòng code mới có được.

Thí dụ: Định dạng, phân trang, sắp xếp, sửa đổi, xóa dữ liệu.

GridView có thể gắn kết dữ liệu với các nguồn như `DataReader`, `SqlDataSource`, `ObjectDataSource` hay bất kỳ nguồn nào có cài đặt `System.CollectionsEnumerable`.

9.1.2. Các thuộc tính và cột thuộc tính

GridView ngoài việc hiển thị thuận túy các trường của một nguồn dữ liệu, nó còn cho phép hiển thị dưới các hình thức khác (dưới dạng nút, dạng HyperLink, dạng checkbox,...), các cột khác hỗ trợ cho việc thao tác dữ liệu như Select, Update, Delete hoàn toàn có thể tùy biến trong GridView.

Để chỉnh sửa các cột dạng này, click chọn "smart tag" của GridView và chọn "Edit Field" hoặc chọn thuộc tính `Columns` của GridView trong cửa sổ thuộc tính.

Loại cột	Mô tả
BoundField	Hiển thị giá trị của một trường thuộc nguồn dữ liệu
ButtonField	Hiển thị một nút lệnh cho mỗi mục trong GridView. Nút này cho phép bạn có thể tạo ra các nút tùy biến kiểu như Add hay Remove
CheckBoxField	Hiển thị một checkbox ứng với mỗi mục trong GridView. Cột này thường được dùng để hiển thị các trường kiểu Boolean (Yes/No)
CommandField	Hiển thị các nút lệnh đã được định nghĩa sẵn để thực hiện các thao tác select, edit, hay delete
HyperLinkField	Hiển thị giá trị của một trường dưới dạng siêu liên kết (hyperlink). Loại cột này cho phép bạn gắn một trường thứ hai vào URL của siêu liên kết
ImageField	Hiển thị một ảnh ứng với mỗi mục trong GridView
TemplateField	Hiển thị nội dung tùy biến của người dùng cho mỗi mục dữ liệu trong GridView, theo như mẫu định sẵn. Loại cột này cho phép ta tạo ra các cột tùy biến

Các thuộc tính

Thuộc tính	Mô tả
GridLines	Ẩn, hiện các đường viền của GridView
ShowHeader	Cho phép hiện/ ẩn phần Header
ShowFooter	Cho phép hiện/ ẩn phần Footer
PageSize	Get/Set cho biết mỗi trang chứa bao nhiêu dòng
PageCount	Cho biết số trang của nguồn dữ liệu mà nó gắn kết
PageIndex	Get/Set chỉ số của trang đang được hiển thị
AllowPaging	Có cho phép phân trang không (true = có)
AllowSorting	Có cho phép sắp xếp không (true=có)
AutoGenerateColumns	Có tự động sinh ra các cột ứng với các cột trong nguồn dữ liệu hay không? Mặc định = true (có)

AutoGenerateDeleteButton	Tự động tạo ra cột Delete (true = tự động)
AutoGenerateUpdateButton	Tự động tạo ra cột Update (true = tự động)
AutoGenerateSelectButton	Tự động tạo ra cột Select (true = tự động)
EditIndex (SelectedIndex)	Đặt hàng nào đó về chế độ edit. EditIndex = 2 → hàng thứ ba (chỉ số 2) sẽ về chế độ edit. Nếu đặt EditIndex = -1 thì sẽ thoát khỏi chế độ Edit
SelectedIndex	Trả về chỉ số của dòng đang chọn
Rows	Một tập hợp chứa các hàng của GridView
Columns	Một tập hợp chứa các cột của GridView

9.1.3 Các style áp dụng cho GridView

GridView rất linh hoạt trong việc trình bày dữ liệu, nó cho phép ta định dạng các phần thông qua style. Thí dụ ta có thể định dạng cho phần Header, Footer, các mục dữ liệu, các hàng chẵn-lẻ v.v...

Bảng dưới đây sẽ giải thích rõ ý nghĩa một số thuộc tính:

Thuộc tính style	Mô tả
AlternatingRowStyle	Style áp dụng cho các hàng dữ liệu chẵn-lẻ trong GridView. Khi đặt thuộc tính này thì các hàng sẽ được hiển thị với định dạng luân phiên giữa <u>RowStyle</u> và <u>AlternatingRowStyle</u>
EditRowStyle	Style để hiển thị hàng hiện đang được sửa (Edit)
FooterStyle	Style áp dụng cho phần Footer
HeaderStyle	Style áp dụng cho phần Header
PagerStyle	Style áp dụng cho phần phân trang (các trang << <u>1</u> <u>2</u> <u>3</u> ... >>)
RowStyle	Style áp dụng cho các hàng dữ liệu trong GridView control. Khi AlternatingRowStyle được thiết lập thì sẽ áp dụng luân phiên giữa RowStyle và <u>AlternatingRowStyle</u>
SelectedRowStyle	Style áp dụng cho hàng đang được chọn (Selected) của GridView

9.1.4. Các sự kiện

GridView có rất nhiều sự kiện quan trọng, các sự kiện này khi kích hoạt sẽ cung cấp cho ta những thông tin hữu ích trong quá trình xử lý. Thí dụ, khi chúng ta click nút Update, nó sẽ kích hoạt sự kiện Updating và trả về cho chúng ta các giá trị mà người dùng vừa sửa....

Dưới đây là bảng tổng hợp một số sự kiện hay dùng nhất:

Tên sự kiện	Mô tả
PageIndexChanged	Xuất hiện khi ta click chọn các nút (<< 1 2 3 >>) trong hàng phân trang
PageIndexChanging	Xuất hiện khi người dùng click chọn các nút (<< 1 2 3 >>) trong hàng phân trang nhưng TRƯỚC khi GridView thực hiện việc phân trang. Ta có thể hủy việc phân trang tại sự kiện này
RowCancelingEdit	Xuất hiện khi nút Cancel được click nhưng trước khi thoát khỏi chế độ Edit
RowCommand	Xuất hiện khi một nút được click
RowCreated	Xuất hiện khi một hàng mới được tạo ra. Thường được sử dụng để sửa nội dung của hàng khi nó vừa được tạo ra
RowDataBound	Xuất hiện khi một hàng dữ liệu được gắn vào GridView. Tại đây ta có thể sửa đổi nội dung của hàng đó
RowDeleted	Xuất hiện khi nút Delete của một hàng được click, nhưng sau khi GridView đã delete bản ghi từ nguồn
RowDeleting	Xuất hiện khi nút Delete được click nhưng trước khi GridView xóa bản ghi từ nguồn. Tại đây có thể Cancel việc Delete
RowEditing	Xuất hiện khi nút Edit được click, nhưng trước khi GridView về chế độ sửa
RowUpdated	Xuất hiện khi nút Update được click, nhưng sau khi GridView update hàng dữ liệu
RowUpdating	Xuất hiện khi nút Update được click, nhưng trước khi GridView update hàng dữ liệu

SelectedIndexChanged	Xuất hiện khi nút Select của hàng được click nhưng sau khi GridView xử lý xong thao tác Select
SelectedIndexChangedChanging	Xuất hiện khi nút Select của hàng được click nhưng trước khi GridView xử lý xong thao tác Select
Sorted	Xuất hiện khi Hyperlink (tiêu đề cột) được click, nhưng sau khi GridView thực hiện việc sắp xếp
Sorting	Xuất hiện khi Hyperlink (tiêu đề cột) được click, nhưng trước khi GridView thực hiện việc sắp xếp. Sự kiện này khi xảy ra, nó sẽ cung cấp cho chúng ta thông tin về tên cột vừa được click. Dựa vào đó ta có thể thực hiện việc sắp xếp một cách dễ dàng

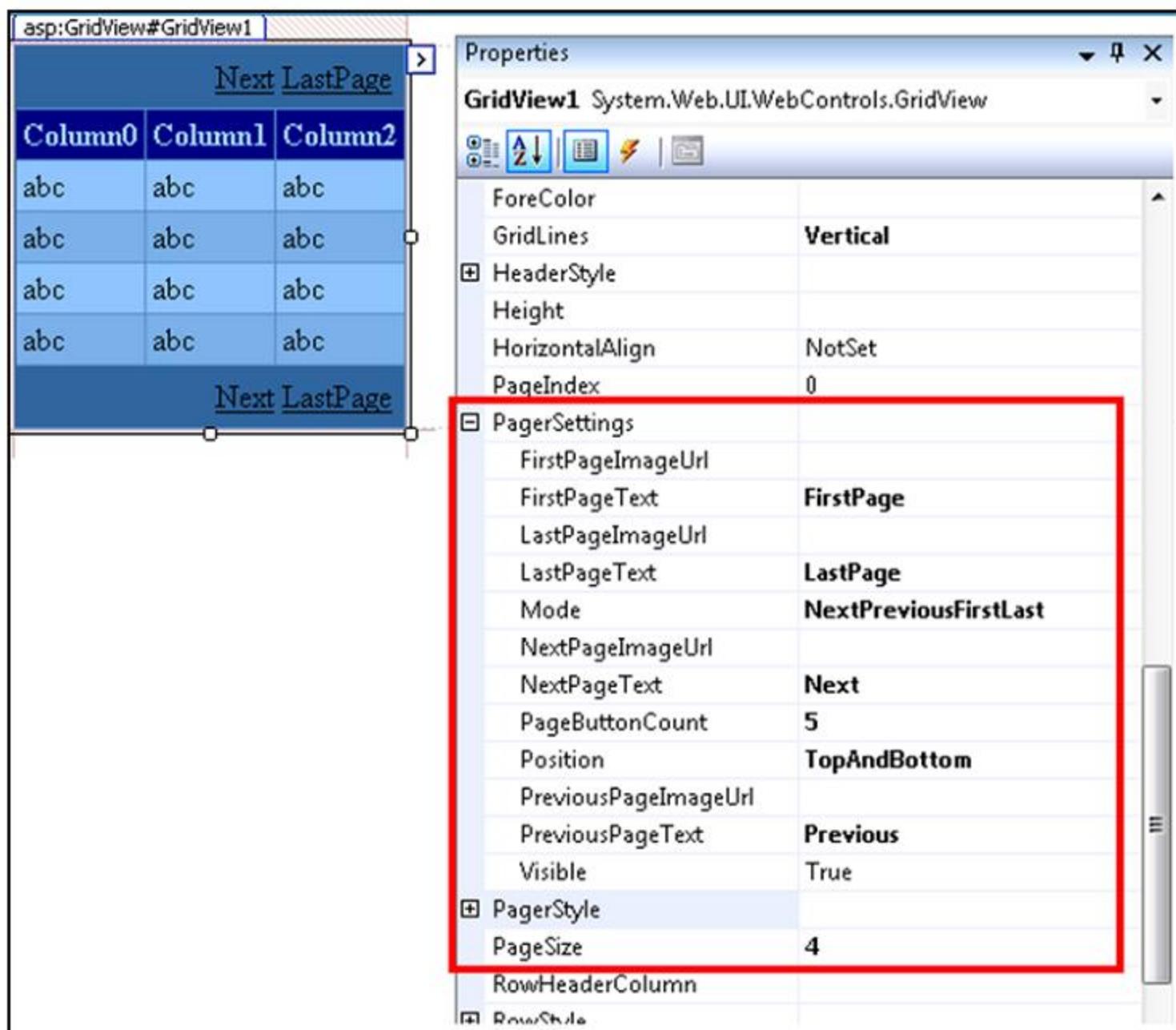
9.1.5. Các phương thức

Tên phương thức	Mô tả
DataBind()	Gắn kết dữ liệu giữa GridView và nguồn dữ liệu (đặt các thuộc tính DataSource, DataTextField hoặc DataSourceID)
DeleteRow(int)	Xóa một dòng trong GridView
UpdateRow(int i, bool Valid)	Cập nhật một dòng trong GridView
Sort(Biểu thức sx, hướng sx)	Sắp xếp dựa trên biểu thức và hướng

9.1.6. Các tính năng hỗ trợ của GridView

9.1.6.1. Phân trang

Để thực hiện phân trang, cần đặt thuộc tính **AllowPaging = True**. Khi phân trang, có thể tùy biến hiển thị các trang (hiển thị dạng các số 1, 2, 3 hay mũi tên <> >>) bằng cách đặt các thuộc tính con trong **PagerSettings**.



Đáp ứng với thao tác click nút chuyển trang gây ra sự kiện **PageIndexChanging** trên GridView. Bắt buộc trong phương thức ủy thác phải có dòng lệnh **GridView1PageIndex= e.NewPageIndex;**

```
protected void GridView1_PageIndexChanging(object sender,
    GridViewEventArgs e)
{
    GridView1PageIndex= e.NewPageIndex;
    .....
}
```

9.1.6.2. Tính năng tự động sắp xếp

Tính năng này cho phép dữ liệu trong GridView sẽ tự động được sắp xếp theo giá trị của cột mà người dùng click. Ở đây, ta có thể sắp xếp theo chiều tăng (**Asscending**) hoặc giảm (**Descending**).

Để bật tính năng này, cần đặt thuộc tính **AllowSorting = true** trong GridView.

Khi người dùng click chuột vào một cột tiêu đề nào đó của GridView thì sự kiện **Sorting** sẽ được kích hoạt, tại đây ta cần phải chỉ rõ cho GridView biết là sắp theo cột nào (**SortExpression**) và theo chiều tăng hay giảm (**SortDirection**).

```
<asp:GridView ID="GridView1" runat="server" AllowPaging="True"
BackColor="#CCCCCC"

    BorderColor="#999999" BorderStyle="Solid" BorderWidth="3px"
    CaptionAlign="Right"

        CellPadding="4"
    OnPageIndexChanging="GridView1_PageIndexChanging"

        PageSize="5" AllowSorting="True" CellSpacing="2" ForeColor="Black"
    OnSorting="GridView1_Sorting">

        <PagerSettings FirstPageText="FirstPage" LastPageText="LastPage"

            NextPageText="Next" PageButtonCount="5"
        Position="TopAndBottom" PreviousPageText="Previous" />

        <FooterStyle BackColor="#CCCCCC" />
        <RowStyle BackColor="White" />
        <SelectedRowStyle BackColor="#000099" Font-Bold="True"
        ForeColor="White" />
        <PagerStyle BackColor="#CCCCCC" ForeColor="Black"
        HorizontalAlign="Left" Wrap="False" />
        <HeaderStyle BackColor="Black" Font-Bold="True" ForeColor="White" />
    </asp:GridView>
```

Phương thức đáp ứng sự kiện Sorting:

```
protected void GridView1_Sorting(object sender, GridViewSortEventArgs e)
{
    bool sortAscending = (ViewState["SortAscending"]==null)?
        true:(bool)ViewState["SortAscending"];
    if (sortAscending)
    {
        e.SortDirection = SortDirection.Ascending;
        sortAscending = false;
    }
    else
```

```

{
    e.SortDirection = SortDirection.Descending;
    sortAscending = true;
}

// do something

ViewState["SortAscending"] = sortAscending;
}

```

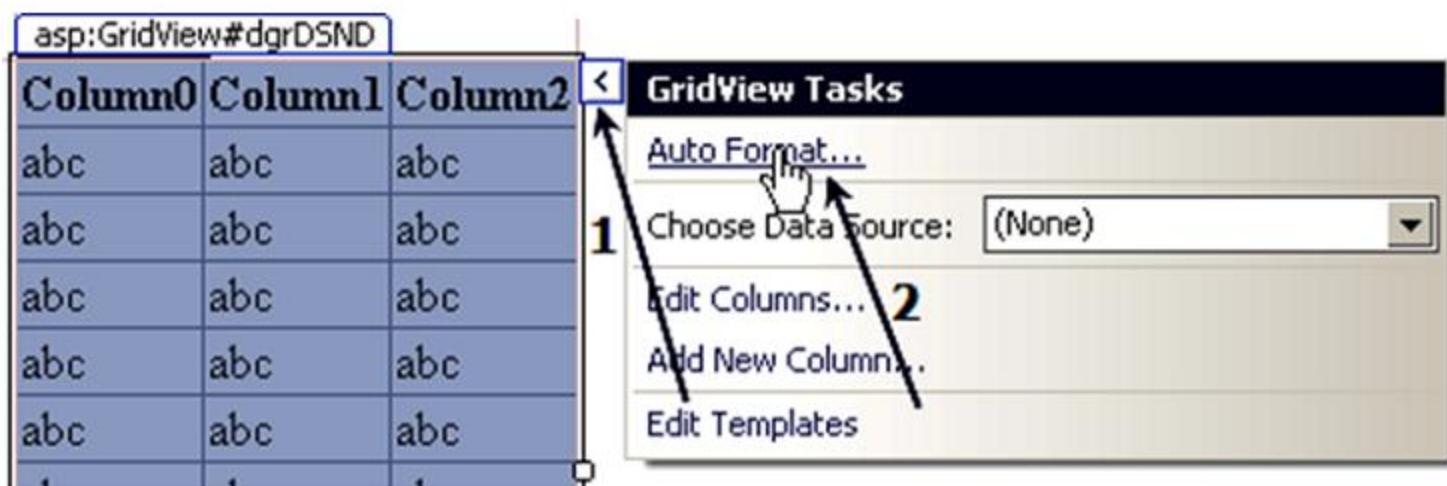
9.1.6.3. Các m^{ẫu} hi^{ển} thi - Template

ASP.NET cung cấp cho chúng ta sẵn một số **Template** (m^{ẫu}) để hiển thi GridView cũng khá đẹp. Vì vậy, bạn có thể sử dụng ngay các template này khi xây dựng ứng dụng.

Cách thức chọn template cho GridView như sau:

Bước 1: Mở trang ở chế độ Design

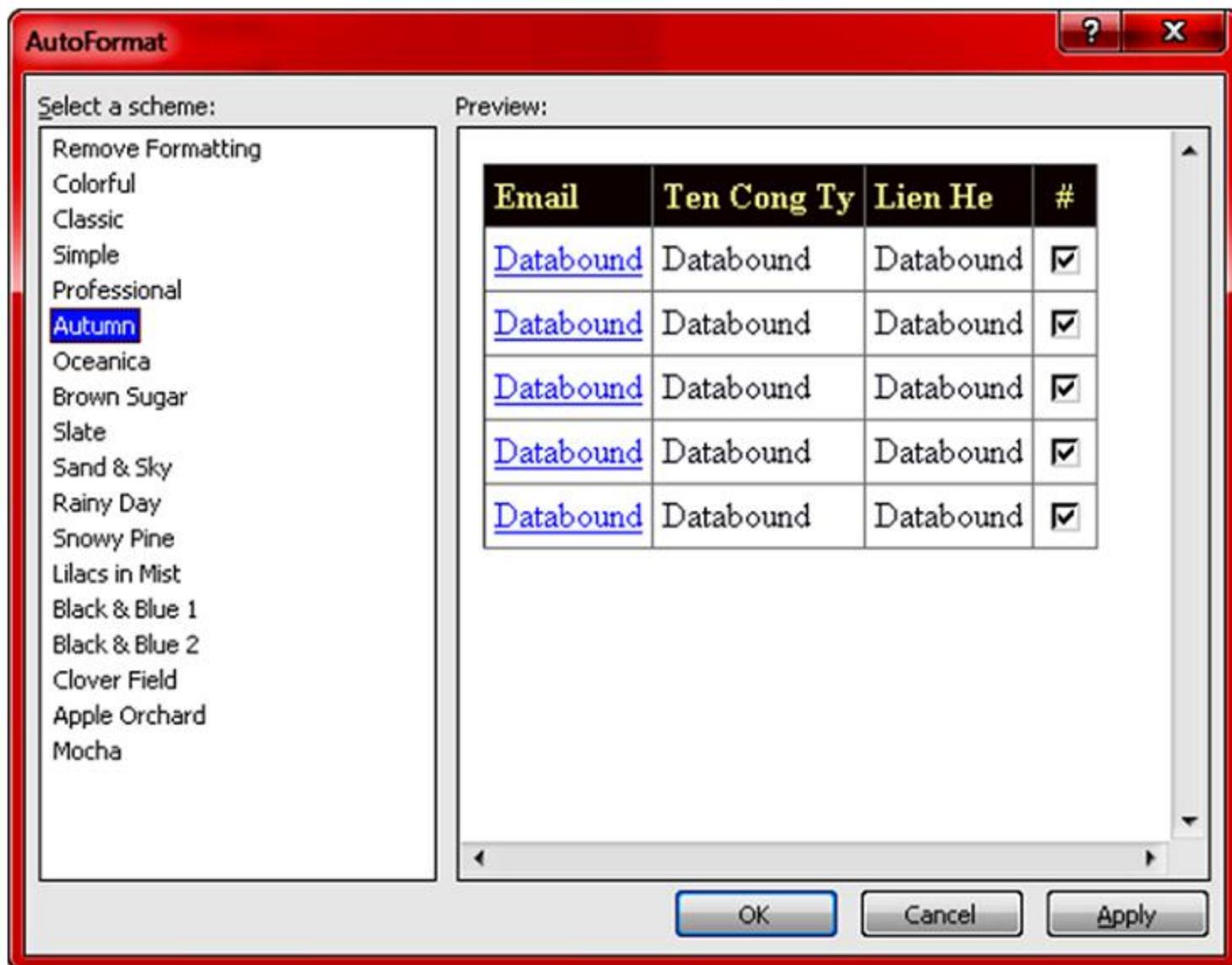
Bước 2: Chọn GridView và chọn smart tag, tiếp theo chọn AutoFormat



Bước 3: Chọn Format trong danh sách

Tô họ^p màu đ^{ược} ch^{ọn} t^ừ Template c^ó sẵn

Sau khi chọn Template, ASP.NET sẽ tự động tạo ra các thuộc tính (thẻ) tương ứng trong GridView, tại đây bạn có thể tiếp tục tùy biến thêm theo ý muốn.



9.1.7. Tạo các cột tùy biến HyperLink, BoundColumn...

9.1.7.1. Tạo cột BoundField thủ công

Để tạo các cột thủ công, cần đặt thuộc tính **AutoGenerateColumns = "False"**, sau đó soạn thủ công các cột trong cửa sổ Edit Columns.

9.1.7.2. Tạo một cột HyperlinkField

Tạo trang ASP.NET GridViewAndHyperLinkPage.aspx sử dụng GridView có cột loại HyperLinkField.

Phản mã lệnh thiết kế trang GridViewAndHyperLinkPage.aspx:

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

<!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>
            <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False">
                <Columns>
                    <asp:HyperLinkField DataNavigateUrlFields="RecruiterID"
DataNavigateUrlFormatString="~/Details.aspx?RecruiterID={0}"
                        DataTextField="Email" HeaderText="Email">
                        <HeaderStyle HorizontalAlign="Left" />
                    </asp:HyperLinkField>
                    <asp:BoundField DataField="CompanyName" HeaderText="Ten Cong Ty">
                        <HeaderStyle HorizontalAlign="Left" />
                    </asp:BoundField>
                    <asp:BoundField DataField="ContactName" HeaderText="Lien He">
                        <HeaderStyle HorizontalAlign="Left" />
                    </asp:BoundField>
                    <asp:CheckBoxField DataField="Activate" HeaderText="#" />
                </Columns>
            </asp:GridView>
        </div>
    </form>
</body>
</html>

```

Phần mã lệnh thực thi trang GridViewAndHyperLinkPage.aspx.cs

```

public partial class GridViewAndHyperLinkPage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        FillData(GridView1);
    }
}

```

```

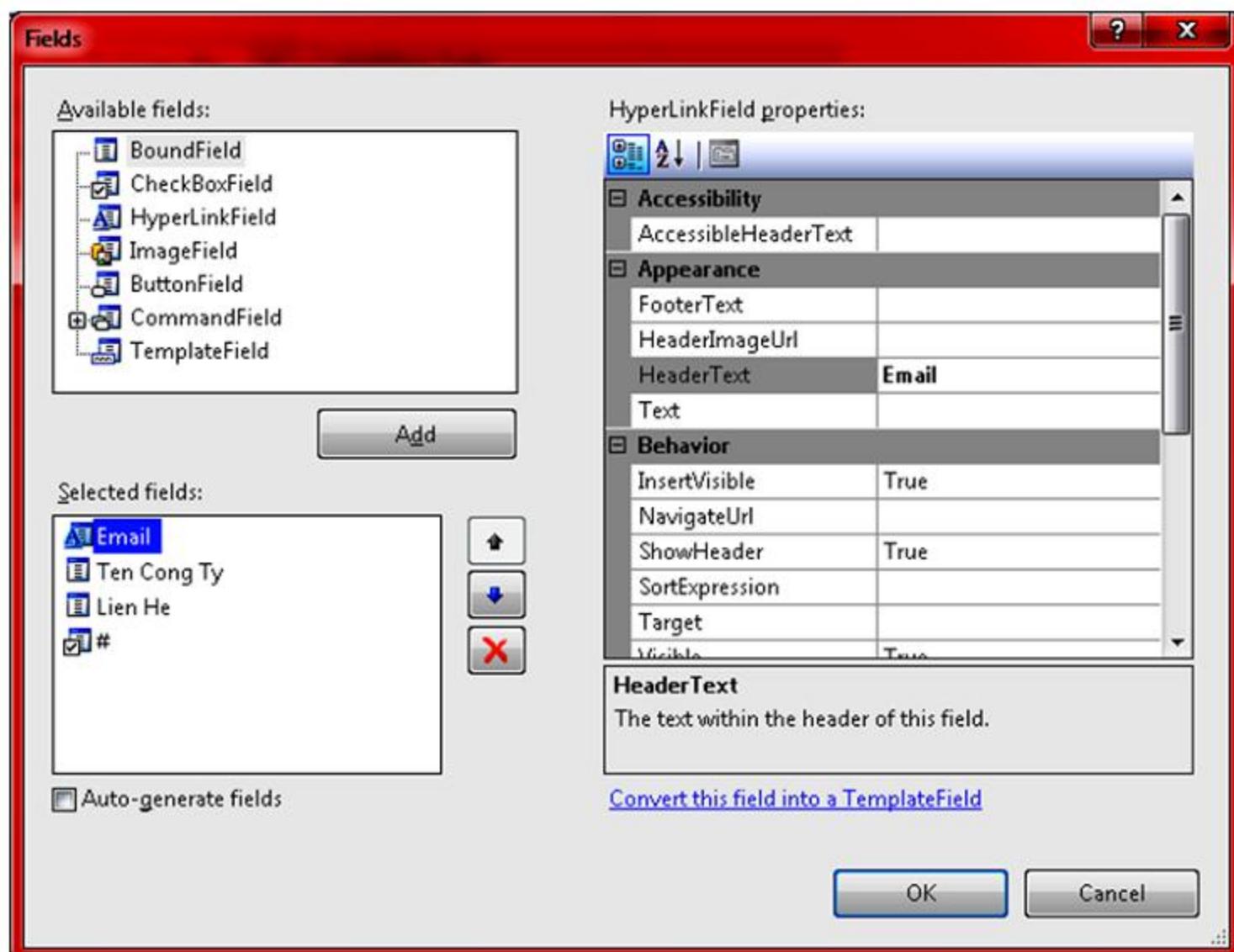
void FillData(GridView gridView)
{
    SqlDataSource sqlDataSource = new SqlDataSource();
    sqlDataSource.ConnectionString =
        WebConfigurationManager.ConnectionStrings[
            "RecruitVietnamDbConnectionString"
        ].ConnectionString;
    sqlDataSource.SelectCommandType =
        SqlDataSourceCommandType.Text;
    sqlDataSource.SelectCommand = "select " +
        "RecruiterID,Email,CompanyName, "
        + "ContactName,Address,Activate from tblRCAccounts";
    gridView.DataSource = sqlDataSource;
    gridView.DataBind();
}

```

Phần thiết kế giao diện trang:

Email	Ten Cong Ty	Lien He	#
Databound	Databound	Databound	<input checked="" type="checkbox"/>
Databound	Databound	Databound	<input checked="" type="checkbox"/>
Databound	Databound	Databound	<input checked="" type="checkbox"/>
Databound	Databound	Databound	<input checked="" type="checkbox"/>
Databound	Databound	Databound	<input checked="" type="checkbox"/>

Trong đó, cột đầu tiên [Email] được chọn dạng HyperLinkField:



Kết quả thực thi:

Email	Ten Cong Ty	Lien He	#
ngocanh@MadridHome.com	Viet Employment	Martin Sommer	<input checked="" type="checkbox"/>
ngocanh@Marseille.com	Bon app'	Laurence Lebihan	<input checked="" type="checkbox"/>
admin@Tswassen.com	Bottom-Dollar Markets	Elizabeth Lincoln	<input type="checkbox"/>
BuenosAires_admin@BuenosAires.com	Cactus Comidas para llevar	Patricia Simpson	<input type="checkbox"/>
Bern_admin@Bern.com	Chop-suey Chinese	Yang Wang	<input type="checkbox"/>
LondonFamily_admin@LondonFamily.com	Consolidated Holdings	Elizabeth Brown	<input type="checkbox"/>
Stuttgart_admin@Stuttgart.com	Die Wandernde Kuh	Rita Müller	<input checked="" type="checkbox"/>
Aachen_admin@Aachen.com	Drachenblut Delikatessen	Sven Ottlieb	<input checked="" type="checkbox"/>
Nantes_admin@Nantes.com	Du monde entier	Janine Labrune	<input checked="" type="checkbox"/>
LondonDistrict_admin@LondonDistrict.com	Eastern Connection	Ann Devon	<input checked="" type="checkbox"/>
Graz_admin@Graz.com	Ernst Handel	Roland Mendel	<input type="checkbox"/>
SaoPauloHome_admin@SaoPauloHome.com	Familia Arquibaldo	Aria Cruz	<input checked="" type="checkbox"/>
MadridCity_admin@MadridCity.com	FISSA Fabrica Inter. Salchichas S.A.	Diego Roel	<input type="checkbox"/>
Lille_admin@Lille.com	Folies gourmandes	Martine Rancé	<input checked="" type="checkbox"/>
Bräcke_admin@Bräcke.com	Folk och fä HB	Maria Larsson	<input type="checkbox"/>

9.1.7.3. Tạo cột ButtonField

Tạo trang ASP.NET GridViewAndButtonPage.aspx sử dụng GridView có cột loại ButtonField.

Phản mã lệnh thiết kế trang GridViewAndButtonPage.aspx:

```

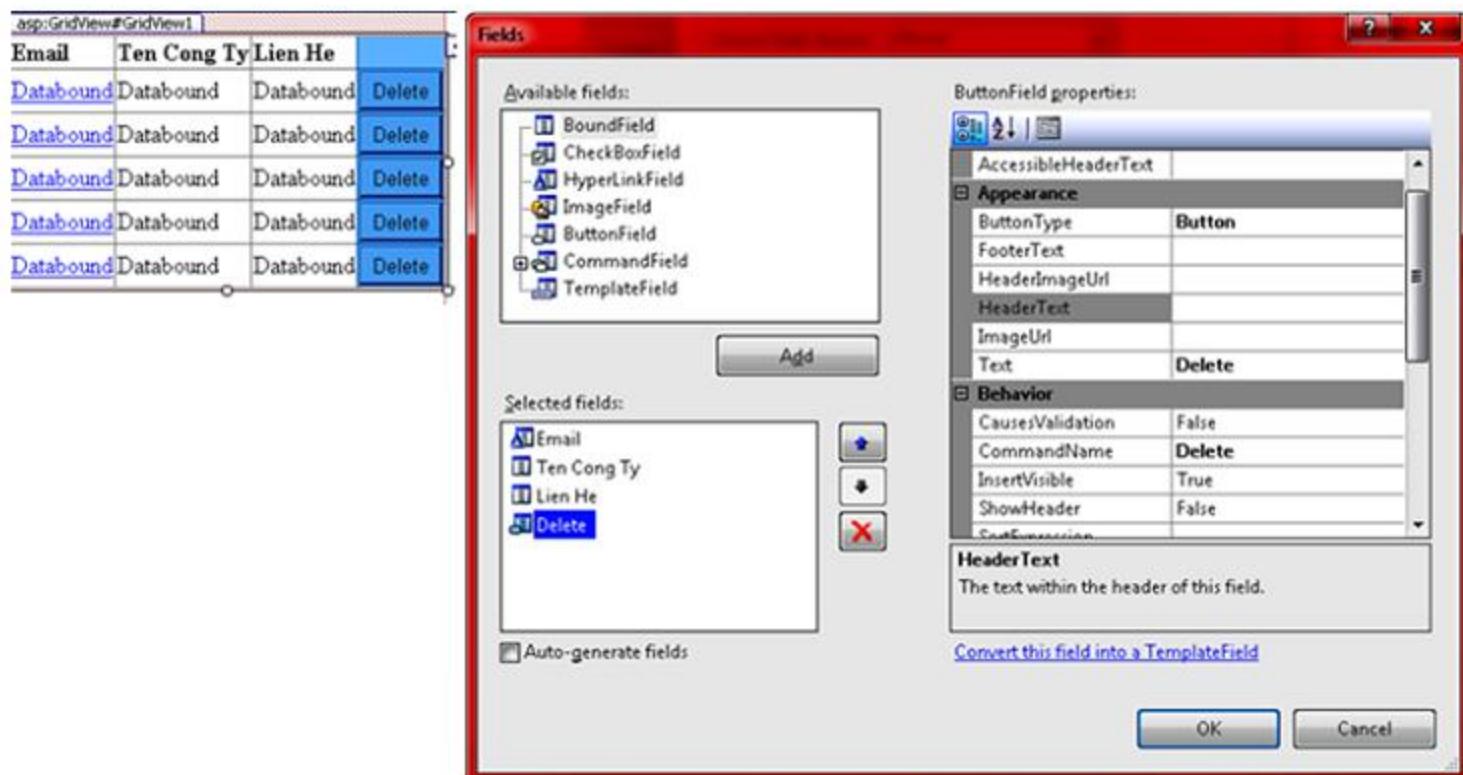
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="GridViewAndButtonPage.aspx.cs" Inherits="GridViewAndButtonPage" %>

<!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>
            <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
OnRowDeleting="GridView1_RowDeleting">
                <Columns>
                    <asp:HyperLinkField DataNavigateUrlFields="RecruiterID"
DataNavigateUrlFormatString="~/Details.aspx?RecruiterID={0}"
DataTextField="Email" HeaderText="Email">
                        <HeaderStyle HorizontalAlign="Left" />
                    </asp:HyperLinkField>
                    <asp:BoundField DataField="CompanyName" HeaderText="Ten Cong Ty">
                        <HeaderStyle HorizontalAlign="Left" />
                    </asp:BoundField>
                    <asp:BoundField DataField="ContactName" HeaderText="Lien He">
                        <HeaderStyle HorizontalAlign="Left" />
                    </asp:BoundField>
                    <asp:ButtonField ButtonType="Button" Text="Delete"
CommandName="Delete" />
                </Columns>
            </asp:GridView>
        </div>
    </form>
</body>
</html>

```

Phần giao diện thiết kế trang GridViewAndButtonPage.aspx:



Phần mã lệnh thực thi trang GridViewAndButtonPage.aspx.cs:

```
public partial class GridViewAndButtonPage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        FillData(GridView1);
    }

    void FillData(GridView gridView)
    {
        SqlDataSource sqlDataSource = new SqlDataSource();
        sqlDataSource.ConnectionString =
            WebConfigurationManager.ConnectionStrings[
                "RecruitVietnamDbConnectionString"
            ].ConnectionString;
        sqlDataSource.SelectCommandType =
            SqlDataSourceCommandType.Text;
        sqlDataSource.SelectCommand = "select " +
            "RecruiterID,Email,CompanyName, " +
            "+ ContactName,Address,Activate from tblRCAccounts";
        gridView.DataSource = sqlDataSource;
        gridView.DataBind();
    }
}
```

```

    }

    protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
    {
        Response.Write(e.RowIndex.ToString());
    }

```

Kết quả thực thi:

Khi click nút Delete tại một dòng nào đó trên GridView sẽ gây ra sự kiện OnRowDeleting="GridView1_RowDeleting" và gọi phương thức sau:

```

protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    Response.Write(e.RowIndex.ToString());
}

```

Email	Ten Cong Ty	Lien He	
ngocanh@MadridHome.com	Viet Employment	Martin Sommer	<input type="button" value="Delete"/>
ngocanh@marseille.com	Bon app'	Laurence Lebihan	<input type="button" value="Delete"/>
admin@Tsawassen.com	Bottom-Dollar Markets	Elizabeth Lincoln	<input type="button" value="Delete"/>
BuenosAires_admin@BuenosAires.com	Cactus Comidas para llevar	Patricia Simpson	<input type="button" value="Delete"/>
Bern_admin@Bern.com	Chop-suey Chinese	Yang Wang	<input type="button" value="Delete"/>

Xóa mẫu tin đầu khi click nút Delete tại dòng đầu tiên trên GridView.
Kết quả như sau:

Email	Ten Cong Ty	Lien He	
ngocanh@MadridHome.com	Viet Employment	Martin Sommer	<input type="button" value="Delete"/>
ngocanh@marseille.com	Bon app'	Laurence Lebihan	<input type="button" value="Delete"/>
admin@Tsawassen.com	Bottom-Dollar Markets	Elizabeth Lincoln	<input type="button" value="Delete"/>
BuenosAires_admin@BuenosAires.com	Cactus Comidas para llevar	Patricia Simpson	<input type="button" value="Delete"/>
Bern_admin@Bern.com	Chop-suey Chinese	Yang Wang	<input type="button" value="Delete"/>

9.1.7.4. Tạo cột ImageField

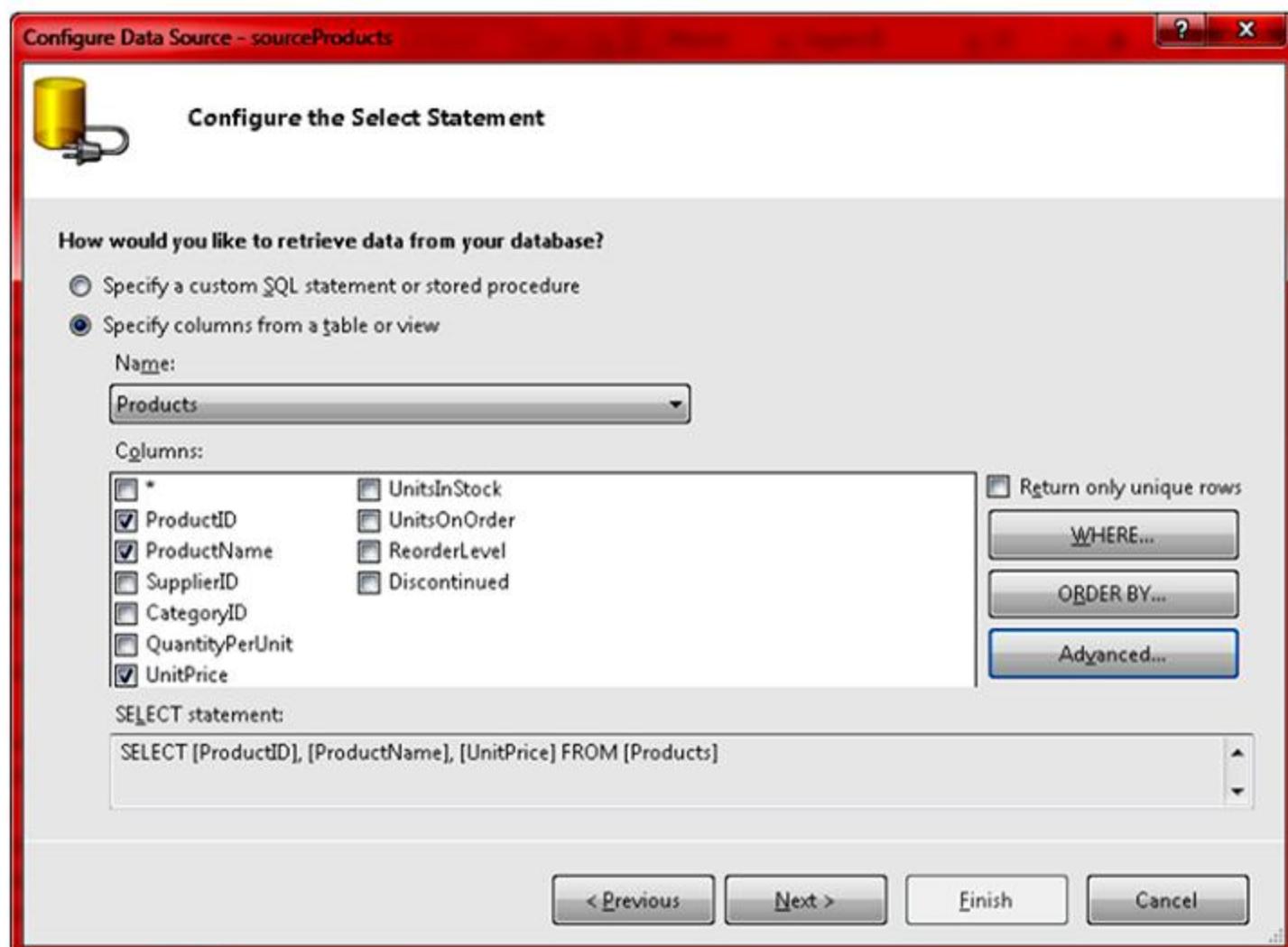
Tương tự như cột HyperLink, GridView cũng có một cột chuyên để hiển thị hình ảnh (ImageField) nếu trường dữ liệu gắn với nó chứa đường dẫn đến ảnh nằm trong ứng dụng.

Để tạo cột cho phép hiển thị Image, dùng thẻ `<asp:ImageField DataImageUrlField.../>`

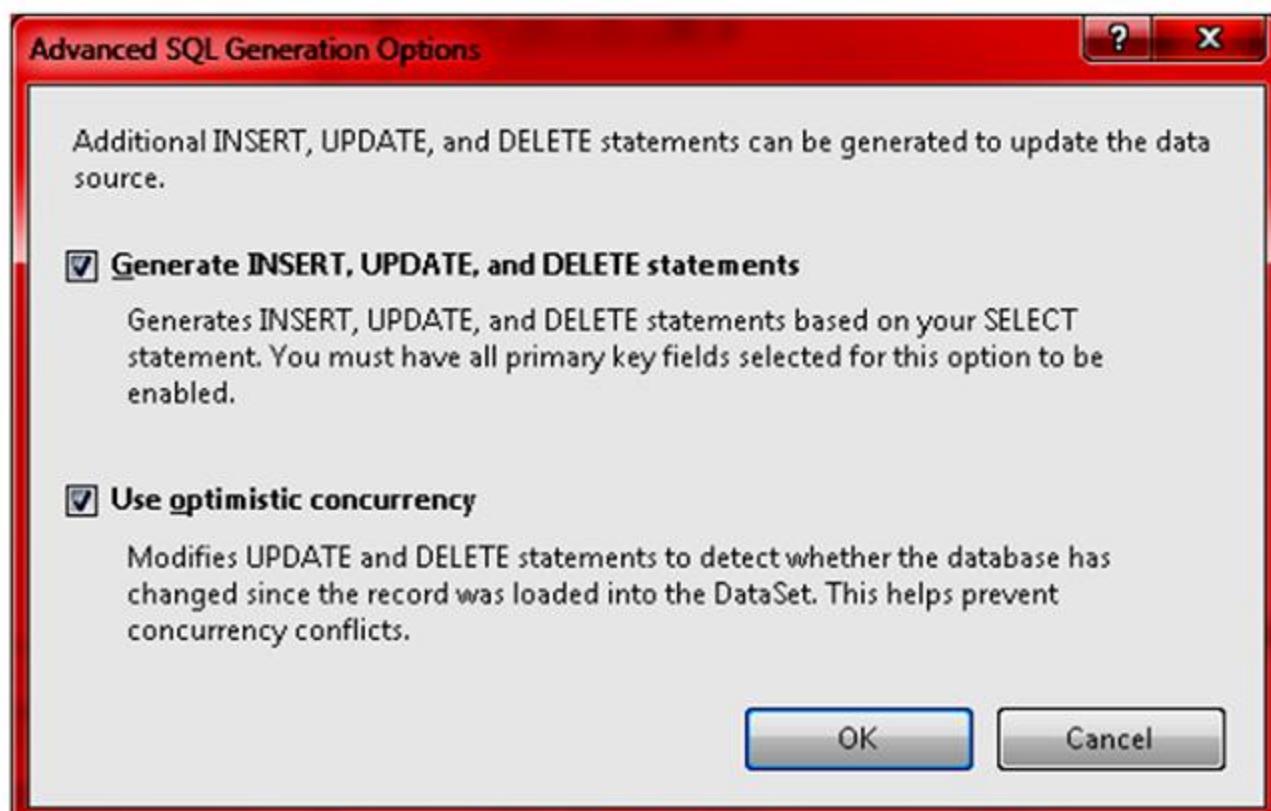
9.1.8. Tạo và xử lý các cột Select, Edit, Delete, Update...

Chú ý

Để thực hiện việc thêm các nút [Select] [Edit] [Delete] trong GridView, khi tạo điều khiển dữ liệu SqlDataSource phải chọn mục Advanced...



và click chọn mục **[X] Generate INSERT, UPDATE and DELETE statements.** Click OK để xác nhận.



Phần mã lệnh của điều khiển SqlDataSource sẽ có thêm phần khai báo sau:

```
<asp:SqlDataSource id="sourceProducts" runat="server"

SelectCommand="SELECT [ProductID], [ProductName], [UnitPrice] FROM
[Products]"
ConnectionString="<%$ ConnectionStrings:NorthwindConnectionString %>"

UpdateCommand="UPDATE [Products] SET [ProductName] =
@ProductName, [UnitPrice] = @UnitPrice WHERE [ProductID] =
@original_ProductID AND [ProductName] = @original_ProductName AND
(([UnitPrice] = @original_UnitPrice) OR ([UnitPrice] IS NULL AND
@original_UnitPrice IS NULL))"

ConflictDetection="CompareAllValues"

DeleteCommand="DELETE FROM [Products] WHERE [ProductID] =
@original_ProductID AND [ProductName] = @original_ProductName AND
(([UnitPrice] = @original_UnitPrice) OR ([UnitPrice] IS NULL AND
@original_UnitPrice IS NULL))"

InsertCommand="INSERT INTO [Products] ([ProductName], [UnitPrice])
VALUES (@ProductName, @UnitPrice)"
OldValuesParameterFormatString="original_{0}""

>
<DeleteParameters>
    <asp:Parameter Name="original_ProductID" Type="Int32" />
    <asp:Parameter Name="original_ProductName" Type="String" />
    <asp:Parameter Name="original_UnitPrice" Type="Decimal" />
</DeleteParameters>
<UpdateParameters>
    <asp:Parameter Name="ProductName" Type="String" />
    <asp:Parameter Name="UnitPrice" Type="Decimal" />
    <asp:Parameter Name="original_ProductID" Type="Int32" />
    <asp:Parameter Name="original_ProductName" Type="String" />
    <asp:Parameter Name="original_UnitPrice" Type="Decimal" />
</UpdateParameters>
<InsertParameters>
    <asp:Parameter Name="ProductName" Type="String" />
    <asp:Parameter Name="UnitPrice" Type="Decimal" />
```

```
</InsertParameters>  
</asp:SqlDataSource>
```

9.1.8.1. Thêm cột Select, Edit - Update, Delete

GridView không chỉ hiển thị được các bảng dữ liệu mà còn hỗ trợ rất tốt trong việc chỉnh sửa và xóa dữ liệu. Đặc biệt khi nguồn dữ liệu là SqlDataSource thì việc sửa và xóa hoàn toàn tự động, không cần phải viết bất kỳ dòng code nào. Để bật tính năng này, cần bổ sung thêm thuộc tính vào GridView với giá trị là **true** cho **AutoGenerateSelectColum**, **AutoGenerateEditColum**, **AutoGenerateDeleteColum**.

Thí dụ:

Tạo trang ASP.NET sử dụng GridView có cột Select.

Thiết kế giao diện trang GridViewSelect.aspx có cột Select
<asp:CommandField ShowSelectButton="True" />

The screenshot shows a web page with two tables generated by a SqlDataSource. The first table, titled "Categories", has columns for CategoryID, CategoryName, and Description. It contains five rows, each with a "Select" link in the first column. The second table, titled "Products in this category:", has columns for ProductID, ProductName, and UnitPrice. It also contains five rows with sample data. Both tables are styled with alternating row colors.

	CategoryID	CategoryName	Description
Select	0	abc	abc
Select	1	abc	abc
Select	2	abc	abc
Select	3	abc	abc
Select	4	abc	abc

	ProductID	ProductName	UnitPrice
	0	abc	0
	1	abc	0.1
	2	abc	0.2
	3	abc	0.3
	4	abc	0.4

Phần mã lệnh thiết kế trang

```

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            Categories:<br />
            <asp:SqlDataSource ID="sourceCategories" runat="server"
ConnectionString="<%$ ConnectionStrings:NorthwindConnectionString %>">
                SelectCommand="SELECT * FROM [Categories]"</asp:SqlDataSource>
            <asp:GridView ID="gridCategories" runat="server"
                DataSourceID="sourceCategories" BackColor="White"
                BorderColor="#CC9966"
                BorderStyle="None" BorderWidth="1px" CellPadding="4"
                DataKeyNames="CategoryID" AutoGenerateColumns="False"
                >
                <Columns>
                    <asp:CommandField ShowSelectButton="True" />>
                    <asp:BoundField DataField="CategoryID" HeaderText="CategoryID"
                        InsertVisible="False" ReadOnly="True" SortExpression="CategoryID" />
                    <asp:BoundField DataField="CategoryName" HeaderText="CategoryName"
                        SortExpression="CategoryName" />
                    <asp:BoundField DataField="Description" HeaderText="Description"
                        SortExpression="Description" />
                </Columns>
                <RowStyle BackColor="White" ForeColor="#330099" />
            
```

```

<SelectedRowStyle BackColor="#FFCC66" Font-Bold="True"
ForeColor="#663399" />

<HeaderStyle BackColor="#990000" Font-Bold="True" ForeColor="#FFFFCC"
/>

</asp:GridView>
<br />

Products in this category:<br />

<asp:SqlDataSource ID="sourceProducts" runat="server" ConnectionString="<%$ConnectionStringNames:NorthwindConnectionString %>">

    SelectCommand="SELECT ProductID, ProductName, UnitPrice FROM
Products WHERE (CategoryID = @CategoryID)">

    <SelectParameters>
        <asp:ControlParameter Name="CategoryID" ControlID="gridCategories"
            PropertyName="SelectedDataKey.Value" DefaultValue="1" />
    </SelectParameters>
</asp:SqlDataSource>

<asp:GridView ID="gridProducts" runat="server"
DataSourceID="sourceProducts"

    BackColor="White" BorderColor="#CC9966" BorderStyle="None"
BorderWidth="1px"

    CellPadding="4" AutoGenerateColumns="False" DataKeyNames="ProductID"
    >

    <RowStyle BackColor="White" ForeColor="#330099" />
    <Columns>

        <asp:BoundField DataField="ProductID" HeaderText="ProductID"
            InsertVisible="False" ReadOnly="True" SortExpression="ProductID" />
        <asp:BoundField DataField="ProductName" HeaderText="ProductName"
            SortExpression="ProductName" />
        <asp:BoundField DataField="UnitPrice" HeaderText="UnitPrice"
            SortExpression="UnitPrice" />
    </Columns>

    <SelectedRowStyle BackColor="#FFCC66" Font-Bold="True"
ForeColor="#663399" />

    <HeaderStyle BackColor="#990000" Font-Bold="True" ForeColor="#FFFFCC"
/>

```

```

</asp:GridView>

</div>
</form>
</body>
</html>

```

Phần mã lệnh thực thi trang GridViewSelect.aspx.cs:

```

public partial class GridViewSelect: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e) { }
}

```

Kết quả thực hiện:

Khi click nút [Select] tại một dòng trên GridView, dựa theo mã CategoryID sẽ lọc dữ liệu của GridView dưới các Product có CategoryID này.

Categories:			
	CategoryID	CategoryName	Description
Select	1	Beverages	Soft drinks, coffees, teas, beers, and ales
Select	2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
Select	3	Confections	Desserts, candies, and sweet breads
Select	4	Dairy Products	Cheeses
Select	5	Grains/Cereals	Breads, crackers, pasta, and cereal
Select	6	Meat/Poultry	Prepared meats
Select	7	Produce	Dried fruit and bean curd
Select	8	Seafood	Seaweed and fish

Products in this category:		
ProductID	ProductName	UnitPrice
4	Chef Anton's Cajun Seasoning	22.0000
5	Chef Anton's Gumbo Mix	21.3500
6	Grandma's Boysenberry Spread	25.0000
8	Northwoods Cranberry Sauce	40.0000
15	Genen Shouyu	15.5000
44	Gula Malacca	19.4500
61	Sirop d'éable	28.5000
63	Vegie-spread	43.9000
65	Louisiana Fiery Hot Pepper Sauce	21.0500
66	Louisiana Hot Spiced Okra	17.0000
77	Original Frankfurter grüne Soße	13.0000

9.1.8.2. Cập nhật dữ liệu

Tạo trang ASP.NET sử dụng GridView có cột Edit với thẻ khai báo sau trong GridView

```
<asp:CommandField ShowEditButton="True" />
```

Thiết kế giao diện trang GridViewEdit.aspx:

ProductID	ProductName	UnitPrice	
0	abc	0	Edit
1	abc	0.1	Edit
2	abc	0.2	Edit
3	abc	0.3	Edit
4	abc	0.4	Edit

SqlDataSource - sourceProducts

Phần mã lệnh thiết kế trang GridViewEdit.aspx:

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:GridView ID="GridView2" runat="server" DataSourceID="sourceProducts"
AutoGenerateColumns="False" DataKeyNames="ProductID">
                <Columns>
                    <asp:BoundField DataField="ProductID" HeaderText="ProductID"
ReadOnly="True"
InsertVisible="False" SortExpression="ProductID" />
                </Columns>
            </asp:GridView>
        </div>
    </form>
</body>
</html>
```

```

<asp:BoundField DataField="ProductName" HeaderText="ProductName"
    SortExpression="ProductName"/>
<asp:BoundField DataField="UnitPrice" HeaderText="UnitPrice"
    SortExpression="UnitPrice" />
<asp:CommandField ShowEditButton="True" />
</Columns>
</asp:GridView>
<asp:SqlDataSource id="sourceProducts" runat="server"
SelectCommand="SELECT [ProductID], [ProductName], [UnitPrice] FROM
[Products]" ConnectionString="<%$  

ConnectionStrings:NorthwindConnectionString %>"
    UpdateCommand="UPDATE [Products] SET [ProductName] =
    @ProductName, [UnitPrice] = @UnitPrice WHERE [ProductID] =
    @original_ProductID AND [ProductName] = @original_ProductName AND
    ([UnitPrice] = @original_UnitPrice) OR ([UnitPrice] IS NULL AND
    @original_UnitPrice IS NULL)"
```

ConflictDetection="CompareAllValues"

```

    DeleteCommand="DELETE FROM [Products] WHERE [ProductID] =
    @original_ProductID AND [ProductName] = @original_ProductName AND
    ([UnitPrice] = @original_UnitPrice) OR ([UnitPrice] IS NULL AND
    @original_UnitPrice IS NULL))"
```

```

    InsertCommand="INSERT INTO [Products] ([ProductName], [UnitPrice])
VALUES (@ProductName, @UnitPrice)"
OldValuesParameterFormatString="original_{0}"
```

>

```

<DeleteParameters>
    <asp:Parameter Name="original_ProductID" Type="Int32" />
    <asp:Parameter Name="original_ProductName" Type="String" />
    <asp:Parameter Name="original_UnitPrice" Type="Decimal" />
</DeleteParameters>
<UpdateParameters>
    <asp:Parameter Name="ProductName" Type="String" />
    <asp:Parameter Name="UnitPrice" Type="Decimal" />
    <asp:Parameter Name="original_ProductID" Type="Int32" />
    <asp:Parameter Name="original_ProductName" Type="String" />
    <asp:Parameter Name="original_UnitPrice" Type="Decimal" />
</UpdateParameters>
```

```

<InsertParameters>
    <asp:Parameter Name="ProductName" Type="String" />
    <asp:Parameter Name="UnitPrice" Type="Decimal" />
</InsertParameters>
</asp:SqlDataSource>
</div>
</form>
</body>
</html>

```

Phần mã lệnh thực thi trang GridViewEdit.aspx.cs:

```

public partial class GridViewEdit : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e) { }
}

```

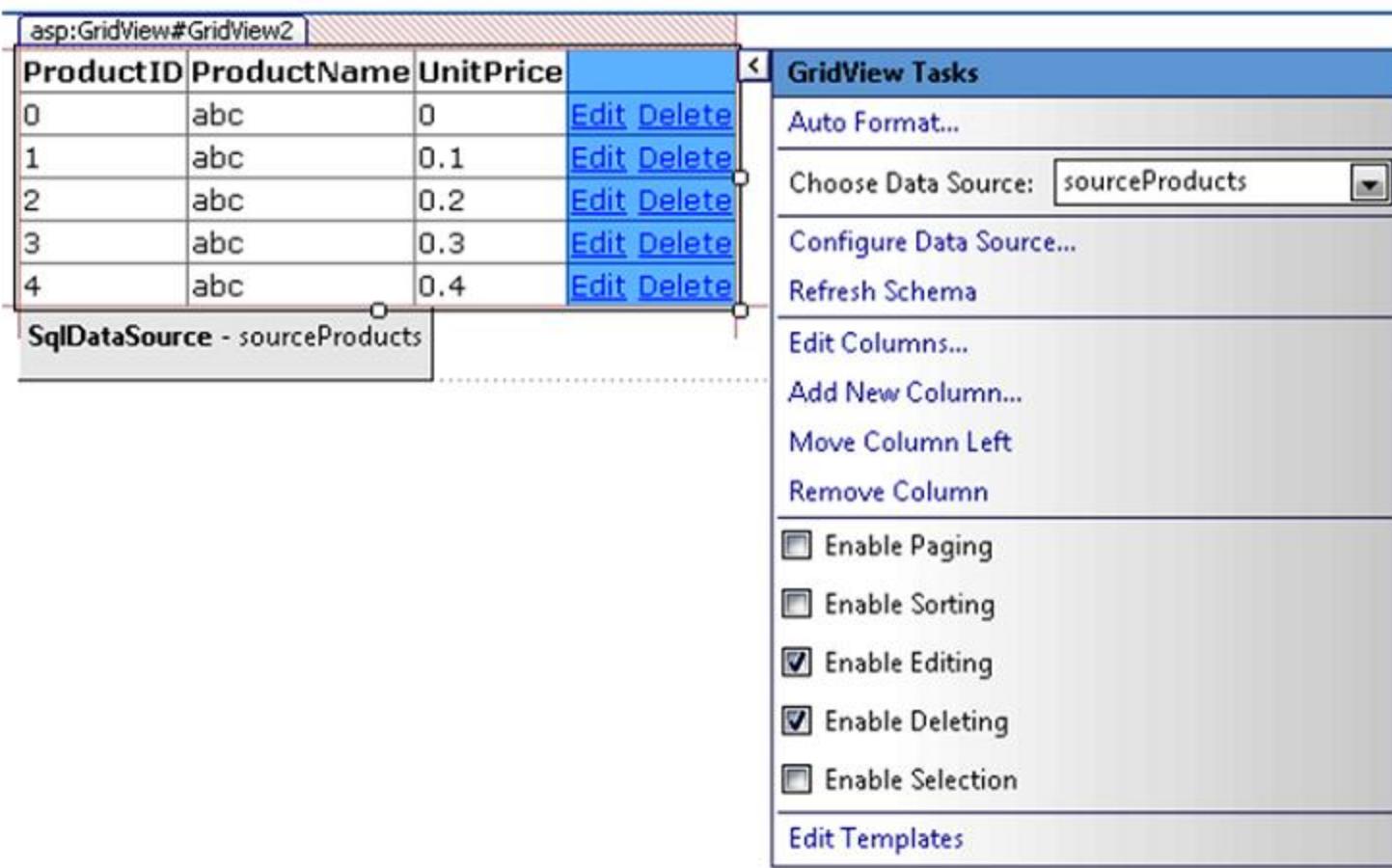
Kết quả thực thi:

Click nút [**Edit**] trên một dòng của GridView, sẽ cho phép bạn cập nhật thông tin các vùng thuộc tính (ngoại trừ thuộc tính khóa). Click [**Update**] để cập nhật, [**Cancel**] để hủy thao tác.

ProductID	ProductName	UnitPrice	
1	Lo	18.0000	Update Cancel
2	Chang	19.0000	Edit
3	Aniseed Syrup	10.0000	Edit
4	Chef Anton's Cajun Seasoning	22.0000	Edit
5	Chef Anton's Gumbo Mix	21.3500	Edit
6	Grandma's Boysenberry Spread	25.0000	Edit
7	Uncle Bob's Organic Dried Pears	30.0000	Edit

9.1.8.3. Xóa dữ liệu

Quay lại thí dụ trên và thiết kế lại giao diện trang, click smart tag và click chọn mục [x]Enable Deleting. Xuất hiện thêm nút [**Delete**]



Phần mã lệnh thiết kế trang GridViewEdit.aspx:

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:GridView ID="GridView2" runat="server" DataSourceID="sourceProducts"
                AutoGenerateColumns="False" DataKeyNames="ProductID">
                <Columns>
                    <asp:BoundField DataField="ProductID" HeaderText="ProductID"
                        ReadOnly="True"
                        InsertVisible="False" SortExpression="ProductID" />
                </Columns>
            </asp:GridView>
        </div>
    </form>
</body>

```

```

<asp:BoundField DataField="ProductName" HeaderText="ProductName"
    SortExpression="ProductName"/>
<asp:BoundField DataField="UnitPrice" HeaderText="UnitPrice"
    SortExpression="UnitPrice" />
<asp:CommandField ShowEditButton="True" ShowDeleteButton="True" />
</Columns>
</asp:GridView>
<asp:SqlDataSource id="sourceProducts" runat="server"
SelectCommand="SELECT [ProductID], [ProductName], [UnitPrice] FROM
[Products]" ConnectionString="<%$  

ConnectionStrings:NorthwindConnectionString %>"

    UpdateCommand="UPDATE [Products] SET [ProductName] =
    @ProductName, [UnitPrice] = @UnitPrice WHERE [ProductID] =
    @original_ProductID AND [ProductName] = @original_ProductName AND
    (([UnitPrice] = @original_UnitPrice) OR ([UnitPrice] IS NULL AND
    @original_UnitPrice IS NULL))"

    ConflictDetection="CompareAllValues"

    DeleteCommand="DELETE FROM [Products] WHERE [ProductID] =
    @original_ProductID AND [ProductName] = @original_ProductName AND
    (([UnitPrice] = @original_UnitPrice) OR ([UnitPrice] IS NULL AND
    @original_UnitPrice IS NULL))"

    InsertCommand="INSERT INTO [Products] ([ProductName], [UnitPrice])
VALUES (@ProductName, @UnitPrice)"
OldValuesParameterFormatString="original_{0}">

<DeleteParameters>
    <asp:Parameter Name="original_ProductID" Type="Int32" />
    <asp:Parameter Name="original_ProductName" Type="String" />
    <asp:Parameter Name="original_UnitPrice" Type="Decimal" />
</DeleteParameters>
<UpdateParameters>
    <asp:Parameter Name="ProductName" Type="String" />
    <asp:Parameter Name="UnitPrice" Type="Decimal" />
    <asp:Parameter Name="original_ProductID" Type="Int32" />
    <asp:Parameter Name="original_ProductName" Type="String" />
    <asp:Parameter Name="original_UnitPrice" Type="Decimal" />

```

```

</UpdateParameters>

<InsertParameters>
    <asp:Parameter Name="ProductName" Type="String" />
    <asp:Parameter Name="UnitPrice" Type="Decimal" />
</InsertParameters>
</asp:SqlDataSource>
</div>
</form>
</body>
</html>

```

Phần mã lệnh thực thi trang:

```

public partial class GridViewEdit: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e) { }
}

```

Kết quả thực thi

ProductID	ProductName	UnitPrice	
1	Lo	18.0000	Edit Delete
2	Chang	19.0000	Edit Delete
3	Aniseed Syrup	10.0000	Edit Delete
4	Chef Anton's Cajun Seasoning	22.0000	Edit Delete
5	Chef Anton's Gumbo Mix	21.3500	Edit Delete

Click nút [Delete] tại một dòng trên GridView, mẫu tin tại dòng sẽ bị xóa.

9.2. ĐỐI TƯỢNG DỮ LIỆU DETAILSVIEW

Dùng để trình bày dữ liệu là một mẫu tin tại một thời điểm. Chúng ta sẽ xem thí dụ dưới đây qua đó nắm vững cách sử dụng đối tượng này.

Thí dụ

Thiết kế giao diện trang DetailsViewTest.aspx:

ProductID	Databound
ProductName	Databound
UnitPrice	Databound
Edit Delete New	
1 2	
SqlDataSource - SqlDataSource1	

Phần mã lệnh thiết kế trang DetailsViewTest.aspx:

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:DetailsView ID="DetailsView1" runat="server" AllowPaging="True"
CellPadding="4"
                DataSourceID="SqlDataSource1" ForeColor="#333333" GridLines="None"
Height="50px"
                Width="336px" AutoGenerateRows="False" DataKeyNames="ProductID"
AutoGenerateEditButton="True" AutoGenerateInsertButton="True"
AutoGenerateDeleteButton="true" OnDataBound="DetailsView1_DataBound">
                <FooterStyle BackColor="#990000" Font-Bold="True" ForeColor="White" />
                <CommandRowStyle BackColor="#FFFFC0" Font-Bold="True" />
                <RowStyle BackColor="#FFFBD6" ForeColor="#333333" />
                <PagerStyle BackColor="#FFCC66" ForeColor="#333333"
HorizontalAlign="Center" />
            </asp:DetailsView>
        </div>
    </form>
</body>
</html>
```

```

<FieldHeaderStyle BackColor="#FFFF99" Font-Bold="True" />
<HeaderStyle BackColor="#990000" Font-Bold="True" ForeColor="White" />
<AlternatingRowStyle BackColor="White" />
<Fields>
    <asp:BoundField DataField="ProductID" HeaderText="ProductID"
InsertVisible="False"
        ReadOnly="True" SortExpression="ProductID" />
    <asp:BoundField DataField="ProductName" HeaderText="ProductName"
SortExpression="ProductName" />
    <asp:BoundField DataField="UnitPrice" HeaderText="UnitPrice"
SortExpression="UnitPrice" />
</Fields>
</asp:DetailsView>

</div>

<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$ ConnectionStrings:Northwind %>" SelectCommand="SELECT ProductID, ProductName, UnitPrice FROM Products" UpdateCommand="UPDATE Products SET ProductName=@ProductName, UnitPrice=@UnitPrice WHERE ProductID=@ProductID" InsertCommand="INSERT Products (ProductName, UnitPrice) VALUES (@ProductName, @UnitPrice)" DeleteCommand="DELETE Products WHERE ProductID=@ProductID" OnInserted="SqlDataSource1_Inserted"></asp:SqlDataSource>
</form>
</body>
</html>

```

Các thuộc tính quan trọng trong thẻ **<asp:DetailsView...>**:

AutoGenerateEditButton="True" // nút [Edit]
 AutoGenerateInsertButton="True" // nút [New]
 AutoGenerateDeleteButton="true" // nút [Delete]

Sự kiện của thẻ **<asp:DetailsView...>**:

```
OnDataBound="DetailsView1_DataBound"
```

```
protected void DetailsView1_DataBound(object sender, EventArgs e)
{
    if (insertComplete)
    {
        // Show the last record (the newly added one).
        DetailsView1PageIndex = DetailsView1PageCount - 1;
    }
}
```

Phần mã lệnh thực thi trang DetailsViewTest.aspx.cs:

```
public partial class DetailsView : System.Web.UI.Page
{
    private bool insertComplete = false;

    protected void SqlDataSource1_Inserted(object sender,
    SqlDataSourceStatusEventArgs e)
    {
        // Flag that a new record is inserted, which we'll show when the grid is bound.
        if (e.AffectedRows > 0) insertComplete = true;

        // You could also get output parameters at this point from e.Command.
        // For example, if you called a stored procedure that returns the newly
        // generated ProductID value, you could get it here.

    }

    protected void DetailsView1_DataBound(object sender, EventArgs e)
    {
        if (insertComplete)
        {
            // Show the last record (the newly added one).
            DetailsView1PageIndex = DetailsView1PageCount - 1;
        }
    }
}
```

Kết quả thực thi:

ProductID	1
ProductName	Lo
UnitPrice	18.0000
<u>Edit</u> <u>Delete</u> <u>New</u>	
1 2 3 4 5 6 7 8 9 10 ...	

Click nút [Edit], sau đó click [] cập nhật hoặc [] để hủy

ProductID	1
ProductName	<input type="text" value="Lo"/>
UnitPrice	18.0000
<u>Update</u> <u>Cancel</u>	
1 2 3 4 5 6 7 8 9 10 ...	

Click nút [New], nhập thông tin sau đó click nút [Insert] để chèn mẫu tin mới hoặc [Cancel] để hủy thao tác.

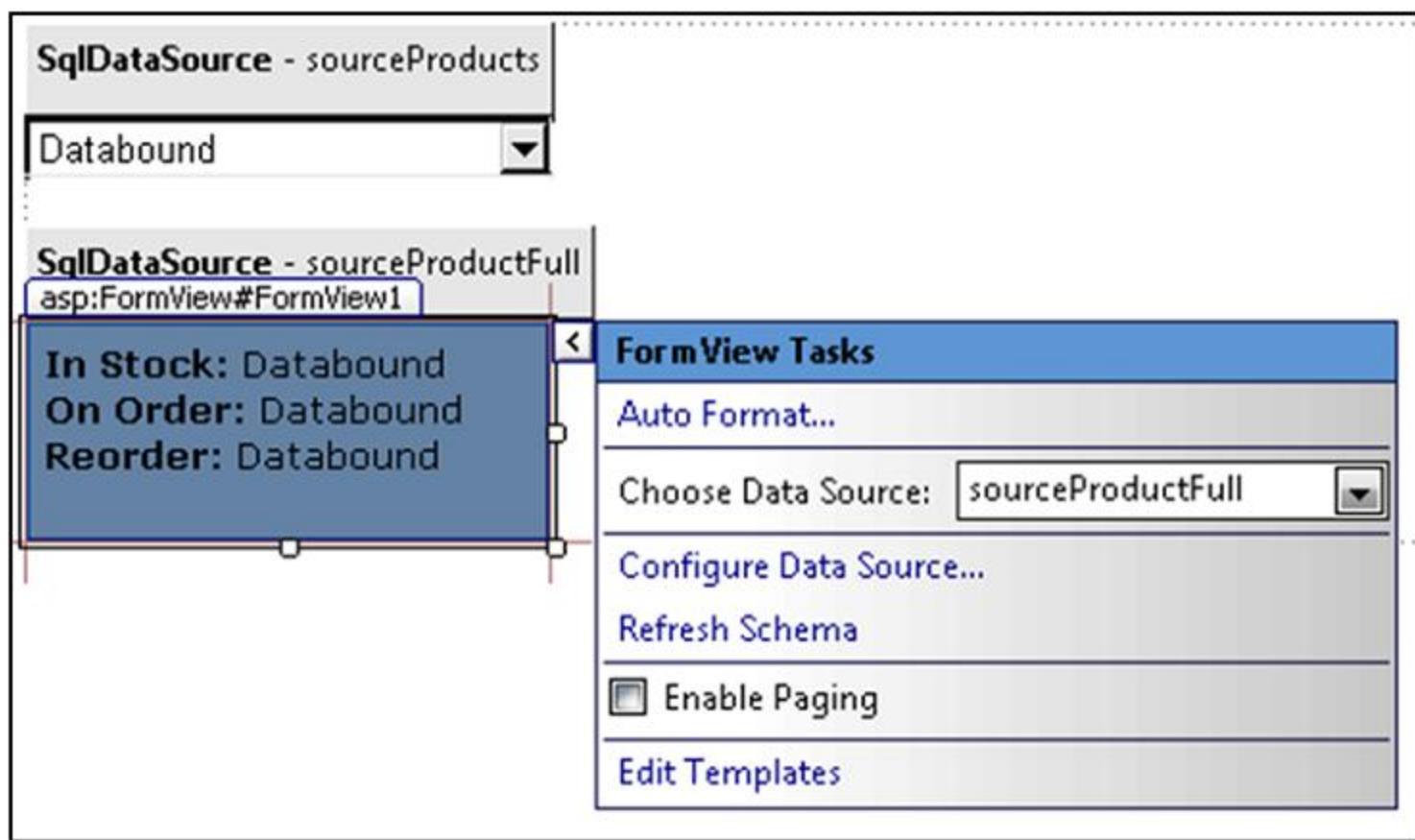
ProductName	<input type="text"/>
UnitPrice	<input type="text"/>
<u>Insert</u> <u>Cancel</u>	

9.3. ĐỐI TƯỢNG DỮ LIỆU FORMVIEW

Dùng để trình bày dữ liệu là một mẫu tin tại một thời điểm. Tương tự như DetailsView nhưng cho phép trình bày dữ liệu theo các mẫu tùy chọn (custom templates). Chúng ta sẽ xem thí dụ dưới đây qua đó nắm vững cách sử dụng đối tượng này.

Thí dụ

Thiết kế giao diện trang



Phần mã lệnh thiết kế trang FormViewTest.aspx:

```

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

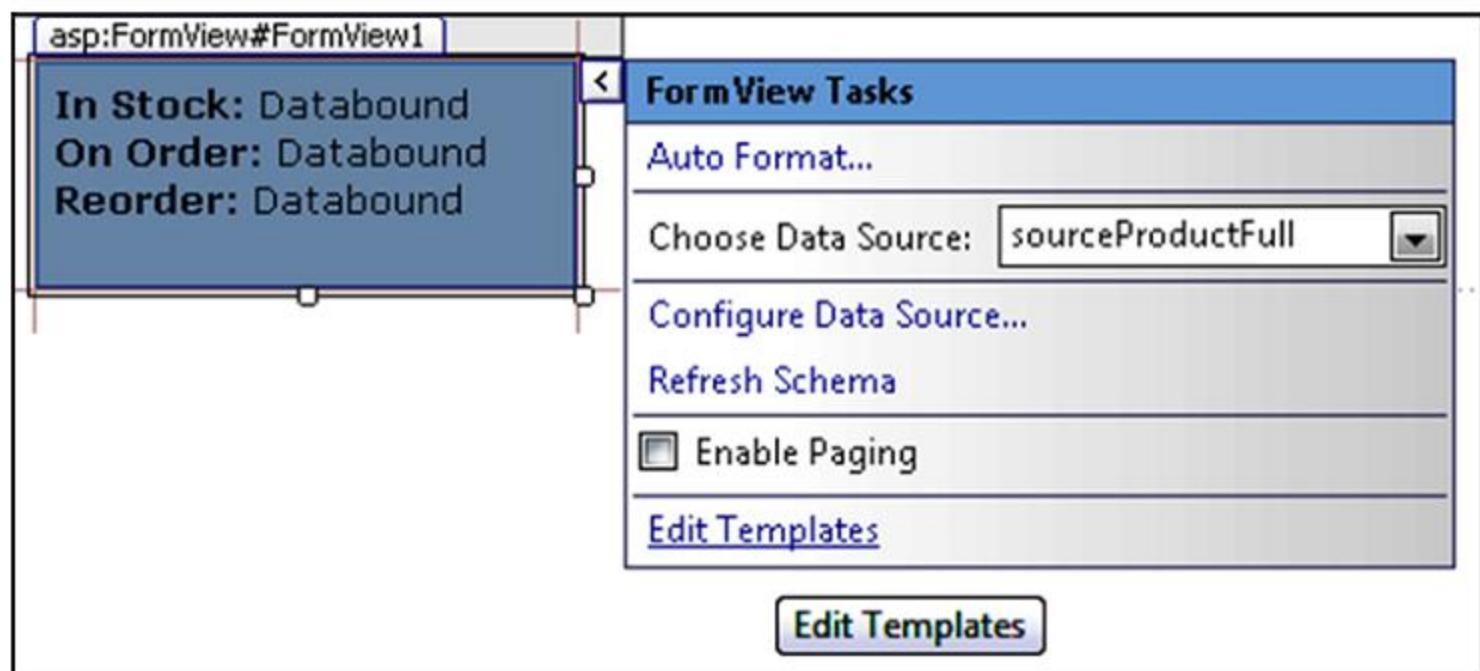
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:SqlDataSource ID="sourceProducts" runat="server" ConnectionString="<%$.ConnectionStrings:Northwind %>">
                SelectCommand="SELECT * FROM Products"></asp:SqlDataSource>
            <br/>
            <asp:DropDownList ID="lstProducts" runat="server" AutoPostBack="True"
                DataSourceID="sourceProducts"
                DataTextField="ProductName" DataValueField="ProductID" Width="184px">
        </div>
    </form>
</body>

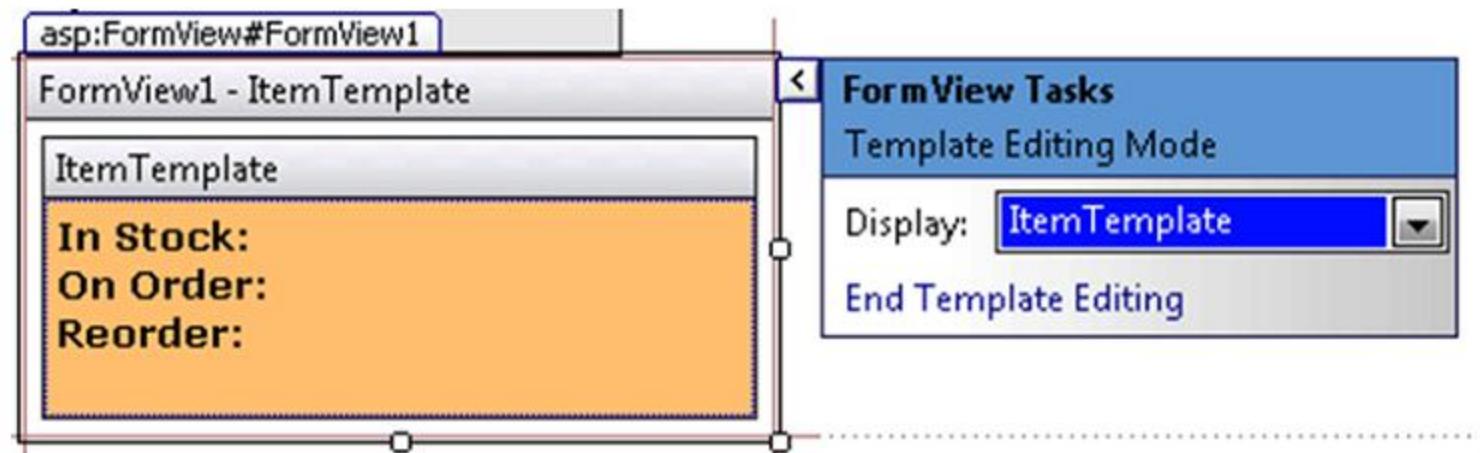
```

```
</asp:DropDownList>
<br />
<br />
<asp:SqlDataSource ID="sourceProductFull" runat="server"
ConnectionString="<%$ ConnectionStrings:Northwind %>">
    SelectCommand="SELECT * FROM Products WHERE
ProductID=@ProductID">
    <SelectParameters>
        <asp:ControlParameter Name="ProductID" ControlID="lstProducts"
PropertyName="SelectedValue" />
    </SelectParameters>
</asp:SqlDataSource>
<asp:FormView ID="FormView1" runat="server"
DataSourceID="sourceProductFull"
Width="184px" BackColor="#FFE0C0" BorderStyle="Solid"
BorderWidth="2px" CellPadding="5">
    <ItemTemplate>
        <b>In Stock:</b>
        <%# Eval("UnitsInStock") %>
        <br />
        <b>On Order:</b>
        <%# Eval("UnitsOnOrder") %>
        <br />
        <b>Reorder:</b>
        <%# Eval("ReorderLevel") %>
        <br />
        <br />
    </ItemTemplate>
</asp:FormView>
</div>
</form>
</body>
</html>
```

Để thiết kế mẫu tùy chọn (custom templates), chọn mục Edit Templates trong FormView Tasks:



Nhập nội dung như hình dưới vào khung Item Template



Trong FormView sẽ có mẫu tùy chọn (custom template) như sau, nhập vào các kết nối dữ liệu lấy giá trị các vùng trong bảng Products của CSDL Northwind:

```
<ItemTemplate>
    <b>In Stock:</b>
    <%# Eval("UnitsInStock") %>
    <br />
    <b>On Order:</b>
    <%# Eval("UnitsOnOrder") %>
    <br />
    <b>Reorder:</b>
    <%# Eval("ReorderLevel") %>
    <br />
    <br />
</ItemTemplate>
```

Click nút **[End Template Editing]** để kết thúc việc tạo các custom templates.

Phần mã lệnh thực thi trang FormViewTest.aspx:

```
public partial class FormView: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e) { }
}
```

Kết quả thực thi FormViewTest.aspx.cs:



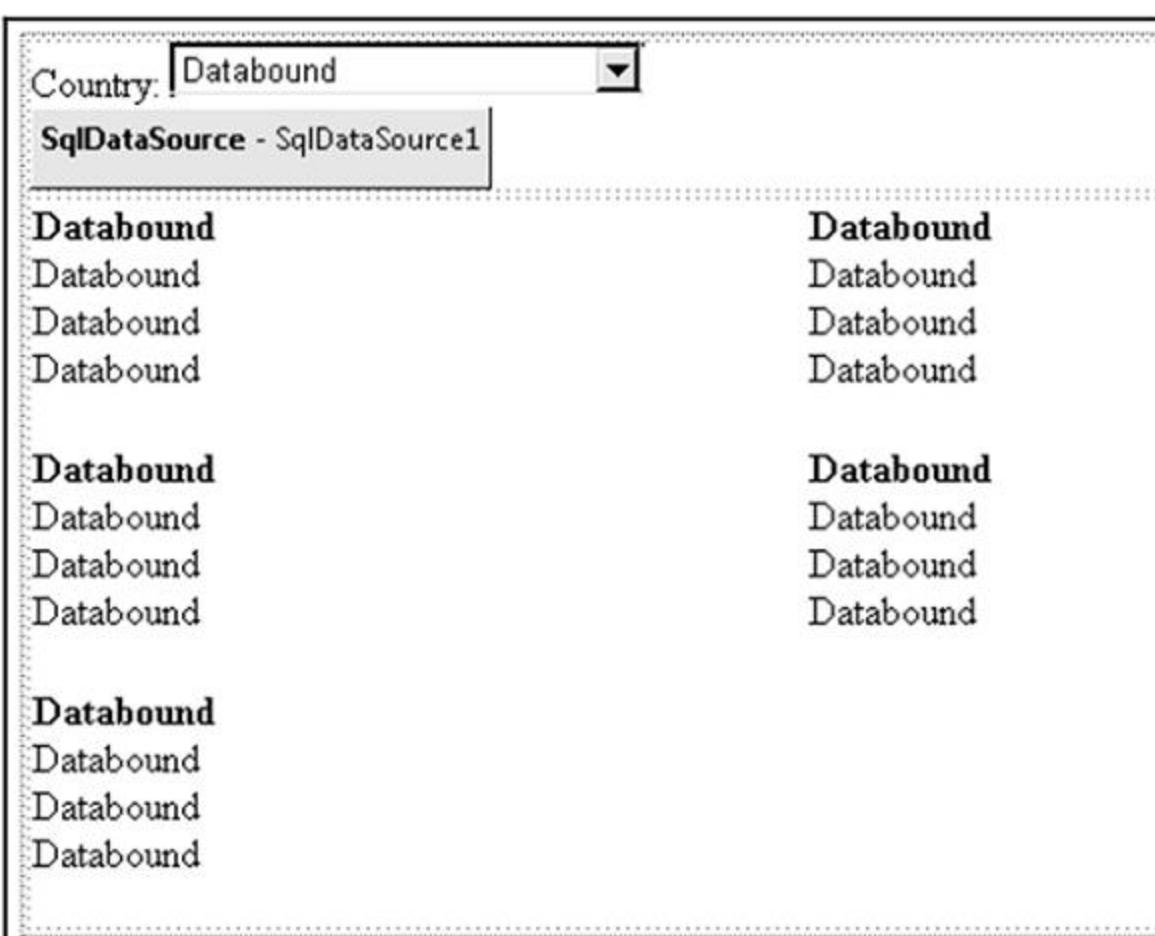
Click chọn một giá trị trong DropDownList, nội dung của FormView sẽ thay đổi theo tương ứng.

9.4. ĐỔI TƯỢNG DỮ LIỆU DATALIST

Dùng để trình bày dữ liệu là một mẫu tin tại một thời điểm. Chúng ta sẽ xem thí dụ dưới đây qua đó nắm vững cách sử dụng đối tượng này.

Thí dụ

Thiết kế giao diện trang



Phần mã lệnh thiết kế trang DataListAndDropDownList.aspx:

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

<!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>
            <div>
                <table width="50%">
                    <tr>
                        <td style="width: 432px; height: 24px">
                            Country:
                            <asp:DropDownList ID="DropDownList1" runat="server"
AutoPostBack="True" DataSourceID="SqlDataSource1"
DataTextField="ProvinceName" DataValueField="ProvinceID"
OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged"
Width="189px">
                                </asp:DropDownList><asp:SqlDataSource ID="SqlDataSource1"
runat="server" ConnectionString="<%$  

ConnectionStrings:RecruitVietnamDbConnectionString %>"  

SelectCommand="SELECT [ProvinceID], [ProvinceName] FROM  

[tblProvinces]"></asp:SqlDataSource>
                        </td>
                    </tr>
                    <tr>
                        <td style="width: 432px">
                            <asp:DataList ID="DataList1" runat="server" RepeatColumns="2"
RepeatDirection="Horizontal"
Width="622px">
                                <ItemTemplate>
                                    <asp:Label ID="Label2" runat="server" Font-Bold="True"
Text='<%# Bind("CompanyName") %>'>
```

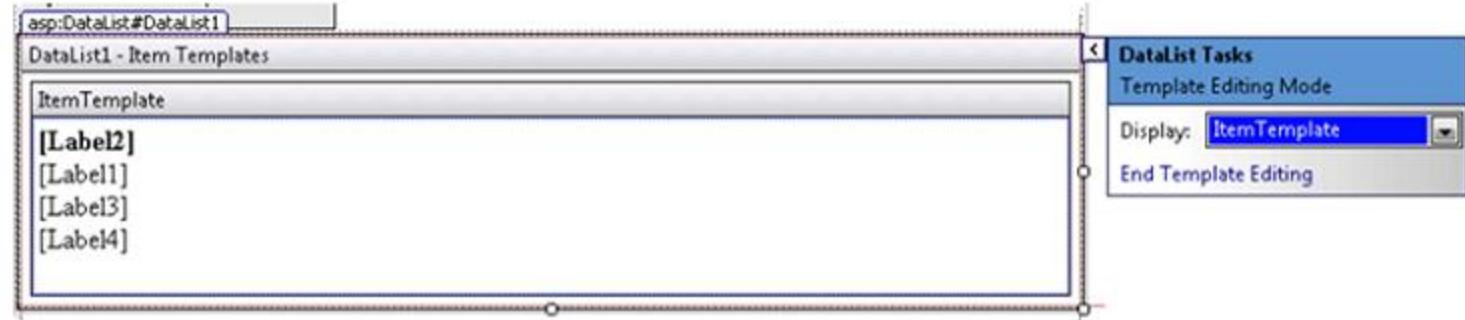
```

        Width="220px"></asp:Label><br />
        <asp:Label ID="Label1" runat="server" Text='<%# Bind("Email") %>' Width="220px"></asp:Label>
        <br />
        <asp:Label ID="Label3" runat="server" Text='<%# Bind("Address") %>' Width="220px"></asp:Label><br />
        <asp:Label ID="Label4" runat="server" Text='<%# Bind("Phone") %>' Width="220px"></asp:Label><br />
        <br />
    </ItemTemplate>
</asp:DataList></td>
</tr>
</table>
 
</div>

</div>
</form>
</body>
</html>

```

Trong DataList trên có tạo custom template



Nội dung trong DataList có thẻ <ItemTemplate> chứa nội dung thiết kế trên.

```

<ItemTemplate>
    <asp:Label ID="Label2" runat="server" Font-Bold="True"
    Text='<%# Bind("CompanyName") %>'>
        Width="220px"></asp:Label><br />
        <asp:Label ID="Label1" runat="server" Text='<%# Bind("Email") %>' Width="220px"></asp:Label>
        <br />
        <asp:Label ID="Label3" runat="server" Text='<%# Bind("Address") %>' Width="220px"></asp:Label><br />
        <asp:Label ID="Label4" runat="server" Text='<%# Bind("Phone") %>' Width="220px"></asp:Label><br />

```

```
<br />  
</ItemTemplate>
```

Phần mã lệnh thực thi trang DataListAndDropDownList.aspx:

```
public partial class DataListAndDropDownList: System.Web.UI.Page  
{  
    protected void Page_Load(object sender, EventArgs e)  
    {  
        if (!IsPostBack)  
            FillRecruiterAccount(this.DataList1, "1");  
    }  
    void FillRecruiterAccount(DataList DataList1, string Province)  
    {  
        try  
        {  
            SqlDataSource sqlDataSource = new SqlDataSource();  
            sqlDataSource.ConnectionString =  
                WebConfigurationManager.ConnectionStrings[  
                    "RecruitVietnamDbConnectionString"]  
                .ConnectionString;  
            sqlDataSource.SelectCommandType =  
                SqlDataSourceCommandType.Text;  
            sqlDataSource.SelectCommand = "select " +  
                "RecruiterID, Email, CompanyName,"  
                + "ContactName, Address, Phone" +  
                " from tblRCAccounts where "  
                +" ProvinceID = '" + Province + "'";  
            DataList1.DataKeyField = "RecruiterID";  
            DataList1.DataSource = sqlDataSource;  
            DataList1.DataBind();  
        }  
        catch (Exception ex) {  
            Response.Write(ex.Message);  
        }  
    }  
    protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)  
    {
```

```
        FillRecruiterAccount(this.DataList1, DropDownList1.SelectedValue);  
    }  
}
```

Kết quả thực thi:

Đầu tiên ta có màn hình thực thi sau:

Country: Ho Chi Minh	
Gourmet Lanchonetes Campinas_admin@Campinas.com Av. Brasil, 442 (11) 555-9482	Great Lakes Food Market Eugene_admin@Eugene.com 2732 Baker Blvd. (503) 555-7555
GROSELLA-Restaurante Caracas_admin@Caracas.com 5 ^a Ave. Los Palos Grandes (2) 283-2951	Al Hanari Carnes RiodeJaneiro_admin@RiodeJaneiro.com Rua do Paço, 67 (21) 555-0091

Sau khi lựa chọn giá trị **[Country: Thua Thien Hue]** trong DropDownList, ta có danh sách mới hiện trong DataList gồm các vùng nội dung định trong phần kết nối dữ liệu trong thẻ **<ItemTemplate>**

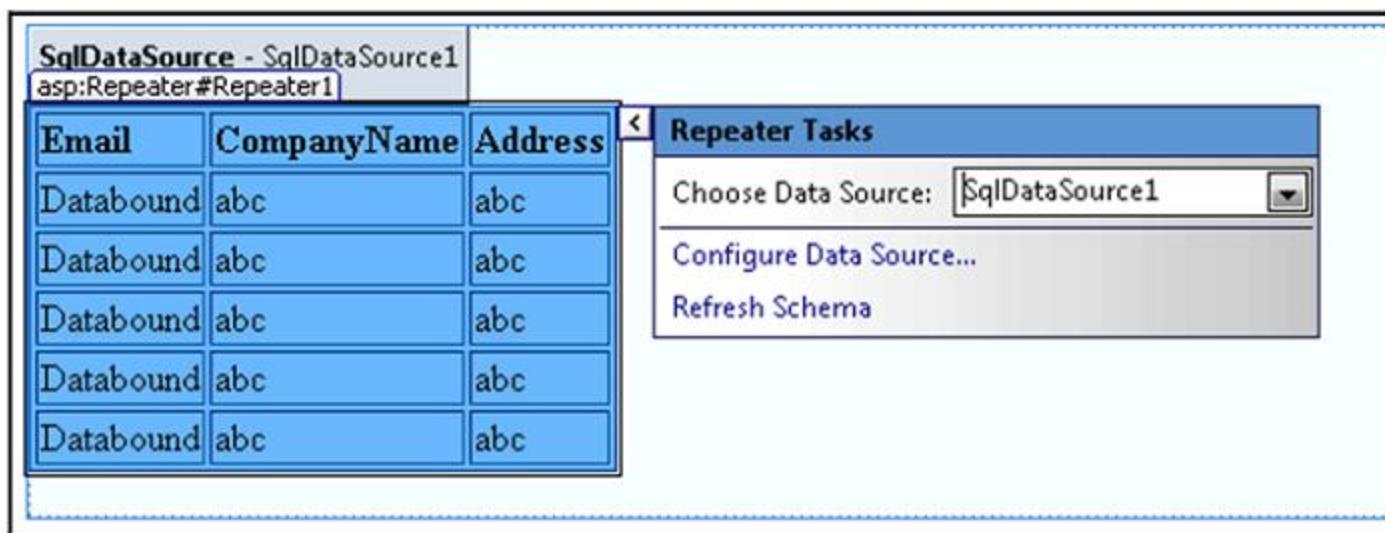
Country: Thua Thien-Hue	
Viet Employment ngocanh@MadridHome.com C/ Araquil, 67 (91) 555 22 82	Bon app' ngochanh@Marseille.com 12, rue des Bouchers 91.24.45.40

9.5. ĐỐI TƯỢNG DỮ LIỆU REPEATER

Dùng để trình bày dữ liệu là một mẫu tin tại một thời điểm. Chúng ta sẽ xem thí dụ dưới đây qua đó nắm vững cách sử dụng đối tượng này.

Thí dụ

Thiết kế giao diện trang



Phần mã lệnh thiết kế trang

```

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

<!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>
            <asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$ ConnectionStrings:RecruitVietnamDbConnectionString
%>">
                SelectCommand="SELECT [RecruiterID], [Email], [CompanyName],
[ContactName], [Address], [Phone] FROM [tblRCAccounts]">
            </asp:SqlDataSource>
        </div>
        <asp:Repeater ID="Repeater1" runat="server" DataSourceID="SqlDataSource1"
DataMember="DefaultView">
            <HeaderTemplate>
                <table border="1">
                    <tr>
                        <td><b>Email</b></td>
                        <td><b>CompanyName</b></td>
                        <td><b>Address</b></td>
                    </tr>
            
```

```

</tr>
</HeaderTemplate>

<ItemTemplate>
<tr>
<td> <%# DataBinder.Eval(Container.DataItem, "Email") %> </td>
<td><asp:Label runat="server" ID="Label1" Text='<%#
Bind("CompanyName") %>' /> </td>
<td> <asp:Label runat="server" ID="Label3" Text='<%# Eval("Address") %>' /></td>
</tr>
</ItemTemplate>

<FooterTemplate>
</table>
</FooterTemplate>

</asp:Repeater>
 
</form>
</body>
</html>

```

Phần mã lệnh thực thi trang

```

public partial class RepeaterPage: System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e) { }
}

```

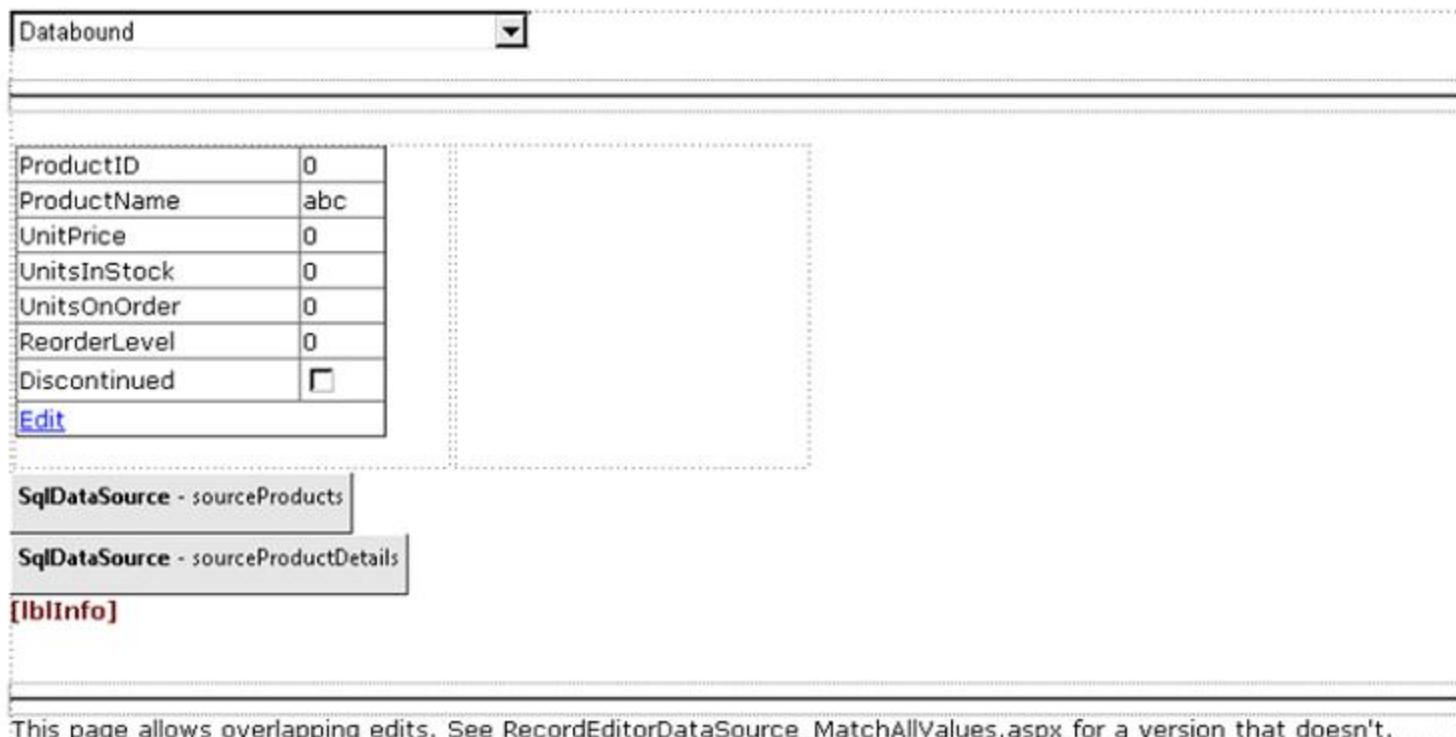
Kết quả thực thi:

Email	CompanyName	Address
ngocanh@MadridHome.com	Viet Employment	C/ Araquil, 67
ngochanh@Marseille.com	Bon app'	12, rue des Bouchers
admin@Tshawassen.com	Bottom-Dollar Markets	23 Tshawassen Blvd.
BuenosAires_admin@BuenosAires.com	Cactus Comidas para llevar	Cerrito 33333
Bern_admin@Bern.com	Chop-suey Chinese	Hauptstr. 29
LondonFamily_admin@LondonFamily.com	Consolidated Holdings	Berkeley Gardens 12 Brewery
Stuttgart_admin@Stuttgart.com	Die Wandernde Kuh	Adenauerallee 900
Aachen_admin@Aachen.com	Drachenblut Delikatessen	Walserweg 21
Nantes_admin@Nantes.com	Du monde entier	67, rue des Cinquante Otages
LondonDistrict_admin@LondonDistrict.com	Eastern Connection	35 King George
Graz_admin@Graz.com	Ernst Handel	Kirchgasse 6
SaoPauloHome_admin@SaoPauloHome.com	Familia Arquibaldo	Rua Orós, 92

BÀI TẬP CHƯƠNG 9

Bài 1: Thực hành lại các thí dụ trong bài học để ôn lại nội dung vừa học.

Bài 2: Thiết kế giao diện trang RecordEditorDataSource.aspx:



Cấu hình SqlDataSource với chuỗi kết nối:

```
<add name="NorthwindConnectionString" connectionString="Data  
Source=localhost\SQLEXPRESS;Initial Catalog=Northwind;Integrated Security=SSPI"  
providerName="System.Data.SqlClient" />
```

Phần mã lệnh thiết kế trang RecordEditorDataSource.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true"  
CodeFile="RecordEditorDataSource.aspx.cs" Inherits="RecordEditorDataSource" %>  
  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"  
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">  
  
<html xmlns="http://www.w3.org/1999/xhtml" >  
<head id="Head1" runat="server">  
    <title>Record Editor</title>  
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />  
</head>  
<body>  
    <form id="form1" runat="server">  
        <div>
```

```
<asp:DropDownList ID="lstProduct" runat="server" AutoPostBack="True"
Width="280px" DataSourceID="sourceProducts" DataTextField="ProductName"
DataValueField="ProductID">
</asp:DropDownList>
<br />
<br />
<hr />
<br />
<table>
<tr>
<td style="width: 234px" valign="top">
<asp:DetailsView ID="DetailsView1" runat="server"
DataSourceID="sourceProductDetails"
Height="50px" Width="200px" AutoGenerateEditButton="True"
AutoGenerateRows="False" DataKeyNames="ProductID">
<Fields>
<asp:BoundField DataField="ProductID" HeaderText="ProductID"
InsertVisible="False" ReadOnly="True"
SortExpression="ProductID" />
<asp:BoundField DataField="ProductName"
HeaderText="ProductName"
SortExpression="ProductName" />
<asp:BoundField DataField="UnitPrice" HeaderText="UnitPrice"
SortExpression="UnitPrice" />
<asp:BoundField DataField="UnitsInStock"
HeaderText="UnitsInStock"
SortExpression="UnitsInStock" />
<asp:BoundField DataField="UnitsOnOrder"
HeaderText="UnitsOnOrder"
SortExpression="UnitsOnOrder" />
<asp:BoundField DataField="ReorderLevel"
HeaderText="ReorderLevel"
SortExpression="ReorderLevel" />
<asp:CheckBoxField DataField="Discontinued"
HeaderText="Discontinued"
SortExpression="Discontinued" />
</Fields>
</asp:DetailsView>
```

```

    &nbsp;
  </td>
  <td style="width: 190px">
    <br />
    <br />
  </td>
</tr>
</table>

<asp:SqlDataSource ID="sourceProducts" runat="server"
ConnectionString="<%$ ConnectionStrings:NorthwindConnectionString %>"
SelectCommand="SELECT [ProductName], [ProductID] FROM [Products]"
/>

<asp:SqlDataSource ID="sourceProductDetails" runat="server"
ConnectionString="<%$ ConnectionStrings:NorthwindConnectionString %>"
SelectCommand="SELECT ProductID, ProductName, UnitPrice, UnitsInStock,
UnitsOnOrder, ReorderLevel, Discontinued FROM Products WHERE
ProductID=@ProductID"

UpdateCommand="UPDATE Products SET ProductName=@ProductName,
UnitPrice=CONVERT(money, @UnitPrice), UnitsInStock=@UnitsInStock,
UnitsOnOrder=@UnitsOnOrder, ReorderLevel=@ReorderLevel,
Discontinued=@Discontinued WHERE ProductID=@ProductID">

<SelectParameters>
  <asp:ControlParameter ControlID="lstProduct" Name="ProductID"
    PropertyName="SelectedValue" />
</SelectParameters>

<UpdateParameters>
  <asp:Parameter Name="ProductName" />
  <asp:Parameter Name="UnitPrice" />
  <asp:Parameter Name="UnitsInStock" />
  <asp:Parameter Name="UnitsOnOrder" />
  <asp:Parameter Name="ReorderLevel" />
  <asp:Parameter Name="Discontinued" />
  <asp:Parameter Name="ProductID" />
</UpdateParameters>

```

```

</asp:SqlDataSource>
    <asp:Label ID="lblInfo" runat="server" EnableViewState="False" Font-
Bold="True" ForeColor="#C00000"></asp:Label>
    <br />
    <br />
    <br />
    <hr />

This page allows overlapping edits. See
RecordEditorDataSource_MatchAllValues.aspx
for a version that doesn't.

</div>
</form>
</body>
</html>

```

Phần mã lệnh thực thi trang RecordEditorDataSource.aspx.cs:

```

public partial class RecordEditorDataSource: System.Web.UI.Page
{
}

```

Kết quả thực thi:

The screenshot shows a dropdown menu with the option "Boston Crab Meat" selected. Below it is a details view table for a product record:

ProductID	40
ProductName	Boston Crab Meat
UnitPrice	18.4000
UnitsInStock	123
UnitsOnOrder	0
ReorderLevel	30
Discontinued	<input type="checkbox"/>
Edit	

At the bottom of the page, there is a note: "This page allows overlapping edits. See RecordEditorDataSource_MatchAllValues.aspx for a version that doesn't."

Chọn ProductName trong DropDownList, nội dung DetailsView sẽ được cập nhật ngay.

Chương 10

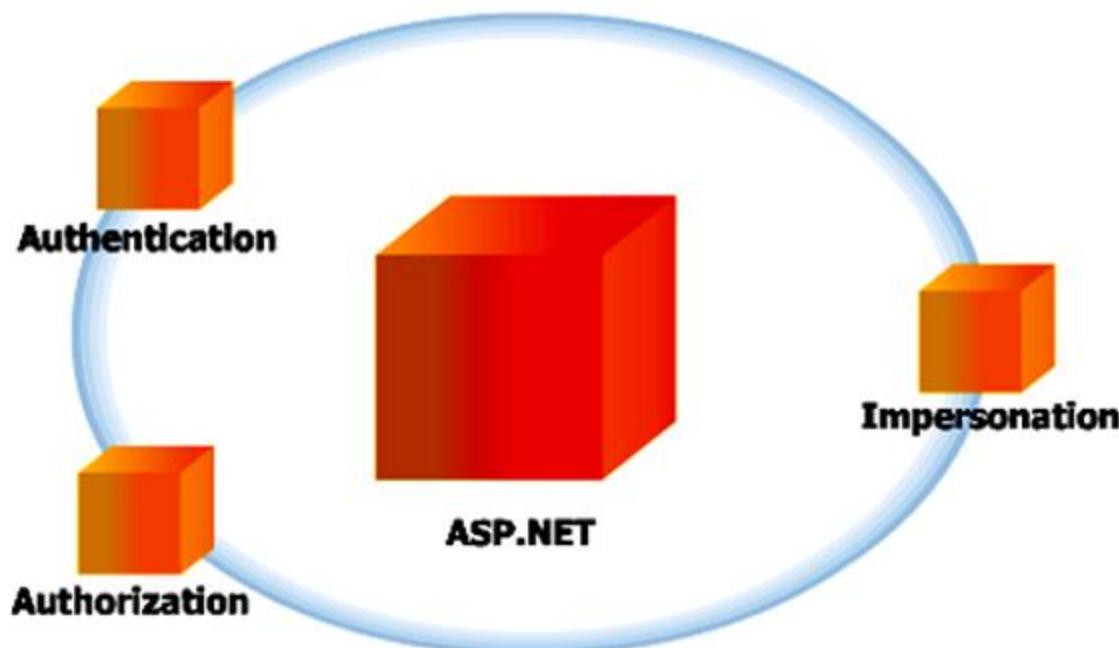
BẢO MẬT ỨNG DỤNG WEB

Kết thúc chương này các bạn có thể:

- *Trình bày được các loại bảo mật trong ASP.NET*
- *Mô tả được Form-Based Authentication*
- *Thực thi được bảo mật sử dụng Form-Based Authentication*

10.1. GIỚI THIỆU VỀ BẢO MẬT TRONG ASP.NET

Bảo mật rất quan trọng cho việc phát triển và bảo trì các ứng dụng Web. Mỗi WebSite có các thiết lập bảo mật khác nhau. Có ba tính năng bảo mật cơ bản cho các ứng dụng ASP.NET:



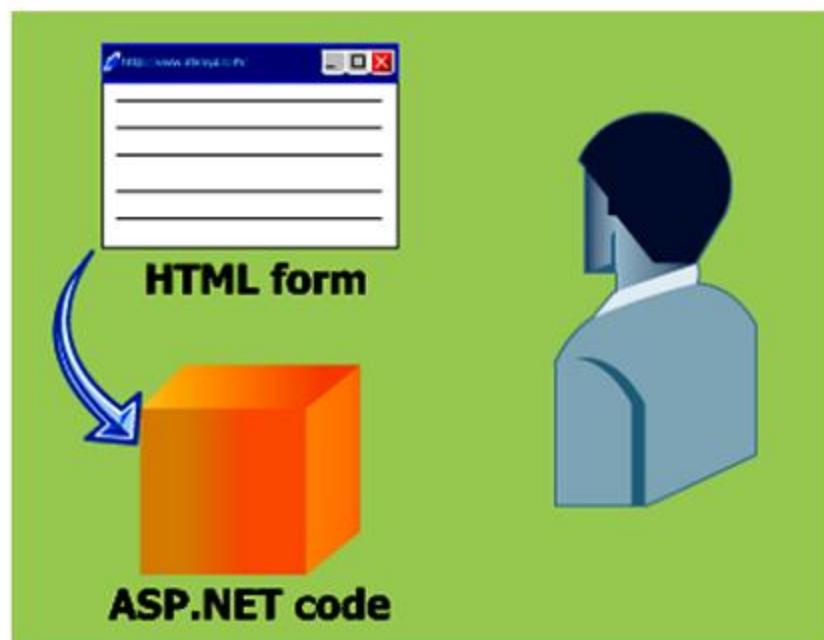
Hình 10.1: Ba tính năng bảo mật cho các ứng dụng ASP.NET

- ❖ **Authentication:** được dùng để xác nhận định danh người dùng trước khi cho phép hoặc từ chối các yêu cầu. Thí dụ: trong ứng dụng e-mail, tên người dùng và mật khẩu phải phù hợp khi so sánh các thông tin này trong cơ sở dữ liệu.
- ❖ **Authorization:** chỉ những người dùng có sự định danh hợp lệ mới có thể truy cập đến các tài nguyên được chỉ định trong ứng dụng. Thí dụ: sinh viên không thể truy cập được thông tin đề thi, điểm thi mà chỉ có giáo viên hoặc người quản trị mới có thể truy cập.
- ❖ **Impersonation:** ứng dụng ASP.NET xử lý các hoạt động đại diện cho người dùng mà được xác nhận bằng IIS (Internet Information Services). IIS truyền dấu hiệu xác nhận đến ứng dụng ASP.NET.

Sau đó, ứng dụng ASP.NET sử dụng dấu hiệu này và hoạt động dưới sự định danh của người dùng đã xác nhận.

Form-Based Authentication

Sử dụng Forms Authentication Provider. Trong forms-base authentication, định dạng HTML thường sử dụng tập hợp các thông tin xác nhận: tên đăng nhập, mật khẩu. Ứng dụng phải viết các lệnh để xác nhận các thông tin được cung cấp phải phù hợp trong cơ sở dữ liệu. Những thông tin đã được xác nhận của người dùng có thể được lưu trong một biến cookie trong suốt phiên làm việc.



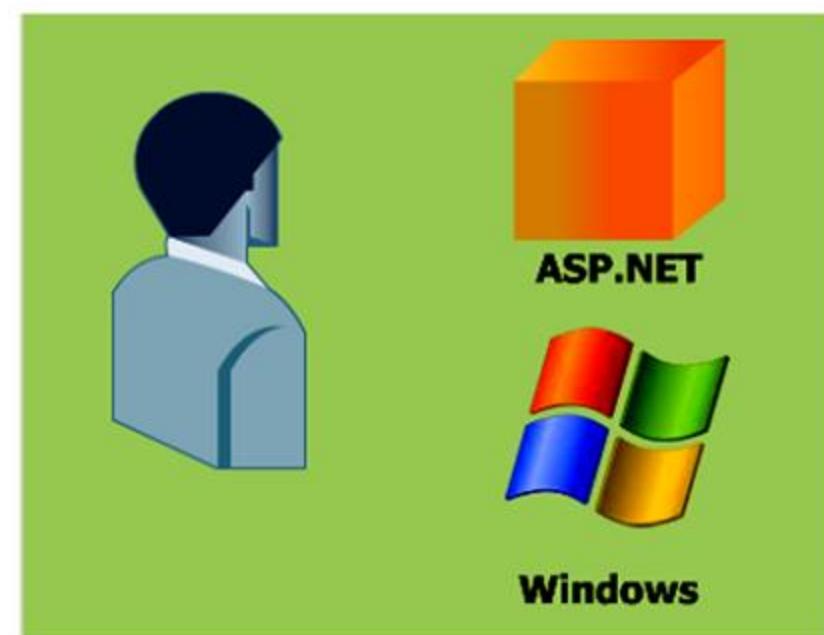
Passport Authentication

Người dùng được xác nhận bằng cách sử dụng Passport Service được cung cấp bởi Microsoft. Tuy nhiên, khi sử dụng loại xác nhận này, chúng ta phải được đăng ký với Microsoft's Passport Service. Passport server sử dụng cơ chế mã hóa cookie để định danh và xác nhận tính hợp lệ của người dùng.



Windows Authentication

Windows authentication là cách xác nhận mặc định của ASP.NET. Loại xác nhận này dựa vào các windows account của người dùng. Windows authentication trong ASP.NET sử dụng II để có thể cấu hình cho phép chỉ các người dùng trong Windows domain đăng nhập vào ứng dụng.



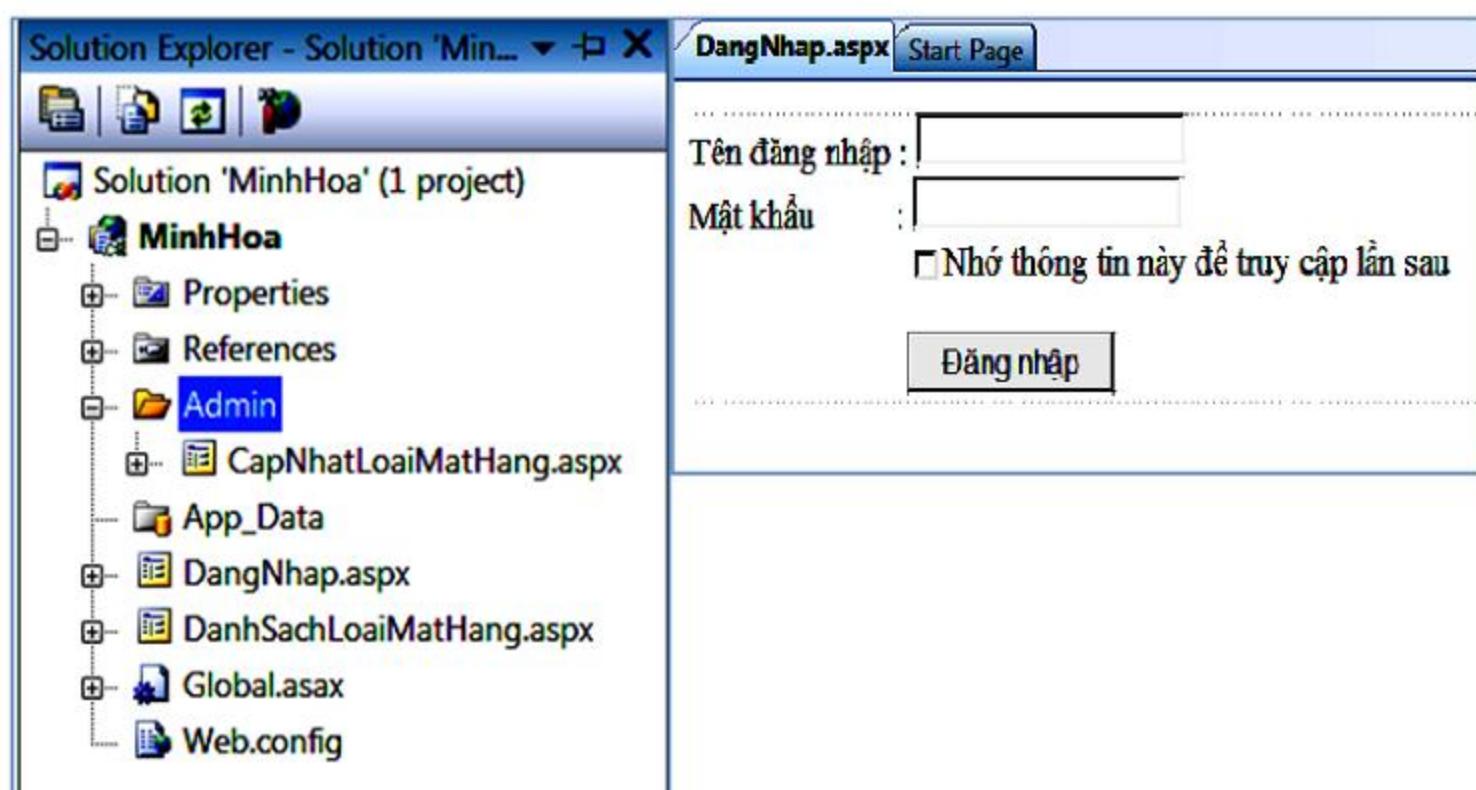
- Có bốn tùy chọn được thiết lập trên IIS:

- **Anonymous Authentication:** cho phép bất kỳ người dùng truy cập đến ứng dụng ASP.NET.
- **Basic Authentication:** yêu cầu sử dụng tên người dùng và mật khẩu của Windows để kết nối đến ứng dụng. Tuy nhiên, mật khẩu được truyền trong dạng text đơn giản. Do đó, loại xác nhận này không được an toàn.
- **Digest Authentication:** tương tự như Basic Authentication. Tuy nhiên, mật khẩu được hashed sau đó mới được truyền đi (giả tăng sự an toàn).
- **Integrated Windows Authentication:** sử dụng Kerberos hoặc giao thức challenge/response để xác nhận người dùng.

10.2. THÍ DỤ MINH HỌA

❖ **Thí dụ:** minh họa sử dụng **Form-based Authentication** trong ứng dụng Web theo yêu cầu: người dùng truy cập vào các trang trong thư mục **Admin** bắt buộc phải đăng nhập, nếu không đăng nhập người dùng chỉ có xem danh sách các loại mặt hàng bằng cách truy cập vào trang **DanhSachLoaiMatHang.aspx**

Bước 1: Xây dựng một ứng dụng Web gồm các trang ASP.NET: **CapNhatLoaiMatHang.aspx**, **DanhSachLoaiMatHang.aspx**, **DangNhap.aspx**. Các trang Web này có cấu trúc ứng dụng web như hình sau:



Hình 10.2: Giao diện trang *DangNhap.aspx*

Hình 10.3: Giao diện trang CapNhatLoaiMatHang.aspx

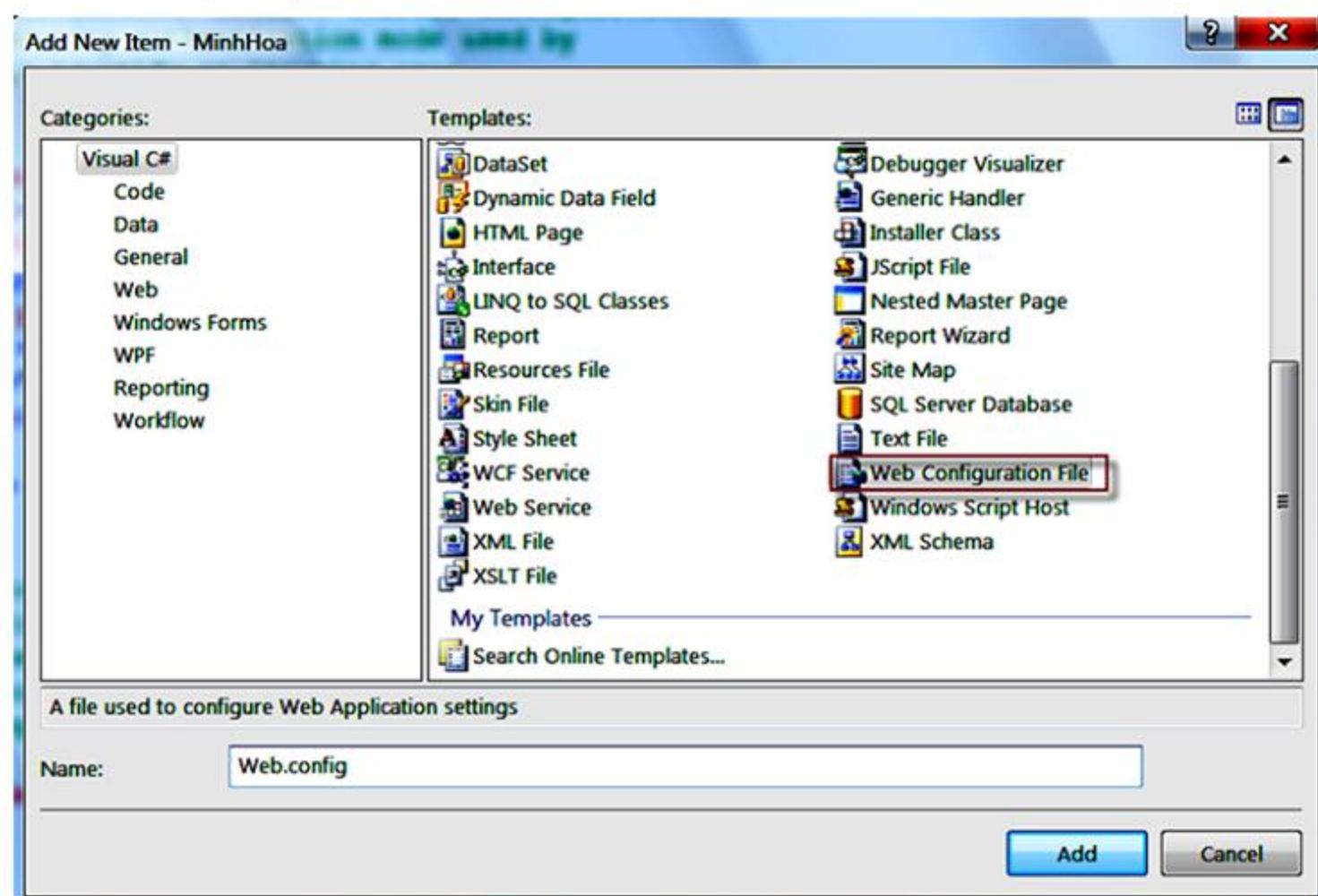
Bước 2: Cấu hình phần <authentication>....</authentication> trong tập tin Web.config như sau:

```
<authentication mode="Forms">
    <forms loginUrl="DangNhap.aspx"
        defaultUrl="~/Admin/CapNhatLoaiMatHang.aspx"
        name="MinhHoa.com" timeout="30">
        <credentials passwordFormat="Clear">
            <user name ="tom" password="tom"/>
            <user name="jerry" password="jerry"/>
        </credentials>
    </forms>
</authentication>
```

- **loginUrl**: chỉ rõ trang người dùng sẽ đăng nhập
 - **defaultUrl**: chỉ rõ trang sẽ được chuyển đến nếu đăng nhập thành công
 - **name**: chỉ rõ tên Cookie sẽ được tạo
 - **timeout**: thời gian cookie sẽ tồn tại (tính bằng phút)
 - **passwordFormat**: không mã hóa mật khẩu (password)

Trong thí dụ này, chúng ta khai báo hai người dùng (user) có tên đăng nhập là Tom và Jerry, các bạn có thể lưu các user này trong cơ sở dữ liệu để kiểm tra khi đăng nhập.

Bước 3. Trên cửa sổ Server Explorer, chọn thư mục Admin, từ menu ngữ cảnh của thư mục chọn Add | New Item. Cửa sổ Add New Item xuất hiện thêm vào một tập tin Web.config để kiểm tra việc truy cập cho thư mục này.



Nội dung của tập tin Web.config trong thư mục **Admin**:

```
<?xml version="1.0"?>
<configuration>
  <appSettings/>
  <connectionStrings/>
  <system.web>
    <authorization>
      <deny users="?" />
    </authorization>
  </system.web>
</configuration>
```

Bước 4. Viết lệnh xử lý các sự kiện cho các trang.

Trang DangNhap.aspx, các bạn viết lệnh cho nút DangNhap như hình sau:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.Security;

namespace MinhHoa
{
    public partial class DangNhap : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }

        protected void btnDangNhap_Click(object sender, EventArgs e)
        {
            //Xác nhận thông tin người dùng
            if (FormsAuthentication.Authenticate(txtTenDangNhap.Text, txtMatKhau.Text))
            {
                //Nếu hợp lệ chuyển qua trang CapNhatLoaiMatHang.aspx
                //Nếu người dùng chọn lưu thông tin đăng nhập thì ASP.Net tạo
                //Cookie để lưu thông tin này.
                FormsAuthentication.RedirectFromLoginPage(txtTenDangNhap.Text,
                    chkLuuThongTinDangNhap.Checked);
            }
        }
    }
}
```

Hình 10.4: Phần mã lệnh trang DangNhap.aspx

Trang CapNhatLoaiMatHang.aspx, các bạn viết thêm vào sự kiện Page_Load và sự kiện cho link button Đăng Xuất như hình sau:

CapNhatLoaiMatHang.aspx.cs Start Page

```

MinhHoa.CapNhatLoaiMatHang CapNhatThongTinLoaiMatHang
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SqlClient;
using System.Data;
using System.Web.Security;

namespace MinhHoa
{
    public partial class CapNhatLoaiMatHang : System.Web.UI.Page
    {

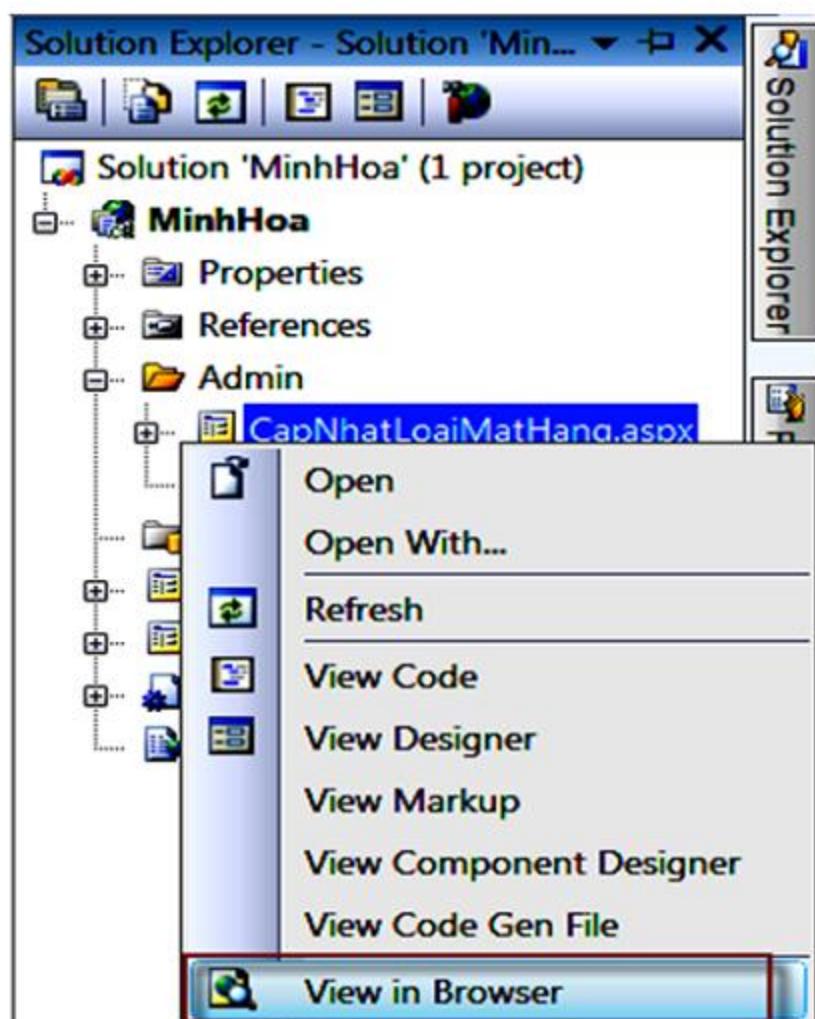
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                LayDanhSachLoaiMatHang();
                //lấy tên người dùng đăng nhập
                lbTenDangNhap.Text = User.Identity.Name;
            }
        }

        protected void lkbDangXuat_Click(object sender, EventArgs e)
        {
            //Hủy tắt cả các thông tin đã lưu và trả về trang đăng nhập
            FormsAuthentication.SignOut();
            FormsAuthentication.RedirectToLoginPage();
        }
    }
}

```

Hình 10.5: Phần mã lệnh trang CapNhatLoaiMatHang.aspx

Bước 5. Trong cửa sổ Solution Explorer, chọn trang *CapNhatLoaiMatHang.aspx* nhấn phải chuột | View in Brower để thi hành trang web. Khi người dùng truy cập vào trang này mà chưa đăng nhập thì ASP.NET tự động chuyển qua trang DangNhap.aspx. Kết quả sẽ xuất hiện như sau:



Tên đăng nhập :

Mật khẩu :

Nhớ thông tin này để truy cập lần sau

Đăng nhập

Hình 10.6: Màn hình hiển thị trang DangNhap.aspx

Nhập vào Tên đăng nhập: “tom” và Mật khẩu là “tom” sau đó nhấn nút **Đăng nhập** để vào trang CapNhatLoaiMatHang.aspx. Nếu người dùng chọn “Nhớ thông tin này.....” thì trong vòng 30 phút có thể truy cập vào trang CapNhatLoaiMatHang.aspx mà không cần phải đăng nhập.

Mã loại mặt hàng	Tên loại mặt hàng	Xuất xứ	Cập nhật	Xóa
1	Nước giải khát	Việt Nam	Cập nhật	Xóa
2	Đồ gia dụng	Thái Lan	Cập nhật	Xóa
3	Hàng Nội Thất	Việt Nam	Cập nhật	Xóa
4	Loại A	Nhật Bản	Cập nhật	Xóa
5	Loại B	Hàn Quốc	Cập nhật	Xóa
6	Loại C	Đài Loan	Cập nhật	Xóa
7	Loại D	Trung Quốc	Cập nhật	Xóa
8	Loại E	Trung Quốc	Cập nhật	Xóa
9	Loại F	Trung Quốc	Cập nhật	Xóa
10	Loại A1	Việt Nam	Cập nhật	Xóa

Hình 10.7: Trang cập nhật thông tin mặt hàng

Sau khi người dùng cập nhật thông tin loại mặt hàng, sau đó nhấp vào **Đăng xuất** để trở về trang DangNhap.aspx. Khi đó, tất cả các thông tin session hay cookie đều được hủy kết thúc một phiên làm việc.

Chương 11

GIỚI THIỆU AJAX

Kết thúc chương này các bạn có thể:

- *Trình bày được kỹ thuật lập trình ASP.NET & Ajax*
- *Mô tả và xây dựng được ứng dụng Ajax ASP.NET với XMLHttpRequest*
- *Mô tả và xây dựng được ứng dụng Ajax ASP.NET với ASPCallBack*
- *Mô tả và xây dựng được ứng dụng Ajax ASP.NET với Ajax Controls: ScriptManager Control, UpdatePanel Control, Timer Controls*
- *Mô tả và xây dựng được ứng dụng Ajax ASP.NET với Ajax Controls Toolkits*

11.1. GIỚI THIỆU AJAX

AJAX là một công nghệ cho phép lập trình bất đồng bộ trong ứng dụng Web. Thông thường, người dùng muốn thay đổi thông tin từ trang Web bằng cách nhấp vào các nút lệnh (button) hay các liên kết (link) để submit yêu cầu về Web Server để thay đổi nội dung trang Web (postback). Như vậy, toàn bộ trang Web phải được xử lý lại do đó tốn khá nhiều thời gian và gia tăng sự phản hồi các trang Web,... Công nghệ Ajax (Asynchronous JavaScript and XML) cho phép chỉ các thông tin nào cần thay đổi được gửi về Sever xử lý, sau đó Server sẽ xử lý và trả kết quả về cho Client. Sau đây là một vài thông tin chung sẽ giúp chúng ta hiểu hơn về Ajax:

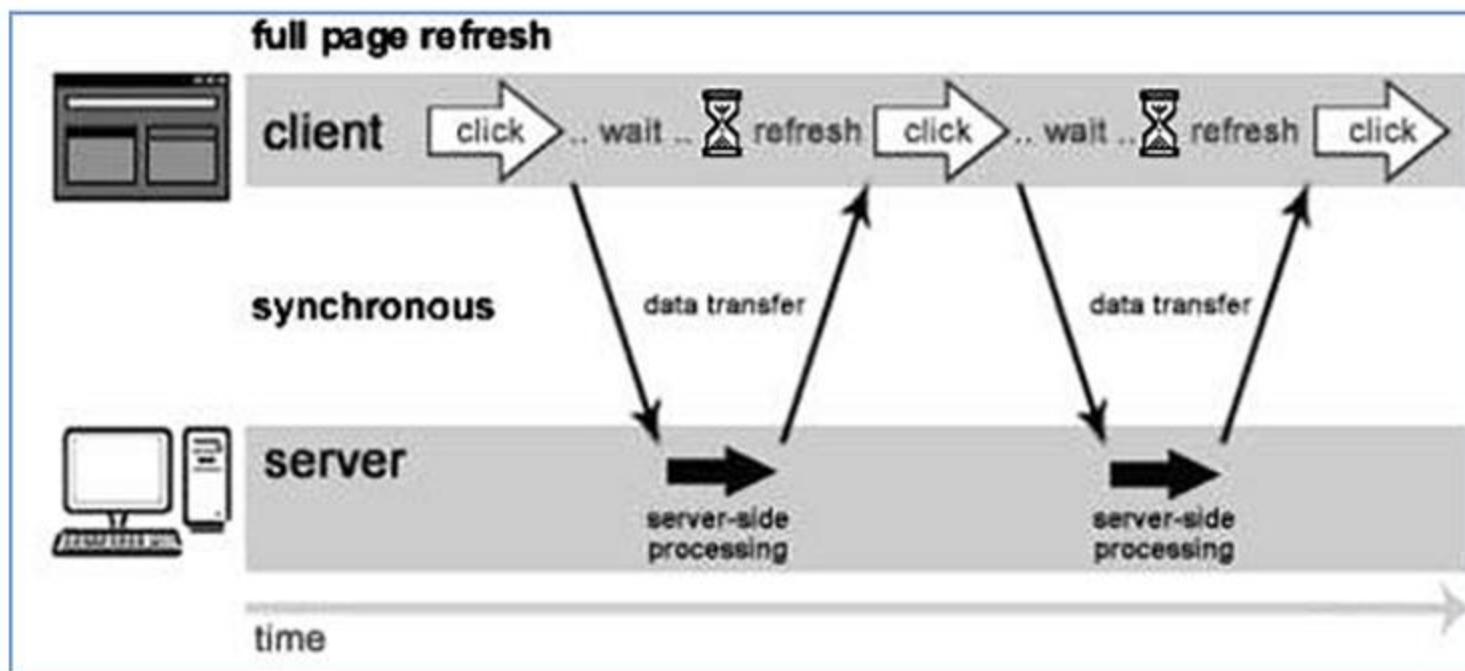
- AJAX bắt đầu phổ biến từ năm 2005 bởi Google (với một ứng dụng Google Suggest, Google Maps, Gmail).
- AJAX không phải là ngôn ngữ lập trình mới, mà nó là một công nghệ mới để tạo ra một ứng dụng web nhỏ hơn, nhanh hơn, tốt hơn và giao diện thân thiện với người dùng hơn.
- Ajax dựa trên các thành phần HTML trước đây:
 - HTML
 - CSS
 - JavaScript (chủ chốt)
 - XML

- AJAX là một công nghệ được hỗ trợ bởi trình duyệt (browser) và nó độc lập với các ứng dụng Web server.
- Với Ajax, Javascript của bạn có thể liên lạc trực tiếp với Web server bằng cách sử dụng đối tượng XMLHttpRequest của Javascript. Với đối tượng này Javascript của bạn có thể trao đổi dữ liệu trực tiếp với Web server mà không cần đệ trình (submit) toàn bộ dữ liệu đến, do đó trang web của bạn không reload lại.
- Ajax sử dụng cơ chế làm việc bất đồng bộ (Asynchronous), tức là trong khi đối tượng XMLHttpRequest thực hiện gửi yêu cầu đến Web server thì Web browser vẫn tiếp tục xử lý các công việc khác mà không cần Web server hoàn thành việc trả lời lại yêu cầu đó. Nhiều công việc được xử lý song song với nhau, điều này khác với cách lập trình web cổ điển trước đây, do đó, ứng dụng web sẽ chạy nhanh hơn.
- Ajax là một kỹ thuật của Web browser và độc lập với Web server
- Tất cả Web có sử dụng Ajax gọi là Web 2.0
- Ajax có thể gửi và nhận dữ liệu với nhiều định dạng khác nhau, bao gồm XML, HTML và thậm chí là file text.

11.2. AJAX LÀM VIỆC NHƯ THẾ NÀO?

Ta hãy phân tích và so sánh cách thức hoạt động của một trang web thông thường và một trang web có ứng dụng Ajax để thấy rõ cách thức thực hiện của Ajax.

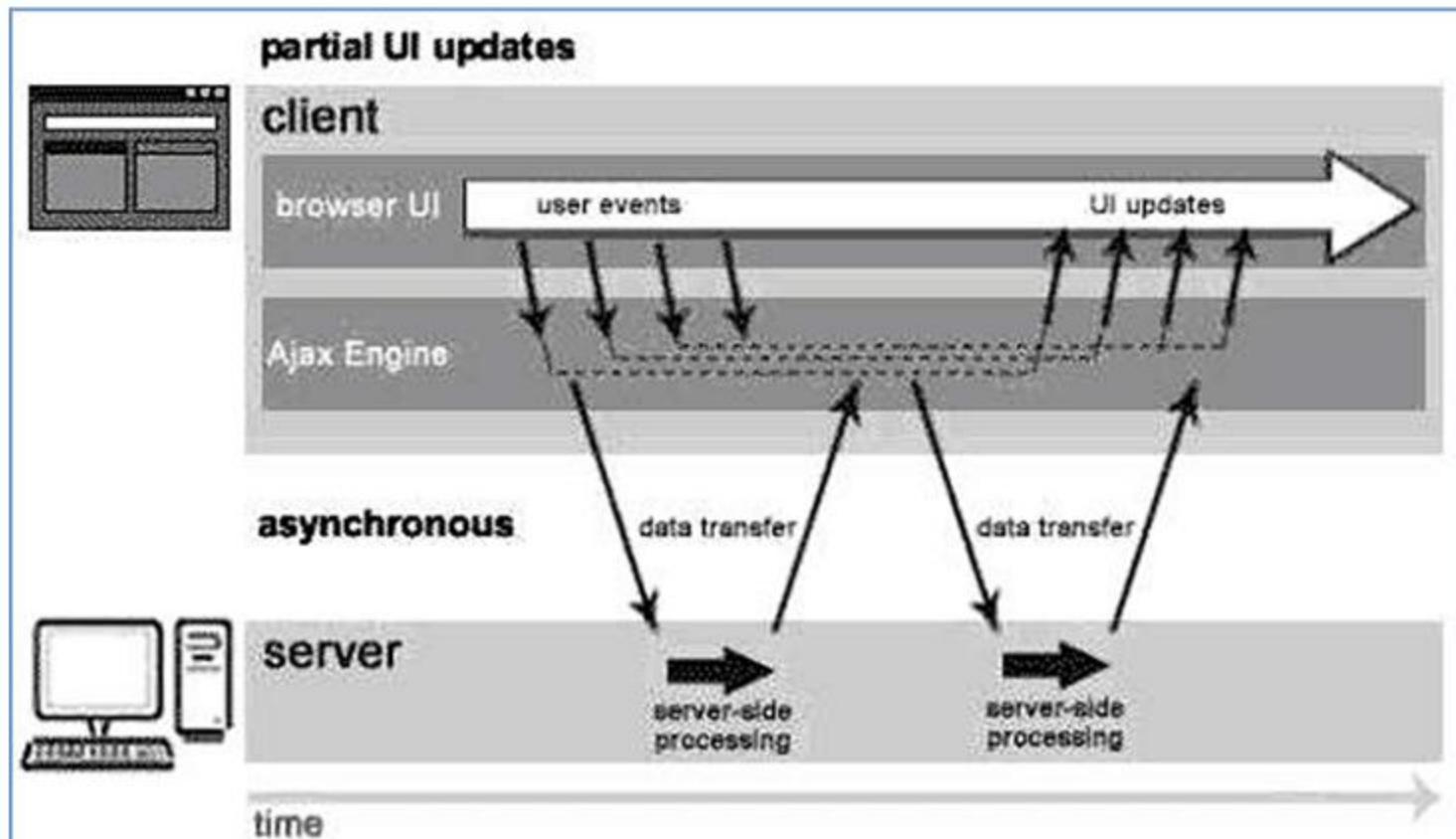
Với cách lập trình Web trước đây (còn gọi là Web 1.0) thì khi người dùng cần cập nhật thông tin (click vào một Button nào đó), thì yêu cầu thay đổi thông tin sẽ được gửi từ phía Client về Server dưới dạng HTTP request, toàn bộ trang web sẽ được gửi chứ không riêng gì một vài thông tin cần thay đổi (dạng này gọi là postback). Lúc đó, Client sẽ rơi vào trạng thái chờ (waiting...), trong lúc này phía Client không thể thực hiện một công việc nào khác. Khi Server xử lý hoàn thành các yêu cầu và thì sẽ gửi trả lại cho phía Client một trang web khác thay thế trang cũ (thông tin mà Server response lại ở dạng HTML và CSS). Quy trình này được mô tả như sau:



Hình 11.1: Cách thức hoạt động trang Web thông thường

Như vậy ta thấy, cách thức hoạt động của một trang web cổ điển là: Click → waiting.. → refresh ... → Do đó, cho dù yêu cầu cập nhật một lượng thông tin nhỏ thì trang web cũng phải tải lại, do đó, các trang web chạy chậm.

Với cách lập trình Web có ứng dụng kỹ thuật Ajax thì Ajax cho phép tạo ra một Ajax Engine nằm giữa UI (user interface – giao diện người dùng) và Server, tức là nằm giữa giao tiếp Client – Server, nhưng phần Ajax Engine này vẫn nằm ở phía Client.



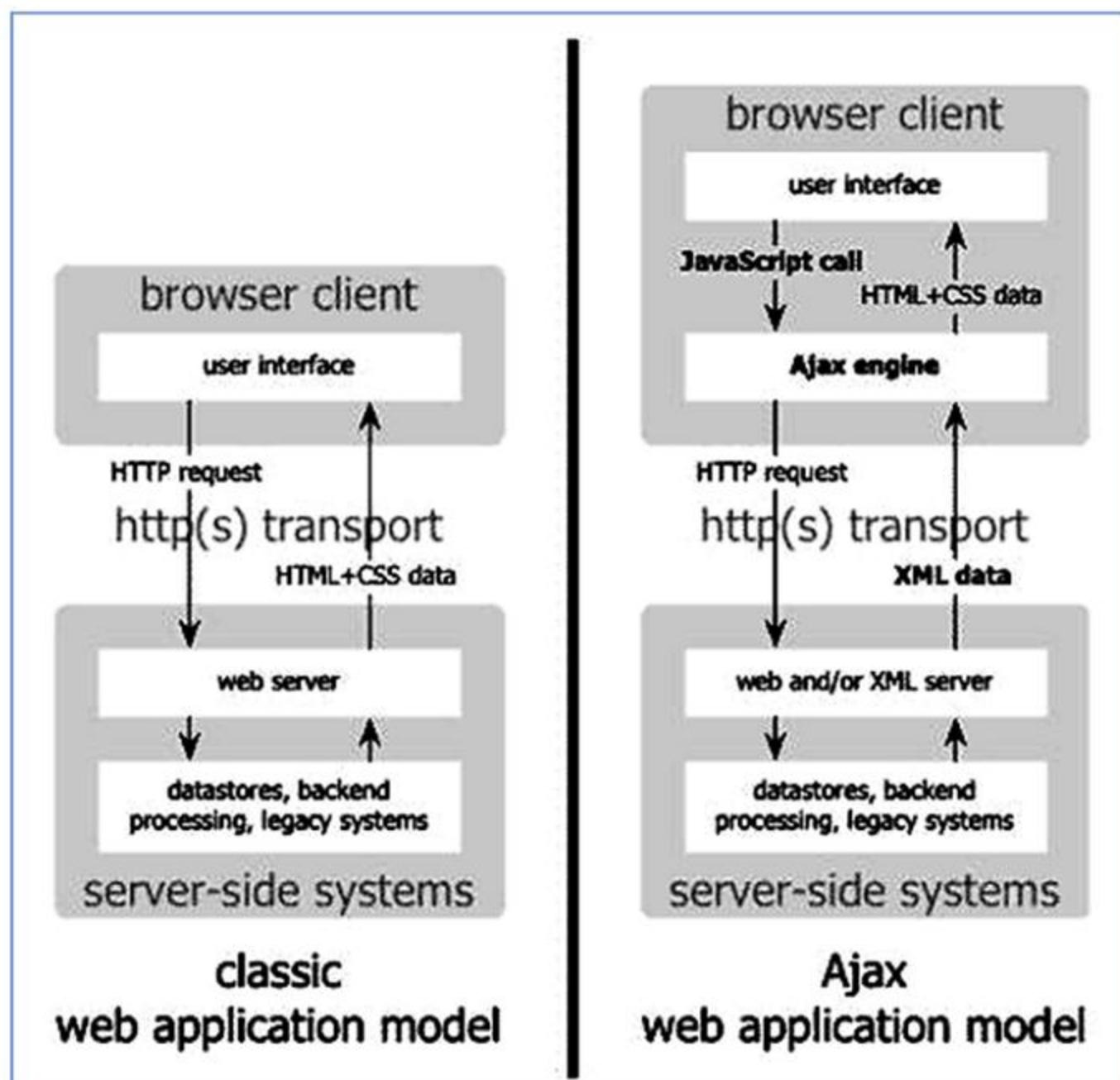
Hình 11.2: Cách thức hoạt động trang Web có Ajax

Khi đó, công việc công việc gửi request và nhận response đều do Ajax Engine thực hiện. Thay vì trả dữ liệu dưới dạng HTML và CSS trực

tiếp cho trình duyệt, Web server có thể gửi trả dữ liệu dạng XML và Ajax Engine sẽ tiếp nhận, phân tích và chuyển đổi thành XHTML + CSS cho trình duyệt hiển thị. Việc phân tích và chuyển đổi này được thực hiện trên Client nên giảm tải rất nhiều cho Server, đồng thời User cảm thấy kết quả xử lý được hiển thị tức thì mà không cần nạp lại toàn bộ trang. Mặt khác, sự kết hợp của các công nghệ web như CSS và XHTML làm cho việc trình bày giao diện trang web tốt hơn nhiều và giảm đáng kể dung lượng trang phải nạp. Đây là những lợi ích hết sức thiết thực mà Ajax đem lại.

- Ajax Engine chỉ gửi đi những thông tin cần thay đổi chứ không phải toàn bộ trang web, do đó giảm được tải qua mạng.
- Việc gửi request và nhận response do Ajax Engine thực. Do đó phía Browser UI không rơi vào trạng thái chờ (waiting...), tức là có thể thực hiện nhiều việc cùng lúc (Asynchronous).

Có thể nhìn vào hai hình sau đây để so sánh hai mô hình ứng dụng Web: truyền thống và sử dụng Ajax.



Hình 11.3: Cách thức hoạt động trang Web không/có Ajax

Sau đây là các bước cài đặt Ajax đơn giản, các bước như sau:

- Phía Client:
 - Viết hàm tạo đối tượng XMLHttpRequest (đối tượng này dùng để gửi request đến Server)
 - Viết hàm gửi yêu cầu (request) đến Server
 - Viết hàm xử lý thông tin sau khi gửi request đến Server. Có thể là thông tin lỗi trả về (nếu việc gửi request thất bại) hay là thông tin do Server trả lời lại
- Phía Server:
 - Viết hàm xử lý các yêu cầu (request) từ Client gửi đến

Thực hiện

Viết hàm tạo đối tượng XMLHttpRequest

- Để bắt đầu viết một ứng dụng web dùng công nghệ Ajax, đầu tiên ta phải khởi tạo đối tượng XMLHttpRequest, đối tượng này được dùng để gửi request đến Server và lấy response từ Server.
- Tùy trình duyệt mà ta có các cách khởi tạo đối tượng XMLHttpRequest khác nhau.
- Cơ bản ta có hai loại trình duyệt:
 - Internet Explorer.
 - Mozilla, Firefox, and Netscape Navigator.

Khởi tạo đối tượng XMLHttpRequest đối với Internet Explorer browser

- IE dùng đối tượng ActiveXObject để khởi tạo.
- Do đó, trước hết phải kiểm tra đối tượng ActiveXObject có tồn tại hay không, ta thực hiện như sau:

```
if (window.ActiveXObject) // nếu true thì là IE
```
- Ta bắt đầu khởi tạo đối tượng XMLHttpRequest với ActiveXObject:

```
XHRObject = new ActiveXObject("Microsoft.XMLHTTP");
```

Khởi tạo đối tượng XMLHttpRequest đối với các browser khác

- Các trình duyệt khác dùng đối tượng XMLHttpRequest để khởi tạo.

- Trước tiên phải kiểm tra đối tượng XMLHttpRequest có tồn tại hay không, ta thực hiện như sau:
`if(window.XMLHttpRequest) // nếu true thì không phải IE`
- Sau đó khởi tạo đối tượng XMLHttpRequest:
`XHRObject = new XMLHttpRequest();`

Code đầy đủ để khởi tạo XMLHttpRequest:

```

var XHRObject;
function CreateXMLHttpRequest()
{
    if (window.ActiveXObject) //nếu là IE
        XHRObject=new
        ctiveXObject("Microsoft.XMLHTTP");
    else if (window.XMLHttpRequest) //Mozilla, FireFox,....
        XHRObject = new XMLHttpRequest();
}

```

Viết hàm gửi yêu cầu (request) đến Server.

Sau khi ta đã khởi tạo đối tượng XMLHttpRequest, ta dùng các phương thức open(), send() và thuộc tính onreadystatechange của đối tượng XMLHttpRequest để thực hiện gửi request đến Server, ta thực hiện theo các bước sau:

Mở một request tới server bằng hàm open():

```
open("method","URL"[,asyncFlag[,“userName”[,”password”]]])
```

Chú giải

- method: Phương thức HTTP được dùng để mở một kết nối, Thí dụ GET hoặc POST
- GET: với giao thức này ta có thể dùng để lấy dữ liệu từ phía Server mà không cần send bất kỳ thông tin nào lên Server. Mặc khác giao thức này có mục đích giảm thiểu giao tiếp không cần thiết qua mạng bằng việc cached tất cả thông tin trả về từ web Server, cho nên những lần sau khi cần lấy dữ liệu cũ thì sẽ giảm thiểu việc truyền thông tin qua mạng bằng cách lấy dữ liệu từ cached.

- POST: Với giao thức này, ta bắt buộc phải truyền thông tin từ Client lên Server trong khi gửi request, Thí dụ như: id=tên. Một khác khi dùng giao thức này thì dữ liệu response từ Server sẽ không được cached, do đó cho dù có request dữ liệu cũ hay mới thì nó đều phải gửi thông tin của form lên web server và lấy dữ liệu mới về.
- URL: Đường dẫn URL request đến server
- asyncFlag: Cách gọi là đồng bộ hay bất đồng bộ, nếu asyncFlag = true: bất đồng bộ (asynchronous), ngược là là đồng bộ (synchronous). Mặc định asyncFlag = true
- userName: Tên đăng nhập tới Server (nếu server yêu cầu)
- password: Mật khẩu tới Server (nếu server yêu cầu)

Lưu ý: Các tham số trong cặp ngoặc [] là optional, nghĩa là có thể có hoặc không

Thí dụ: XHRObject.open("GET","trang2.aspx",true);

XHRObject.open("POST","trang2.aspx",true);

(chú ý: **GET** và **POST** phải viết bằng chữ in hoa)

Khai báo hàm sẽ xử lý dữ liệu trả về từ Server với thuộc tính onreadystatechange:

XHRObject.onreadystatechange = handleStateChanged;

Ta có thể đặt handleStateChanged thành một tên khác bất kỳ. Đây là một con trỏ sẽ trỏ tới hàm có tên tương ứng để xử lý dữ liệu khi Server response lại.

Kế đến ta thực hiện gửi request đến Server với phương thức send():

XHRObject.send(null);

- Nếu trong hàm open() ta dùng method **Get**, thì trong hàm send này ta bắt buộc phải truyền dữ liệu lên server:

Thí dụ:

var queryString = "id=123&cateID=abc";

XHRObject.send(queryString);

- Nếu trong hàm open() ta dùng method **POST**, thì trong hàm send này ta không cần phải truyền dữ liệu lên server:

XHRObject.send(null);

Code hoàn chỉnh để gửi yêu cầu về Server

```
function CallServer() {  
    CreateXMLHttpRequest();  
    if(XHRObject) {  
        XHRObject.open("POST", "page1.aspx");  
        XHRObject.onreadystatechange = handleStateChanged;  
        XHRObject.send(null);  
    }  
}
```

Hay

```
function CallServer () {  
    if(XHRObject) {  
        XHRObject.open("GET", "page1.aspx");  
        XHRObject.onreadystatechange = handleStateChanged;  
        XHRObject.send("id=123&name=tung");  
    }  
}
```

Viết hàm xử lý thông tin khi Server response lại

Hàm này phải có tên giống như đã khai báo trong thuộc tính onreadystatechange ở trên. Đầu tiên là ta kiểm tra thử request có được gửi đến Server hay chưa với thuộc tính status, nếu bằng bốn là request đã được gửi đến Server.

Kế đến ta kiểm tra xem Server có response lại hay chưa với thuộc tính readyState, nếu bằng 200 tức là Server đã response lại hoàn tất.

Thí dụ:

```
function handleStateChanged() {  
    if (XHRObject.status == 200) //status OK  
        if (XHRObject.readyState==4) //complete  
            document.getElementById("myDiv").innerHTML =  
                XHRObject.responseText;  
}
```

❖ **Thí dụ 1.** Minh họa sử dụng Ajax sử dụng XMLHttpRequest

Bước 1. Tạo một trang LayGioHienHanh.aspx có giao diện như sau:



Bước 2. Tạo một trang XuLyLayGioHienHanh.aspx, trang này chỉ để xử lý yêu cầu cho trang LayGioHienHanh.aspx. Do đó, trong thí dụ này, chúng ta không cần thiết kế giao diện chỉ viết lệnh cho phần Code-Behind trong tập tin LayGioHienHanh.aspx.cs

```
XuLyLayGioHienHanh.aspx.cs* Start Page
MinhHoa.XuLyLayGioHienHanh

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace MinhHoa
{
    public partial class XuLyLayGioHienHanh : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            Response.Write(DateTime.Now.ToString("hh:mm:ss"));
        }
    }
}
```

Hình 11.4: Phần mã lệnh trang LayGioHienHanh.aspx

Bước 3. Mở trang LayGioHienHanh.aspx, trong phần mã HTML viết các lệnh javascript

```

<script language ="javascript" type ="text/javascript" >
    var XHRObject;
    function CreateXMLHttpRequest() {
        if (ActiveXObject) //IE
            XHRObject = new ActiveXObject("Microsoft.XMLHTTP");
        else if (XMLHttpRequest) //Mozilla, FireFox,....
            XHRObject = new XMLHttpRequest();
    }
    function LayDuLieu() {
        if (XHRObject) {
            var URL = "XuLyLayGioHienHanh.aspx";
            XHRObject.open("GET", URL);
            XHRObject.onreadystatechange = handleStateChanged;
            XHRObject.send(null);
        }
    }
    function handleStateChanged() {
        if (XHRObject.readyState == 4)
            if (XHRObject.status == 200) {
                document.write(XHRObject.responseText);
            }
    }
</script>

```

như hình sau:

```

<body onload ="CreateXMLHttpRequest();">
    <form id="form1" runat="server">
        <input id="btHienThi" type="button"
            value="Hiển thị giờ hiện hành" onclick="LayDuLieu();" />
    </form>
</body>

```

Bước 4. Nhấn **Ctrl+F5** hoặc **View in Browser** trang LayGioHienHanh.aspx và nhấn nút **Hiển thị giờ hiện hành** để lấy giờ hiện hành từ Server nhưng trang web không postback.



Hình 11.5: Kết quả thực thi

11.3. ASP.NET AJAX & ASP CALLBACK

Trong phần trước, chúng ta đã biết để xây dựng một ứng dụng ASP.NET Ajax thông qua XMLHttpRequest thì chúng ta phải xây dựng hai trang ASP.NET và viết mã javascript rất nhiều. ASP CallBack hỗ trợ xây dựng ứng dụng ASP.NET Ajax một cách đơn giản hơn (chỉ viết trong một trang ASP.NET) bằng kỹ thuật CallBack. Để sử dụng được ASP Callback trong trang ASP.NET chúng ta phải thực thi các phương thức **RaiseCallbackEvent** và **GetCallbackResult** của giao diện (interface) **IcallbackEventHandler**.

- ❖ **Thí dụ 2.** Trang LayGioHienHanh.aspx trong Thí dụ 1, chúng ta viết lại theo ASP CallBack như sau:

Bước 1: Thiết kế giao diện



Bước 2: Viết mã javascript trong phần HTML:

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>lấy giờ hiện hành với ASP CallBack</title>
</head>
<script type="text/ecmascript">
    function ReceiveServerData(rValue)
    {
        document.getElementById("HienThi").innerHTML = rValue;
    }
</script>
<body>
    <form id="form1" runat="server">
        <div>
            Giờ hiện hành:  <span id="HienThi" runat="server"></span><br>
            <input id="btHienThi" type="button"
                value="Hiển thị giờ hiện hành" onclick="CallServer();"/>
        </div>
    </form>
</body>
</html>
```

Bước 3: Viết lệnh cho phía Sever trong tập tin LayGioHienHanh.aspx.cs

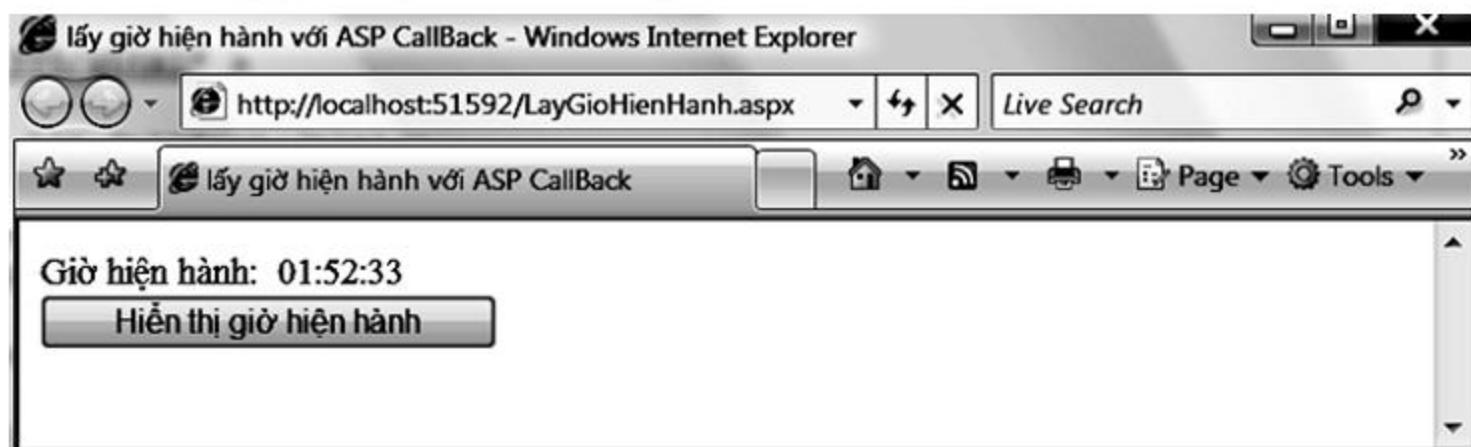
```

LayGioHienHanh.aspx.cs Start Page
MinhHoa.LayGioHienHanh RaiseCallbackEvent(String eventArgument)
namespace MinhHoa
{
    public partial class LayGioHienHanh : System.Web.UI.Page, ICallbackEventHandler
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            //Đăng ký các lệnh CallBack
            String cbReference =
                Page.ClientScript.GetCallbackEventReference(this,
                "", "ReceiveServerData", "");
            String callbackScript;
            callbackScript = "function CallServer() " +
            "{ " + cbReference + ";"};
            Page.ClientScript.RegisterClientScriptBlock(this.GetType(),
            "CallServer", callbackScript, true);
        }

        //Phương thức này nhận dữ liệu được gửi từ Client
        public void RaiseCallbackEvent(String eventArgument)
        {
            //
        }
        //Phương thức này trả kết quả về cho Client
        public String GetCallbackResult()
        {
            //Trả về giờ hiện hành cho Client
            return DateTime.Now.ToString("hh:mm:ss");
        }
    }
}

```

Bước 4. Nhấn **Ctrl+F5** hoặc **View in Browser** để thi hành. Nhấn nút **Hiển thị giờ hiện hành** kết quả như hình sau.



11.4. ASP.NET AJAX SERVER CONTROL

ASP.NET Ajax server controls bao gồm các mã của server và client tương tác với nhau để xử lý các yêu cầu từ client. Khi các bạn thêm các Ajax control vào trong trang ASP.NET nó sẽ tự động gửi các script đến trình duyệt để thực hiện các chức năng Ajax. Chúng ta có

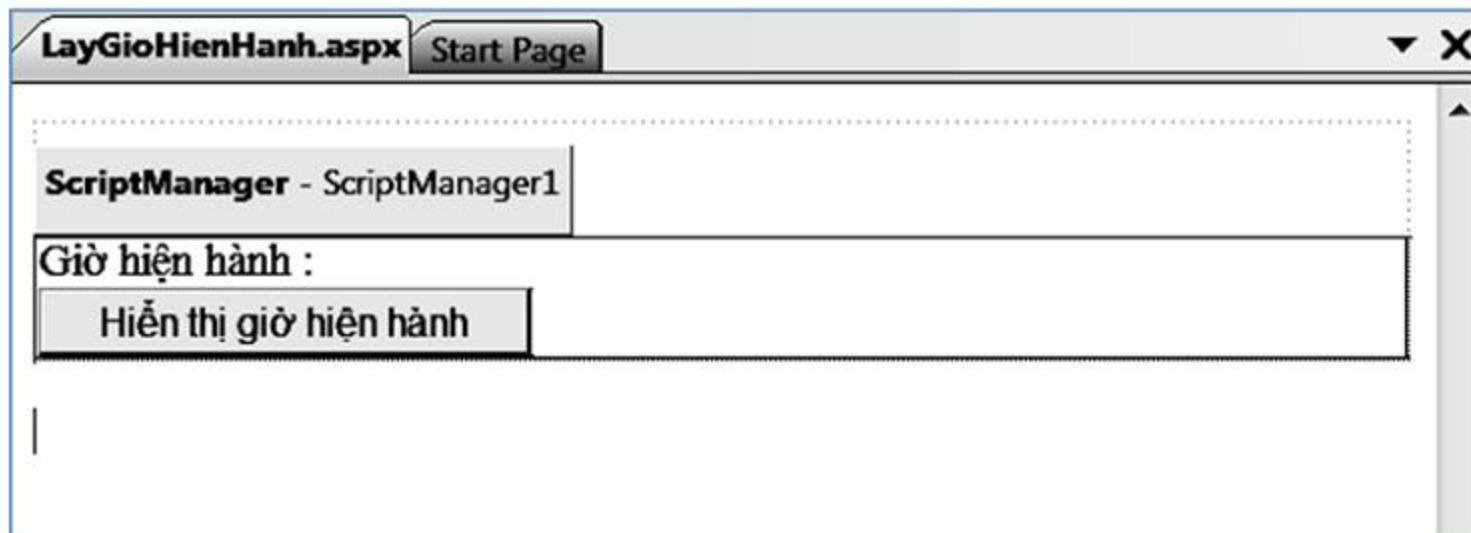


thể cung cấp thêm các đoạn mã ở phía client để tùy biến các chức năng cho Ajax control.

Ajax control gồm các control sau đây:

- **ScriptManager control:** quản lý các script cho các thành phần client, partial-page rendering, localization, globalization và scripts của người dùng. Control này được yêu cầu sử dụng cho các controls: UpdatePanel, UpdateProgress và Timer controls.
 - **UpdatePanel:** cho phép các bạn refresh một phần của trang ASP.NET thay vì refresh toàn bộ trang sử dụng postback.
 - **UpdateProgress:** cung cấp thông tin các trang thái về partial-page update trong UpdatePanel control.
 - **Timer:** thực hiện việc postback trong một khoảng thời gian được chỉ định. Chúng ta có thể sử dụng control này để post toàn bộ trang hoặc sử dụng cùng với Update Panel control thực hiện partial-page update (chỉ một phần nào đó trên trang ASP.NET được thay đổi) với một khoảng thời gian chỉ định.
- ❖ **Thí dụ 3.** Trang LayGioHienHanh.aspx trong Thí dụ 1 và 2, chúng ta viết lại theo ASP.NET ajax server control như sau:

Bước 1: Thiết kế giao diện



Bước 2: Thiết lập phần mã HTML

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Lấy giờ hiện hành với ASP.Net ajax server control</title>
</head>
<body>
    <form id="form1" runat="server">
        <div style="padding-top: 10px">
            <asp:ScriptManager ID="ScriptManager1" runat="server">
            </asp:ScriptManager>
            <asp:UpdatePanel ID="UpdatePanel1" runat="server" RenderMode="Inline">
                <ContentTemplate>
                    <fieldset>
                        <asp:Label ID="lbGioHienHanh" runat="server" Text="Giờ hiện hành :"></asp:Label>
                        <br />
                        <asp:Button ID="btnLayGioHienHanh" runat="server"
                            OnClick="btnLayGioHienHanh_Click" Text="Hiển thị giờ hiện hành" Width="201px" />
                    </fieldset>
                </ContentTemplate>
            </asp:UpdatePanel>
            <br />
        </div>
    </form>
</body>
</html>

```

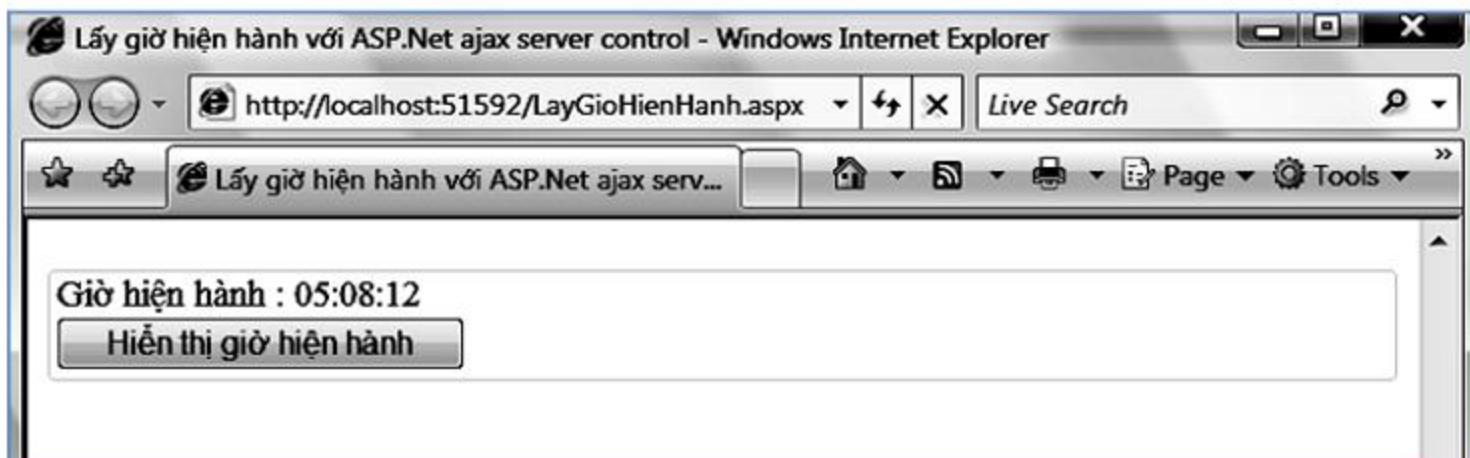
Bước 3: Viết lệnh xử lý sự kiện

```

public partial class LayGioHienHanh : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void btnLayGioHienHanh_Click(object sender, EventArgs e)
    {
        lbGioHienHanh.Text = "Giờ hiện hành : " + DateTime.Now.ToString("hh:mm:ss");
    }
}

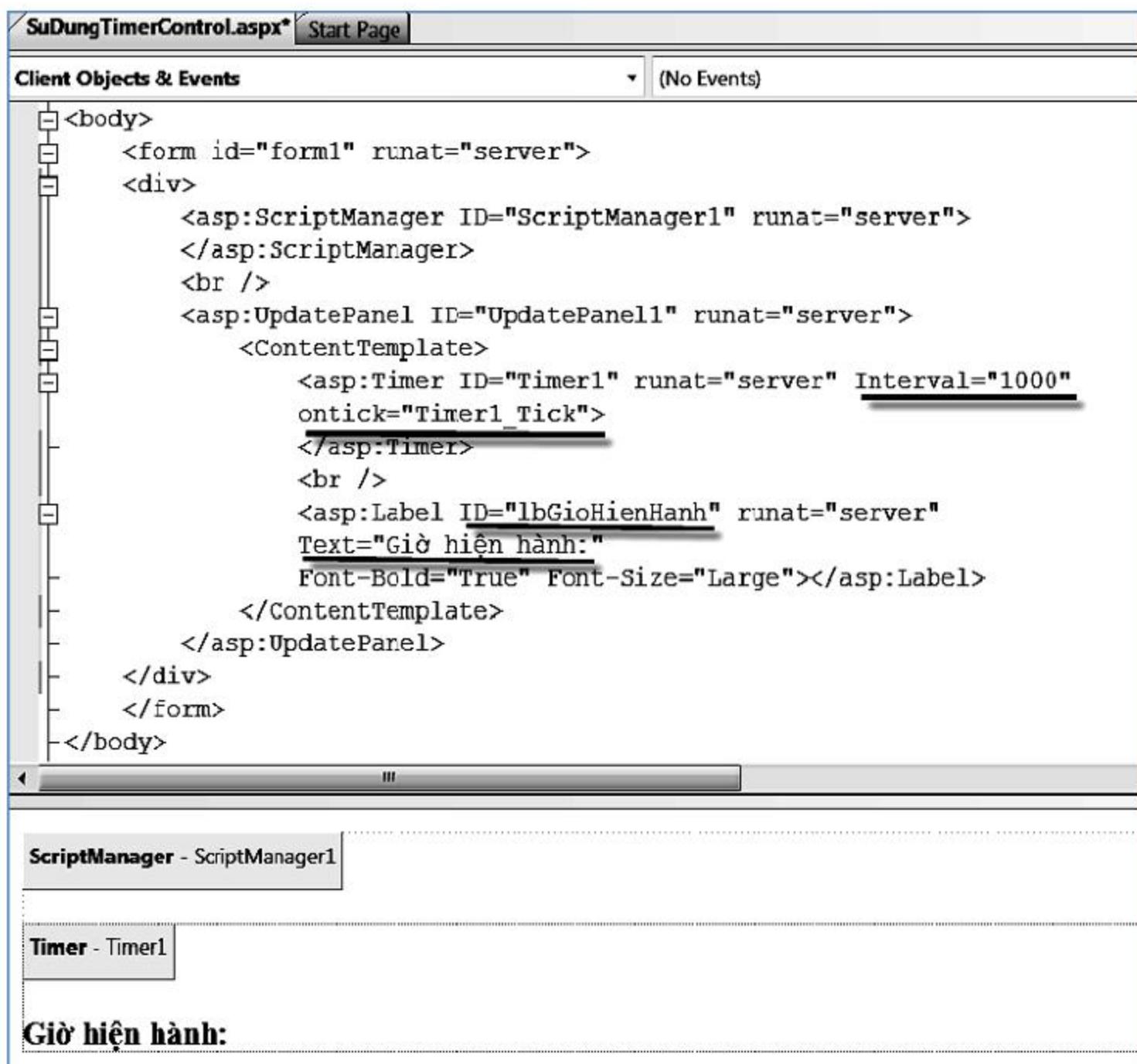
```

Bước 4: Nhấn Ctrl+F5 hoặc View in Browser để thi hành. Nhấn nút Hiển thị giờ hiện hành kết quả như hình sau.



- ❖ **Thí dụ 4.** Tạo trang SuDungTimerControl.aspx trong Thí dụ 1, 2, 3 chúng ta viết lại theo ASP.NET ajax server control hiển thị đồng hồ hiện hành từ server sử dụng Timer Control.

Bước 1: Thiết kế giao diện



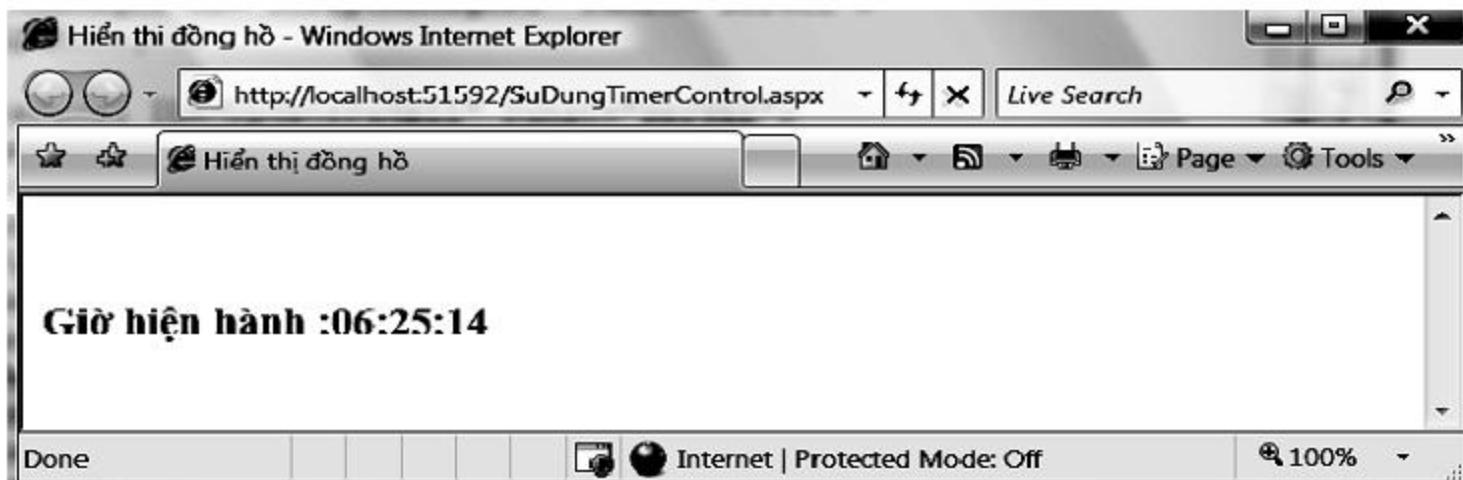
Bước 2. Viết lệnh xử lý sự kiện

```

public partial class SuDungTimerControl : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    //Phương thức này mỗi giây sẽ được gọi 1 lần
    protected void Timer1_Tick(object sender, EventArgs e)
    {
        lbGioHienHanh.Text = "Giờ hiện hành :" +
            DateTime.Now.ToString("hh:mm:ss");
    }
}

```

Bước 3. Nhấn **Ctrl+F5** hoặc **View in Browser** để thi hành. Các bạn sẽ thấy giờ hiện hành được cập nhật liên tục mỗi giây 1 lần.



11.5. GIỚI THIỆU AJAX CONTROL TOOLKIT

Ajax Control Toolkit chứa một tập phong phú các điều khiển mà bạn có thể sử dụng để xây dựng các ứng dụng ASP.NET Web Forms cho phép đáp ứng và tương tác cao với Ajax. Thực hiện theo các bước dưới đây để tải về và bắt đầu sử dụng Ajax Control Toolkit với Visual Studio:

11.5.1. Tải Ajax Control Toolkit

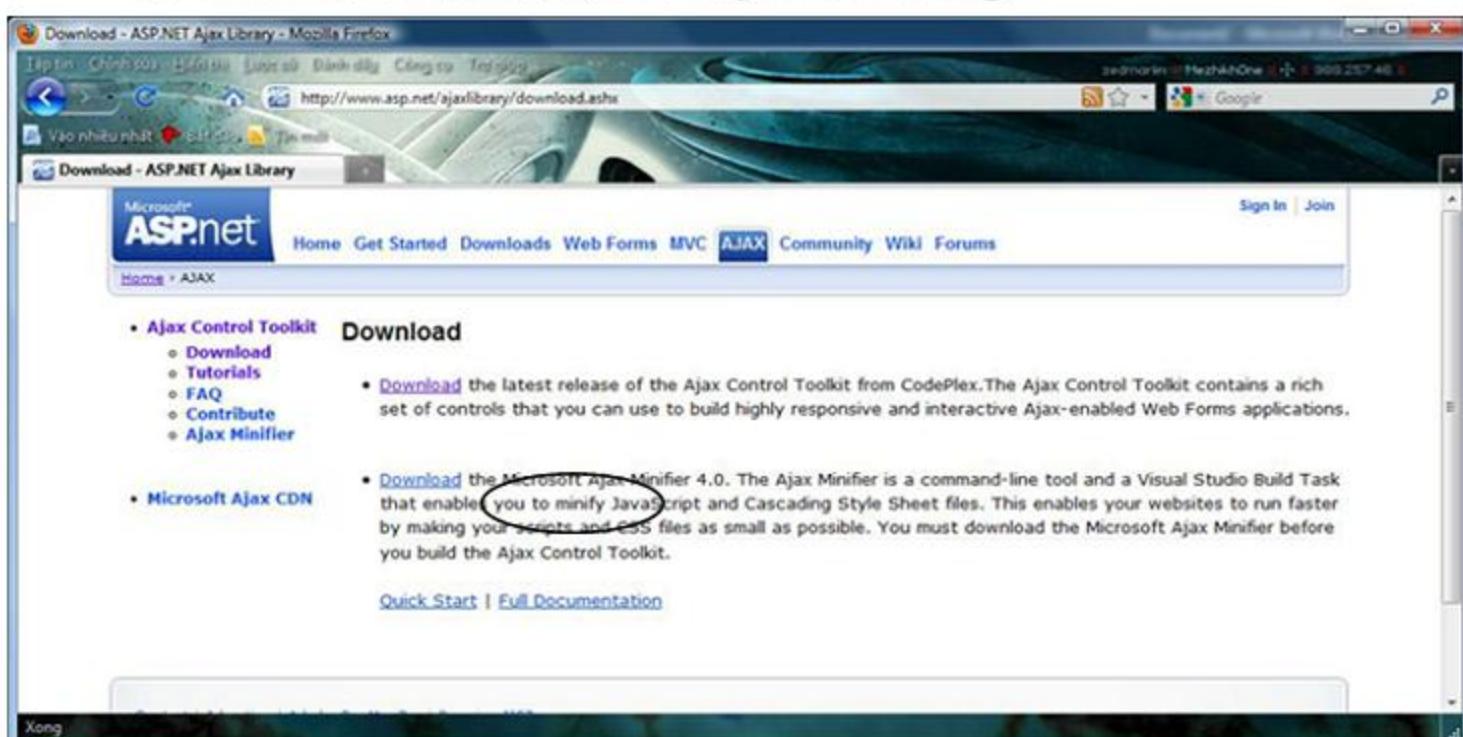
Bạn gõ địa chỉ sau <http://ajaxcontroltoolkit.codeplex.com/> để vào trang:

A screenshot of a Mozilla Firefox browser window displaying the "AJAX Control Toolkit" page on CodePlex. The URL in the address bar is "http://ajaxcontroltoolkit.codeplex.com/". The page features the "ASP.net" logo and the word "AJAX" prominently. It includes links for "Home", "Downloads", "Issue Tracker", "Source Code", "People", and "License". On the right side, there is a large "Download" button with the number "40412" and a "Stable" status. Below the download stats, there's a section for "Recent reviews" and some activity metrics like "Page Views" and "Visits". The footer of the page shows the URL "http://ajaxcontroltoolkit.codeplex.com/".

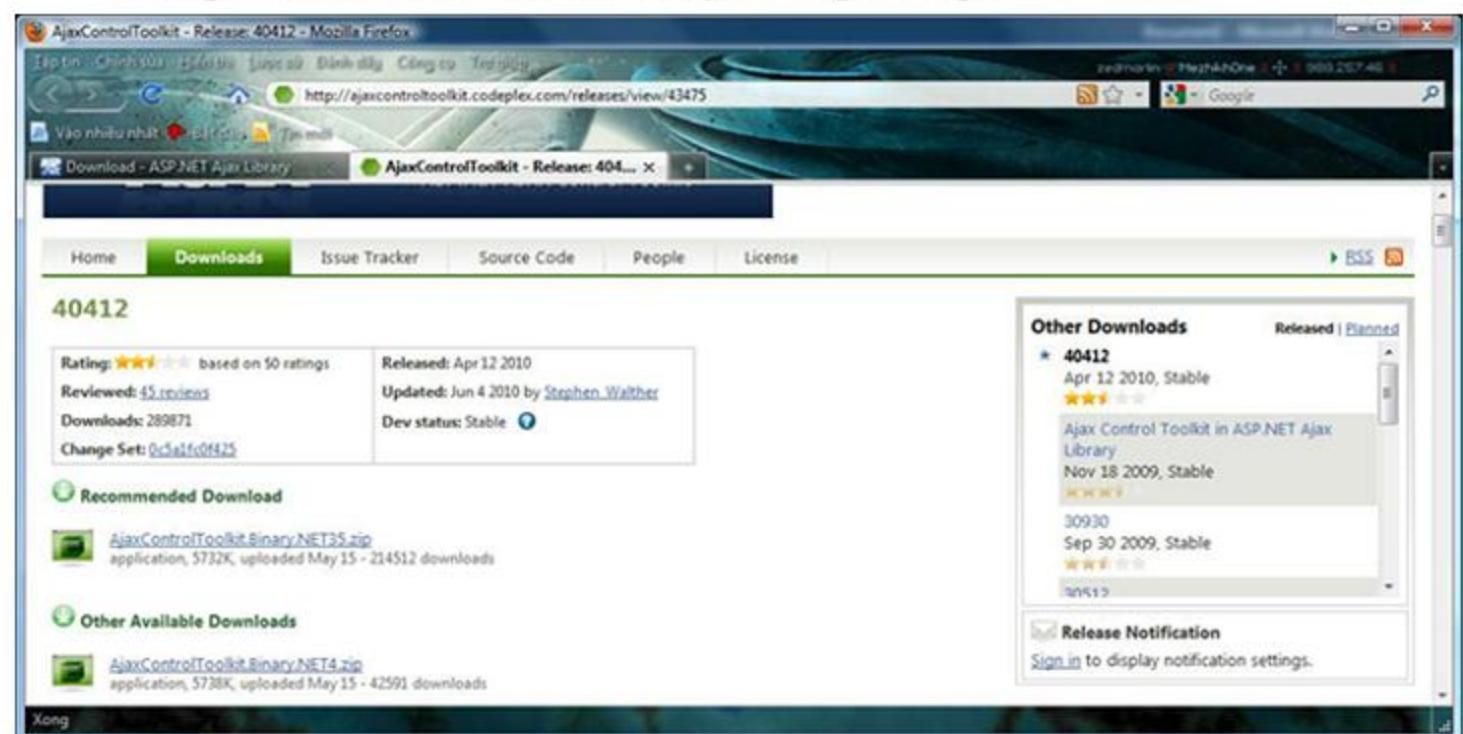
Nhấp vào liên kết **Ajax Control Toolkit Sample Site**, chuyển sang trang:



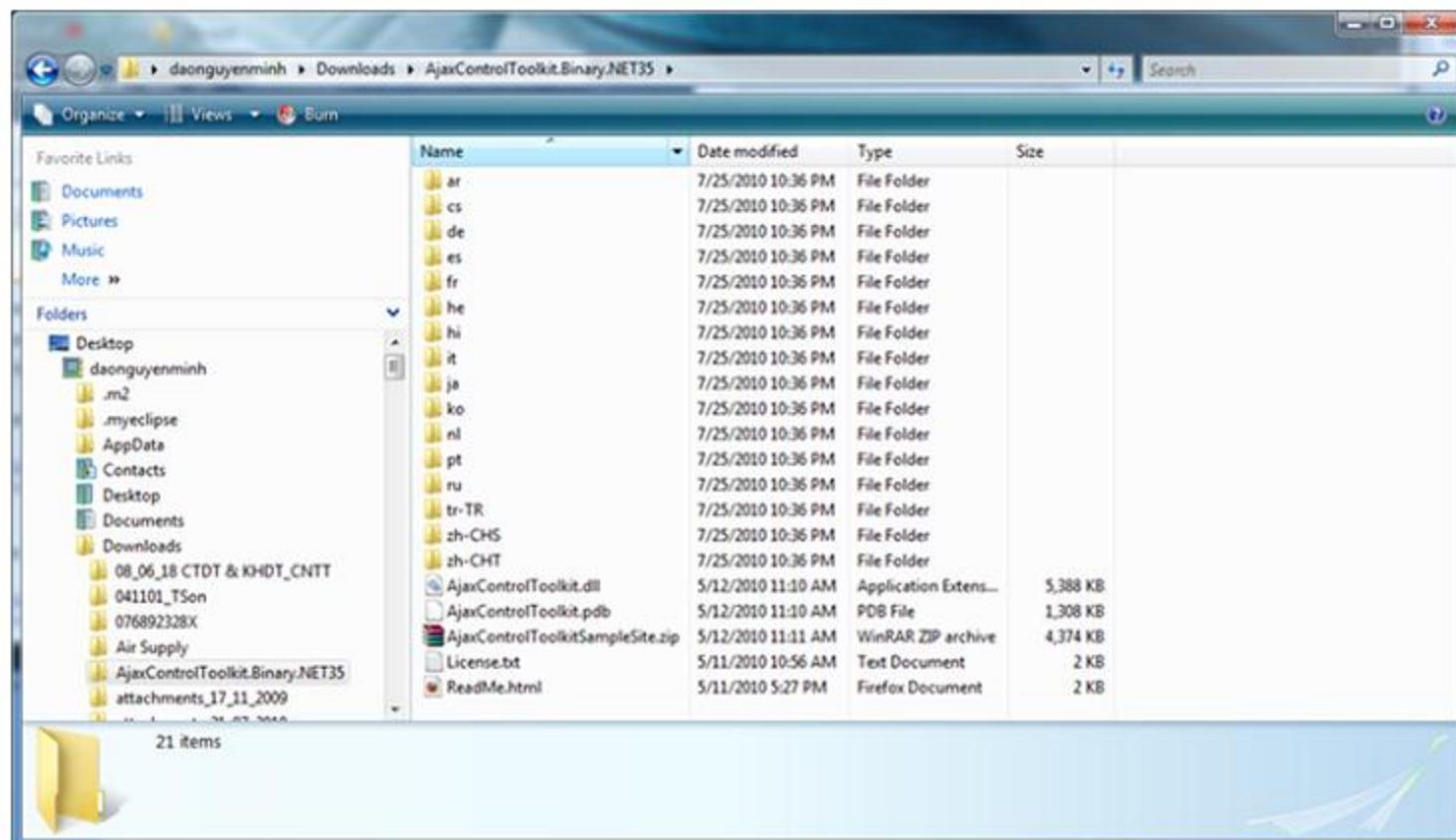
Và click nút Download, sẽ chuyển đến trang:



Nhấp mục Download để chuyển tiếp trang:



Nhấp chọn Ajax Control Toolkit Binary.NET35.zip để tải gói Ajax Control Toolkit 3.5 về máy. Sau đó, bạn giải nén vào trong thư mục trên ổ đĩa, giả sử ổ đĩa F:\ trong thư mục sau F:\Downloads\AjaxControlToolkit.Binary.NET35

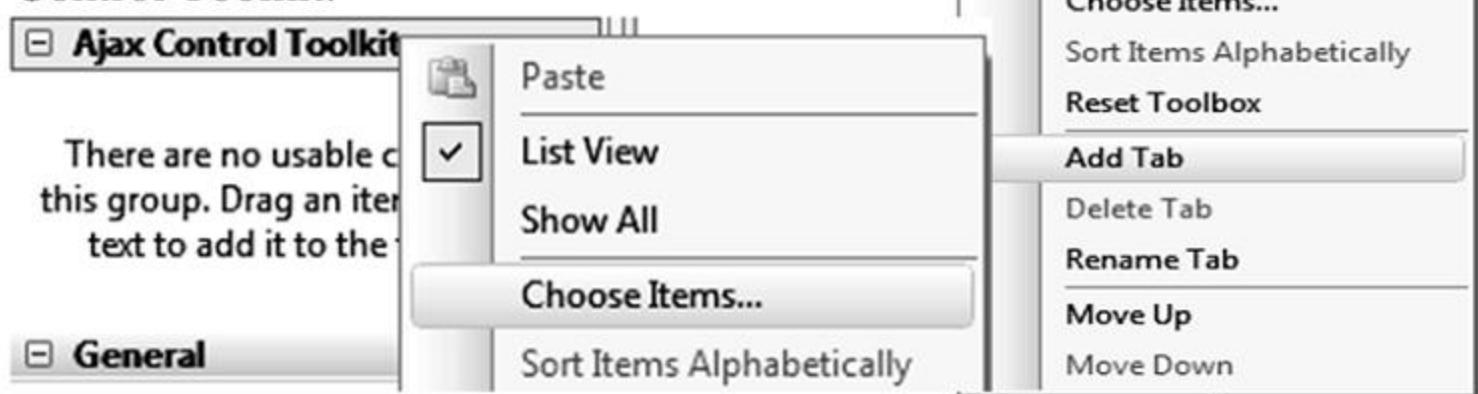


Để ý bạn sẽ thấy có gói AjaxControlToolkitSampleSite.zip, đây là toàn bộ các thí dụ đi kèm với gói Ajax Control Toolkit 3.5 để bạn có thể tìm hiểu toàn bộ các Ajax controls qua các bài tập tương ứng.

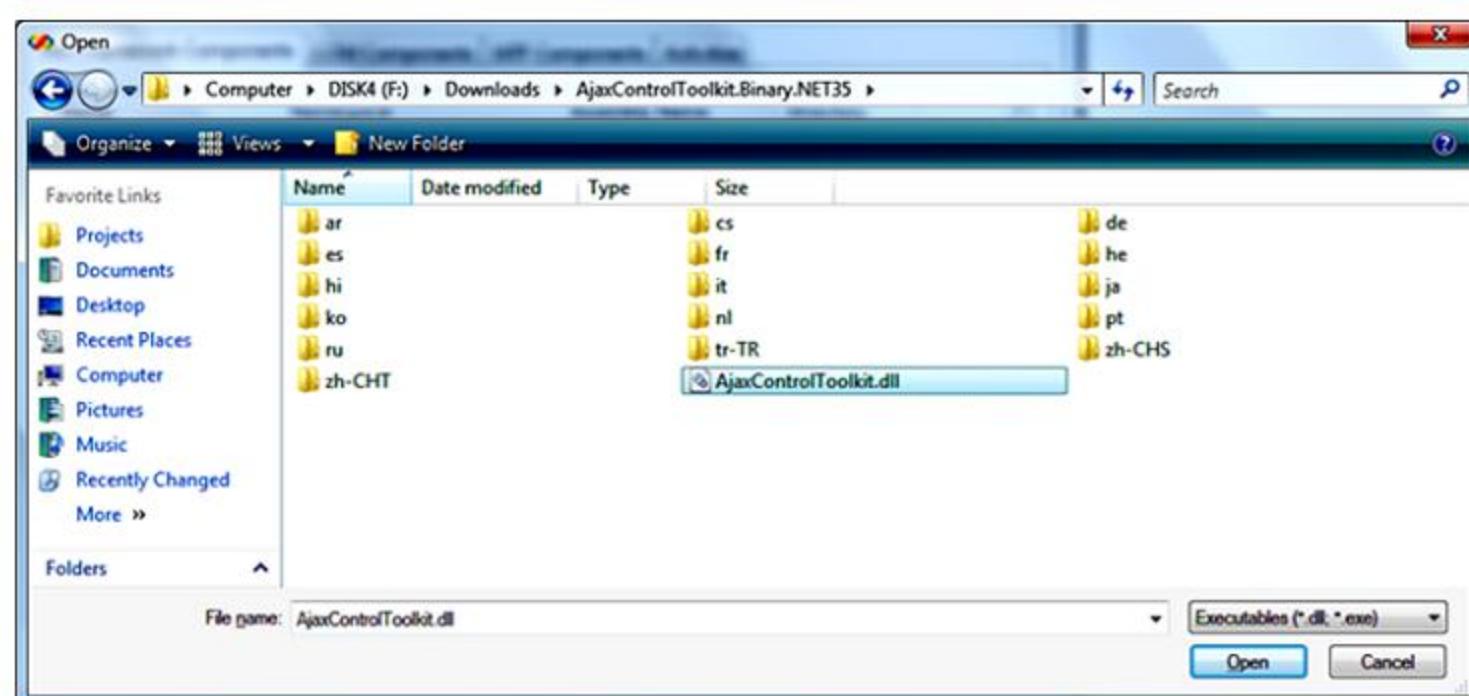
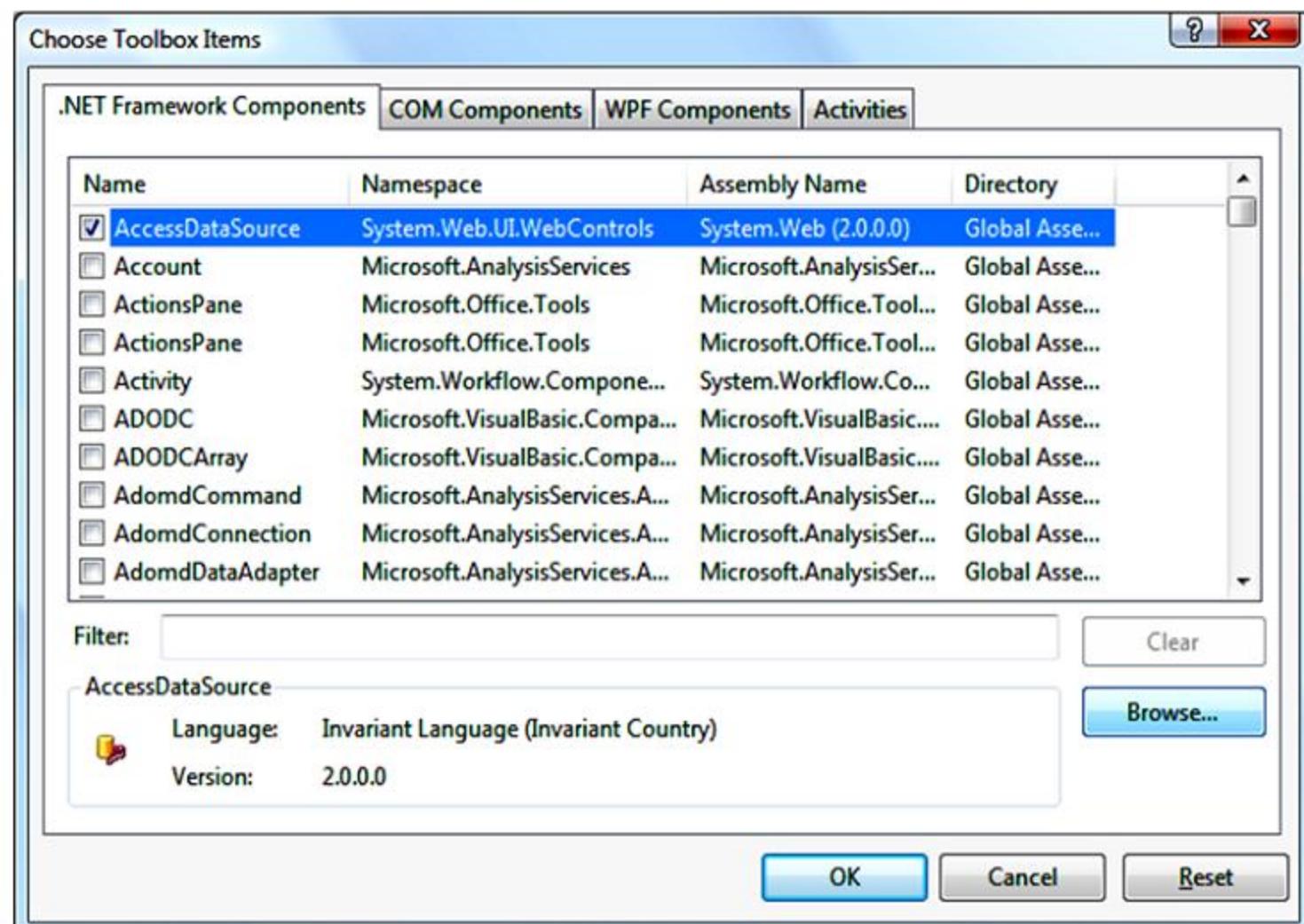
11.5.2. Thêm Ajax Control Toolkit vào trong Visual Studio Toolbox

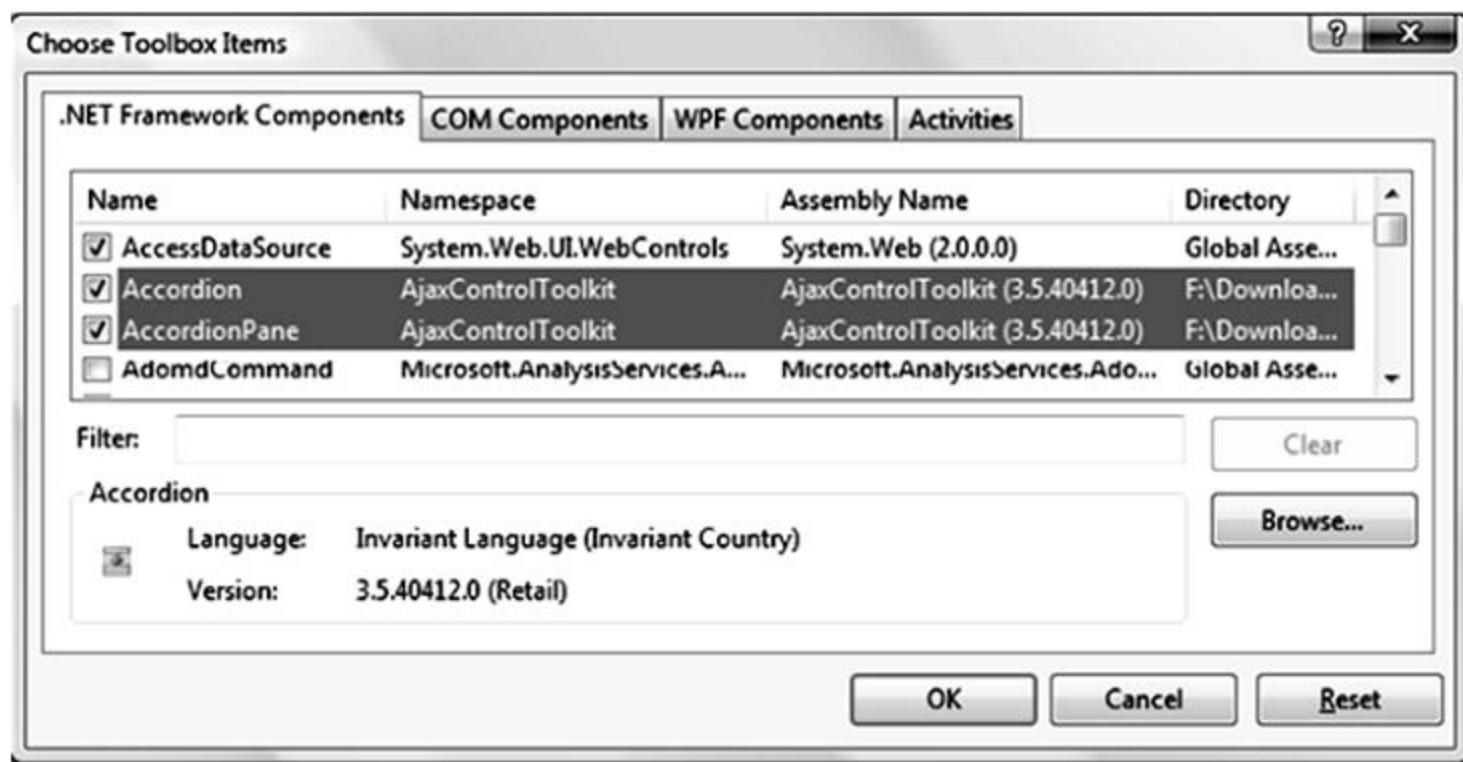
Thực hiện theo các bước sau:

- Khởi động Visual Studio và tạo ra một website ASP.NET mới. Mở trang Default.aspx trong trình soạn thảo Visual Studio.
- Mở Hộp công cụ (toolbox) và tạo ra một tab bằng cách nhấn chuột phải và chọn mục Add Tab. Tên tab mới là Ajax Control Toolkit.

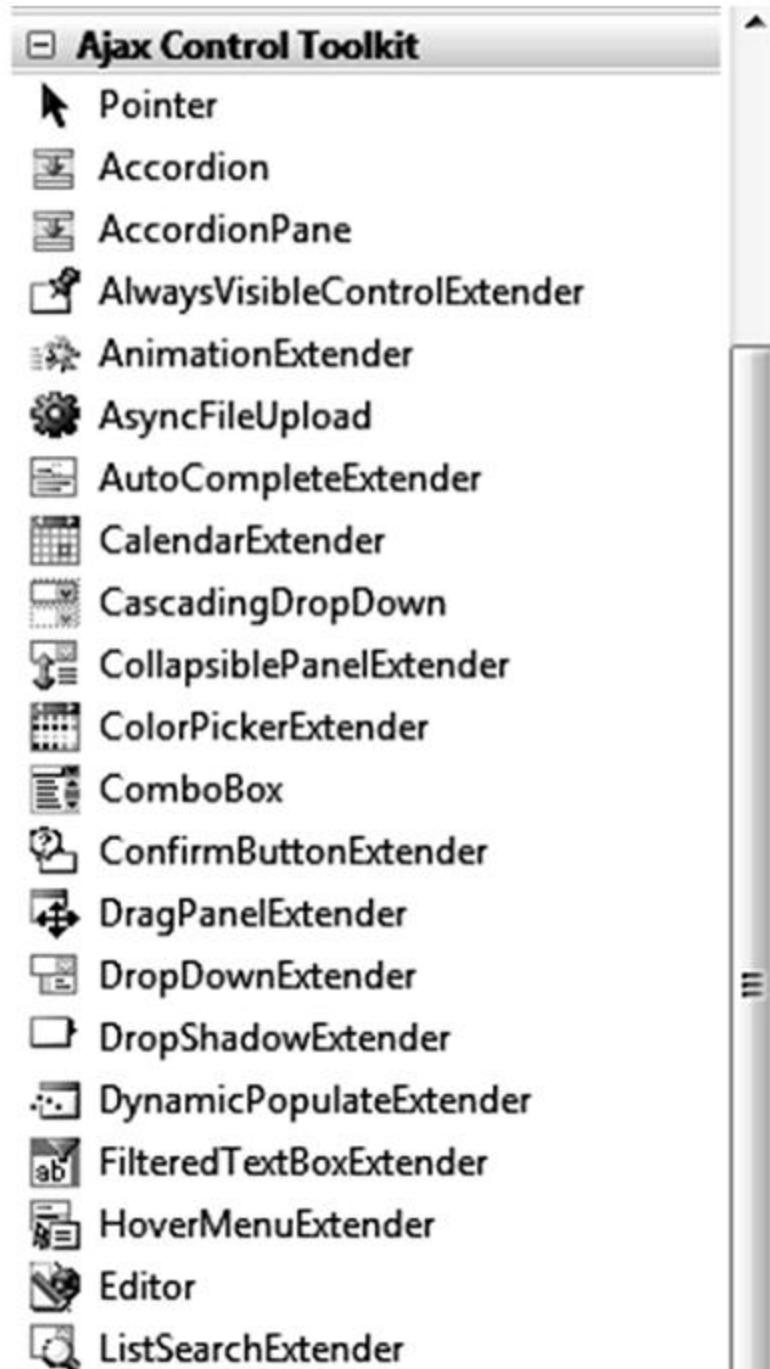


3. Nhấp chuột phải vào bên dưới tab mới và chọn Choose Items..., xuất hiện hộp thoại Choose Toolbox Items, nhấp vào nút Browse và duyệt đến thư mục mà bạn giải nén gói Ajax Control Toolkit. Chọn tập tin AjaxControlToolkit.dll và nhấp vào nút Open để chọn các Ajax Controls đưa vào và chuyển về lại hộp thoại Choose Toolbox Items, nhấp nút OK để hoàn tất.





Lúc này, trong cửa sổ Toolbox, tab Ajax Control Toolkit vừa mới thêm vào đã có các Ajax controls trong đó như hình:



11.6. TÌM HIỂU MỘT SỐ ĐIỀU KHIỂN TRONG AJAX CONTROL TOOLKIT 3.5

11.6.1. Accordion Control

Accordion control giúp bạn định nghĩa nhiều lớp và trình bày chúng từng cái một. Nó giống như là có nhiều CollapsiblePanel controls mà chỉ có một có thể mở rộng tại một thời điểm xác định. Accordion control chứa một hoặc nhiều AccordionPane controls. Mỗi AccordionPane control có chứa phần header và phần content của nó.

Accordion control hỗ trợ chế độ AutoSize sau để tùy biến với nhiều cỡ trang khác nhau:

- None: Accordion control mở rộng và thu hẹp không hạn chế
- Limit: Accordion control mở rộng và thu hẹp không quá giá trị của thuộc tính Height
- Fill: Accordion control bị cố định bởi giá trị của thuộc tính Height

Accordion cũng có thể là data-bound, để gắn dữ liệu vào control, chỉ ra nguồn dữ liệu sử dụng thuộc tính Datasource hoặc DataSourceID, và sau đó đặt các gói dữ liệu vào thuộc tính HeaderTemplate và ContentTemplate. Bạn phải gọi phương thức DataBind để gắn dữ liệu vào điều khiển.

❖ Phía server

```
<ajaxToolkit:Accordion  
    ID="MyAccordion"  
    runat="Server"  
    SelectedIndex="0"  
    HeaderCssClass="accordionHeader"  
    HeaderSelectedCssClass="accordionHeaderSelected"  
    ContentCssClass="accordionContent"  
    AutoSize="None"  
    FadeTransitions="true"  
    TransitionDuration="250"  
    FramesPerSecond="40"  
    RequireOpenedPane="false"  
    SuppressHeaderPostbacks="true">
```

```

<Panes>
  <ajaxToolkit:AccordionPane
    HeaderCssClass="accordionHeader"
    HeaderSelectedCssClass="accordionHeaderSelected"
    ContentCssClass="accordionContent">
    <Header>... </Header>
    <Content>... </Content>
  </ajaxToolkit:AccordionPane>
  .
  .
</Panes>
<HeaderTemplate>...</HeaderTemplate>
<ContentTemplate>...</ContentTemplate>
</ajaxToolkit:Accordion>

```

Thuộc tính

- SelectedIndex – Đối tượng AccordionPane được hiển thị.
- HeaderCssClass – Tên của lớp CSS dùng cho headers. Nó có thể được áp dụng cho Accordion control như là mặc định cho tất cả đối tượng AccordionPane, hoặc nó có thể áp dụng cho một đối tượng Accordion riêng.
- HeaderSelectedCssClass – Tên của lớp CSS dùng cho header được chọn. Nó có thể được áp dụng cho Accordion control như là mặc định cho tất cả đối tượng AccordionPane, hoặc nó có thể áp dụng cho một đối tượng Accordion riêng.
- ContentCssClass – Tên lớp CSS dùng cho phần nội dung. Nó có thể được áp dụng cho Accordion control như là mặc định cho tất cả đối tượng AccordionPane, hoặc nó có thể áp dụng cho một đối tượng Accordion riêng.
- FadeTransitions - true để sử dụng hiệu ứng chuyển đổi; ngược lại, false.
- TransitionDuration – Số giây thực hiện hiệu ứng chuyển đổi.
- FramesPerSecond – Số khung trên một giây được dùng để chuyển đổi hiệu ứng.

- AutoSize – Giá trị chỉ ra giới hạn của phần thể hiện.
- RequireOpenedPane – Giá trị chỉ ra rằng phần đang mở sẽ không đóng lại khi mà header của nó được click, để đảm bảo một pane luôn luôn mở, mặc định là true.
- SuppressHeaderPostbacks – Giá trị chỉ ra khi mà một phần tử bên trong header được gọi, nó hữu ích khi mà bạn muốn đặt hyperlinks trong header.
- Panes – Tập hợp AccordionPane.
- HeaderTemplate – Các yếu tố có chứa đánh dấu, nên được sử dụng cho tiêu đề của một cửa sổ khi có ràng buộc dữ liệu.
- ContentTemplate – Phần tử chứa phần mở rộng sẽ thể hiện cho phần nội dung trong khi dữ liệu được gắn.
- DataSource – Nguồn dữ liệu được sử dụng. Bạn phải gọi phương thức Databind để điều khiển đọc dữ liệu từ nguồn dữ liệu.
- DataSourceID – ID của datasource.
- DataMember – Đối tượng được gắn vào khi sử dụng DataSourceID cho việc gắn dữ liệu

Phương thức

Tên	Mô tả
Constructor	Constructor mặc định để báo cho ASP.NET trả lại nó như là thẻ DIV
ClearPanes	Xóa toàn bộ nội dung các Pane
ConnectToDataSourceView	Nối điều khiển ranh giới dữ liệu này tới DataSourceView thích hợp và móc trình nghe sự kiện thích hợp (cho) sự kiện DataSourceViewChanged. Giá trị trả về là thẻ hiện mới (nếu có) mà được nối tới. Một exception sẽ được ném ra nếu có vấn đề trong việc tìm kiếm thẻ hiện hoặc nguồn dữ liệu
CreateChildControls	Tạo một AccordionExtender và gắn nó vào thẻ div sẽ được khởi tạo cho điều khiển này

CreateControlHierarchy(System.Boolean)	Tạo ra sự phân cấp điều khiển mới (của) AccordionPanes (sử dụng DataSource nếu được chỉ ra)
CreateDataSourceSelectArguments	Tạo ra DataSourceSelectArguments (với giá trị mặc định là Empty bởi vì chúng ta ko muốn sắp xếp, lọc,...)
CreateItem(System.Object, System.Int32, AjaxControlToolkit.AccordionItemType, AjaxControlToolkit.AccordionContentPanel, System.Web.UI.ITemplate, System.Boolean)	Tạo một mục AccordionPane (cả Header và Content) và xuất hiện sự kiện ItemCreated
.DataBind	Gắn Accordion với nguồn dữ liệu của nó
DoSelect(System.Collections.IEnumerable)	Chọn dữ liệu
EnsureDataBound	Chắc chắn rằng Accordion đã gắn với một nguồn dữ liệu nếu nó bắt buộc như thế
FindControl(System.String)	Override FindControl để tìm control này, sau đó kiểm vào các phần con của nó
GetData	Trả về một Ienumerable của nguồn dữ liệu, có thể từ thuộc tính DataSource hoặc từ thuộc tính DataSourceID
OnBubbleEvent(System.Object, System.EventArgs)	Gói CommandArgs của một sự kiện ItemCommand với AccordionCommandEventEventArgs
OnDataBinding(System.EventArgs)	Gắn nội dung vào các phần của nó
OnDataPropertyChanged	Phương thức này được gọi khi DataMember, DataSource hoặc DataSourceID bị thay đổi
OnDataSourceViewChanged(System.Object, System.EventArgs)	Báo cho chúng ta biết cần phải báo dữ liệu khi DataSourceView bị thay đổi

OnInit(System.EventArgs)	Điều khiển sự kiện PreLoad của trang
OnItemCommand(AjaxControlToolkit.AccordionCommandEventArgs)	Báo sự kiện ItemCommand
OnItemCreated(AjaxControlToolkit.AccordionEventArgs)	Báo sự kiện ItemCreated
OnItemDataBound(AjaxControlToolkit.AccordionEventArgs)	Báo sự kiện ItemDatabound
OnLoad(System.EventArgs)	Kết nối tới DataSourceView và phát hiện nếu chúng ta vẫn cần bao dữ liệu
OnPagePreLoad(System.Object, System.EventArgs)	OnPreLoad được sử dụng để phát hiện chúng ta còn cần bao dữ liệu hay ko
OnPreRender(System.EventArgs)	Đánh dấu AccordionPane sao cho nó không bị thu gọn

Sự kiện

Tên	Mô tả
ItemCommand	Sự kiện xảy ra khi lệnh được thực thi
ItemCreated	Sự kiện xảy ra khi một mục được tạo khi bao dữ liệu
ItemDataBound	Sự kiện xảy ra khi bao dữ liệu

❖ Phía Client

Thuộc tính

Tên	Mô tả
AutoSize	Chế độ autosize
Count	
FadeTransitions	Làm mờ khi thay đổi các pane
FramesPerSecond	Số lượng các bước trong một giây trong quá trình chuyển đổi. Mặc định là 30 khung hình trong một giây
HeaderCssClass	Thiết lập CSS cho phần header

HeaderSelectedCssClass	Thiết lập CSS cho phần header được chọn
Pane	Lấy giá trị index của Accordion được chỉ ra, nếu không có thì lấy giá trị index của pane đang được chọn
requireOpenedPane	Thiết lập chế độ bắt buộc phải có một pane được mở
SelectedIndex	Index của phần đang được chọn

Phương thức

Tên	Mô tả
addPane	Tạo một thẻ Accordion mới tham chiếu đến header và nội dung của nó và thêm nó vào tập hợp panes
dispose	Vứt bỏ các AccordionBehavior
initialize	Các chức năng khởi tạo có trách nhiệm nhận các chỉ số được lựa chọn từ cơ chế ClientState và duyệt qua tất cả các phần tử con sau đó xây dựng một tập các phần đó, đóng các phần không được chọn, chỉ trình bày phần được chọn hoặc mặc định
raiseSelectedIndexChanged	Xử lý sự kiện raiseSelectedIndexChanged
raiseSelectedIndexChanging	Xử lý sự kiện raiseSelectedIndexChanging

Sự kiện

Tên	Mô tả
selectedIndexChanged	Thêm một điều khiển sự kiện cho sự kiện selectedIndexChanged
selectedIndexChanging	Thêm một điều khiển sự kiện cho sự kiện selectedIndexChanging

11.6.2. AlwaysVisible Control

AlwaysVisible Control là một extender được dùng để hiển thị một cách liên tục đến một điều khiển ASP.NET. Điều khiển được mở rộng sẽ luôn luôn di chuyển đến một vị trí cố định trên trang bất kể trang thị thay đổi kích thước hay bị cuộn.

❖ Phía Server

Thuộc tính

Tên	Mô tả
HorizontalOffset	Khoảng cách tới cạnh ngang của trình duyệt tính bằng pixel cùng phía với điều khiển đích, mặc định là 0px
HorizontalSide	Phía của điều khiển phụ thuộc vào, mặc định là bên trái
ScrollEffectDuration	Thời gian tính bằng giây cho hiệu ứng cuộn khi điều khiển đích thay đổi vị trí, mặc định là 0.1s
UseAnimation	Cho phép hay không hiệu ứng một phần từ vào vị trí
VerticalOffset	Khoảng cách tới cạnh dọc của trình duyệt tính bằng pixel cùng phía với điều khiển đích, mặc định là 0px
VerticalSide	Phía của điều khiển phụ thuộc vào, mặc định là bên trên

Phương thức

Tên	Mô tả
EnsureValid	Xác nhận các tính hợp lệ

❖ Phía Client

Các hành vi AlwaysVisibleControl được sử dụng để cố định điều khiển liên quan với một khoảng cách đã chỉ ra trước tính từ góc trên bên trái dù người dùng có thay đổi kích thước hay kéo thanh cuộn.

Thuộc tính

Tên	Mô tả
HorizontalOffset	Khoảng cách tới cạnh ngang của trình duyệt tính bằng pixel cùng phía với điều khiển đích, mặc định là 0px
HorizontalSide	Phía của điều khiển phụ thuộc vào, mặc định là bên trái
ScrollEffectDuration	Thời gian tính bằng giây cho hiệu ứng cuộn khi điều khiển đích thay đổi vị trí, mặc định là 0.1s

useAnimation	Cho phép hay không hiệu ứng một phần tử vào vị trí (với IE6 giá trị mặc định là <i>true</i>)
VerticalOffset	Khoảng cách tới cạnh dọc của trình duyệt tính bằng pixel cùng phía với điều khiển đích, mặc định là 0px
VerticalSide	Phía của điều khiển phụ thuộc vào, mặc định là bên trên

Phương thức

Tên	Mô tả
dispose	Hủy bỏ các hành vi
initialize	Khởi tạo các hành vi
raiseRepositioned	Mô tả sự kiện thay đổi vị trí
raiseRepositioning	Mô tả sự kiện tái định vị

Sự kiện

Tên	Mô tả
repositioned	Thêm một xử lý sự kiện cho sự kiện thay đổi vị trí
repositioning	Thêm một xử lý sự kiện cho sự kiện tái định vị

Thí dụ

```
<asp:ToolkitScriptManager ID="ToolkitScriptManager1"
runat="server"> </asp:ToolkitScriptManager>

<asp:Panel ID="Panel1" CssClass="staticPanel" runat="server">
<h2>Hello World!</h2> </asp:Panel>

<asp:AlwaysVisibleControlExtender
    ID="AlwaysVisibleControlExtender1"
    TargetControlID="Panel1"      VerticalSide="Top"
    VerticalOffset="10"          HorizontalSide="Right"
    HorizontalOffset="10"         runat="server" />
```

11.6.3. Animation control

Phần mở rộng được sử dụng để tạo các hiệu ứng cho điều khiển khi xảy ra sự kiện.

❖ Phía Server

Thuộc tính

Tên	Mô tả
OnClick	Hiệu ứng tương ứng với hành động Click
OnHoverOut	Hiệu ứng tương ứng với hành động hoverOut
OnHoverOver	Hiệu ứng tương ứng với hành động HoverOver
Onload	Hiệu ứng tương ứng với hành động Onload
OnMouseOut	Hiệu ứng tương ứng với hành động MouseOut
OnMouseOver	Hiệu ứng tương ứng với hành động MouseOver

Phương thức

Tên	Mô tả
OnPreRender(System.EventArgs)	Thay đổi bất kỳ animation target nào từ ID máy chủ điều khiển sang ClientIDs mà mã lệnh hiệu ứng hướng tới

❖ Phía Client

Animation là một lớp cơ sở trừu tượng được dùng như là một điểm khởi đầu cho tất cả các hiệu ứng khác. Nó cung cấp các cơ chế cơ bản cho hiệu ứng (playing, pausing, stopping, timing,...) và các hiệu ứng thực tế sẽ hoàn tất trong các phương thức trừu tượng getAnimatedValue và setValue.

Thuộc tính

Tên	Miêu tả
duration	Độ dài thực thi hiệu ứng, được tính bằng giây, mặc định là 1s
Events	Tập hợp các điều khiển sự kiện cho hành vi, thuộc tính này chỉ nên được gọi bởi các hành vi, không nên gọi bởi các mã lệnh khác
fps	Số khung hình trong 1 giây, mặc định là 25
Id	
isActive	True nếu hiệu ứng đang hoạt động, false nếu ngược lại
isInitialized	

isPlaying	True nếu hiệu ứng đang chạy, false nếu ngược lại
isUpdating	
percentComplete	Phần trăm hiệu ứng đã được chạy
target	

Phương thức:

Tên	Miêu tả
beginUpdate	
Dispose	Hủy bỏ các hành động
endUpdate	
getAnimatedValue	Xác định trạng thái của hiệu ứng sau một lượng thời gian trôi qua
Initialize	
Interpolate	Hàm này được dùng để xác định giá trị thích hợp giữa giá trị bắt đầu và kết thúc để đưa ra phần trăm hiện tại
onEnd	Phương thức này chỉ được gọi sau mỗi lần hiệu ứng thực thi xong
onStart	Phương thức này chỉ được gọi trước mỗi lần hiệu ứng thực thi xong
onStep	Phương thức này được gọi lặp lại mỗi bước tiến triển của hiệu ứng qua mỗi khung hình
pause	Dừng hiệu ứng nếu nó đang được thực thi, gọi play để chơi tiếp
Play	Chạy hiệu ứng hoặc chơi tiếp nếu pause
Play	Tạo hiệu ứng, chạy hết và hủy nếu đã xong
raiseEnded	Kích hoạt sự kiện ended
raisePropertyChanged	Thay đổi khai báo sự kiện
raiseStarted	Kích hoạt sự kiện started
raiseStep	Kích hoạt sự kiện step
setOwner	Làm cho hiệu ứng này là con của hiệu ứng khác

setValue	Thiết lập trạng thái hiện tại của hiệu ứng
Stop	Dừng chạy hiệu ứng
updated	

Sự kiện

Tên	Miêu tả
disposing	
Ended	Thêm một điều khiển sự kiện cho sự kiện ended
PropertyChanged	
Started	Thêm một điều khiển sự kiện cho sự kiện started
step	Thêm một điều khiển sự kiện cho sự kiện step

Thí dụ

```
<asp:ToolkitScriptManager ID="ToolkitScriptManager1" runat="server">
</asp:ToolkitScriptManager>

<asp:Button ID="btn1" runat="server" Text="Button" OnClientClick="return false;" />
<asp:Panel ID="panel1" runat="server" CssClass="panel1">
    ASP.NET AJAX is a free framework for quickly creating a new generation of
    more efficient,
    more interactive and highly-personalized Web experiences that work across all the
    most popular browsers.<br />
</asp:Panel>
<asp:AnimationExtender ID="AnimationExtender1" runat="server"
    TargetControlID="btn1">
    <Animations>
        <OnClick>
        <Sequence>
            <EnableAction enabled="false"/>
            <Sequence AnimationTarget="Panell">
                <Resize Width="500" Height="110" Unit="px" />
                <FadeOut Duration="1.5" Fps="24"/>
            </Sequence>
            <EnableAction enabled="true"/>
        
```

```
</Sequence>  
</OnClick>  
</Animations></asp:AnimationExtender>
```

Các hiệu ứng

➤ Hiệu ứng:

- ParentAnimation:
 - ParallelAnimation
 - SequenceAnimation
- SelectionAnimation:
 - ConditionAnimation
 - CaseAnimation
- FadeAnimation:
 - FadeInAnimation
 - FadeOutAnimation
- PulseAnimation
- PropertyAnimation:
 - DiscreteAnimation
 - InterpolatedAnimation
 - ColorAnimation
 - LengthAnimation

- MoveAnimation
- ResizeAnimation
- ScaleAnimation

➤ Action:

- EnableAction
- HideAction
- StyleAction
- OpacityAction
- ScriptAction

11.6.4. AsyncFileUpload Control

AsyncFileUpload cho phép bạn upload file lên server một cách bất đồng bộ, kết quả của file upload có thể được kiểm tra cả ở phía server lẫn client, bạn có thể lưu file được upload bằng cách gọi phương thức SaveAs() trong điều khiển sự kiện UploadedComplete ở server.

❖ Phía Server

Thuộc tính

Tên	Mô tả
CompleteBackColor	Lấy hoặc thiết lập giá trị màu nền khi file đang được upload, mặc định là Lime
ContentType	Lấy kiểu nội dung của file upload
ErrorBackColor	Lấy hoặc thiết lập giá trị màu nền khi file upload bị lỗi, mặc định là red
FileBytes	Xác định dung lượng file vừa upload
FileContent	Tạo một đối tượng Stream trả về file vừa được upload để chuẩn bị cho việc đọc file
FileName	Xác định tên file vừa upload
HasFile	Trả về giá trị true nếu file đã được upload
IsUploading	Trả về giá trị true nếu file đang được upload
OnClientUploadComplete	Lấy hoặc thiết lập đoạn mã trên client sẽ được thực thi sau khi file đã được upload
OnClientUploadError	Lấy hoặc thiết lập đoạn mã trên client sẽ được thực thi nếu có lỗi xảy ra trong khi upload
OnClientUploadStarted	Lấy hoặc thiết lập đoạn mã trên client sẽ được thực thi khi file bắt đầu được upload
PersistedStoreType	Xác lập bộ lưu trữ để điều khiển, mặc định là Session
PostedFile	Lấy một đối tượng HttpPostedFile để truy xuất file vừa upload
ThrobberID	Lấy hoặc thiết lập ID của điều khiển được thể hiện trong khi file đang được upload
UploaderStyle	Xác định hoặc thiết lập thể hiện của điều khiển (Traditional, Modern), mặc định là Traditional

UploadingBackColor	Xác định hoặc thiết lập màu nền của điều khiển trong khi file đang upload, mặc định là white
Width	Xác định hoặc thiết lập chiều rộng của điều khiển

Phương thức

Tên	Mô tả
Constructor	Khởi tạo một AsyncFileUpload mới
ClearAllFilesFromPersistedStore	Xóa toàn bộ file trong bộ lưu trữ
ClearFileFromPersistedStore	Xóa một tập tin cụ thể trong bộ lưu trữ
GetBytesFromStream(System.IO.Stream)	Nhận một chuỗi byte từ luồng file
SaveAs(System.String)	Lưu file đã upload với tên cụ thể

Sự kiện

Tên	Mô tả
UploadedComplete	Sự kiện xảy ra ở server nếu file upload thành công
UploadedFileError	Sự kiện xảy ra ở server nếu file upload bị lỗi

❖ Phía Client (không có).

11.6.5. AutoComplete Control

❖ Phía Server

Phần mở rộng cung cấp gợi ý để hoàn tất phần nhập vào textbox.

Thuộc tính

Tên	Mô tả
CompletionInterval	Thời gian tính bằng ms để nhận được gợi ý sử dụng web service
CompletionListCssClass	Lớp Css được sử dụng để thể hiện phần nội dung hiện ra

CompletionListElementID	ID của phần tử sẽ là danh sách hoàn thành
CompletionListHighlightedItemCssClass	Css Class sẽ được sử dụng để tô đậm một mục trong danh sách tự động hoàn chỉnh
CompletionListItemCssClass	Css Class sẽ được sử dụng để định dạng style một mục trong danh sách tự động hoàn chỉnh
CompletionSetCount	Số lượng các đề xuất để được cung cấp
ContextKey	
DelimiterCharacters	Xác định hoặc thiết lập ký tự dùng để phân cách các từ cho autocomplete
EnableCaching	Cho phép bộ nhớ đệm phía client hoạt động
FirstRowSelected	Xác định nếu dòng đầu tiên của kết quả tìm kiếm được xác định
MinimumPrefixLength	Số lượng ít nhất của chuỗi ký tự trước khi webservice đưa ra gợi ý
OnClientHidden	Điều khiển gắn vào client sự kiện hidden
OnClientHiding	Điều khiển gắn vào client sự kiện hidding
OnClientItemOut	Điều khiển gắn vào client sự kiện item out
OnClientItemOver	Điều khiển gắn vào client sự kiện item over
OnClientItemSelected	Điều khiển gắn vào client sự kiện item selected

OnClientPopulated	Điều khiển gắn vào client sự kiện populated
OnClientPopulating	Điều khiển gắn vào client sự kiện populating
OnClientShowing	Điều khiển gắn vào client sự kiện showing
OnClientShown	Điều khiển gắn vào client sự kiện shown
OnHide	Hiệu ứng onhide
OnShow	Hiệu ứng onshow
ServiceMethod	Phương thức dịch vụ web được gọi
ServicePath	Đường dẫn đến web service mà extender sẽ lấy từ hoặc câu, nếu không được cung cấp, mặc định là phương thức trang
ShowOnlyCurrentWordInCompletionListItem	Nếu được thiết lập là true thì phần gợi ý chỉ cho các từ hiện hành, ngược lại thì đưa ra chuỗi chứa các từ hiện hành, đây là mặc định
UseContextKey	Cho dù thuộc tính ContextKey có được dùng hay không thì thuộc tính cũng được enable, nếu ContextKey được thiết lập thì giá trị này có tham số contextKey cùng giá trị kiểu string

Phương thức

Tên	Mô tả
CreateAutoCompleteItem(System.String, System.String)	Tạo một đối tượng JSON thể hiện text/value có thể được trả về bởi webservice

OnPreRender(System.EventArgs)	Chuyển đổi server IDs thành client IDs cho hiệu ứng
-------------------------------	---

❖ Phía Client

Thuộc tính

Tên	Mô tả
CompletionInterval	Tự động hoàn thành tính bằng ms
completionList	Danh sách phần tử dom
CompletionListCssClass	Lớp Css được sử dụng để thể hiện phần nội dung hiện ra
CompletionListElementID	ID của phần tử sẽ là danh sách hoàn thành
CompletionListItemCssClass	Css Class sẽ được sử dụng để định dạng style một mục trong danh sách tự động hoàn chỉnh
completionSetCount	Kích thước tối đa của completion
contextKey	
delimiterCharacters	Xác định hoặc thiết lập ký tự dùng để phân cách các từ cho autocomplete
EnableCaching	Cho phép bộ nhớ đệm phía client hoạt động
FirstRowSelected	Xác định nếu dòng đầu tiên của kết quả tìm kiếm được xác định
highlightedItemCssClass	Lớp css được dùng để định dạng phần tử được tô đậm trong danh sách
isMultiWord	Chế độ multi-word
minimumPrefixLength	Số ký tự ít nhất để web service đưa ra gợi ý
onHide	Định nghĩa JSON của hiệu ứng onHide

onHideBehavior	Hành vi của hiệu ứng onHide
onShow	Định nghĩa JSON của hiệu ứng onShow
onShowBehavior	Hành vi của hiệu ứng onShow
serviceMethod	Phương thức web service
servicePath	Web service url
showOnlyCurrentWordInCompletionList Item	Nếu được thiết lập là true thì phần gợi ý chỉ cho các từ hiện hành, ngược lại thì đưa ra chuỗi chứa các từ hiện hành, đây là mặc định
useContextKey	Cho dù thuộc tính ContextKey có được dùng hay không thì thuộc tính cũng được enable, nếu ContextKey được thiết lập thì giá trị này có tham số contextKey cùng giá trị kiểu string

Phương thức

Tên	Mô tả
dispose	Xử lý hành vi autocomplete
hidePopup	Dấu danh sách hoàn tự động hoàn tất
initialize	Khởi tạo hành vi autocomplete
initializeCompletionList	Khởi tạo danh sách các phần tử autocomplete
initializeTextBox	Khởi tạo textbox
initializeTimer	Khởi tạo thiết bị tính giờ
onHide	Chạy hiệu ứng onHide
onShow	Chạy hiệu ứng onShow
raiseHidden	Kích hoạt sự kiện hidden
raiseHiding	Kích hoạt sự kiện hiding
raiseItemOut	Kích hoạt sự kiện item out
raiseItemOver	Kích hoạt sự kiện item over

raiseItemSelected	Kích hoạt sự kiện item selected
raisePopulated	Kích hoạt sự kiện populated
raisePopulating	Kích hoạt sự kiện populating
raiseShowing	Kích hoạt sự kiện showing
raiseShown	Kích hoạt sự kiện shown
showPopup	Hiện danh sách gợi ý

Sự kiện

Tên	Mô tả
Hidden	Thêm điều khiển sự kiện cho sự kiện Hidden
hiding	Thêm điều khiển sự kiện cho sự kiện hiding
itemOut	Thêm điều khiển sự kiện cho sự kiện itemOut
itemOver	Thêm điều khiển sự kiện cho sự kiện itemOver
itemSelected	Thêm điều khiển sự kiện cho sự kiện itemSelected
Populated	Thêm điều khiển sự kiện cho sự kiện populated
Populating	Thêm điều khiển sự kiện cho sự kiện populating
Showing	Thêm điều khiển sự kiện cho sự kiện showing
shown	Thêm điều khiển sự kiện cho sự kiện shown

Thí dụ

```
<ajaxToolkit:AutoCompleteExtender
    runat="server"
    ID="autoComplete1"
    TargetControlID="myTextBox"
    ServiceMethod="GetCompletionList"
    ServicePath="AutoComplete.asmx"
    MinimumPrefixLength="2"
    CompletionInterval="1000"
    EnableCaching="true"
    CompletionSetCount="20"
    CompletionListCssClass="autocomplete_completionListElement" />
```

```

CompletionListItemCssClass="autocomplete_listItem"
CompletionListHighlightedItemCssClass="autocomplete_highlightedListItem"
    DelimiterCharacters=";,:"
    ShowOnlyCurrentWordInCompletionListItem="true">
<Animations>
<OnShow> ... </OnShow>
<OnHide> ... </OnHide>
</Animations>
</ajaxToolkit:AutoCompleteExtender>

```

11.6.6. Calender Control

❖ Phía Server

Điều khiển Calendar extender có thể được gắn với bất kỳ điều khiển textbox nào của ASP.NET. Nó cung cấp hàm định dạng ngày phía client với định dạng ngày tùy biến với giao diện trong một điều khiển popup. Bạn có thể tương tác với calendar bằng cách click vào một ngày để chọn, hoặc Today để chọn ngày hiện tại. Ngoài ra, mũi tên trái phải có thể được sử dụng để chuyển tới tháng trước hoặc tháng sau. Bằng cách click vào tiêu đề của calendar bạn có thể thay đổi hiển thị của các ngày trong tháng hiện tại thành các tháng trong năm hiện tại. Click khác sẽ chuyển các năm trong thập kỷ hiện tại. Hành động này cho phép bạn dễ dàng chuyển đến một ngày trong quá khứ hoặc tương lai từ điều khiển calendar.

Thuộc tính

Tên	Mô tả
Animated	Chuyển sang chế độ hoạt hình
ClearTime	Xóa thời gian
CssClass	Lớp Css được dùng để định dạng calendar
DaysModeTitleFormat	Định dạng ngày được hiển thị, giá trị mặc định là MMMM/YYYY
DefaultView	Xác định hoặc thiết lập hiển thị mặc định của calendar. Mặc định là thể hiện các ngày
EnabledOnClient	Hành vi này có sẵn cho phần tử hiện tại
FirstDayOfWeek	Xác định hoặc thiết lập ngày đầu tiên của tuần

Format	Định dạng chuỗi được dùng để thể hiện ngày được chọn, mặc định là ‘d’
OnClientDateSelectionChanged	Xác định hoặc thiết lập đoạn mã được thực thi khi một ngày mới được chọn
OnClientHidden	Thiết lập đoạn mã được thực hiện ngay sau khi calendar vừa ẩn
OnClientHiding	Thiết lập đoạn mã thực hiện ngay trước khi calendar ẩn
OnClientShowing	Thực thi đoạn mã lệnh ngay trước khi calendar vừa hiện
OnClientShown	Thực thi đoạn mã lệnh ngay sau khi calendar vừa hiện
PopupButtonID	ID của điều khiển để hiện calendar khi click, nếu giá trị không được thiết lập, calendar sẽ được hiện khi đưa trỏ chuột vào text box
PopupPosition	Thiết lập vị trí hiển thị của calendar, mặc định là góc dưới bên trái
SelectedDate	Thiết lập ngày mà calendar được khởi tạo cùng
TodaysDateFormat	Định dạng chuỗi hiển thị ngày hiện tại, mặc định là MMMM d YYYY

❖ Phía Client

Thuộc tính

Tên	Mô tả
animated	Chuyển sang chế độ hoạt hình
button	Button được dùng để hiển thị calendar(tùy chọn)
clearTime	Xóa thời gian trong khung sửa đổi ngày giờ
cssClass	Lớp css được dùng để chỉnh sửa hiển thị calendar
dayCell	Lấy một ngày trong cột và hàng được chỉ ra
daysModeTitleFormat	Định dạng được sử dụng cho tiêu đề trong chế độ ngày
defaultView	Hiển thị mặc định của calendar khi nó hiện ra

enabled	Bật chức năng calendar cho phần tử hiện hành
firstDayOfWeek	Xác lập ngày đầu tiên của tuần
format	Định dạng được dùng cho giá trị ngày
isOpen	Cho phép calendar hiển thị
popupPosition	Vị trí của calendar khi nó hiển thị. Có thể là BottomLeft (mặc định), BottomRight, TopLeft, TopRight
selectedDate	Giá trị ngày hiển thị trong textbox
todayButton	Button được dùng để chọn ngày hiện tại
todaysDate	Ngày sử dụng cho “Today”
todaysDateFormat	Định dạng của todaysDate
visibleDate	Ngày hiện tại được nhìn thấy trong calendar

Phương thức

Tên	Mô tả
Blur	
Dispose	Vô hiệu hóa các hành vi
focus	
hide	Ẩn calendar
initialize	Khởi tạo thành phần và tham số cho hành vi
invalidate	Thực hiện bố trí của hành vi trừ phi bố trí bị treo
raiseDateSelectionChanged	Kích hoạt sự kiện DateSelectionChanged
raiseHidden	Kích hoạt sự kiện Hidden
raiseHiding	Kích hoạt sự kiện Hidding
raiseShowing	Kích hoạt sự kiện Showing
raiseShown	Kích hoạt sự kiện Shown
resumeLayout	Tiếp tục bố trí hành vi và thực hiện bất kỳ yêu cầu bố trí nào
show	Hiển thị calendar
suspendLayout	Dừng hiển thị

Sự kiện

Tên	Mô tả
dateSelectionChanged	Thêm điều khiển sự kiện cho sự kiện SelectionChanged
hidden	Thêm điều khiển sự kiện cho sự kiện hidden
hiding	Thêm điều khiển sự kiện cho sự kiện hidding
showing	Thêm điều khiển sự kiện cho sự kiện showing
shown	Thêm điều khiển sự kiện cho sự kiện shown

Thí dụ

```
<ajaxToolkit:Calendar runat="server">  
    TargetControlID="Date1"  
    CssClass="ClassName"  
    Format="MMMM d, yyyy"  
    PopupButtonID="Image1" />
```

11.6.7. CascadingDropDown Control.

❖ Phía Server

Định nghĩa lớp mở rộng CascadingDropdown.

Thuộc tính

Tên	Mô tả
Category	Đề mục của DropDownList (sử dụng khi liên hệ với một web service)
ContextKey	Trạng thái của người dùng/trang được chỉ ra cung cấp đến một phương thức được mô tả bởi ServiceMethod/ServicePath. Nếu context key được sử dụng, nó sẽ có cùng một kiểu với một tham số có tên là contextKey kiểu string
EmptyText	Chuỗi ký tự hiển thị khi danh sách rỗng
EmptyValue	Giá trị hiển thị khi danh sách rỗng
LoadingText	Chuỗi ký tự hiển thị khi danh sách load dữ liệu
ParentControlID	ID của phần chứa DropDownList (phần nội dung mà DropDownList sử dụng)

PromptText	Chuỗi được hiển thị khi người dùng không lựa chọn, nếu bỏ qua, mục đầu tiên sẽ được chọn
PromptValue	Giá trị của lựa chọn được hiển thị bởi DropDownList thay thế cho PromptText
SelectedValue	Giá trị được lựa chọn
ServiceMethod	Tên của phương thức web service
ServicePath	Đường dẫn đến web service
UseContextKey	Sử dụng hay không thuộc tính ContextKey, nó sẽ tự động sử dụng nếu thuộc tính ContextKey chưa được thiết lập (cả ở client và server), nếu ContextKey được sử dụng, nó sẽ có cùng giá trị với tham số contextKey với kiểu String

Phương thức

Tên	Mô tả
Constructor	Constructor
CascadingDropDown_ClientStateValuesLoaded(System.Object, System.EventArgs)	Hiển thị DropDownList với các giá trị của nó
ParseKnownCategoryValuesString(System.String)	Phương thức dùng để chuyển đổi định dạng riêng được dùng để liên hệ với các đề mục
QuerySimpleCascadingDropDownDocument(System.Xml.XmlDocument, System.String[], System.Collections.Specialized.StringDictionary, System.String)	Phương thức cung cấp một hiện thực đơn giản của phương thức dùng để truy vấn tập dữ liệu và trả về thành phần của dropdown
QuerySimpleCascadingDropDownDocument(System.Xml.XmlDocument, System.String[], System.Collections.Specialized.StringDictionary, System.String, System.Text.RegularExpressions.Regex)	Phương thức cung cấp một hiện thực đơn giản của phương thức dùng để truy vấn tập dữ liệu và trả về thành phần của dropdown
ShouldSerializeServicePath	Ngăn dịch vụ chạy khi nó rỗng

❖ Phía Client

Phương thức

Tên	Mô tả
Category	Đề mục của drop down
contextKey	Trạng thái của người dùng/trang được chỉ ra cung cấp đến một phương thức được mô tả bởi ServiceMethod/ServicePath. Nếu context key được sử dụng, nó sẽ có cùng một kiểu với một tham số có tên là contextKey kiểu string
EmptyText	Chuỗi ký tự hiển thị khi danh sách rỗng
EmptyValue	Giá trị hiển thị khi danh sách rỗng
LoadingText	Chuỗi được hiển thị khi drop down lấy giá trị từ web service
ParentControlID	ID của phần tử cha trong hệ thống các drop down
PromptText	Hiển thị như là mục đầu tiên trong drop down
PromptValue	Giá trị của lựa chọn được hiển thị bởi Dropdown thẻ hiện PromtText
SelectedValue	Giá trị được lựa chọn trong drop down
ServiceMethod	Tên của phương thức gọi web service
ServicePath	Đường dẫn đến web service
useContextKey	Sử dụng hay không thuộc tính ContextKey, nó sẽ tự động sử dụng nếu thuộc tính ContextKey chưa được thiết lập (cả ở client và server), nếu ContextKey được sử dụng, nó sẽ có cùng giá trị với tham số contextKey với kiểu String

Phương thức

Tên	Mô tả
dispose	Vô hiệu các hành vi
initialize	Khởi tạo các hành vi
raisePopulated	Kích hoạt sự kiện Populated
raisePopulating	Kích hoạt sự kiện Populating
raiseSelectionChanged	Kích hoạt sự kiện SelectionChanged

Sự kiện

Tên	Mô tả
populated	Thêm điều khiển sự kiện cho sự kiện populated
populating	Thêm điều khiển sự kiện cho sự kiện populating
selectionChanged	Thêm điều khiển sự kiện cho sự kiện selectionChanged

Thí dụ

```
<ajaxToolkit:CascadingDropDown ID="CDD1" runat="server"
    TargetControlID="DropDownList2"
    Category="Model"
    PromptText="Please select a model"
    LoadingText="[Loading models...]"
    ServicePath="CarsService.asmx"
    ServiceMethod="GetDropDownContents"
    ParentControlID="DropDownList1"
    SelectedValue="SomeValue" />
```

11.6.8. CollapsiblePanel Control

Một lớp mở rộng cho phép thêm hành vi mở rộng hay thu hẹp một điều khiển ASP.NET. Phần nội dung được gắn mở rộng sau đó có thể mở rộng ra hoặc thu hẹp lại bởi người dùng, một cách thủ công như hiển thị hoặc ẩn nội dung hay mở rộng toàn bộ khoảng trống hiện có.

❖ Phía Server

Phương thức

Tên	Mô tả
AutoCollapse	Nếu giá trị là true và trạng thái của panel là “expanded” thì khi di chuyển con trỏ chuột ra khỏi panel thì nó tự động collapse
AutoExpand	Ngược lại, nếu giá trị là true và trạng thái của panel là “collapse” thì khi di chuyển chuột vào nó sẽ tự động expanded
CollapseControlID	ID của panel được gắn với điều khiển. Panel sẽ thu gọn khi điều khiển kích hoạt sự kiện “onclick” phía client. Nếu nó cùng giá trị với ExpandControlID, CollapsiblePanel sẽ tự động chuyển chế độ khi click vào điều khiển này

Collapsed	Tín hiệu khởi tạo trạng thái thu gọn của điều khiển. Lưu ý là nó không làm cho điều khiển đã mở rộng thu hẹp lại khi khởi tạo, mà nó thông báo với extender trạng thái của panel
CollapsedImage	Hình ảnh được hiển thị khi panel được thu hẹp và ImageControlID được thiết đặt
CollapsedSize	Kích thước của panel khi nó ở trạng thái thu hẹp. Để tránh bị co giãn trang khi panel được khởi tạo, thiết lập chiều cao (hoặc chiều rộng) của panel với một giá trị nào đó và thiết lập thuộc tính Collapse là true. Giá trị mặc định là -1, chỉ ra rằng Panel sẽ khởi tạo giá trị CollapsedSize dựa trên kích thước của đối tượng
CollapsedText	Đoạn text để hiển thị trong trạng thái đóng. Khi panel ở trạng thái đóng, nội dung bên trong điều khiển tham chiếu tới thuộc tính TextLabelID sẽ được thay thế bởi đoạn text này. Thuộc tính này cũng được dùng như một đoạn text thay thế một hình ảnh nếu ImageControlID được thiết lập
ExpandControlID	ID của điều khiển phía server. Panel sẽ được mở nếu điều khiển này kích hoạt sự kiện onclick, nếu nó có cùng giá trị với CollapsedControlID, panel sẽ tự động chuyển chế độ khi click vào điều khiển này
ExpandDirection	Kích thước dùng để mở rộng hoặc thu hẹp theo chiều ngang hoặc chiều dọc
ExpandedImage	Hình ảnh được hiển thị khi panel được mở rộng và thuộc tính ImageControlID được thiết lập
ExpandedSize	Kích thước của panel khi nó ở trạng thái mở, để tránh trang bị co giãn khi khởi tạo, thiết lập chiều rộng của panel với một giá trị nào đó, và thuộc tính Collapse là false, giá trị mặc định là -1, điều đó có nghĩa là panel sẽ được khởi tạo ExpandedSize trên cơ sở kích thước của đối tượng
ExpandedText	Văn bản được hiển thị trong trạng thái mở rộng, khi panel được mở, nội dung bên trong của điều khiển tham chiếu bởi thuộc tính TextLabelID sẽ được thay thế bởi đoạn text này, phần text mở rộng cũng được sử dụng như là phần thay thế cho hình ảnh nếu ImageControlID được thiết lập

ImageControlID	ID của hình ảnh thể hiện trạng thái hiện tại của panel, khi trạng thái thu gọn của panel thay đổi, hình ảnh nguồn sẽ thay đổi từ ExpandedImage sang CollapsedImage. Chúng ta cũng sử dụng ExpandedText và CollapsedText như là đoạn text thay thế hình ảnh nếu chúng được yêu cầu
ScrollContents	Xác định phần nội dung của panel sẽ được cuộn hoặc che đi nếu chúng không nằm gọn bên trong kích thước đã mở rộng
SuppressPostBack	Xác định CollapsiblePanelBehavior sẽ ngăn chặn xử lý sự kiện click của điều khiển tham chiếu đến CollapseControlID hoặc ExpandControlID. Mặc định giá trị này là false, ngoại lệ đối với tag “a”
TextLabelID	ID của một label thể hiện trạng thái hiện tại của Panel. Khi trạng thái thu gọn của panel bị thay đổi, phần nội dung HTML sẽ được thay thế bởi đoạn text mô tả trạng thái của panel

Phương thức

Tên	Mô tả
Constructor	Constructor mặc định

❖ Phía Client

CollapsiblePanelBehavior cho phép bạn thêm một phần có thể thu gọn vào trang web của bạn.

Thuộc tính

Tên	Mô tả
AutoCollapse	Cho phép panel tự động thu hẹp nếu con trỏ chuột di chuyển ra ngoài panel
AutoExpand	Cho phép panel tự động mở rộng nếu trỏ chuột di chuyển vào bên trong panel
CollapseControlID	ID của điều khiển được dùng để thu gọn phần nội dung khi click
Collapsed	Thiết lập panel có thu gọn hay không
CollapsedImage	Đường dẫn đến hình ảnh sẽ được thể hiện ở phần tử được chỉ ra ở ImageControlID khi panel ở trạng thái thu gọn

CollapsedSize	Kích thước của panel tính bằng pixel khi nó ở trạng thái thu gọn
CollapsedText	Phần text được thể hiện khi panel ở trạng thái thu gọn
ExpandControlID	ID của điều khiển dùng để mở rộng phần nội dung khi click
ExpandDirection	Chỉ định hướng mở rộng hoặc thu gọn của panel (hoặc theo chiều ngang “Horizontal” hoặc theo chiều dọc “Vertical”)
ExpandedImage	Đường dẫn đến hình ảnh sẽ hiển thị ở phần tử được chỉ ra bởi ImageControlID khi panel ở trạng thái mở rộng
ExpandedSize	Kích thước của panel tính bằng pixel khi nó ở trạng thái mở rộng
ExpandedText	Phần text hiển thị khi panel ở trạng thái mở rộng
ImageControlID	
ScrollContents	Cho phép thêm hay không thanh cuộn khi phần nội dung hiển thị quá giới hạn khung panel (phần nội dung sẽ bị mất nếu thiết lập là false)
SuppressPostBack	Cho phép hay không tạo postbacks khi CollapseControlID và ExpandControlID được click
TargetHeight	Ràng buộc chiều cao của panel
TargetWidth	Ràng buộc chiều rộng của panel
TextLabelID	ID của phần tử mà trạng thái của panel thể hiện

Phương thức

Tên	Mô tả
collapsePanel	Thu gọn panel. Một hàm public cho phép người dùng gọi khi họ muốn thu gọn panel bằng lệnh
Dispose	Vô hiệu hóa các hành vi
expandPanel	Mở rộng panel
Initialize	Khởi tạo các hành vi
raiseCollapseComplete	Kích hoạt sự kiện CollapseComplete
raiseCollapsed	Kích hoạt sự kiện Collapsed
raiseCollapsing	Kích hoạt sự kiện Collapsing
raiseExpandComplete	Kích hoạt sự kiện ExpandCompete

raiseExpanded	Kích hoạt sự kiện Expanded
raiseExpanding	Kích hoạt sự kiện Expanding
togglePanel	Quản lý sự kiện mở rộng hay thu gọn panel (trên trạng thái hiện tại của nó). Đây là một hàm public sẽ được gọi thay cho _toggle nếu bạn muốn mở hay đóng panel bằng lệnh

Sự kiện

Tên	Mô tả
collapseComplete	Thêm điều khiển sự kiện cho sự kiện collapseComplete
collapsed	Thêm điều khiển sự kiện cho sự kiện collapsed
collapsing	Thêm điều khiển sự kiện cho sự kiện collapsing
expandComplete	Thêm điều khiển sự kiện cho sự kiện expandComplete
expanded	Thêm điều khiển sự kiện cho sự kiện expanded
expanding	Thêm điều khiển sự kiện cho sự kiện expanding

Thí dụ

```
<ajaxToolkit:CollapsiblePanelExtender ID="cpe" runat="Server"
    TargetControlID="Panel1"
    CollapsedSize="0"
    ExpandedSize="300"
    Collapsed="True"
    ExpandControlID="LinkButton1"
    CollapseControlID="LinkButton1"
    AutoCollapse="False"
    AutoExpand="False"
    ScrollContents="True"
    TextLabelID="Label1"
    CollapsedText="Show Details..."
    ExpandedText="Hide Details"
    ImageControlID="Image1"
    ExpandedImage("~/images/collapse.jpg")
    CollapsedImage "~/images/expand.jpg"
    ExpandDirection="Vertical" />
```

11.6.9. ColorPicker Control

Phần mở rộng ColorPicker cho phép bạn hiển thị một bảng pop-up màu khi con trỏ di chuyển đến phần tử input. Bạn có thể gắn ColorPicker vào bất cứ textbox nào của ASP.NET. Nó cung cấp một hàm cho phép lựa chọn màu ở phía client với giao diện người dùng. Ngoài ra, bạn có thể chỉ ra một button để hiển thị một popup lựa chọn màu và một điều khiển cho phép xem trước màu từ bảng màu. Bạn có thể cung cấp một textbox để khi mà người dùng nhập vào một giá trị màu thì ColorPicker có thể hiển thị màu tương ứng nếu màu đó không có trong bảng màu mặc định.

❖ Phía Server

Thuộc tính

Tên	Mô tả
TargetControlID	ID của textbox
PopupButtonID	ID của điều khiển được dùng để hiển thị bảng popup chọn màu, nếu giá trị không được thiết lập, bảng màu sẽ tự động hiển thị khi textbox chỉ định ở TargetControlID được chọn
SampleControlID	ID của điều khiển dùng để hiển thị màu được chọn. Nếu giá trị này được thiết lập và trang chọn màu đang mở, màu nền của điều khiển được chỉ ra sẽ hiện màu mà trỏ chuột đang trỏ tới, nếu giá trị không được thiết lập, màu được chọn sẽ không hiển thị
PopupPosition	Xác định vị trí mà bảng lựa chọn màu sẽ hiển thị so với vị trí của textbox mà extender này gắn vào, có thể là BottomLeft, BottomRight, TopLeft, TopRight
SelectedColor	Giá trị màu mà bảng màu ColorPicker tạo ra lúc khởi tạo

Phương thức

Tên	Mô tả
OnClientColorSelectionChanged	Hàm javascript sẽ được gọi khi sự kiện colorSelectionChanged xảy ra

❖ Phía Client

Thuộc tính

Tên	Mô tả
button	Thiết lập một đối tượng kiểu Sys.UI.DomElement thể hiện một button dùng để hiện bảng màu, thuộc tính này là tùy chọn

sample	Thiết lập một đối tượng kiểu Sys.UI.DomElement thể hiện một phần tử dùng để xem trước màu vừa được chọn hay màu mà trỏ chuột đang trỏ vào, thuộc tính này là tùy chọn
selectedColor	Thiết lập một chuỗi chứa giá trị màu được thể hiện ở trong textbox
enabled	Thiết lập một giá trị kiểu bool chỉ ra rằng bảng màu có hiệu lực đối với phần tử hiện tại hay không
popupPosition	Thiết lập một đối tượng kiểu Sys.UI.DomElement thể hiện vị trí mà bảng màu sẽ xuất hiện tương ứng với textbox, các giá trị có thể là BottomLeft, BottomRight, TopLeft, TopRight

Phương thức

Tên	Mô tả
Initialize	Khởi tạo bảng màu
dipose	Vô hiệu hóa bảng màu
raiseColorSelectionChanged	Kích hoạt sự kiện ColorSelectionChanged
raiseShowing	Kích hoạt sự kiện Showing
raiseShown	Kích hoạt sự kiện Shown
raiseHiding	Kích hoạt sự kiện Hiding
raiseHidden	Kích hoạt sự kiện Hidden
show	Hiển thị bảng màu
hide	Ẩn bảng màu

Sự kiện

Tên	Mô tả
colorSelectionChanged	Thêm một điều khiển sự kiện cho sự kiện colorSelectionChanged
showing	Thêm một điều khiển sự kiện cho sự kiện Showing
shown	Thêm một điều khiển sự kiện cho sự kiện Shown
hiding	Thêm một điều khiển sự kiện cho sự kiện Hiding
hidden	Thêm một điều khiển sự kiện cho sự kiện Hidden

Thí dụ

```
<asp:TextBoxID="TextBox1"runat="server"></asp:TextBox>  
<asp:PanelID="Panel1"runat="server"style="width:18px;height:18px;border:1px solid  
#000;margin:0 3px;float:left">  
</asp:Panel>  
<asp:ColorPickerExtender  
ID="ColorPickerExtender1"runat="server"  
TargetControlID="TextBox1"PopupPosition="TopLeft"SampleControlID="Panel1">  
</asp:ColorPickerExtender>
```

11.6.10. ComboBox Control

ComboBox là một điều khiển của ASP.NET AJAX, nó giống như AutoCompleteExtender, kết hợp linh hoạt với một Textbox với một danh sách tùy chọn cho phép người dùng có thể lựa chọn. Nó có các thuộc tính, hành vi hay quy ước đặt tên tương tự như trên Form combobox, và có cùng lớp cơ sở như là ListBox, BulletedList và DropDownList của web control. Thực vậy, ComboBox được xem như là DropDownList nhưng lại có thể gõ trực tiếp vào như là textbox.

Làm việc với ComboBox cũng giống như làm việc với DropDownList. Nó có cùng tất cả các thuộc tính và sự kiện của DropDownList, với một vài thuộc tính và sự kiện khác nữa. Đầu tiên, nó có thể được cấu hình để ngăn chặn hoặc cho phép người dùng nhập chuỗi mà không trùng khớp với các mục trong danh sách. Khi người dùng gõ một chuỗi trùng khớp với một mục trong danh sách, ComboBox cũng có thể được cấu hình để tự động hoàn tất chuỗi đó dựa trên cơ sở những ký tự đã được gõ, để hiển thị danh sách và đánh dấu phần tử đầu tiên trùng khớp, hoặc làm cả hai đồng thời. Khi người dùng gõ một chuỗi không trùng khớp với một mục nào trong danh sách, ComboBox kích hoạt sự kiện ItemInserting và ItemInserted mà có thể điều khiển được trong quá trình postback. Ngoài các hành vi đặc biệt đó, ComboBox hoạt động như một DropDownList.

ComboBox giống như là một phần bổ sung chứ không phải là thay thế cho AutoCompleteExtender. Mặc dù nó cũng có thể đáp ứng cùng một yêu cầu về giao diện người dùng.

❖ Thuộc tính

Tên	Mô tả
DropDownStyle	Xác định xem người dùng có được phép gõ một chuỗi mà không trùng khớp với một mục nào trong danh sách hay không, và cho phép danh sách luôn luôn được hiển thị. Nếu giá trị là DropDownList, người dùng không được phép gõ chuỗi mà không trùng khớp với một mục trong danh sách. Nếu là DropDown (giá trị mặc định) thì bất kỳ chuỗi nào cũng được chấp nhận. Nếu là Simple thì bất kỳ chuỗi nào cũng được chấp nhận và danh sách luôn luôn được hiển thị bất kể giá trị nào của AutoCompleteMode
AutoCompleteMode	Xác định bằng cách nào ComboBox tự động hoàn tất chuỗi vừa nhập vào, nếu giá trị là Suggest, ComboBox sẽ hiển thị danh sách, tô đậm mục trùng khớp đầu tiên tìm thấy, và nếu cần thiết, mục được đánh dấu sẽ được cuộn tới, nếu giá trị là Append, ComboBox sẽ nối các phần còn lại của mục đầu tiên trùng khớp với phần mà người dùng gõ vào và tô đậm phần được nối, nếu giá trị là SuggestAppend, cả hai hành vi trên được áp dụng, nếu là None (giá trị mặc định), tính năng tự động hoàn tất của ComboBox sẽ bị vô hiệu
CaseSensitive	Có phân biệt ký tự hoa – thường hay không, mặc định là không (false)
RenderMode	Chỉ ra là ComboBox là một phần tử HTML ở mức độ Inline hay Block, mặc định là Inline
ItemInsertLocation	Xác định một item mới được thêm vào là Append hay là Prepend khi chúng được thêm vào trong danh sách, hoặc thêm chúng chèn vào theo thứ tự Alphabet trên cơ sở Text hay Value, mặc định là Append
ListItemHoverCssClass	Khi chúng được thiết lập, thay thế style mặc định được áp dụng để tô đậm mục trong danh sách với một lớp css

ListItem	Một hoặc nhiều điều khiển con được dùng để khai báo các mục sẽ được thêm vào danh sách ComboBox, khi gắn nguồn dữ liệu, tất cả ListItems sẽ bị xóa bỏ cho đến khi thuộc tính AppendDataBoundItem được đặt là true
-----------------	---

❖ Lớp CSS

Tên	Mô tả
.ajax__combobox_inputcontainer	Vị trí của button và textbox để nhập
.ajax__combobox_textboxcontainer	Css cho phần textbox
.ajax__combobox_buttoncontainer	Css cho phần button
.ajax__combobox_itemlist	Css cho danh sách mục

❖ Thí dụ

```
<ajaxToolkit:ComboBox ID="ComboBox1" runat="server"
    DropDownStyle="DropDown"
    AutoCompleteMode="None"
    CaseSensitive="false"
    RenderMode="Inline"
    ItemInsertLocation="Append"
    ListItemHoverCssClass="ComboBoxListItemHover">
<asp:ListItem>...</asp:ListItem>
    ...
</ajaxToolkit:ComboBox>
.CustomComboBoxStyle.ajax__combobox_textboxcontainer input {
    background-color: #ADD8E6;
    border: solid 1px Blue;
    border-right: 0px none;
}
.CustomComboBoxStyle.ajax__combobox_buttoncontainer button {
    background-color: #ADD8E6;
    border: solid 1px Blue;
}
```

11.6.11. ConfirmButton Control

ConfirmButton là một extender đơn giản bắt sự kiện click một button và hiển thị một thông báo đến người dùng. Nếu button OK được click, hàm sẽ được xử lý bình thường, nếu không, sự kiện click sẽ bị bỏ qua và button sẽ không thực hiện hành vi mặc định của nó, thay vào đó, một đoạn mã sẽ được thực thi nếu thuộc tính OnClientCancel được thiết lập. Nó thật sự hữu ích để xóa liên kết hoặc bắt cứ thứ gì khác yêu cầu xác thực từ người dùng.

❖ Phía Server

Thuộc tính

Tên	Mô tả
ConfirmOnFormSubmit	True nếu hộp thoại xác thực sẽ được chạy trên form
ConfirmText	Chuỗi sẽ được hiển thị bên trong hộp thoại
DisplayModalPopupID	Chỉ ra ID của ModalPopupExtender được dùng trong window.confirm
OnClientCancel	Thiết lập mã lệnh phía client sẽ được thực thi khi button cancel được click trên hộp thoại
PostBackScript	Chỉ ra đoạn mã được chạy để khởi tạo postback

Phương thức

Tên	Mô tả
OnLoad(System.EventArgs)	Onload override
RegisterDisplayModalPopup	Đăng ký phần của DisplayModalPopupID được dùng với ConfirmButton

❖ Phía Client

Thuộc tính

Tên	Mô tả
ConfirmOnFormSubmit	True nếu hộp thoại xác thực sẽ được chạy trên form
ConfirmText	Chuỗi sẽ được hiển thị bên trong hộp thoại
DisplayModalPopupID	Chỉ ra ID của ModalPopupExtender được dùng trong window.confirm
OnClientCancel	Thiết lập mã lệnh phía client sẽ được thực thi khi button cancel được click trên hộp thoại
PostBackScript	Chỉ ra đoạn mã được chạy để khởi tạo postback

Phương thức

Tên	Mô tả
dispose	Vô hiệu các hành vi
initialize	Khởi tạo các hành vi
raiseHidden	Kích hoạt sự kiện Hidden
raiseShowing	Kích hoạt sự kiện Showing
WebForm_OnSubmit	

Sự kiện

Tên	Mô tả
hidden	Thêm điều khiển sự kiện cho sự kiện hidden
showing	Thêm điều khiển sự kiện cho sự kiện showing

Thí dụ

```
<ajaxToolkit:ConfirmButtonExtender ID="cbe" runat="server">
    TargetControlID="LinkButton1"
    ConfirmText="Are you sure you want to click this?"
    OnClientCancel="CancelClick" />
```

11.6.12. DragPanel Control

DragPanel extender cho phép người dùng dễ dàng thêm một phần có thể kéo đi được trong điều khiển của họ. DragPanel điều khiển bất kỳ panel nào của ASP.NET và thêm một tham số nói lên điều khiển được dùng như là có thể kéo đi được. Khi được khởi tạo, người dùng có thể thoải mái kéo panel quanh trang web sử dụng extender này.

Thuộc tính

Tên	Mô tả
TargetControlID	ID của panel muốn kéo được
DragHandleID	ID của một điều khiển được xem như là điều khiển kéo thả của panel, khi người dùng click và kéo điều khiển này, panel sẽ di chuyển theo

11.6.13. DynamicPopulate Control.

DynamicPopulate là một extender đơn giản thay thế nội dung của một control với một kết quả của web service hay phương thức được gọi. Phương thức được gọi trả về một chuỗi HTML để đặt vào phần tử đích.

❖ Phía Server

Thuộc tính

Tên	Mô tả
CacheDynamicResults	Dùng để lưu kết quả của lần truy vấn trước đó và không lấy thêm lần nữa sau lần nạp đầu tiên
ClearContentsDuringUpdate	Cho phép chúng ta xóa nội dung của phần tử đích hay không khi việc cập nhật bắt đầu
ContextKey	Key giúp bạn có thể gọi được web service, thuộc tính này là tùy chọn
CustomScript	Customscript có thể được sử dụng để eval hàm JavaScript sẽ gửi trả chuỗi để thay đổi nội dung điều khiển. Phương pháp này phải gửi trả chuỗi và sẽ được gọi là thay vì phương thức Service hay Page
PopulateTriggerControlID	ID của điều khiển gọi thủ tục thay đổi nội dung của phần tử đích, sự thay đổi sẽ xảy ra bởi sự kiện click
ServiceMethod	Phương thức của web service để gọi
ServicePath	Đường dẫn đến web service được gọi, hoặc nếu thuộc tính này để trống, phương thức của trang sẽ được gọi
UpdatingCssClass	Lớp css được dùng trong quá trình update

Phương thức

Tên	Mô tả
CheckIfValid(System.Boolean)	Kiểm tra nếu các thuộc tính được thiết lập không đúng
ShouldSerializeServicePath	Ngăn khởi tạo đường dẫn đến dịch vụ nếu không có ServiceMethod nào được cung cấp

❖ Phía Client

Thuộc tính

Tên	Mô tả
CacheDynamicResults	Dùng để lưu kết quả của lần truy vấn trước đó và không lấy thêm lần nữa sau lần nạp đầu tiên
ClearContentsDuringUpdate	Cho phép chúng ta xóa nội dung của phần tử đích hay không khi việc cập nhật bắt đầu
ContextKey	Key giúp bạn có thể gọi được web service, thuộc tính này là tùy chọn
CustomScript	Customscript có thể được sử dụng để eval hàm JavaScript sẽ gửi trả chuỗi để thay đổi nội dung điều khiển. Phương pháp này phải gửi trả chuỗi và sẽ được gọi là thay vì phương thức Service hay Page
PopulateTriggerControlID	ID của điều khiển gọi thủ tục thay đổi nội dung của phần tử đích, sự thay đổi sẽ xảy ra bởi sự kiện click
ServiceMethod	Phương thức của web service để gọi
ServicePath	Đường dẫn đến web service được gọi, hoặc nếu thuộc tính này để trống, phương thức của trang sẽ được gọi
UpdatingCssClass	Lớp css được dùng trong quá trình update

Phương thức

Tên	Mô tả
dispose	Vô hiệu các hành vi
initialize	Khởi tạo các hành vi
populate	Lấy nội dung và gắn chúng vào phần tử đích
raisePopulated	Kích hoạt sự kiện Populated
raisePopulating	Kích hoạt sự kiện Populating
setStyle	Thiết lập kiểu trình diễn

Sự kiện

Tên	Mô tả
populated	Thêm điều khiển sự kiện cho sự kiện populated
populating	Thêm điều khiển sự kiện cho sự kiện populating

Thí dụ

```
<ajaxToolkit:DynamicPopulateExtender ID="dp" runat="server">  
    TargetControlID="Panel1"  
    ClearContentsDuringUpdate="true"  
    PopulateTriggerControlID="Label1"  
    ServiceMethod="GetHtml"  
    UpdatingCssClass="dynamicPopulate_Updating" />
```

11.6.14. HTMLEditor Control

HTMLEditor là một điều khiển ASP.NET AJAX cho phép bạn dễ dàng tạo hoặc sửa đổi nội dung HTML. Các button khác trong thanh công cụ được dùng để biên tập nội dung. Bạn có thể thấy các thẻ HTML hoặc tài liệu xem trước.

Thuộc tính

Tên	Mô tả
ActiveMode	Kích hoạt panel dùng để biên tập khi điều khiển được nạp
AutoFocus	Nếu giá trị là true, khung biên tập sẽ được chọn và con trỏ sẽ được đặt vào trong nó (Design hoặc HTML text) khi trang được khởi tạo hoặc khung soạn thảo thay đổi
Content	Thiết lập nội dung của HTML Editor
CssClass	Lớp css dùng để định nghĩa giao diện cho HTMLEditor
DesignPanelCssPath	Thiết lập đường dẫn của file css được dùng để nội dung trong HTMLEditor hiển thị ở chế độ Design. Nếu không thiết lập, file css mặc định sẽ được dùng
DocumentCssPath	Thiết lập đường dẫn của file css được dùng để nội dung trong HTMLEditor hiển thị ở chế độ Design hoặc Preview. Nếu không thiết lập, file css mặc định sẽ được dùng

Height	Thiết lập chiều cao của khung soạn thảo
HtmlPanelCssClass	Lớp css định nghĩa giao diện cho chế độ HTML text
IgnoreTab	Nếu giá trị là true, nút tab sẽ không có hiệu lực
InitialCleanUp	Nếu giá trị là true, nội dung của HTMLEditor sẽ bị xóa khi được nạp lúc khởi tạo
NoScript	Nếu giá trị là true, javascript sẽ bị vô hiệu hóa
NoUnicode	Nếu giá trị là true, tất cả các ký tự Unicode sẽ được thay thế bằng &#code
SuppressTabInDesignMode	Nếu giá trị là true, sẽ không có khoảng trắng nào được đặt khi nhấn tab trong chế độ Design. Nút tab mặc định sẽ được xử lý trong trường hợp này
Width	Thiết lập chiều rộng của khung biên tập

Phương thức

Tên	Mô tả
OnClientActiveModeChanged	Phần mã sẽ được thực hiện sau khi chế độ hoạt động thay đổi
OnClientBeforeActiveModeChanged	Phần mã sẽ được thực thi trước khi chế độ hoạt động thay đổi

Các lớp css của HTMLEditor

Tên	Mô tả
ajax__htmleditor_editor_container	Phần tử bao gồm mọi phần tử khác trong HTMLEditor
ajax__htmleditor_editor_toptoolbar	Thiết lập cho button
ajax__htmleditor_editor_editpanel	Thiết lập cho khung soạn thảo
ajax__htmleditor_editor_bottomtoolbar	Thiết lập cho button bên dưới(các chế độ)
ajax__htmleditor_toolbar_button	Thiết lập cho một nút của toolbar
ajax__htmleditor_toolbar_button_hover	Thiết lập cho một button của toolbar khi đưa chuột vào nó

div.ajax__htmleditor_toolbar_button_label	Thiết lập cho <label> của toolbar
div.ajax__htmleditor_toolbar_button_select	Thiết lập cho <select> của toolbar
div.ajax__htmleditor_toolbar_button_select option	

11.6.15. ListSearch Control

ListSearch extender cho phép bạn tìm kiếm một mục trong ListBox hoặc DropDownList trên cơ sở các ký tự mà bạn đã đánh. Một thông điệp sẽ được hiển thị khi bạn click vào danh sách có thể được tùy chỉnh phụ thuộc vào lớp css và vị trí của nó.

Thuộc tính:

Tên	Mô tả
PromptText	Lời nhắn hiển thị khi ListBox hay DropDownList được trỏ vào. Mặc định là “Type to search”. Dòng text này sẽ được thay thế bởi các ký tự khi người dùng nhập vào
PromptCssClass	Tên của lớp Css được áp dụng vào PromtText
PromptPosition	Xác định nơi mà tin nhắn sẽ được hiển thị ở phía trên hay dưới của ListBox, mặc định là phía trên
QueryPattern	Xác định bằng cách nào các ký tự được gõ vào sẽ được sử dụng trong các truy vấn tìm kiếm, mặc định là kết quả được truy vấn từ những ký tự được gõ vào đầu tiên
IsSorted	Xác định nếu một mục được thêm vào trong danh sách sẽ được sắp xếp, mặc định là không sắp xếp, nếu được thiết lập là true, nó cho phép mã lệnh sẽ tìm kiếm nhanh hơn thay vì xác định sự trùng khớp trước khi tìm kiếm
QueryTimeout	Chỉ ra rằng các truy vấn tìm kiếm sẽ bị reset sau khoảng thời gian định mức cho phép mà vẫn không có kết quả nào được tìm thấy, mặc định là 0, nghĩa là không tự động reset
Animations	Tạo hiệu ứng cho ListSearch extender

Phương thức

Tên	Mô tả
OnShow	Hiệu ứng sẽ được chạy mỗi khi thông báo được hiện ra
OnHide	Hiệu ứng được chạy khi thông báo ẩn đi

Thí dụ

```
<ajaxToolkit:ListSearchExtender id="LSE" runat="server">  
    TargetControlID="ListBox1"  
    PromptText="Type to search"  
    PromptCssClass="ListSearchExtenderPrompt"  
    PromptPosition="Top"  
    AutoResetTimeout="0"  
    IsSorted="true"/>
```

11.6.16. MaskedEdit Control

MaskedEdit gắn vào một textbox để hạn chế các loại văn bản được nhập vào. MaskedEdit áp một mặt nạ vào khung nhập mà chỉ cho phép một số loại ký tự / văn bản được nhập vào. Các dữ liệu được hỗ trợ định dạng là: Số, Ngày, Thời gian, và datetime. MaskedEdit sử dụng các thiết lập culture quy định tại các thuộc tính CultureName. Nếu không có quy định các thiết lập culture sẽ được giống như trang: Tiếng Việt (Việt Nam).

Thuộc tính

Tên	Mô tả
MaskType	Loại hợp lệ để thực hiện: <ul style="list-style-type: none">• None – không cần hợp lệ• Number – chấp nhận kiểu số• Date – chấp nhận kiểu ngày• Time – chấp nhận kiểu thời gian• DateTime – chấp nhận cả ngày và thời gian
Mask Characters and Delimiters	9 – chỉ có các ký tự số L – chỉ có các chữ cái \$ - chỉ có các chữ cái và khoảng trắng C – chỉ có các ký tự và phân biệt hoa

	<p>thường</p> <p>A – chỉ có từ hoặc ký tự</p> <p>N – chỉ có số hoặc ký tự</p> <p>? – bất kỳ ký tự nào</p> <p>/ - dấu phân cách ngày</p> <p>: - dấu phân cách thời gian</p> <p>. – dấu phân cách thập phân</p> <p>, - dấu phân cách hàng ngàn</p> <p>\ - ký tự ESC</p> <p>{ - bắt đầu ký tự xác định sự lặp lại của mặt nạ</p> <p>} – ký tự kết thúc sự lặp lại của mặt nạ</p>
AcceptAMPM	Nếu giá trị là true sẽ hiển thị ký hiệu AM/PM
AcceptNegative	<p>True nếu chấp nhận ký tự ‘-’</p> <ul style="list-style-type: none"> • None: không hiển thị ký tự “-” • Left: hiển thị ký tự - phía bên trái của mặt nạ • Right: hiển thị ký tự - bên phải mặt nạ
AutoComplete	<p>Tự động điền các ký tự trống nếu người dùng không gõ vào</p> <ul style="list-style-type: none"> • MaskType=Number – điền vào các ký tự 0 • MaskType=Time – điền vào giờ hiện tại • MaskType=Date – điền vào ngày hiện tại • MaskType=DateTime – điền vào ngày giờ hiện tại
AutoCompleteValue	Ký tự mặc định được dùng khi AutoComplete có hiệu lực
Century	Thế kỷ mặc định được dùng khi mặt nạ ngày chỉ có hai ký tự thể hiện năm
ClearMaskOnLostFocus	Bỏ mặt nạ khi textbox không được trỏ vào

ClearTextOnInvalid	Xóa textbox khi nhập vào chuỗi không hợp lệ
ClipboardEnabled	Cho phép copy/paste với clipboard
ClipboardText	Chuỗi hiển thị được dùng khi thực hiện dán từ clipboard
DisplayMoney	Chỉ ra bằng cách nào ký tự tiền tệ được thể hiện <ul style="list-style-type: none"> • None: không hiển thị ký tự \$ • Left: hiển thị ký tự \$ phía bên trái của mặt nạ • Right: hiển thị ký tự \$ bên phải mặt nạ
ErrorTooltipCssClass	Lớp css cho phần tooltip
ErrorTooltipEnabled	Hiển thị tin nhắn kiểu tooltip khi rê chuột vào textbox
Filtered	Ký tự hợp lệ cho mặt nạ kiểu C (phân biệt hoa thường)
InputDirection	Kiểu nhập <ul style="list-style-type: none"> • LeftToRight – trái qua phải • RightToLeft – phải qua trái
MessageValidatorTip	Tin nhắn hiển thị khi sửa đổi textbox
PromptChararacter	Dấu nhắc cho ký tự không chỉ định
UserDateFormat	Định dạng ngày tùy chỉnh
UserTimeFormat	Định dạng thời gian tùy chỉnh
OnFocusCssClass	Lớp css được dùng khi textbox được trỏ vào
OnFocusCssNegative	Lớp css được dùng khi textbox được trỏ vào với giá trị âm
OnBlurCssNegative	Lớp css được dùng khi textbox không được trỏ vào với giá trị âm
OnInvalidCssClass	Lớp css được dùng khi chuỗi không hợp lệ
CultureName	Tên của culture được sử dụng

Thí dụ

```
<ajaxToolkit:MaskedEditExtender  
    TargetControlID="TextBox2"  
    Mask="9,999,999.99"  
    MessageValidatorTip="true"  
    OnFocusCssClass="MaskedEditFocus"  
    OnInvalidCssClass="MaskedEditError"  
    MaskType="Number"  
    InputDirection="RightToLeft"  
    AcceptNegative="Left"  
    DisplayMoney="Left"  
    ErrorTooltipEnabled="True"/>
```

11.6.17. ModalPopup Control

Modalpopup extender cho phép một trang có thể hiển thị nội dung cho người dùng theo kiểu modal, tức là ngăn cản người dùng tác động vào phần còn lại của trang. Nội dung modal có thể được hiển thị trên nền của trang và với kiểu tùy chọn áp dụng vào nó. Khi hiển thị, chỉ có nội dung modal có thể tương tác được, click vào phần còn lại của trang không có hiệu tượng gì. Khi người dùng tương tác với nội dung kiểu modal, click vào nút Ok hoặc Cancel sẽ tắt nội dung modal và chạy đoạn mã mà người dùng thiết lập. Đoạn mã được áp dụng với bất kỳ sự thay đổi nào trong khi chế độ modal đang hoạt động. Nếu postback được yêu cầu, đơn giản cho phép điều khiển OK/Cancel và trang sẽ được nạp lại. Bạn cũng có thể tùy chỉnh vị trí hiển thị của popup với thuộc tính X và Y, mặc định là chính giữa màn hình, tuy nhiên nếu chỉ cần X hoặc Y được xác định sau đó nó là trung tâm theo chiều dọc hoặc chiều ngang.

Thuộc tính:

Tên	Mô tả
TargetControlID	ID của phần tử kích hoạt popup modal
PopupControlID	ID của phần tử hiển thị như là popup modal
BackgroundCssClass	Lớp css áp dụng cho nền khi popup modal hiển thị
DropShadow	Tự động thêm rỗng cho popup modal
OkControlID	ID của phần tử hủy bỏ modal popup

OnOkScript	Mã lệnh được chạy khi modal popup bị hủy bỏ bởi OkControlID
CancelControlID	ID của phần tử bỏ qua modal popup
OnCancelScript	Mã lệnh được chạy khi hủy bỏ modal popup bởi CancelControlID
PopupDragHandleControlID	ID của phần tử chứa tiêu đề được dùng như là điều khiển popup
X	Tọa độ theo chiều ngang
Y	Tọa độ theo chiều dọc
RepositionMode	Tùy chọn xác định nếu popup cần phải thay đổi vị trí khi cửa sổ thay đổi kích thước hoặc được cuộn

Thí dụ

```
<ajaxToolkit:ModalPopupExtender ID="MPE" runat="server"
    TargetControlID="LinkButton1"
    PopupControlID="Panel1"
    BackgroundCssClass="modalBackground"
    DropShadow="true"
    OkControlID="OkButton"
    OnOkScript="onOk()"
    CancelControlID="CancelButton"
    PopupDragHandleControlID="Panel3" />
```

11.6.18. MultiHandleSlider Control

Mở rộng MultiHandleSlider cung cấp một tính năng mở rộng cho một asp phổ biến: textbox. Nó cho phép bạn chọn một giá trị duy nhất, hay nhiều giá trị trong một phạm vi nhất định, thông qua một thanh trượt. Nó hỗ trợ một xử lý, xử lý kép, hoặc số bất kỳ xử lý ràng buộc với các giá trị của asp: TextBox hay asp:Label. Nó cũng cung cấp tùy chọn cho truy cập chỉ đọc, tùy biến giao diện, di chuột và kéo để xử lý, cũng như hỗ trợ khả năng truy cập của chuột và bàn phím.

Nhiều MultiHandleSlider thiết kế là dựa trên bản gốc Slider, do đó, bạn có thể xem lại những lời khuyên và gợi ý thiết kế được cung cấp bởi các trang trình diễn của extender đó. Thí dụ, MultiHandleSlider giữ lại các chức năng tooltip giống như Slider gốc.

Bằng cách khai báo TextBox đã được mở rộng như là một phần của một UpdatePanel, các MultiHandleSlider có thể kích hoạt sự kiện cập nhật bất cứ khi nào tiến hành xử lý. Bằng cách thiết lập các thuộc tính RaiseChangeOnlyOnMouseUp là false, cập nhật trên được xảy ra ngay sau khi thay đổi giá trị của MultiHandleSlider này.

Thuộc tính

Tên	Mô tả
Minimum	Giá trị tối thiểu cho phép
Maximum	Giá trị tối đa cho phép
Length	Chiều dài của thanh trượt tính bằng pixel
Decimals	Số lượng số thập phân cho giá trị
Steps	Số lượng các giá trị rời rạc trong phạm vi thanh trượt
Orientation	xác định xem định hướng của thanh trượt là nằm ngang hoặc thẳng đứng
CssClass	Phong cách tổng thể để áp dụng cho thanh trượt
EnableHandleAnimation	Nếu giá trị là true, click để di chuyển thanh trượt sẽ cho phép tạo hiệu ứng khi di chuyển
EnableRailClick	Cho phép click vào thanh trượt sẽ di chuyển điều khiển gần nhất đến vị trí vừa click
EnableInnerRangeDrag	Cho thanh trượt có nhiều điều khiển, xác định việc click và kéo thanh trượt giữa hai điều khiển sẽ di chuyển cả hai cùng lúc
EnableKeyboard	Cho phép giá trị thanh trượt có thể được thay đổi bằng bàn phím
EnableMouseWheel	Cho phép giá trị thanh trượt có thể được thay đổi bằng cách lăn con lăn của chuột
ShowInnerRail	Với thanh trượt có nhiều điều khiển, xác định cách thể hiện thị phần giữa hai điều khiển
ShowHandleHoverStyle	Cho phép hiển thị một kiểu css khi người dùng di chuyển chuột lên trên điều khiển

ShowHandleDragStyle	Kiểu css khi người dùng kéo điều khiển
InnerRailStyle	Cho thanh trượt có nhiều điều khiển và phong cách tùy chỉnh, xác định bằng cách nào thể hiện InnerRailCss như nó đã được mô tả, hoặc sử dụng phương pháp ảnh trượt
ReadOnly	Cho phép người dùng có thể thay đổi giá trị thanh trượt
Increment	Cho thanh trượt sử dụng bàn phím hoặc chuột, xác định số lượng điểm để tăng hoặc giảm giá trị thanh trượt
HandleAnimationDuration	Quá trình của hiệu ứng xử lý, tính bằng giây
BoundControlID	Cho tính tương thích ngược, cho phép sử dụng Slider cho một điều khiển
HandleCssClass	Cho tính tương thích ngược, chỉ ra kiểu của xử lý đơn
RaiseChangeOnlyOnMouseUp	Nếu giá trị là true, kích hoạt sự kiện thay đổi textbox chỉ khi nhả phím trái chuột
TooltipText	Text hiển thị khi rê chuột đến điều khiển, giá trị {0} sẽ được thay thế cho giá trị hiện tại của thanh trượt
MultiHandleSliderTargets	<p>Thuộc tính bên trong mô tả mỗi xử lý trên thanh trượt:</p> <ul style="list-style-type: none"> • ControlID: Textbox hoặc Label gắn với điều khiển • HandleCssClass: lớp css được dùng khi người dùng cho điều khiển • HandleHoverCssClass: lớp css được dùng khi người dùng rê chuột vào điều khiển • HandleDragCssClass: lớp css được dùng khi người dùng kéo thanh trượt • DecimalPlaces: số chữ số cho định dạng của giá trị • Offset

Sự kiện

Tên	Mô tả
OnClientLoad	Sự kiện khi khởi tạo thanh trượt
OnClientDragStart	Sự kiện khi người dùng bắt đầu kéo thanh trượt
OnClientDrag	Sự kiện khi người dùng kéo thanh trượt
OnClientDragEnd	Sự kiện khi người dùng kết thúc kéo thanh trượt
OnClientValueChanged	Sự kiện khi giá trị thay đổi

Thí dụ

```
<asp:TextBox ID="sliderOne" runat="server" AutoPostBack="true" Text="0"/>
<ajaxToolkit:MultiHandleSliderExtender ID="multiHandleSliderExtenderOne"
runat="server"
    BehaviorID="multiHandleSliderExtenderOne"
    TargetControlID="sliderOne"
    Minimum="-100"
    Maximum="100"
    Steps="5"
    Length="140"
    BoundControlID="lblSliderOne"
    ToolTipText="{0}">
</ajaxToolkit:MultiHandleSliderExtender>
<asp:Label ID="lblSliderOne" runat="server" style="text-align:right" Text="0" />
```

11.6.19. MutuallyExclusiveCheckBox Control

MutuallyExclusiveCheckBox là một ASP.NET AJAX extender có thể được gắn vào bất kỳ CheckBox nào trong ASP.NET. Bằng cách thêm vào một số hộp kiểm tra có cùng “Key”, chỉ có một checkbox với key được chỉ ra mới có thể được chọn. Extender này rất hữu ích khi một số lựa chọn là có sẵn nhưng chỉ có một cái có thể được lựa chọn, tương tự như một nút radio. Việc sử dụng các checkbox tuy nhiên cho phép bạn chọn hoặc bỏ chọn một giá trị mà không thể làm được với các nút radio. Điều này cũng cung cấp một giao diện nhất quán và được mong đợi nhiều hơn sử dụng javascript để cho phép bỏ chọn một mục RadioButton.

Thuộc tính

Tên	Mô tả
TargetControlID	ID của checkbox để sửa đổi
Key	Khóa duy nhất được dùng, gắn với checkbox

Thí dụ

```
<ajaxToolkit:MutuallyExclusiveCheckboxExtender runat="server"
    ID="MustHaveGuestBedroomCheckBoxEx"
    TargetControlID="MustHaveGuestBedroomCheckBox"
    Key="GuestBedroomCheckBoxes" />
```

11.6.20. NoBot Control

NoBot là một điều khiển giống như CAPTCHA như phòng, chống thư rác/bot mà không cần bất kỳ tương tác người dùng. Phương pháp này dễ dàng để bỏ qua hơn nhiều so với một yêu cầu xác thực con người, nhưng NoBot có lợi ích hơn là việc đang được thực hiện hoàn toàn vô hình. NoBot có lẽ là thích hợp nhất cho các trang web lưu lượng truy cập thấp, nơi mà spam bài viết, nhận xét là một vấn đề và không cần phải đạt hiệu quả 100%.

NoBot sử dụng một vài kỹ thuật chống bot khác nhau:

- Buộc trình duyệt khách hàng phải thực hiện một phép tính JavaScript và xác nhận kết quả như là một phần của các postback (Thí dụ: việc tính toán có thể là một số đơn giản, hoặc cũng có thể liên quan đến DOM để bảo đảm thêm rằng có liên quan đến một trình duyệt).
- Làm một khoảng trống giữa khi một form được yêu cầu và khi nó có thể được postback (Thí dụ: Con người không thể hoàn thành một form trong vòng chưa đầy hai giây).
- Cấu hình một giới hạn với số lượng yêu cầu chấp nhận được cho mỗi địa chỉ IP trên một đơn vị thời gian (Thí dụ: một con người thường như không thể xác nhận một form nhiều hơn năm lần trong một phút).

NoBot có thể được kiểm tra bằng cách vi phạm bất kỳ các kỹ thuật nói trên: postback một cách nhanh chóng, đăng lại nhiều lần, hoặc vô hiệu hóa JavaScript trong trình duyệt.

Thuộc tính

Tên	Mô tả
OnGenerateChallengeAndResponse	Tùy chọn EventHandler <NoBotEventArgs> cung cấp một tùy chỉnh thực hiện các mã challenge/response
ResponseMinimumDelaySeconds	Tùy chọn số giây tối thiểu trước khi có một phản ứng (postback) được gọi là hợp lệ
CutoffWindowSeconds	Tùy chọn số giây của cửa sổ cắt theo dõi postbacks trước đó từ mỗi địa chỉ IP
CutoffMaximumInstances	Tùy chọn số lượng tối đa postbacks cho phép bởi một địa chỉ IP duy nhất

Thí dụ

```
<ajaxToolkit:NoBot  
ID="NoBot2"  
runat="server"  
OnGenerateChallengeAndResponse="CustomChallengeResponse"  
ResponseMinimumDelaySeconds="2"  
CutoffWindowSeconds="60"  
CutoffMaximumInstances="5" />
```

11.6.21. PagingBulletedList Control

PagingBulletedList là một ASP.NET AJAX extender có thể được gắn vào một BulletedList ASP.NET kiểm soát và cung cấp cho phía máy khách hình thức sắp xếp phân trang. Nó rất linh hoạt và cho phép bạn chỉ định, hoặc số lượng các ký tự được sử dụng trong các chỉ số nhóm hoặc số lượng tối đa của các mục để hiển thị mỗi chỉ mục. Nếu đầu vào chưa được sắp xếp (hoặc trên máy chủ hoặc máy khách), nó sẽ tạo ra các chỉ số tiêu đề hơn nhưng vẫn còn chức năng thích hợp.

Thuộc tính

Tên	Mô tả
TargetControlID	ID của BulletedList
ClientSort	Cho phép các mục sẽ được sắp xếp phía client hay không

IndexSize	Số lượng ký tự trong các nhóm chỉ số (bỏ qua nếu MaxItemPerPage được thiết lập)
MaxItemPerPage	Số mục tối đa trên mỗi trang (bỏ qua thuộc tính IndexSize)
Separator	Dấu ngăn được đặt giữa các chỉ số
SelectIndexCssClass	CSS class cho các chỉ số được lựa chọn
UnselectIndexCssClass	CSS class cho các chỉ số mà không được lựa chọn

Thí dụ

```
<ajaxToolkit:PagingBulletedListExtender ID="PBLE1" runat="server"
    TargetControlID="BulletedList1"
    ClientSort="true"
    IndexSize="1"
    MaxItemPerPage="20"
    Separator=" – "
    SelectIndexCssClass="selectIndex"
    UnselectIndexCssClass="unSelectIndex" />
```

11.6.22. PasswordStrength Control

PasswordStrength là một ASP.NET AJAX extender có thể được gắn vào một TextBox điều khiển ASP.NET được sử dụng cho các mục nhập của mật khẩu. Các extender PasswordStrength cho thấy độ mạnh của mật khẩu trong TextBox và cập nhật chính nó khi người dùng đánh mật khẩu. Các chỉ số có thể hiển thị độ mạnh của mật khẩu dưới dạng tin nhắn văn bản hoặc với một chỉ số thanh tiến trình. Kiểu dáng và vị trí của cả hai loại chỉ số có thể được tùy chỉnh. Yêu cầu về độ mạnh của mật khẩu cũng có thể được tùy chỉnh, cho phép chúng ta tùy chỉnh yêu cầu về độ mạnh của mật khẩu theo nhu cầu. Các tin nhắn văn bản mô tả độ mạnh hiện tại của mật khẩu cũng có thể được tùy chỉnh và giá trị mặc định của chúng có hỗ trợ sẵn cục bộ hóa. Chúng ta không có đủ các chuỗi cho tất cả ngôn ngữ hiện tại để có thể hiển thị nên chúng ta có thể dùng một ngôn ngữ chung cho một số ngôn ngữ khác. Một chỉ báo có thể được sử dụng để cung cấp hướng dẫn rõ về những gì thay đổi là cần thiết để đạt được một mật khẩu mạnh. Chỉ số này được hiển thị khi người dùng bắt đầu gõ vào TextBox và được ẩn đi mỗi khi textbox không được trỏ đến.

Thuộc tính

Tên	Mô tả
TargetControlID	ID của TextBox để đính kèm vào
DisplayPosition	Xác định vị trí hiển thị độ mạnh so với textbox
StrengthIndicatorType	Kiểu hiển thị độ mạnh của password (kiểu text hay là bar)
PreferredPasswordLength	Độ dài của mật khẩu ưu tiên
PrefixText	Văn bản có tiền tố để hiển thị khi StrengthIndicatorType = text
TextCssClass	Lớp CSS áp dụng cho các hiển thị văn bản khi StrengthIndicatorType = Text
MinimumNumericCharacters	Số ký tự tối thiểu
MinimumSymbolCharacters	Số biểu tượng tối thiểu
RequiresUpperAndLower CaseCharacters	Yêu cầu có cả ký tự hoa và thường
MinimumLowerCaseCharacters	Chỉ có hiệu lực nếu thuộc tính RequiresUpperAndLowerCaseCharacters là true. Chỉ định số ký tự thường tối thiểu.
MinimumUpperCaseCharacters	Chỉ có hiệu lực nếu thuộc tính RequiresUpperAndLowerCaseCharacters là true. Chỉ định số ký tự hoa tối thiểu.
TextStrengthDescriptions	Danh sách các mô tả được ngăn cách bởi dấu chấm phẩy được dùng khi StrengthIndicatorType = Text (tối thiểu là 2, tối đa là 10, sắp xếp từ yếu nhất đến mạnh nhất)
CalculationWeightings	Danh sách các giá trị số được ngăn cách bởi dấu chấm phẩy được dùng để xác định tỷ trọng độ mạnh của các ký tự. Yêu cầu phải có bốn giá trị được chỉ ra mà có tổng là 100. Các giá trị mặc định là 50; 11; 11; 20. Điều này có nghĩa là chiều dài mật khẩu được tính

	là chiếm 50% trong việc tính toán độ mạnh của mật khẩu, có ký tự số chiếm 11%, hoa -thường chiếm 11% và các biểu tượng chiếm 20%
BarBorderCssClass	Lớp css được áp dụng cho biên của thanh chỉ số khi StrengthIndicatorType=BarIndicator
BarIndicatorCssClass	Lớp css được áp dụng cho phần bên trong của thanh chỉ số khi StrengthIndicatorType=BarIndicator
StrengthStyles	Danh sách lớp CSS được ngăn cách bởi dấu chấm phẩy được dùng tùy thuộc vào độ mạnh của mật khẩu. Thuộc tính này override lại thuộc tính BarIndicatorCssClass / TextIndicatorCssClass. Thuộc tính BarIndicatorCssClass / TextIndicatorCssClass khác ở chỗ nó là một thuộc tính kiểu CSS của độ mạnh kiểu text hay bar (tùy thuộc vào kiểu lựa chọn) bắt kể độ mạnh của mật khẩu. Thuộc tính này sẽ thay đổi kiểu hiển thị dựa trên độ mạnh của mật khẩu cũng như số kiểu được chỉ ra trong thuộc tính này. Thuộc tính này có thể có tối đa 10 kiểu
HelpStatusLabelID	ID của label được dùng để hiển thị phần văn bản trợ giúp
HelpHandleCssClass	Lớp css được áp dụng cho phần tử trợ giúp được dùng để hiển thị hộp thông báo mô tả yêu cầu về mật khẩu
HelpHandlePosition	Vị trí của phần tử trợ giúp so với điều khiển đích

Thí dụ

```
<ajaxToolkit:PasswordStrength ID="PS" runat="server"
    TargetControlID="TextBox1"
    DisplayPosition="RightSide"
```

```

StrengthIndicatorType="Text"
PreferredPasswordLength="10"
PrefixText="Strength:"
TextCssClass="TextIndicator_TextBox1"
MinimumNumericCharacters="0"
MinimumSymbolCharacters="0"
RequiresUpperAndLowerCaseCharacters="false"
TextStrengthDescriptions="Very Poor;Weak;Average;Strong;Excellent"
TextStrengthDescriptionStyles="cssClass1;cssClass2;cssClass3;cssClass4;cssClass5
CalculationWeightings="50;11;11;20" />

```

11.6.23. Popup Control

PopupControl là một ASP.NET AJAX extender có thể được đính kèm vào bất kỳ điều khiển nào để mở một cửa sổ popup có hiển thị nội dung bổ sung. Cửa sổ popup này sẽ được tương tác và sẽ được đặt vào bên trong một ASP.NET AJAX UpdatePanel, vì vậy, nó có thể thực hiện các xử lý phức tạp dựa trên máy chủ (bao gồm cả postbacks) mà không ảnh hưởng đến phần còn lại của trang. Các cửa sổ popup có thể chứa bất kỳ nội dung nào, bao gồm cả điều khiển ASP.NET ở phía server, các phần tử HTML. Sau khi công việc của các cửa sổ popup hoàn tất, phía server chỉ việc gọi lệnh hủy bỏ nó và chạy một đoạn mã lệnh có liên quan ở phía máy khách để cập nhật trang.

Thuộc tính

Tên	Mô tả
TargetControlID	ID của điều khiển muốn gắn vào
PopupControlID	ID của điều khiển hiển thị
Position	Thiết lập tùy chỉnh vị trí của hiển thị popup so với điều khiển đích (top, left, bottom, center)
CommitProperty	Thiết lập tùy chọn chỉ ra rằng điều khiển được gắn extender này sẽ nhận kết quả từ popup
CommitScript	Thiết lập tùy chỉnh đoạn mã lệnh sẽ được thực hiện sau khi thiết lập kết quả ở popup
OffsetX/OffsetY	Số pixel chênh lệch của popup từ vị trí mặc định của nó được chỉ định trong thuộc tính position
Animations	Hiệu ứng khi hiển thị popup

Thí dụ

```
<ajaxToolkit:PopupControlExtender ID="PopEx" runat="server"
    TargetControlID="DateTextBox"
    PopupControlID="Panel1"
    Position="Bottom" />
```

11.6.24. Rating Control

Điều khiển Rating cung cấp một thẻ hiện đánh giá trực quan cho phép người dùng chọn số lượng các ngôi sao đại diện cho đánh giá của họ. Các nhà thiết kế trang có thể chỉ định đánh giá ban đầu, sự đánh giá tối đa cho phép, chiều và hướng của các ngôi sao, và tùy chỉnh style cho các trạng thái khác nhau có thể có của một ngôi sao. Đánh giá cũng hỗ trợ sự kiện ClientCallBack cho phép một đoạn code chạy sau khi người sử dụng đánh giá.

Thuộc tính

Tên	Mô tả
AutoPostBack	Thiết lập là true để gửi một postback khi click vào item để đánh giá
CurrentRating	Khởi tạo giá trị đánh giá
MaxRating	Giá trị đánh giá tối đa
ReadOnly	Cho phép sửa đổi đánh giá hay không
StarCssClass	Lớp css cho các ngôi sao hiển hiện
WaitingStarCssClass	Lớp css cho các ngôi sao ở trạng thái chờ (chưa được chọn)
FilledStarCssClass	Lớp css cho các ngôi sao ở trạng thái đầy
EmptyStarCssClass	Lớp css cho các ngôi sao ở trạng thái trống
RatingAlign	Cách sắp các ngôi sao (ngang hay dọc)
RatingDirection	Hướng các ngôi sao là từ trái qua phải hay ngược lại (LeftToRightTopToBottom hoặc RightToLeftBottomToTop)
OnChanged	Sự kiện ClientCallBack được gọi khi đánh giá được thay đổi
Tag	Tham biến theo ý riêng để chuyển đến clientcallback

Thí dụ

```
<ajaxToolkit:Rating ID="ThaiRating" runat="server"
    CurrentRating="2"
    MaxRating="5"
    StarCssClass="ratingStar"
    WaitingStarCssClass="savedRatingStar"
    FilledStarCssClass="filledRatingStar"
    EmptyStarCssClass="emptyRatingStar"
    OnChanged="ThaiRating_Changed" />
```

11.6.25. ReorderList Control

ReorderList là một điều khiển ASP.NET AJAX thực hiện ràng buộc dữ liệu với danh sách liệt kê các mục có thể được sắp xếp lại. Để sắp xếp lại các mục trong danh sách, người dùng chỉ cần kéo thanh điều khiển của mặt hàng đó đến vị trí mới. Thông tin phản hồi được hiển thị dưới dạng đồ họa, nơi mục này được đặt bởi người sử dụng. Các nguồn dữ liệu được cập nhật sau khi mục đó được kéo tới vị trí mới.

Khi bị ràng buộc vào dữ liệu, điều khiển ReorderList sẽ hoạt động giống như nhiều điều khiển databound khác. Nếu dữ liệu bạn đang hiển thị có một trường được sắp xếp thứ tự (Thí dụ như truy vấn sắp xếp theo cột này), và cột đó thuộc kiểu số nguyên, các ReorderList có thể thực hiện sắp xếp lại tự động nếu thuộc tính SortOrderField được thiết lập.

Thuộc tính:

Tên	Mô tả
DataSourceID	Nguồn dữ liệu được dùng để gắn vào điều khiển
DataKeyField	Khóa riêng cho dữ liệu
SortOrderField	Trường đại diện cho thứ tự sắp xếp các mục
ItemInsertLocation	Xác định nơi mà một phần tử mới được thêm vào danh sách có thể là begin hoặc end
DragHandleAlignment	Nơi điều khiển kéo được gắn so với phần tử, có thể là left, right, top, bottom
AllowReorder	Cho phép kéo thả để sắp xếp hay không, mặc định là true
ItemTemplate	Mẫu để hiển thị các phần tử trong danh sách
EditItemTemplate	Mẫu để hiển thị cho một dòng trong chế độ sửa đổi

ReorderTemplate	Mẫu được dùng để hiển thị vị trí được thả xuống khi thực hiện sắp xếp lại, mẫu này không được gắn dữ liệu
InsertItemTemplate	Mẫu được dùng để thể hiện một phần tử mới được thêm vào
DragHandleTemplate	Mẫu cho mục khi người dùng kéo để sắp xếp lại mục
EmptyListTemplate	Mẫu được dùng khi danh sách rỗng, mục này cũng không được gắn dữ liệu
PostBackOnReorder	Xác định cho phép postback hay callback khi được sắp xếp lại, nếu dùng bất kỳ hàm sửa đổi hay xóa nào thì nên thiết lập là enable

Thí dụ

```
<ajaxToolkit:ReorderList ID="ReorderList1" runat="server">
    DataSourceID="ObjectDataSource1"
    DragHandleAlignment="Left"
    ItemInsertLocation="Beginning"
    DataKeyField="ItemID"
    SortOrderField="Priority"
    AllowReorder="true">
    <ItemTemplate>...</ItemTemplate>
    <ReorderTemplate>...</ReorderTemplate>
    <DragHandleTemplate>...</DragHandleTemplate>
    <InsertItemTemplate>...</InsertItemTemplate>
</ajaxToolkit:ReorderList>
```

11.6.26. Resizable Control

ResizableControl là một extender có thể gắn vào bất kỳ phần tử nào trên một trang web và cho phép người dùng thay đổi kích cỡ với một xử lý gắn vào góc dưới bên phải của kiểm soát. Việc xử lý thay đổi kích cỡ cho phép người dùng thay đổi kích cỡ các phần tử như thế nó là một cửa sổ. Phần thể hiện cho phép thay đổi kích cỡ có thể được thiết kế bởi các nhà thiết kế thông qua CSS. Các nội dung bên trong phần tử có thể sử dụng CSS để tự động thay đổi kích cỡ để phù hợp với kích thước mới. Ngoài ra, ResizableControl có hai sự kiện (onresizing và onresize) mà ta có thể đính kèm đoạn mã script để cho phép bố trí logic phức tạp hơn. Kích thước Element được bảo tồn qua postbacks đến máy chủ và thuộc

tính “kích thước” có thể truy cập cả trên máy khách và máy chủ đều có thể được sử dụng để cho phép thay đổi kích thước. ResizableControl có thể giới hạn chiều rộng và chiều cao tối thiểu, tối đa của mục tiêu.

Thuộc tính

Tên	Mô tả
TargetControlID	ID của phần tử đích
HandleCssClass	Tên của lớp css được dùng để áp dụng vào điều khiển
ResizableCssClass	Tên của lớp css được dùng để áp dụng vào điều khiển khi đang thay đổi kích thước
MinimumWidth/MinimumHeight	Kích thước tối thiểu của phần tử đích
MaximumWidth/MaximumHeight	Kích thước tối đa của phần tử đích
HandleOffsetX/HandleOffsetY	Vị trí của phần tử đích

Sự kiện

Tên	Mô tả
OnClientResizeBegin	Sự kiện xảy ra khi phần tử bắt đầu thay đổi kích thước
OnClientResizing	Sự kiện xảy ra khi phần tử đang được thay đổi kích thước
OnClientResize	Sự kiện xảy ra sau khi phần tử được thay đổi kích thước

Thí dụ

```
<ajaxToolkit:ResizableControlExtender ID="RCE" runat="server"
    TargetControlID="PanelImage"
    HandleCssClass="handleImage"
    ResizableCssClass="resizingImage"
    MinimumWidth="50"
    MinimumHeight="20"
    MaximumWidth="260"
    MaximumHeight="130"
    OnClientResize="OnClientResizeImage"
    HandleOffsetX="3"
    HandleOffsetY="3" />
```

11.6.27. RoundedCorners Control

Các extender RoundedCorners làm tròn góc cho các phần tử. Để thực hiện điều này nó chèn vào các phần tử trước và sau khi các phần tử đó được chọn, do đó, chiều cao tổng thể của phần tử này sẽ thay đổi một chút. Bạn có thể chọn các góc của panel đích cần phải được làm tròn bằng cách thiết lập thuộc tính Corners trên extender là None, TopLeft, TopRight, BottomRight, BottomLeft, Top, Right, Bottom, Left, hoặc tất cả.

Thuộc tính

Tên	Mô tả
TargetControlID	ID của điều khiển đích
Radius	Bán kính của các góc, mặc định là 5px
Corners	Góc được làm tròn, có thể là None, TopLeft, TopRight, BottomRight, BottomLeft, Top, Right, Bottom, Left, hoặc tất cả

Thí dụ

```
<ajaxToolkit:RoundedCornersExtender ID="rce" runat="server"
    TargetControlID="Panel1"
    Radius="6"
    Corners="All" />
```

11.6.28. Tab

TabContainer là một ASP.NET AJAX Control giúp tạo ra một tập hợp các Tabs có thể được dùng để tổ chức các trang nội dung. TabContainer lưu trữ một số kiểm soát TabPanel.

Mỗi TabPanel định nghĩa một HeaderText hoặc HeaderTemplate cũng như ContentTemplate định nghĩa nội dung của nó. Các tab gần đây nhất vẫn chọn sau khi postback, và trạng thái Enabled của các tab vẫn còn sau khi postback.

Thuộc tính của TabContainer

Tên	Mô tả
ActiveTabChanged	Sự kiện xảy ra ở máy chủ khi tab bị thay đổi sau khi postback
OnClientActiveTabChanged	Tên của hàm javascript gắn vào sự kiện tabChanged phía client
CssClass	Lớp css định dạng cho các tab

ActiveTabIndex	Tab được hiển thị đầu tiên
Height	Thiết lập chiều cao của panel chưa tính phần header
Width	Chiều rộng của panel
ScrollBars	Cho phép hiển thị thanh cuộn (None, Horizontal, Vertical, Both, Auto) trong phần hàm của tabContainer
TabStripPlacement	Xác định chỗ đặt các tabs ở trên hay dưới tabContainer

Thuộc tính cẩu TabPanel

Tên	Mô tả
Enabled	Cho phép hiển thị các tab cho tabPanel theo mặc định, có thể thay đổi ở phía client
OnClientClick	Tên hàm javascript được gắn vào sự kiện click tab phía client
HeaderText	Phần chuỗi hiển thị trên tab
HeaderTemplate	Một TemplateInstance.Single ITemplate được dùng cho tiêu đề
ContentTemplate	Một TemplateInstance.Single ITemplate được dùng cho nội dung

Thí dụ

```
<ajaxToolkit:TabContainer runat="server"
    OnClientActiveTabChanged="ClientFunction"
    Height="110px">
<ajaxToolkit:TabPanel runat="server"
    HeaderText="Signature and Bio">
    <ContentTemplate>
        ...
    </ContentTemplate>
    />
</ajaxToolkit:TabContainer>
```

11.6.29. TextBoxWatermark Control

TextBoxWatermark là một ASP.NET AJAX extender có thể được gắn vào một TextBox trong ASP.NET để có được hành vi "watermark". Khi một watermark TextBox rỗng, nó sẽ hiển thị một thông điệp tới người dùng với kiểu được tùy chỉnh bởi CSS. Một khi người dùng đã gõ một số ký tự vào TextBox, watermarked sẽ không xuất hiện nữa. Mục đích chính của một watermark là cung cấp thêm thông tin cho người sử dụng về các TextBox mà không gây nên sự xáo trộn lên phần còn lại của trang.

Thuộc tính

Tên	Mô tả
TargetControlID	ID của textbox
WatermarkText	Phần chuỗi hiển thị khi textbox không có nội dung
WatermarkCssClass	Lớp css được áp dụng cho textbox khi nó không có nội dung

Thí dụ

```
<ajaxToolkit:TextBoxWatermarkExtender ID="TBWE2" runat="server"
    TargetControlID="TextBox1"
    WatermarkText="Type First Name Here"
    WatermarkCssClass="watermarked" />
```

BÀI TẬP CHƯƠNG 11

Một số thí dụ với Ajax Toolkit 3.5

Khi sử dụng AjaxToolkit thì phải khai báo ScriptManager hoặc là ToolkitScriptManager

```
<asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
<asp:ToolkitScriptManager ID="ScriptManager1" runat="server">
    </asp:ToolkitScriptManager>
```

1. Accordion Control

Accordion control giúp bạn định nghĩa nhiều lớp và trình bày chúng từng cái một. Nó giống như là có nhiều CollapsiblePanel controls mà chỉ có một cái có thể mở rộng tại một thời điểm. Accordion control chứa một hoặc nhiều AccordionPane controls. Mỗi AccordionPane control có một mẫu cho phần header và phần content của nó.

1.1. Bước 1: kéo thả Accordion Control và tạo các AccordionPane.

```
<asp:Accordion
    ID="Accordion1"
    CssClass="accordion"
    HeaderCssClass="accordionHeader"
    HeaderSelectedCssClass="accordionHeaderSelected"
    ContentCssClass="accordionContent"
    runat="server">
    <Panes>
        <asp:AccordionPane ID="AccordionPanel1" runat="server">
            <Header>Pane 1</Header>
            <Content>
                Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor
                congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada
                libero, sit amet commodo magna eros quis urna.
            </Content>
        </asp:AccordionPane>
        <asp:AccordionPane ID="AccordionPanel2" runat="server">
            <Header>Pane 2</Header>
            <Content>
                Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor
                congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada
                libero, sit amet commodo magna eros quis urna.
            </Content>
    </Panes>

```

Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci.

```
</Content>
</asp:AccordionPane>
<asp:AccordionPane ID="AccordionPane3" runat="server">
    <Header>Pane 3</Header>
    <Content>
        Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor
        congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada
        libero, sit amet commodo magna eros quis urna.
        Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus.
        Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac
        turpis egestas. Proin pharetra nonummy pede. Mauris et orci.
    </Content>
</asp:AccordionPane>
</Panes>
</asp:Accordion>
```

1.2. Bước 2: tạo CSS cho nội dung trong AccordionPane

```
<style type="text/css">
    .accordion {
        width: 400px;
    }
    .accordionHeader {
        border: 1px solid #2F4F4F;
        color: white;
        background-color: #2E4d7B;
        font-family: Arial, Sans-Serif;
        font-size: 12px;
        font-weight: bold;
        padding: 5px;
        margin-top: 5px;
        cursor: pointer;
    }
    .accordionHeaderSelected {
        border: 1px solid #2F4F4F;
        color: white;
        background-color: #5078B3;
```

```

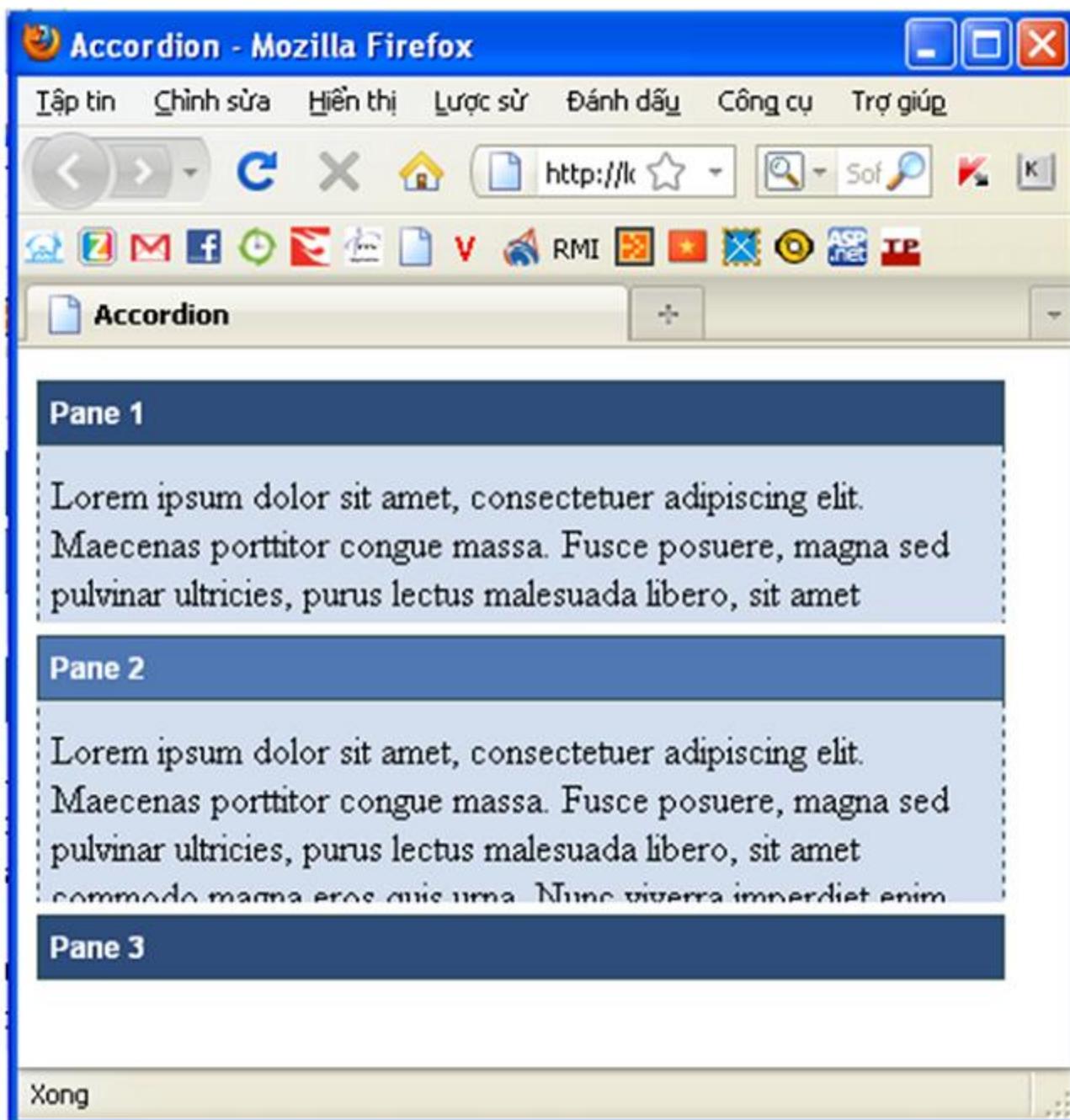
font-family: Arial, Sans-Serif;
font-size: 12px;
font-weight: bold;
padding: 5px;
margin-top: 5px;
cursor: pointer;
}

.accordionContent {
background-color: #D3DEEF;
border: 1px dashed #2F4F4F;
border-top: none;
padding: 5px;
padding-top: 10px;
}

</style>

```

Kết quả:



2. AlwaysVisible Control

AlwaysVisible Control là một extender được dùng để hiển thị một cách liên tục đến một điều khiển ASP.NET. Điều khiển được mở rộng sẽ luôn luôn di chuyển đến một vị trí cố định trên trang bất kể trang thị thay đổi kích thước hay bị cuộn.

2.1. Bước 1: Kéo thả một Panel:

```
<asp:Panel ID="Panel1" runat="server" CssClass="staticPanel">  
    <h2> Hello Words!!!</h2>  
</asp:Panel>
```

2.2. Bước 2: Trong phần design, từ Panel trên chọn Add Extender, chọn AlwaysVisibleControlExtender. Ta có đoạn code như sau:

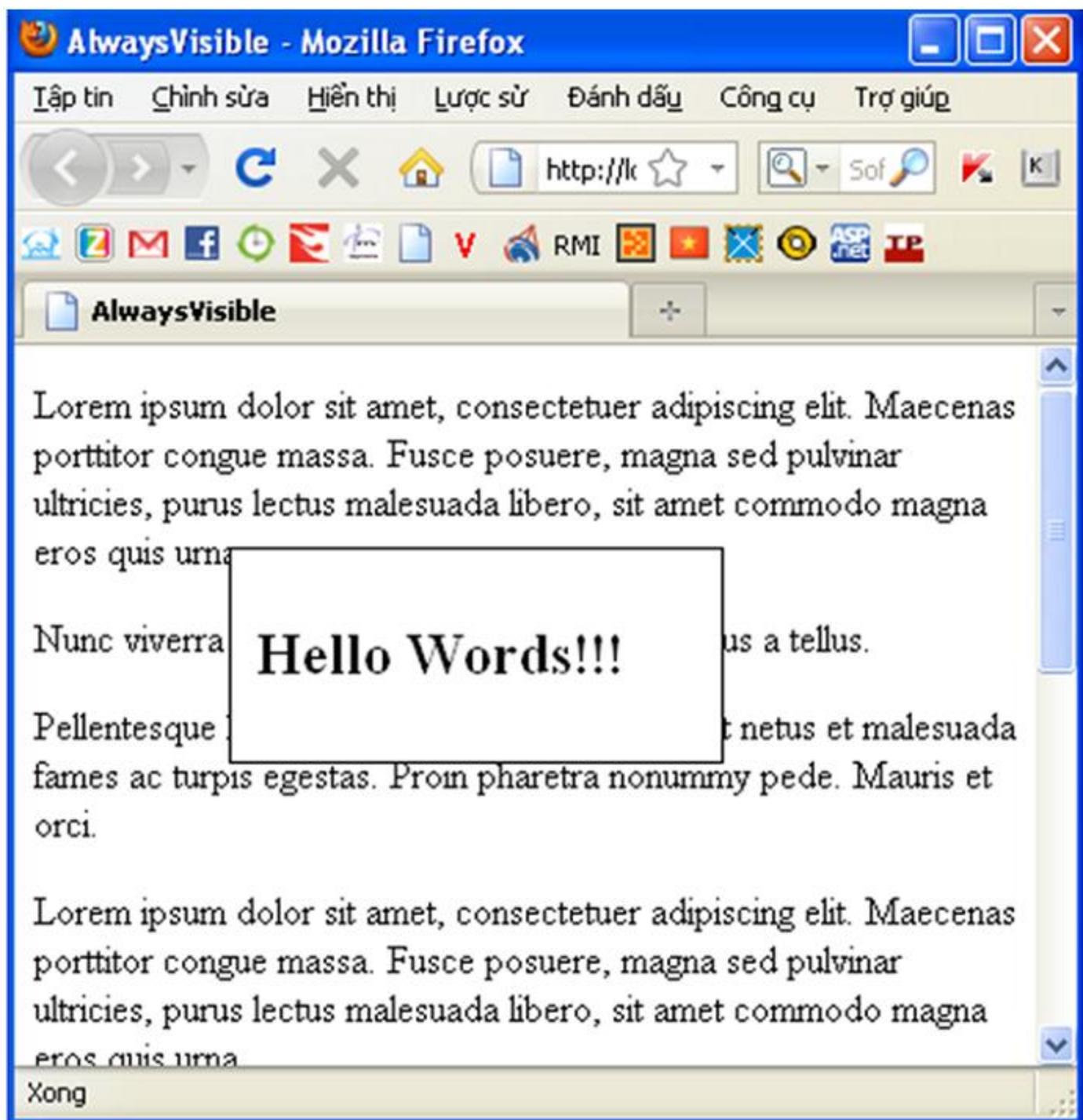
```
<asp:AlwaysVisibleControlExtender  
    ID="Panel1_AlpwaysVisibleControlExtender"  
    runat="server"  
    Enabled="True"  
    VerticalSide="Middle"  
    VerticalOffset="20"  
    HorizontalSide="Center"  
    HorizontalOffset="20"  
    TargetControlID="Panel1">  
</asp:AlwaysVisibleControlExtender>
```

2.3. Bước 3: Tạo nội dung bên trong trang để thấy rõ chức năng của Control này.

2.4. Bước 4: Tạo CSS cho Panel:

```
<style type="text/css">  
.staticPanel {  
    width: 110px;  
    background-color: White;  
    border: solid 1px black;  
    padding: 10px;  
}  
</style>
```

Kết quả:



3. Animation Control

Phần mở rộng được sử dụng để tạo các hiệu ứng cho điều khiển khi xảy ra sự kiện.

3.1. Bước 1: Tạo một Panel, là đối tượng hiển thị Animation.

```
<asp:Panel  
    ID="Message"  
    runat="server">  
    Pay attention to me!  
</asp:Panel>
```

3.2. Bước 2: Tạo một LinkButton để thực hiện Animation.

```
<asp:LinkButton ID="lnkYellowFade" OnClientClick="return false;"  
    runat="server">Play Animation</asp:LinkButton>
```

3.3. Bước 3: Từ LinkButton Add Extender, chọn AnimationExtender.

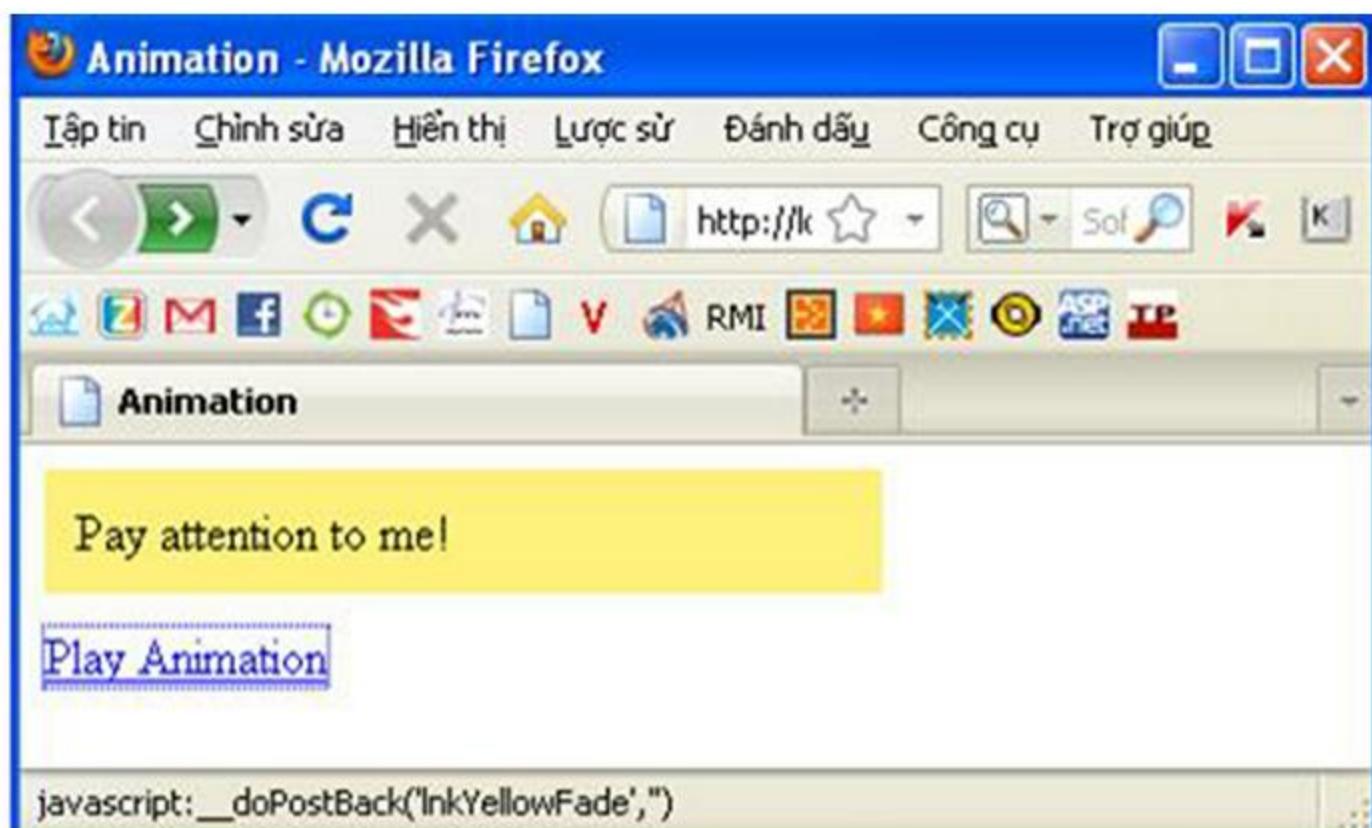
```
<asp:AnimationExtender  
    ID="lnkYellowFade_AnimationExtender"  
    runat="server"  
    Enabled="True"  
    TargetControlID="lnkYellowFade">  
    <Animations>  
        <OnClick>  
            <Sequence>  
                <Color AnimationTarget="Message" Duration="2" Property="style"  
                    PropertyKey="backgroundColor"  
                    StartValue="#FFEF66"  
                    EndValue="#FFFFFF" />  
            </Sequence>  
        </OnClick>  
    </Animations>  
</asp:AnimationExtender>
```

Trong thẻ AnimationExtender tạo các Animation. Bên trong là chức năng Click của LinkButton.

3.4. Bước 4: Tạo CSS cho Panel.

```
<style type="text/css">  
    #Message {  
        width: 250px;  
        padding: 10px;  
        margin-bottom:10px;    }  
</style>
```

Kết quả:



4. ColorPicker Control

Phần mở rộng ColorPicker cho phép bạn hiển thị một bảng pop-up màu khi con trỏ di chuyển đến phần tử input. Bạn có thể gắn ColorPicker vào bất cứ textbox nào của ASP.NET. Nó cung cấp một hàm cho phép lựa chọn màu ở phía client với giao diện người dùng. Ngoài ra, bạn có thể chỉ ra một button để hiển thị một popup lựa chọn màu và một điều khiển cho phép xem trước màu từ bảng màu. Bạn có thể cung cấp một textbox để khi mà người dùng nhập vào một giá trị màu thì ColorPicker có thể hiện thị màu tương ứng nếu màu đó không có trong bảng màu mặc định.

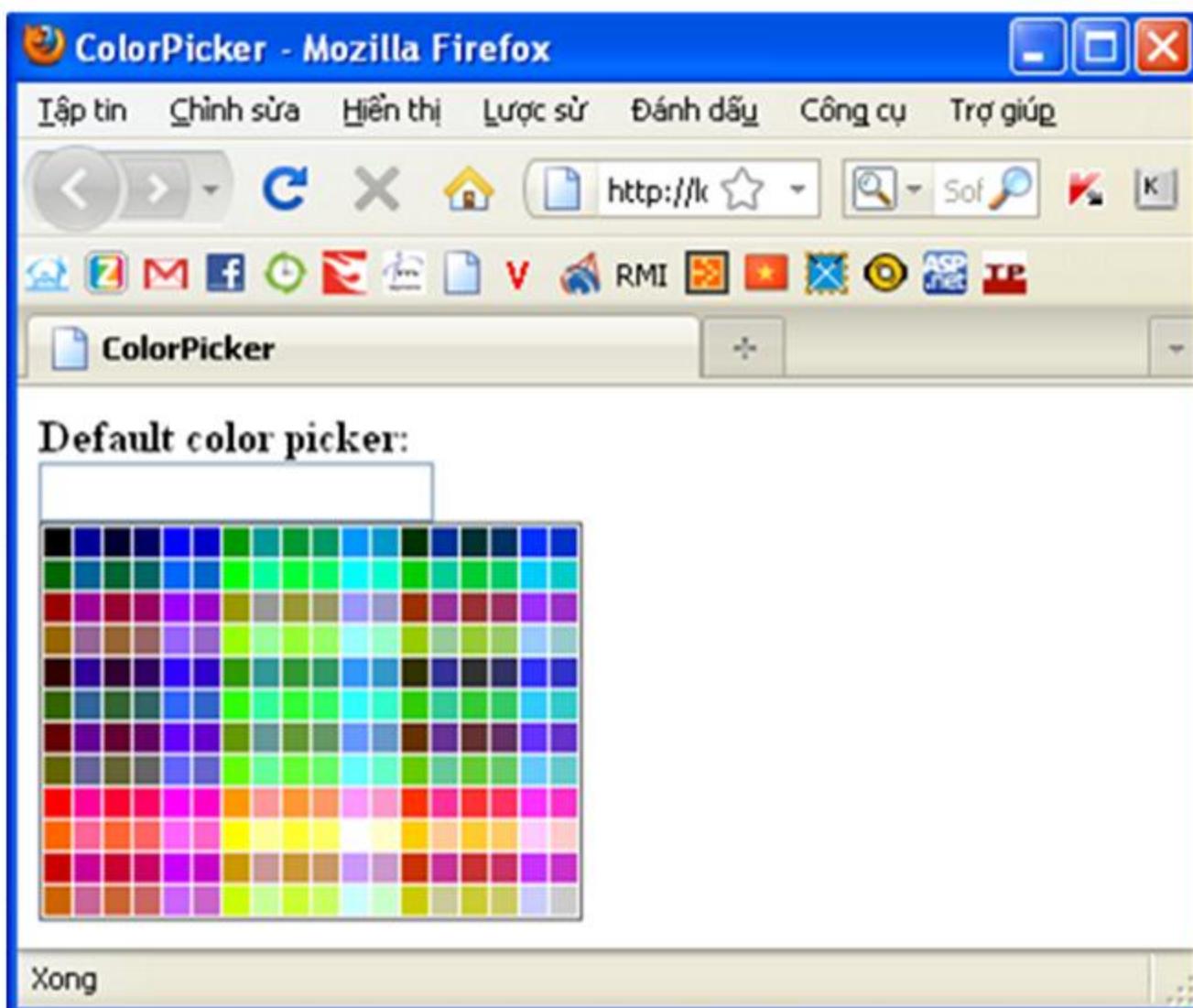
4.1. Bước 1: Kéo thả một TextBox

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
```

4.2. Bước 2: Từ TextBox AddExtender

```
<asp:ColorPickerExtender ID="TextBox1_ColorPickerExtender" runat="server"
    Enabled="True" TargetControlID="TextBox1">
</asp:ColorPickerExtender>
```

Kết quả:



5. ComboBox Control

ComboBox là một điều khiển của ASP.NET AJAX, nó giống như AutoCompleteExtender, kết hợp linh hoạt với một Textbox với một danh sách tùy chọn cho phép người dùng có thể lựa chọn. Nó có các thuộc tính, hành vi hay quy ước đặt tên tương tự như trên Form combobox, và có cùng lớp cơ sở như là ListBox, BulletedList và DropDownList của web control. Thật vậy, ComboBox được xem như là DropDownList nhưng lại có thể gõ trực tiếp vào như là textbox.

5.1. Bước 1: Kéo thả một ComboBox với các ListItem

```
<asp:ComboBox ID="ComboBox1" runat="server">
    <asp:ListItem Value="0">Cao</asp:ListItem>
    <asp:ListItem Value="1">Trung Bình</asp:ListItem>
    <asp:ListItem Value="2">Thap</asp:ListItem>
</asp:ComboBox>
```

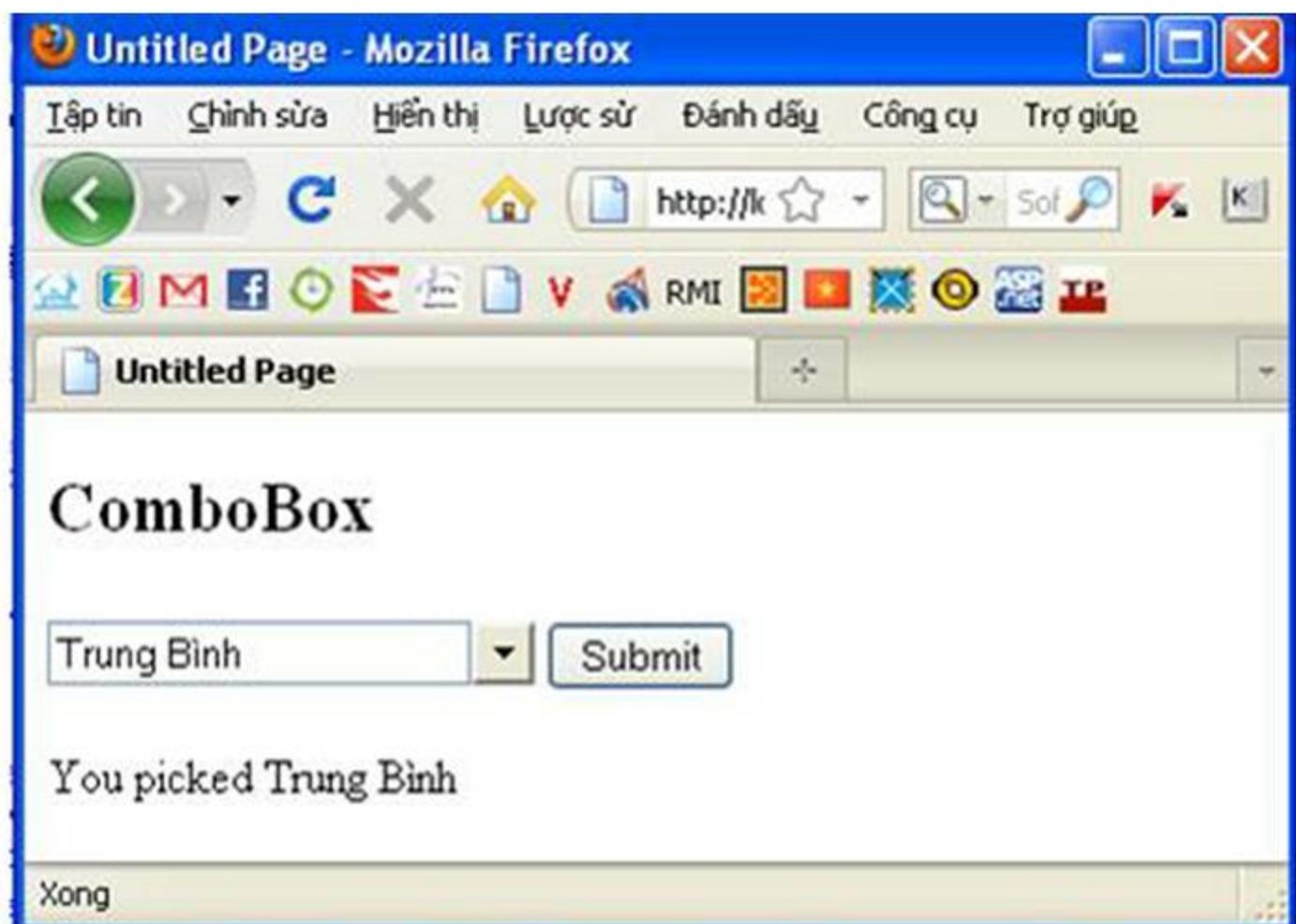
5.2. Bước 2: Tạo một Button và một Label để hiển thị nội dung trong ComboBox được chọn

```
<asp:Button ID="btnSubmit" Text="Submit" Runat="server"
    OnClick="btnSubmit_Click"/> <br /> <br />
<asp:Label ID="lblSelection" Runat="server" />
```

5.3. Tạo một Script là Button_Click

```
<script runat="server">  
    public void btnSubmit_Click(object sender, EventArgs e) {  
        lblSelection.Text = "You picked " + ComboBox1.SelectedItem.Text;  
    }  
</script>
```

Kết quả:



6. PopupCalendar Control

Điều khiển Calendar extender có thể được gắn với bất kỳ điều khiển textbox nào của ASP.NET. Nó cung cấp hàm định dạng ngày phía client với định dạng ngày tùy biến với giao diện trong một điều khiển popup. Bạn có thể tương tác với calendar bằng cách click vào một ngày để chọn, hoặc Today để chọn ngày hiện tại. Ngoài ra, mũi tên trái phải có thể được sử dụng để chuyển tới tháng trước hoặc tháng sau. Bằng cách click vào tiêu đề của calendar bạn có thể thay đổi hiển thị của các ngày trong tháng hiện tại thành các tháng trong năm hiện tại. Click khác sẽ chuyển các năm trong thập kỷ hiện tại. Hành động này cho phép bạn dễ dàng chuyển đến một ngày trong quá khứ hoặc tương lai từ điều khiển calendar.

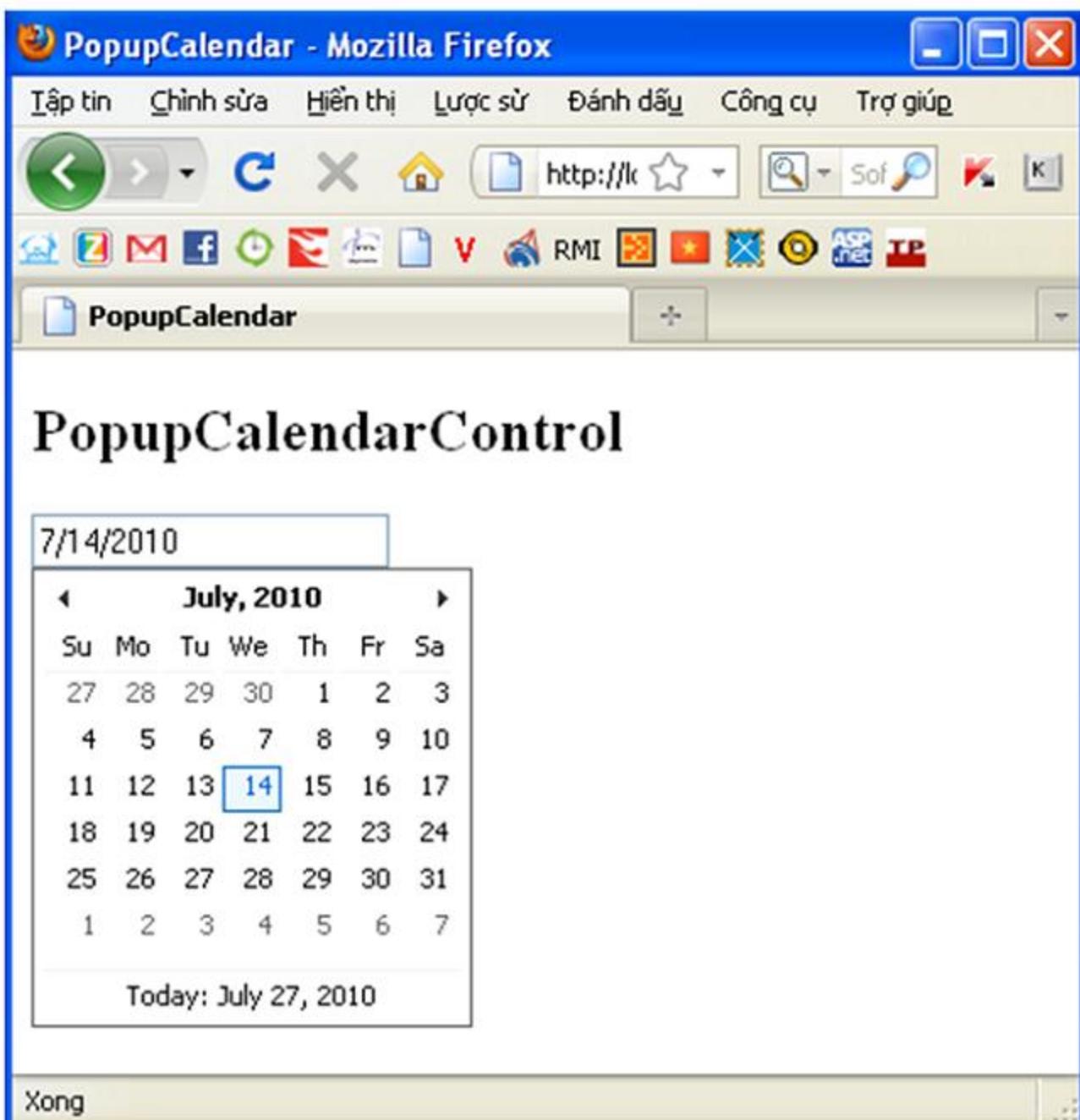
6.1. Bước 1: Tạo một TextBox.

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
```

6.2. Bước 2: Từ TextBox đã tạo ở trên Add Extender là CalendarExtender

```
<asp:CalendarExtender ID="TextBox1_CalendarExtender" runat="server"  
    Enabled="True" TargetControlID="TextBox1">  
</asp:CalendarExtender>
```

Kết quả:



7. AsyncFileUpload Control

AsyncFileUpload cho phép bạn upload file lên server một cách bất đồng bộ, kết quả của file upload có thể được kiểm tra cả ở phía server lẫn client, bạn có thể lưu file được upload bằng cách gọi phương thức SaveAs() trong điều khiển sự kiện UploadedComplete ở server.

7.1. Bước 1: Tạo một AsyncFileUpload

```
<ajaxToolkit:AsyncFileUpload ID="AsyncFileUpload1" Width="400px"
runat="server"

    OnClientUploadError="uploadError"
    OnClientUploadStarted="StartUpload"
    OnClientUploadComplete="UploadComplete"
    CompleteBackColor="Lime" UploaderStyle="Modern"
    ErrorBackColor="Red" ThrobberID="Throbber"
    onuploadedcomplete="AsyncFileUpload1_UploadedComplete"
    UploadingBackColor="#66CCFF" />
```

7.2. Bước 2: Tạo một Label để thể hiện một Image khi đang upload

```
<asp:Label ID="Throbber" runat="server" Style="display: none">
    
</asp:Label>
```

7.3. Bước 3: Đưa các Script

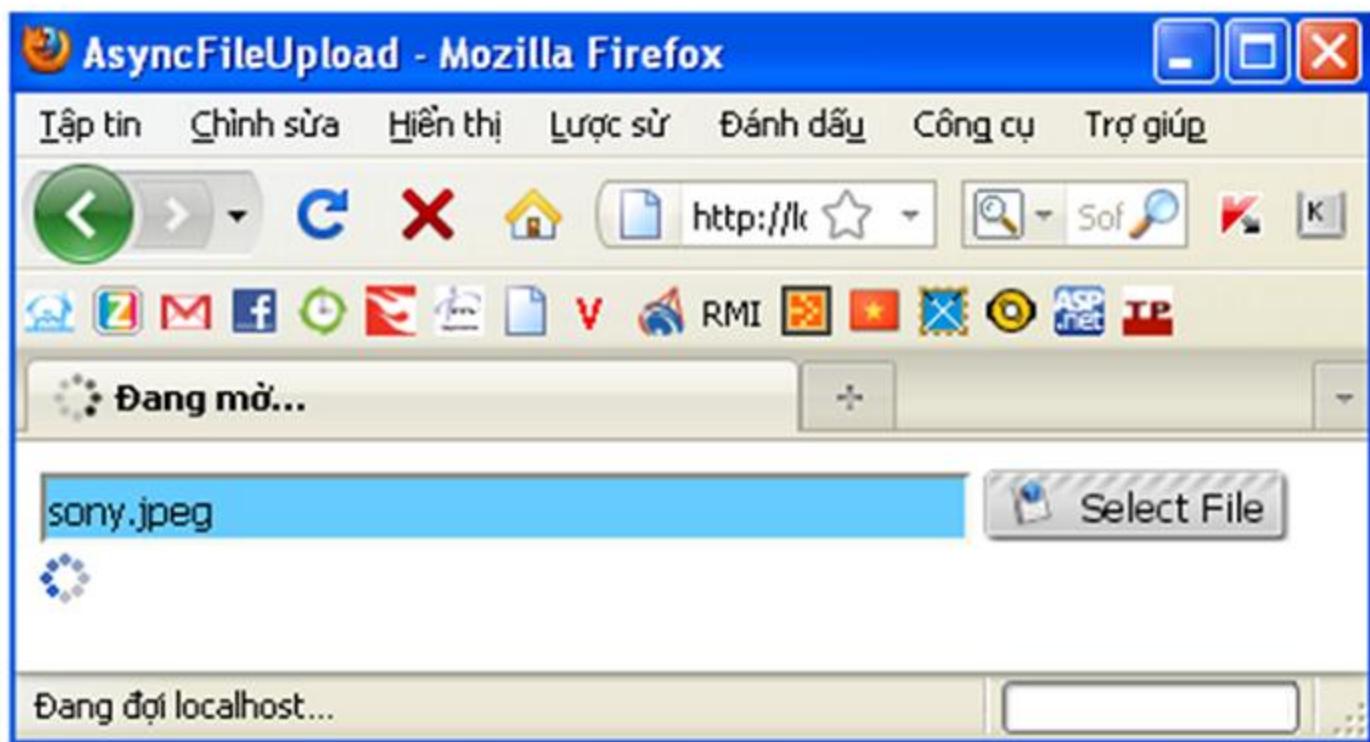
```
<script type="text/javascript">

    function uploadError(sender, args) {
        document.getElementById('lblStatus').innerText = args.get_fileName() + "<span style='color:red;'>" + args.get_errorMessage() + "</span>";
    }

    function StartUpload(sender, args) {
        document.getElementById('lblStatus').innerText = 'Uploading Started.';
    }

    function UploadComplete(sender, args) {
        var filename = args.get_fileName();
        var contentType = args.get_contentType();
        var text = "Size of " + filename + " is " + args.get_length() + " bytes";
        if (contentType.length > 0) {
            text += " and content type is " + contentType + ".";
        }
        document.getElementById('lblStatus').innerText = text;
    }
</script>
```

Kết quả:



8. DragPanelExtender

DragPanel extender cho phép người dùng dễ dàng thêm một phần có thể kéo đi được trong điều khiển của họ. DragPanel điều khiển bất kỳ panel nào của ASP.NET và thêm một tham số nói lên điều khiển được dùng như là có thể kéo đi được. Khi được khởi tạo, người dùng có thể thoải mái kéo panel quanh trang web sử dụng extender này.

8.1. Bước 1: Tạo Panel

```
<asp:Panel ID="PnlContainer" runat="server" cssclass="dragContainer">
    <asp:Panel ID="PnlHeader" runat="server" CssClass="dragHeader">
        Click and Drag Here To Move ME.....around the page</asp:Panel>
    <asp:Panel ID="PnlDetail" runat="server" CssClass="dragDetail">
        Here is just some content where you can add as much text as you like<br />
        line 1<br />        line 2<br />
        Remmeber that you can add any html text here<br /><br />
    </asp:Panel>
</asp:Panel>
```

8.2. Bước 2: Tạo một DragPanelExtender

```
<asp:DragPanelExtender ID="PnlContainer_DragPanelExtender" runat="server"
    DragHandleID="PnlHeader" Enabled="True"
    TargetControlID="PnlContainer">
</asp:DragPanelExtender>
```

8.3. Bước 3: Tạo CSS cho các nội dung bên trong trang

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

.dragContainer{
    background-color: #FFC0FF;           height: 282px;
    width: 357px;                      border-bottom-color: black;
}

.dragHeader{
    background-color: #8080FF;           height: 48px;
    width: 358px;
}

.dragDetail{
    background-color: #FFC0FF;           height: 213px;
    width: 357px;
}

</style>
```

8.4. Bước 4: Script

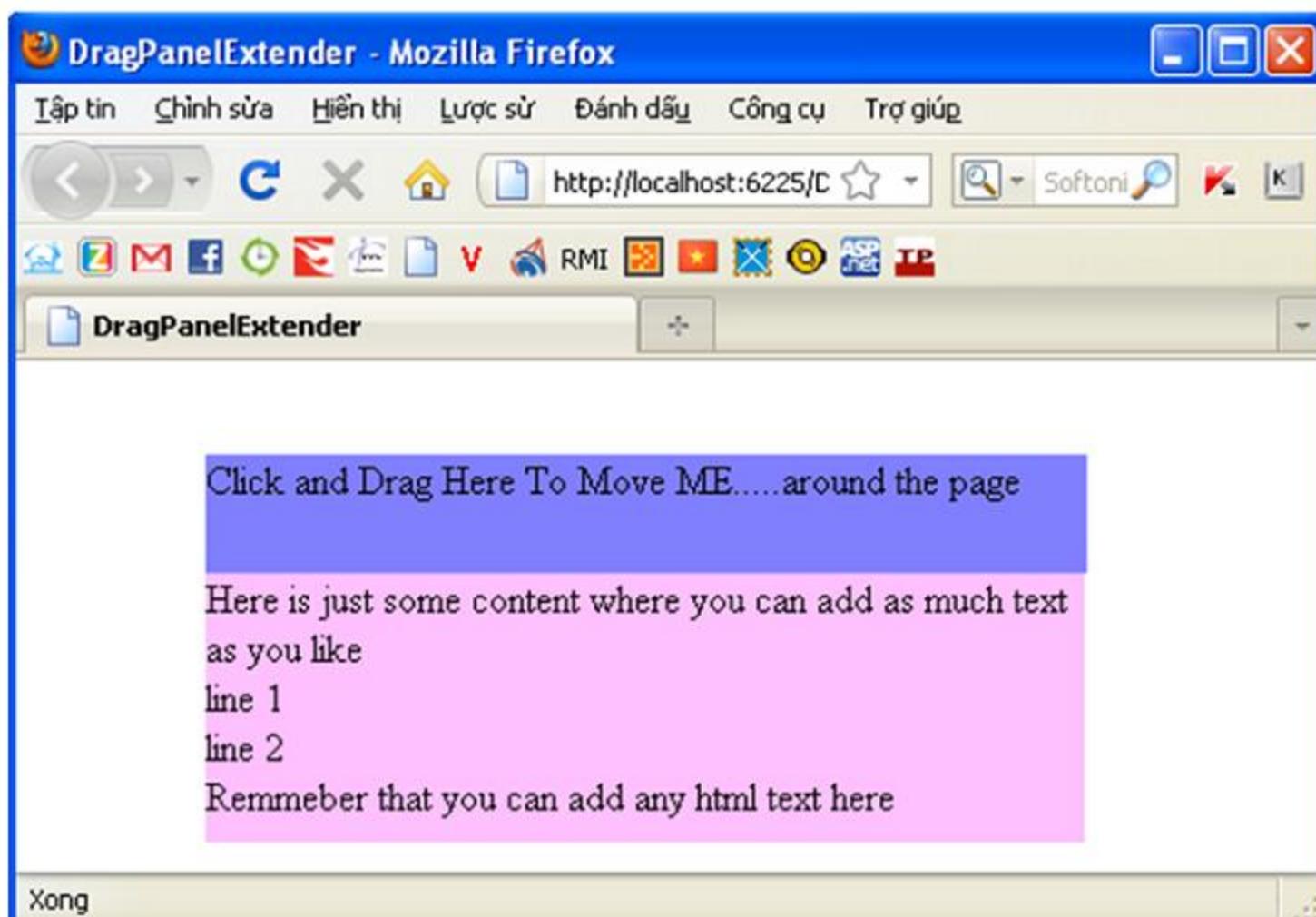
```
<script type="text/javascript">

    function setBodyHeightToContentHeight() {
        document.body.style.height =
Math.max(document.documentElement.scrollHeight,
        document.body.scrollHeight) + "px";
    }

    setBodyHeightToContentHeight();

</script>
```

Kết quả:



9. DropShaDown

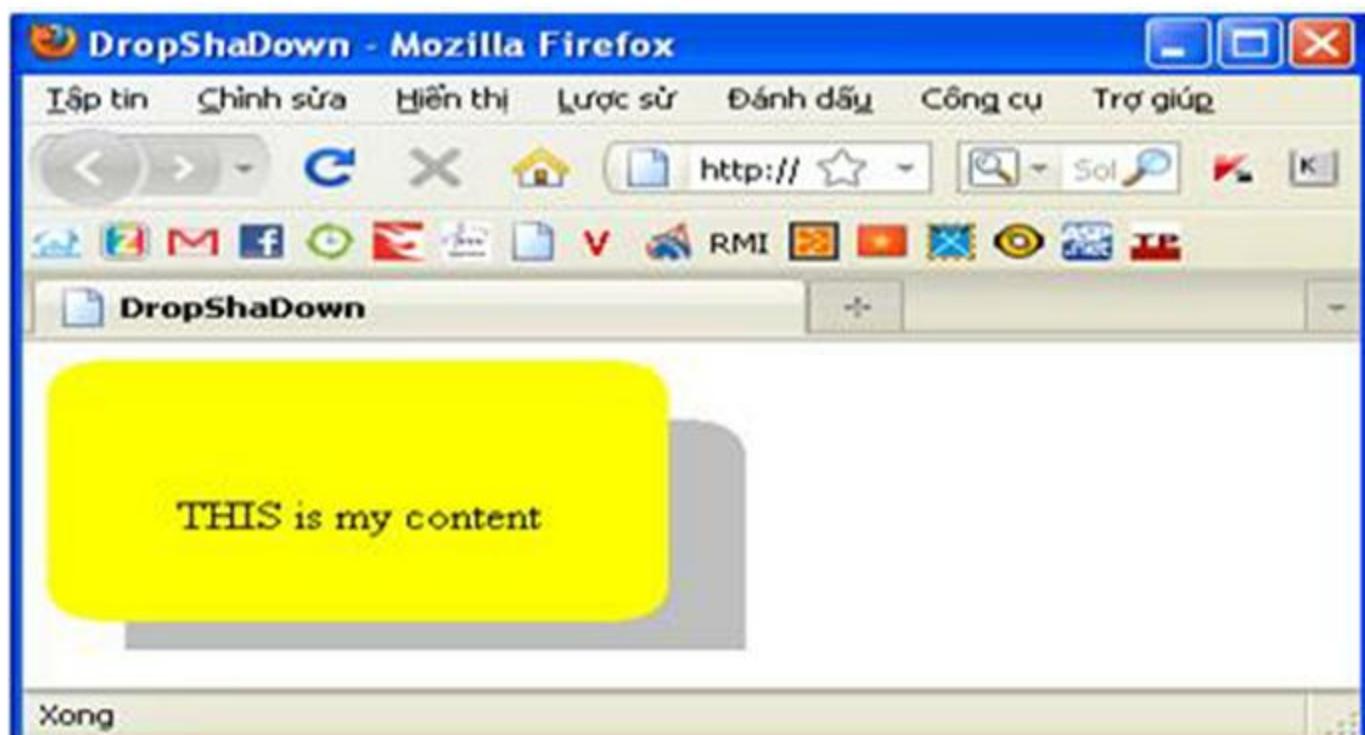
9.1. Bước 1: Tạo một Panel

```
<asp:Panel ID="Panel1" runat="server" BackColor="Yellow" ForeColor="Black" Width="200px">  
    <br /><br />  
    <center>  
        THIS is my content  
        <br />  
        <br />  
    </center>  
</asp:Panel>
```

9.2. Bước 2: Đưa vào một DropShadowExtender cho Panel trên

```
<asp:DropShadowExtender ID="Panel1_DropShadowExtender" runat="server"  
    Enabled="True" TargetControlID="Panel1" Opacity="0.25" Rounded="true"  
    Radius="11" TrackPosition="true"  
    Width="25">  
</asp:DropShadowExtender>
```

Kết quả:



10. DynamicPopulateExtender

DynamicPopulate là một extender đơn giản thay thế nội dung của một control với một kết quả của web service hay phương thức được gọi. Phương thức được gọi trả về một chuỗi HTML để đặt vào phần tử đích.

10.1. Bước 1: Tạo một Panel

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

10.2. Bước 2: Đưa vào một DynamicPopulateExtender cho Panel trên

```
<asp:DynamicPopulateExtender ID="Label1_DynamicPopulateExtender"
runat="server"
    Enabled="True" PopulateTriggerControlID="" TargetControlID="Label1"
    BehaviorID="dp1"
    ClearContentsDuringUpdate="true"
    ServiceMethod="GetHtml" >
</asp:DynamicPopulateExtender>
```

10.3. Bước 3: Tạo 2 radio button để lựa chọn

```
<input type="radio" name="rbFormat" id="r0" value='radio 1'
    onclick="updateDateKey(this.value);"
    checked="checked" />click to set 'radio 1'<br/>
<input type="radio"
    name="rbFormat"
    id="Radio1"
```

```
onclick="updateDateKey(this.value);"  
value='radio 2' />click to set 'radio 2'
```

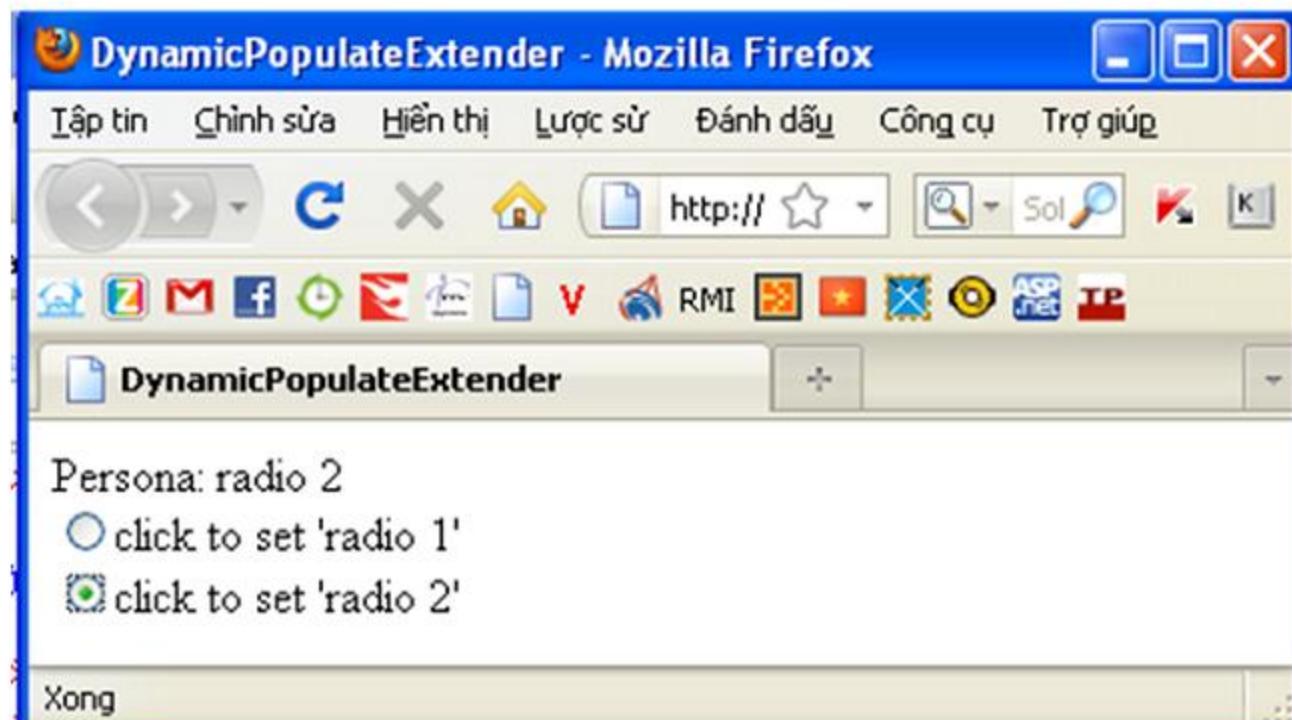
10.4. Bước 4: Script

```
<script type="text/javascript">  
  
function updateDateKey(value) {  
    var behavior = $find('dp1');  
    if (behavior) {  
        behavior.populate(value);  
    }  
}  
  
Sys.Application.add_load(function () { updateDateKey('radio 1 radio 2'); });  
</script>
```

10.5. Bước 5: CodeBehind

```
[System.Web.Services.WebMethod]  
[System.Web.Script.Services.ScriptMethod]  
public static string GetHtml(string contextKey)  
{  
    System.Threading.Thread.Sleep(350);  
    return String.Format("Persona: {0}", contextKey);  
}
```

Kết quả:



11. HoverMenuExtender

11.1. Bước 1: Tạo một GridView và đưa HoverMenuExtender vào trong Column

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
    CellPadding="4" ForeColor="#333333" GridLines="None"
    BorderColor="#628BD7" BorderStyle="Solid" BorderWidth="2px">

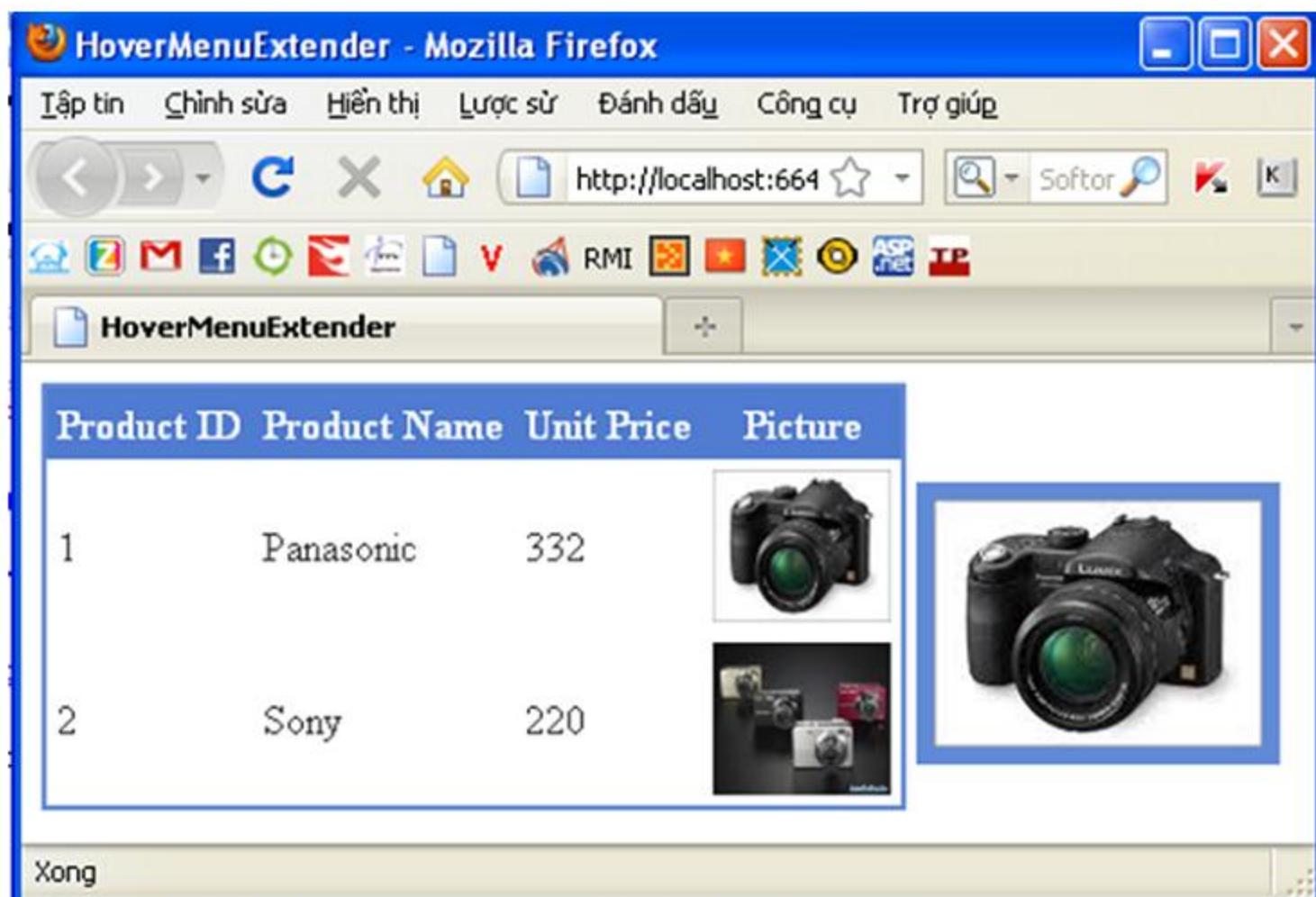
    <Columns>
        <asp:BoundField DataField="ProductID" HeaderText="Product ID" />
        <asp:BoundField DataField="ProductName" HeaderText="Product Name" />
        <asp:BoundField DataField="UnitPrice" HeaderText="Unit Price" />
        <asp:TemplateField HeaderText="Picture">
            <ItemTemplate>
                <asp:Image runat="server" ID="thumbnailImage"
                    ImageUrl='<%# Eval("Picture") %>' Width="70" Height="60" />
                <asp:HoverMenuExtender ID="HoverMenuExtender1" runat="server"
                    PopupControlID="popupImage"
                    TargetControlID="thumbnailImage"
                    OffsetX="10" OffsetY="5"
                    PopupPosition="Right"
                    PopDelay="100" HoverDelay="100">
                </asp:HoverMenuExtender>
                <asp:Panel runat="server" ID="popupImage" BorderColor="#628BD7"
                    BorderStyle="Solid" BorderWidth="7px">
                    <asp:Image runat="server" ID="mainImage"
                        ImageUrl='<%# Eval("Picture") %>' />
                </asp:Panel>
            </ItemTemplate>
        </asp:TemplateField>
    </Columns>
    <HeaderStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />
</asp:GridView>
```

11.2. Bước 2: CodeBehind

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack) {
        LoadData();
    }
}

private void LoadData()
{
    DataClasses1DataContext db = new DataClasses1DataContext();
    var table = from t in db.Products select t;
    GridView1.DataSource = table;
    GridView1.DataBind();
}
```

Kết quả:



Chương 12

LẬP TRÌNH LINQ

Kết thúc chương này các bạn có thể:

- *Trình bày được kiến trúc của LINQ*
- *Trình bày được các khái niệm cơ bản trong LINQ*
- *Mô tả và thực thi được LINQ to SQL*
- *Mô tả và thực thi được LINQ to Object*
- *Mô tả và thực thi LINQ to DataSet*

12.1. GIỚI THIỆU LINQ

Công nghệ LINQ (Language Integrated Query) là giải pháp lập trình hợp nhất đem đến khả năng truy vấn dữ liệu theo cú pháp SQL trực tiếp trong C# hay VB.NET, áp dụng cho tất cả các dạng dữ liệu từ đối tượng đến CSDL quan hệ và XML.

Xử lý thông tin hay dữ liệu là nhiệm vụ quan trọng nhất của bất kỳ phần mềm nào và một trong những trở ngại chính mà các nhà phát triển hiện nay phải đối mặt là khác biệt giữa ngôn ngữ lập trình hướng đối tượng và ngôn ngữ truy vấn dữ liệu, vấn đề càng phức tạp hơn với sự xuất hiện của XML (eXtensible Markup Language - ngôn ngữ đánh dấu mở rộng).

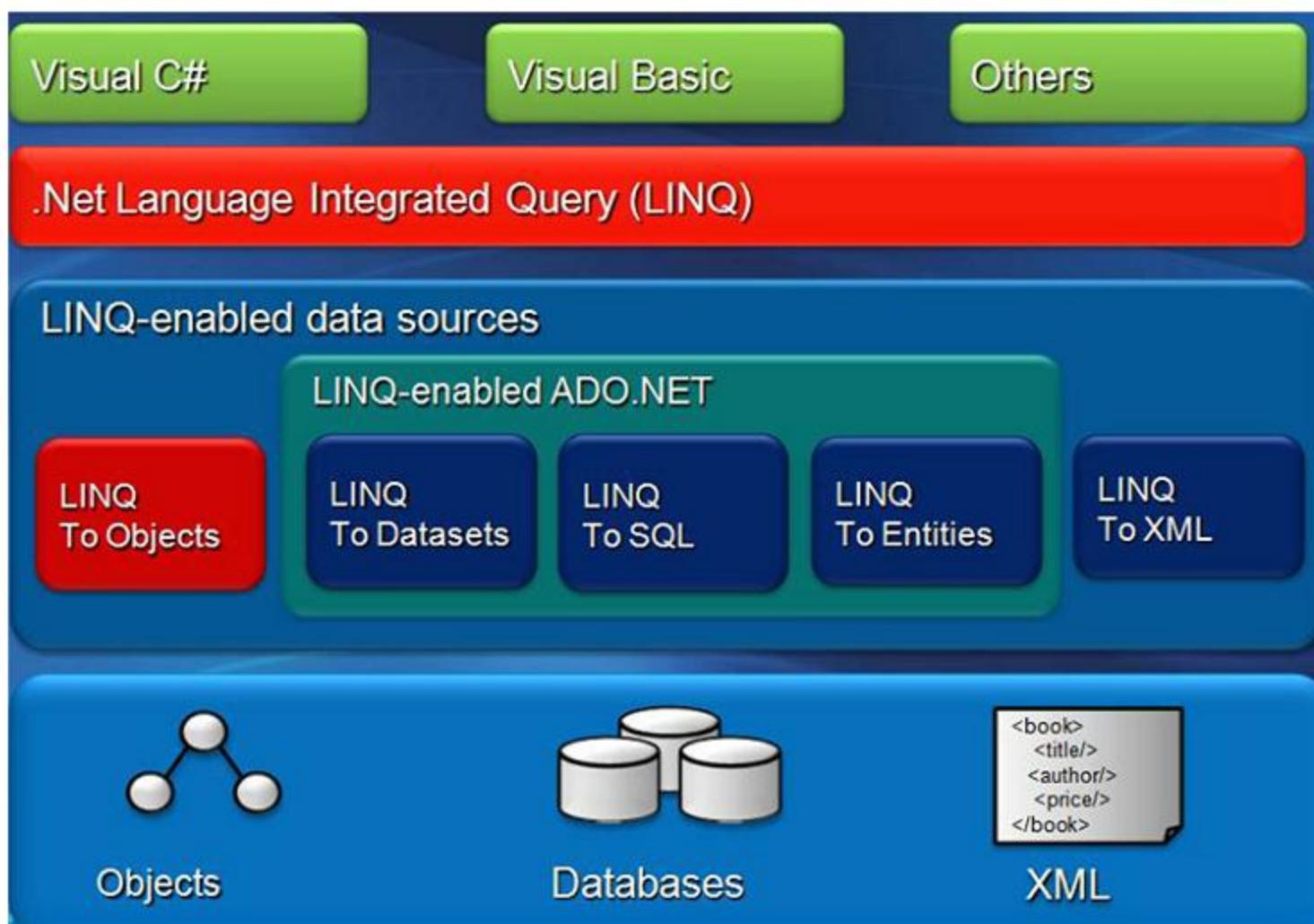
Hiện tại, cách phổ biến nhất để ứng dụng lấy dữ liệu từ các hệ cơ sở dữ liệu (CSDL) là sử dụng SQL (Structure Query Language - ngôn ngữ truy vấn cấu trúc). SQL có cú pháp rất khác với những ngôn ngữ lập trình phổ biến như C# và VB.NET, do vậy, người lập trình mất nhiều thời gian để sử dụng hai thực thể khác biệt này với nhau trong mỗi dự án phần mềm.

Một vấn đề khác với SQL là nó chỉ dùng để truy vấn dữ liệu trong các CSDL dạng quan hệ. Nếu muốn truy cập dữ liệu XML hay dạng khác (như trang HTML, email...), nhà phát triển lại phải sử dụng cú pháp truy vấn khác (XPath/XQuery).

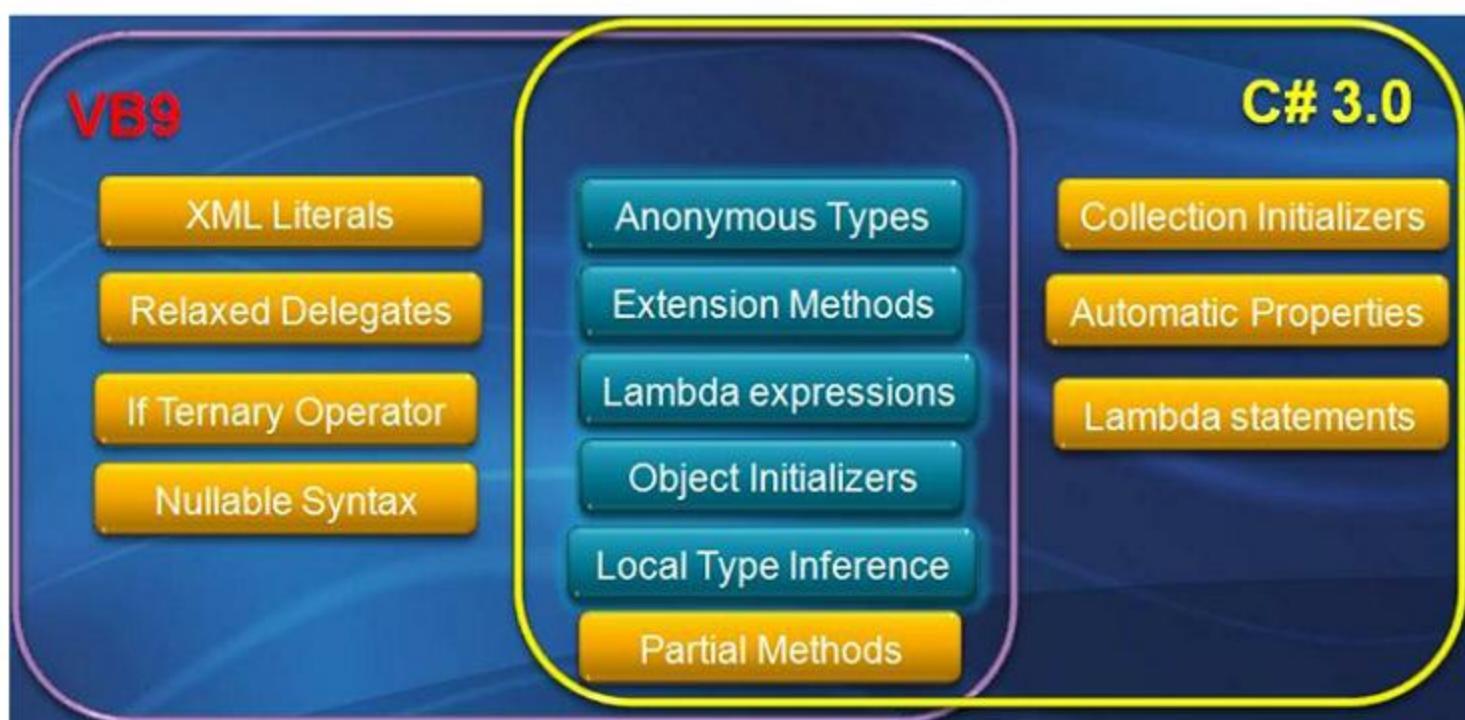
Để giảm gánh nặng thao tác trên nhiều ngôn ngữ khác nhau và cải thiện năng suất lập trình, Microsoft đã phát triển giải pháp tích hợp dữ liệu LINQ cho .NET Framework, đây là thư viện mở rộng cho các ngôn ngữ lập trình C# và Visual Basic.NET (có thể mở rộng cho các ngôn ngữ

khác) cung cấp khả năng truy vấn trực tiếp dữ liệu đối tượng, CSDL và XML. Để truy vấn dữ liệu đối tượng trong bộ nhớ, dữ liệu cần phải đổ vào bộ nhớ để xử lý, nhưng một khi tách khỏi nơi gốc của nó thì khả năng truy vấn rất kém. Bạn có thể dễ dàng truy vấn thông tin khách hàng móc nối với thông tin đơn hàng của họ từ CSDL SQL Server nhưng không dễ gì thực hiện tương tự với thông tin trong bộ nhớ. Trong môi trường .NET, thông tin (trong bộ nhớ) thường được thể hiện ở dạng các đối tượng và trước LINQ, không có cách nào để móc nối các đối tượng hay thực hiện bất kỳ thao tác truy vấn nào. LINQ chính là giải pháp cho vấn đề này.

- ❖ Với Microsoft.NET platform, ngôn ngữ hỗ trợ chính là C# và VB.NET. Những người lập trình viên họ thường gặp rắc rối, và cảm thấy khó chịu với việc truy cập dữ liệu ở những nguồn khác nhau. Đặc biệt là hai loại dữ liệu XML và CSDL. Với CSDL là mạnh nhất về dữ liệu lưu trữ.
- ❖ Các vấn đề về truy xuất dữ liệu như sau:
 - Chúng ta không lập trình tương tác với CSDL tại cấp độ native language. Vì thế lỗi thường khó phát hiện rõ. Khó khăn trong việc quản lý lỗi xảy ra.
 - Kiểu dữ liệu khác nhau trong mỗi nguồn dữ liệu ở XML và CSDL. Đặc biệt date và time.
- ❖ Các nhà phát triển ở Microsoft đưa ra LINQ là một nền tảng mới trong việc truy vấn dữ liệu ở bất kể các nguồn khác nhau (Object, XML, CSDL). Đây là công nghệ hỗ trợ cơ chế truy vấn dữ liệu ở tất cả các kiểu. Những kiểu này bao gồm mảng (List, Vector), XML, CSDL...
- ❖ Điều quan trọng nhất, LINQ là tất cả về truy vấn, kết quả sau khi truy vấn có thể là tập hợp các đối tượng cùng loại, có thể là một đối tượng đơn, có thể là tập hợp con của các trường (field) từ một đối tượng. Kết quả trả về của LINQ người ta gọi là sequence. Hầu hết sequence là `IEnumerable<T>` với T là kiểu dữ liệu của những đối tượng trong sequence.
- ❖ LINQ nó sẽ cung cấp cách duy nhất để truy cập dữ liệu từ bất kể nguồn dữ liệu nào với cú pháp giống nhau.



Hình 12.1: Kiến trúc LINQ



Hình 12.2: Một số tính năng mới trong C# 3.0 sử dụng cho LINQ



Hình 12.3: Minh họa ngôn ngữ LINQ

12.2 CÁC KHÁI NIỆM CƠ BẢN

- **Extension Methods:** là một phương thức tĩnh của một lớp tĩnh mà chúng ta có thể từ một thê hiện của phương thức trong một lớp khác.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ExtensionMethods
{
    //Khai báo lớp tĩnh
    static class MyExtensions {
        //Khai báo phương thức tĩnh dùng để đảo ngược 1 chuỗi
        public static string Reverse(this string s) {
            char[] c = s.ToCharArray();
            Array.Reverse(c);
            return new string(c);
        }
    }
}
  
```

```

class Program
{
    static void Main(string[] args)
    {
        string name = "ABCDEF";
        string reversed = name.Reverse();
        Console.WriteLine("Kết quả : {0}-->{1}", name, reversed);
        Console.ReadLine();
    }
}

```

❖ Kết quả thi hành

```
C:\Windows\system32\cmd.exe
Ket qua : ABCDEF-->
```

➤ Automatic Properties

Thông thường chúng ta phải khai báo một lớp và khai báo các thành phần dữ liệu sau đó khai báo properties để truy cập các thành phần này.

```

class KhachHang
{
    private string mMakhachhang;
    public KhachHang()
    {
        mMakhachhang = "KH0001";
    }

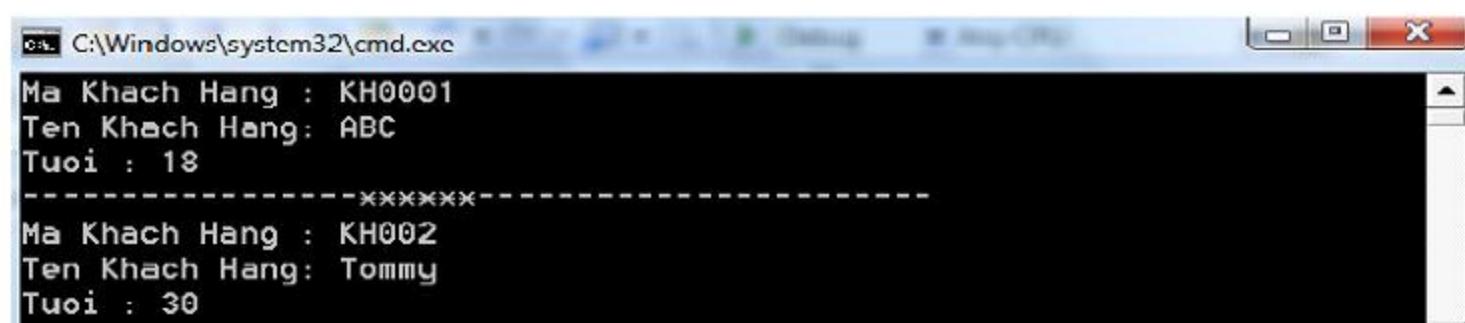
    public string MaKhachHang
    {
        get
        {
            return mMakhachhang;
        }
        set
        {
            mMakhachhang = value;
        }
    }
}
```

Automatic Properties phát sinh tự động phát sinh các thành phần dữ liệu khi chúng ta khai báo properties.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ComponentModel;
using System.Reflection;
namespace MinhHoa_Automatic_Properties
{
    class KhachHang
    {
        //Khai tạo các thành phần dữ liệu thông qua Automatic properties
        public KhachHang()
        {
            MaKhachHang = "KH0001";
            TenKhachHang = "ABC";
            Tuoi = 18;
        }
        //Khai báo các Automatic properties
        public string MaKhachHang { get; set; }
        public string TenKhachHang { get; set; }
        public int Tuoi { get; set; }
    }

    class Program
    {
        static void Main(string[] args)
        {
            //Tạo đối tượng Khách Hàng
            KhachHang c1 = new KhachHang();
            //Xuất các giá trị mặc định
            Console.WriteLine("Ma Khach Hang : {0}", c1.MaKhachHang);
            Console.WriteLine("Ten Khach Hang: {0}", c1.TenKhachHang);
            Console.WriteLine("Tuoi : {0}", c1.Tuoi);
            Console.WriteLine("-----*****-----");
            //Thay đổi các giá trị mới
            c1.MaKhachHang = "KH002";
            c1.TenKhachHang = "Tommy";
            c1.Tuoi = 30;
            //Xuất các giá trị sau khi thay đổi
            Console.WriteLine("Ma Khach Hang : {0}", c1.MaKhachHang);
            Console.WriteLine("Ten Khach Hang: {0}", c1.TenKhachHang);
            Console.WriteLine("Tuoi : {0}", c1.Tuoi);
            Console.ReadLine();
        }
    }
}
```

❖ Kết quả thi hành



```
C:\Windows\system32\cmd.exe
Ma Khach Hang : KH0001
Ten Khach Hang: ABC
Tuoi : 18
-----
Ma Khach Hang : KH002
Ten Khach Hang: Tommy
Tuoi : 30
```

➤ Object Initialization

Object initialization cho phép chúng ta khởi tạo các giá trị cho các thành phần public và các properties của lớp trong suốt quá trình khởi động.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Object_initializers
{
    class KhachHang {
        public string TenKhachHang { get; set; }
        public int Tuoi { get; set; }
    }
    class Program
    {
        static void Main(string[] args)
        {
            KhachHang c = new KhachHang() { TenKhachHang = "Tom", Tuoi = 24 };
            Console.WriteLine("Ten Khach Hang : {0}", c.TenKhachHang);
            Console.WriteLine("Tuoi : {0}", c.Tuoi);
            Console.ReadLine();
        }
    }
}
```

❖ Kết quả thi hành



```
C:\Windows\system32\cmd.exe
Ten Khach Hang : Tom
Tuoi : 24
```

➤ Anonymous Types

Anonymous Types là kiểu không tên được phát sinh khi biên dịch dựa vào quá trình khởi động của đối tượng đang được khởi động và không thể trả về từ một phương thức, chỉ được sử dụng như là biến tham chiếu cục bộ.

```
class Program
{
    static void Main(string[] args)
    {
        //Khai bao kiểu AnonymousType KhachHang sử dụng từ khóa var
        var KhachHang = new {TenKhachHang = "Tom", Tuoi = 24 };
        Console.WriteLine("Ten Khach Hang : {0}", KhachHang.TenKhachHang);
        Console.WriteLine("Tuoi : {0}", KhachHang.Tuoi);
        Console.ReadLine();
    }
}
```

Kết quả thi hành



```
Ten Khach Hang : Tom
Tuoi : 24
```

➤ Lambda Expressions

Biểu thức lambda được chỉ định như là các tham số được phân cách bởi dấu “;” theo sau là toán tử lambda “=>” hoặc một biểu thức hoặc một khối lệnh.

Cú pháp:

```
(param1, param2, ...paramN) => expr  
hay:  
(param1, param2, ...paramN) =>  
{  
    statement1;  
    statement2;  
    ...  
    statementN;  
    return(lambda_expression_return_type);  
}
```

```
class Program  
{  
    //Khai báo delegate gọi đến phương thức cộng  
    public delegate int GoiPhuongThuc(int a, int b);  
    public static int Calc(GoiPhuongThuc f, int a, int b)  
    {  
        return f(a, b);  
    }  
    public static int Cong(int a, int b)  
    {  
        return a + b;  
    }  
    static void Main(string[] args)  
    {  
        |  
        Console.WriteLine("Minh họa Lambda Expressions");  
        int x = 1, y = 2;  
        //Khai báo biểu thức lambda  
        int ketqua = Calc((a, b) => a + b, x, y);  
        Console.WriteLine("Kết quả : {0} + {1} = {2}", x, y, ketqua);  
    }  
}
```

❖ Kết quả thi hành



```
on C:\Windows\system32\cmd.exe  
Minh họa Lambda Expressions  
Kết quả : 1 + 2 = 3
```

➤ Query Expressions

Biểu thức Query cho phép chúng ta viết lệnh truy vấn trong LINQ gần giống với các lệnh truy vấn trong T-SQL

Cú pháp

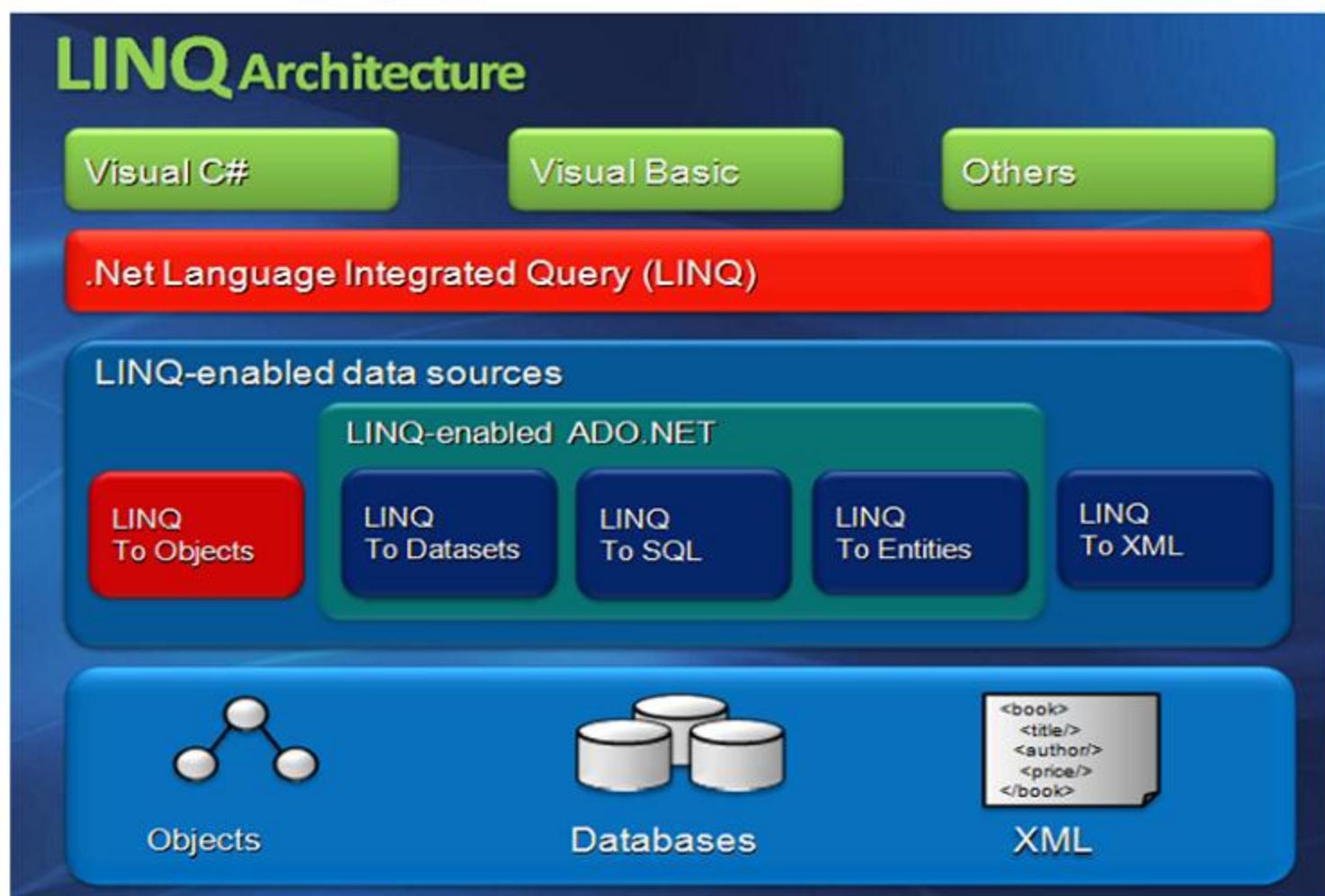
```
from itemName in srcExpr  
join itemName in srcExpr on keyExpr equals keyExpr  
    (into itemName)?  
let itemName = selExpr  
where predExpr  
orderby (keyExpr (ascending | descending)?)*  
select selExpr  
group selExpr by keyExpr  
into itemName query-body
```

- ❖ Kết quả thi hành



12.3. LINQ TO OBJECTS

LINQ to Object dùng để truy vấn, sắp xếp, lọc dữ liệu trong các đối tượng như: mảng, tập hợp.



Hình 12.4: Minh họa ngôn ngữ LINQ to Objects

Thí dụ. Minh họa sử dụng LINQ to Object để chuyển một mảng các chuỗi số sang một mảng các số nguyên.

```
class Program
{
    static void Main(string[] args)
    {
        //Khai báo mảng các chuỗi số
        string[] MangChuoiSo = { "0042", "010", "9", "27" };
        //Chuyển sang mảng các số nguyên
        int[] MangSoNguyen = MangChuoiSo
            .Select(s => Int32.Parse(s))
            .OrderBy(s => s).ToArray();
        //Xem danh sách mảng sau khi chuyển
        foreach (int SoNguyen in MangSoNguyen)
            Console.WriteLine("{0}", SoNguyen);

        Console.ReadLine();
    }
}
```

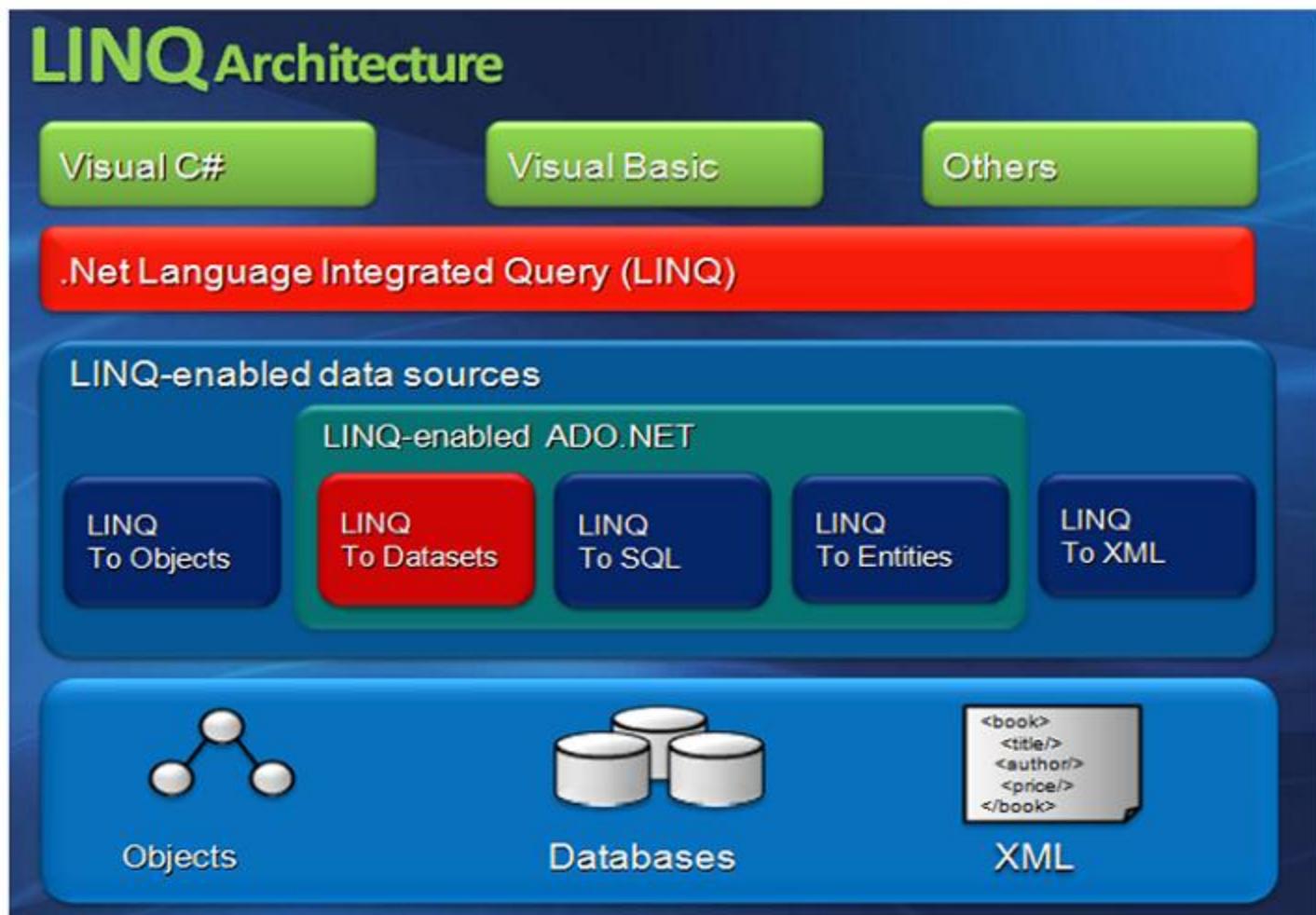
❖ Kết quả thi hành



12.4. LINQ TO DATASET

LINQ to DataSet dùng để thực hiện truy vấn, sắp xếp, lọc dữ liệu,... nhanh và dễ dàng hơn trong đối tượng DataSet và cho phép các bạn viết các câu lệnh truy vấn trong ngôn ngữ lập trình thay vì phải viết bằng T-SQL.

LINQ to DataSet cũng được sử dụng để truy vấn dữ liệu từ một hoặc nhiều nguồn dữ liệu với nhau.



Hình 12.5: Minh họa ngôn ngữ LINQ to DataSets

Thí dụ. Minh họa lọc dữ liệu với LINQ to DataSet.

Tạo một trang ASP.NET hiển thị danh sách loại mặt hàng và cho phép người dùng lọc danh sách này theo tên loại mặt hàng.

Bước 1: Thiết kế giao diện

MìnhHoaLINQtoDataSet.aspx

Column0	Column1	Column2
abc	abc	abc

Lọc theo tên loại mặt hàng :

Bảng mô tả danh sách các control

Loại Control	Thuộc tính	Giá trị thiết lập
GridView	ID	gvDanhSachLoaiMatHang
TextBox	ID	txtTenLoaiMatHang
Button	ID	btnHienThiDanhSach

Bước 2: Viết lệnh xử lý sự kiện.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SqlClient;
using System.Data;

namespace MinhHoa
{
    public partial class MinhHoaLINQtcDataSet : System.Web.UI.Page
    {
        string strConnectionString = "server = .\\sql2k8; "+
            " database=minhhoa;integrated security=true";
        public DataTable LayDanhSachLoaiMatHang()
        {
            string strSQL = " SELECT * from LoaiMatHang";
            SqlConnection objConnection = new SqlConnection(strConnectionString);
            SqlCommand objCommand = new SqlCommand(strSQL, objConnection);
            SqlDataAdapter objAdapter = new SqlDataAdapter(objCommand);
            DataTable dtLoaiMatHang = new DataTable();
            objAdapter.Fill(dtLoaiMatHang);
            return dtLoaiMatHang;
        }

        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                DataTable dtDanhSach = LayDanhSachLoaiMatHang();
                gvDanhSachLoaiMatHang.DataSource = dtDanhSach;
                gvDanhSachLoaiMatHang.DataBind();
            }
        }

        protected void btnHienThiDanhSach_Click(object sender, EventArgs e)
        {
            DataTable dtDanhSach = LayDanhSachLoaiMatHang();
            var danhSachChonLoc = from LoaiMatHang in dtDanhSach.AsEnumerable()
                where LoaiMatHang.Field<string>("TenLoaiMatHang").
                    Contains(txtTenLoaiMatHang.Text)
                select new
                {
                    MaLoaiMatHang = LoaiMatHang.Field<int>("MaLoaiMatHang"),
                    TenLoaiMatHang = LoaiMatHang.Field<string>("TenLoaiMatHang"),
                    XuatXu = LoaiMatHang.Field<string>("XuatXu")
                };
            gvDanhSachLoaiMatHang.DataSource = danhSachChonLoc;
            gvDanhSachLoaiMatHang.DataBind();
        }
    }
}
```

Bước 3. Nhấn Ctrl+F5 thi hành ứng dụng. Kết quả như sau

MaLoaiMatHang	TenLoaiMatHang	XuatXu
1	Nước giải khát	Việt Nam
2	Đồ gia dụng	Thái Lan
3	Hàng Nội Thất	Việt Nam
4	Loại A	Nhật Bản
5	Loại B	Hàn Quốc
6	Loại C	Đài Loan
7	Loại D	Trung Quốc
8	Loại E	Trung Quốc
9	Loại F	Trung Quốc
10	Loại A1	Việt Nam
11	Loại B1	Việt Nam
12	Loại B3	Thái Lan

Lọc theo tên loại mặt hàng :

Hiển thị danh sách

Nhập vào tên loại mặt hàng “n”, nhấn nút **Hiển thị danh sách** để xem danh sách các loại hàng mà tên có chứa chuỗi “n”, kết quả lọc dữ liệu như sau:

MaLoaiMatHang	TenLoaiMatHang	XuatXu
2	Đồ gia dụng	Thái Lan
3	Hàng Nội Thất	Việt Nam

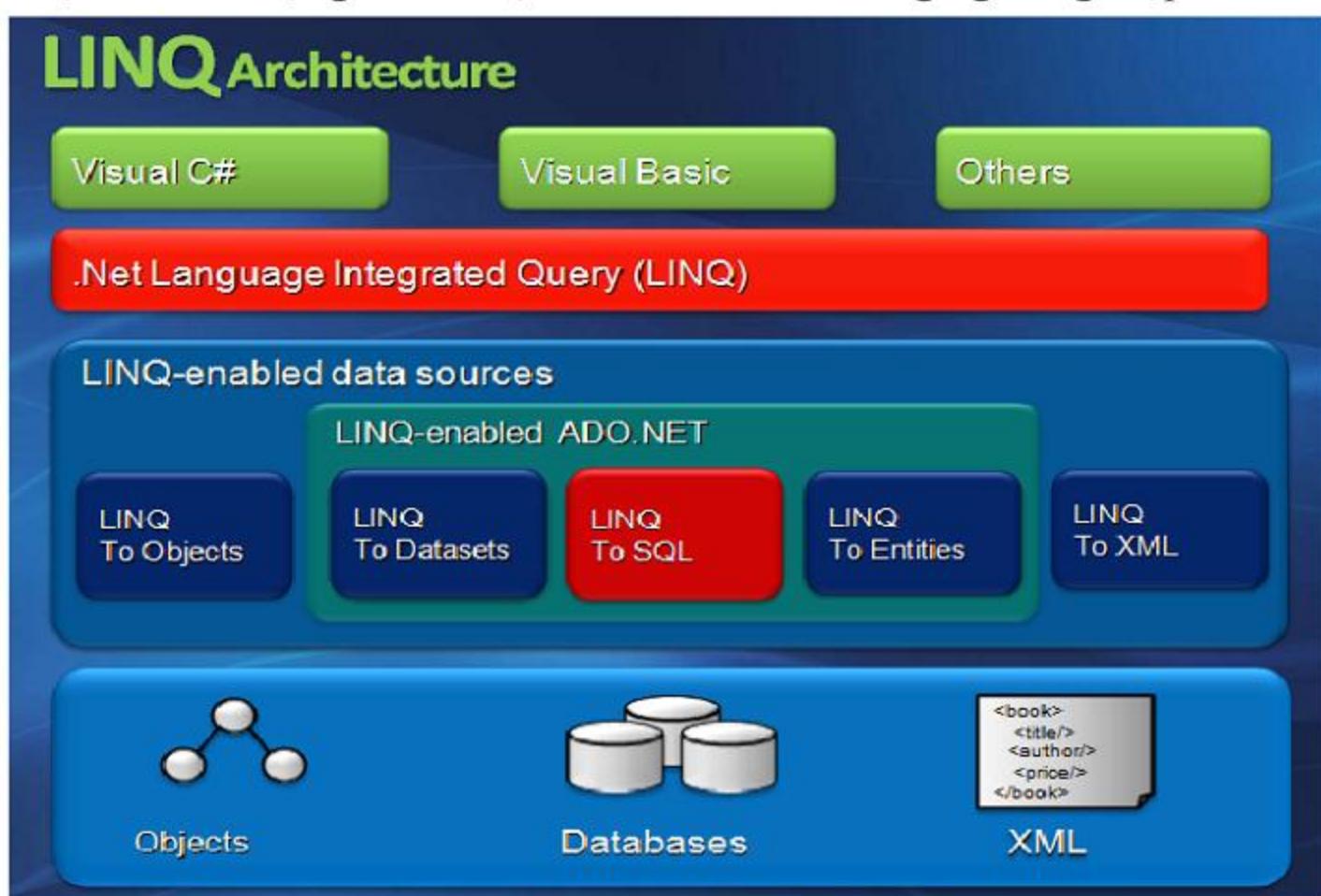
Lọc theo tên loại mặt hàng :

Hiển thị danh sách

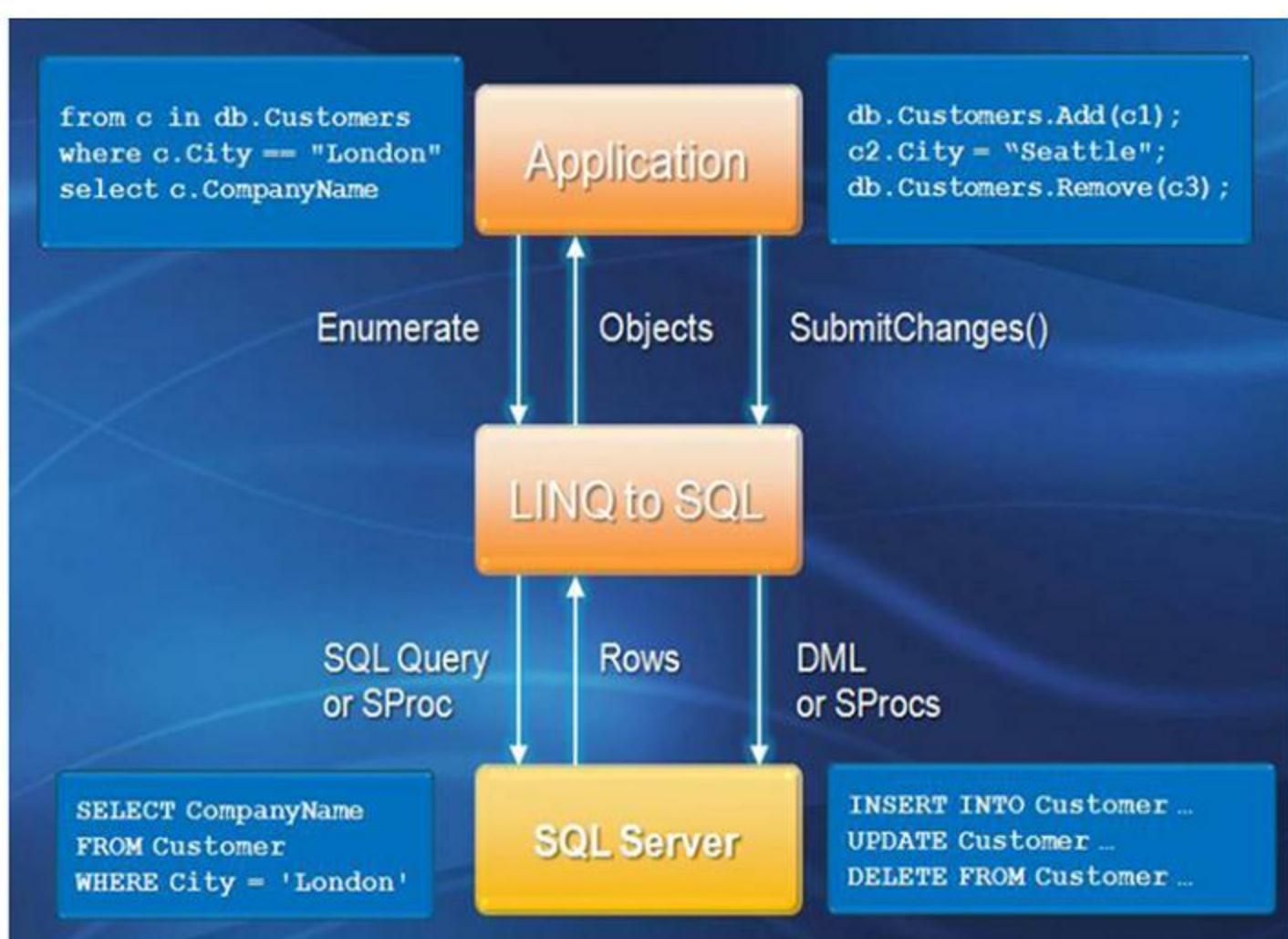
12.5. LINQ TO SQL

Trong LINQ to SQL, một mô hình cơ sở dữ liệu quan hệ được ánh xạ thành một mô hình đối tượng đã được thiết kế đặc biệt trong ngôn ngữ lập trình.

Khi ứng dụng thi hành LINQ to SQL biên dịch thành ngôn ngữ truy vấn tích hợp SQL trong mô hình đối tượng và gửi đến cơ sở dữ liệu để thực thi, đến khi cơ sở dữ liệu trả về kết quả, LINQ to SQL chuyển kết quả về lại các đối tượng để các bạn có thể thao tác trong ngôn ngữ lập trình.



Hình 12.6: Minh họa ngôn ngữ LINQ



Hình 12.7: Mô hình hoạt động của LINQ to SQL

Thí dụ. Tạo một trang ASP.NET dùng để xem, thêm, xóa, sửa danh sách các loại mặt hàng sử dụng LINQ to SQL.

Bước 1. Thiết kế giao diện

MinhHoaLINQtoSQL.aspx*

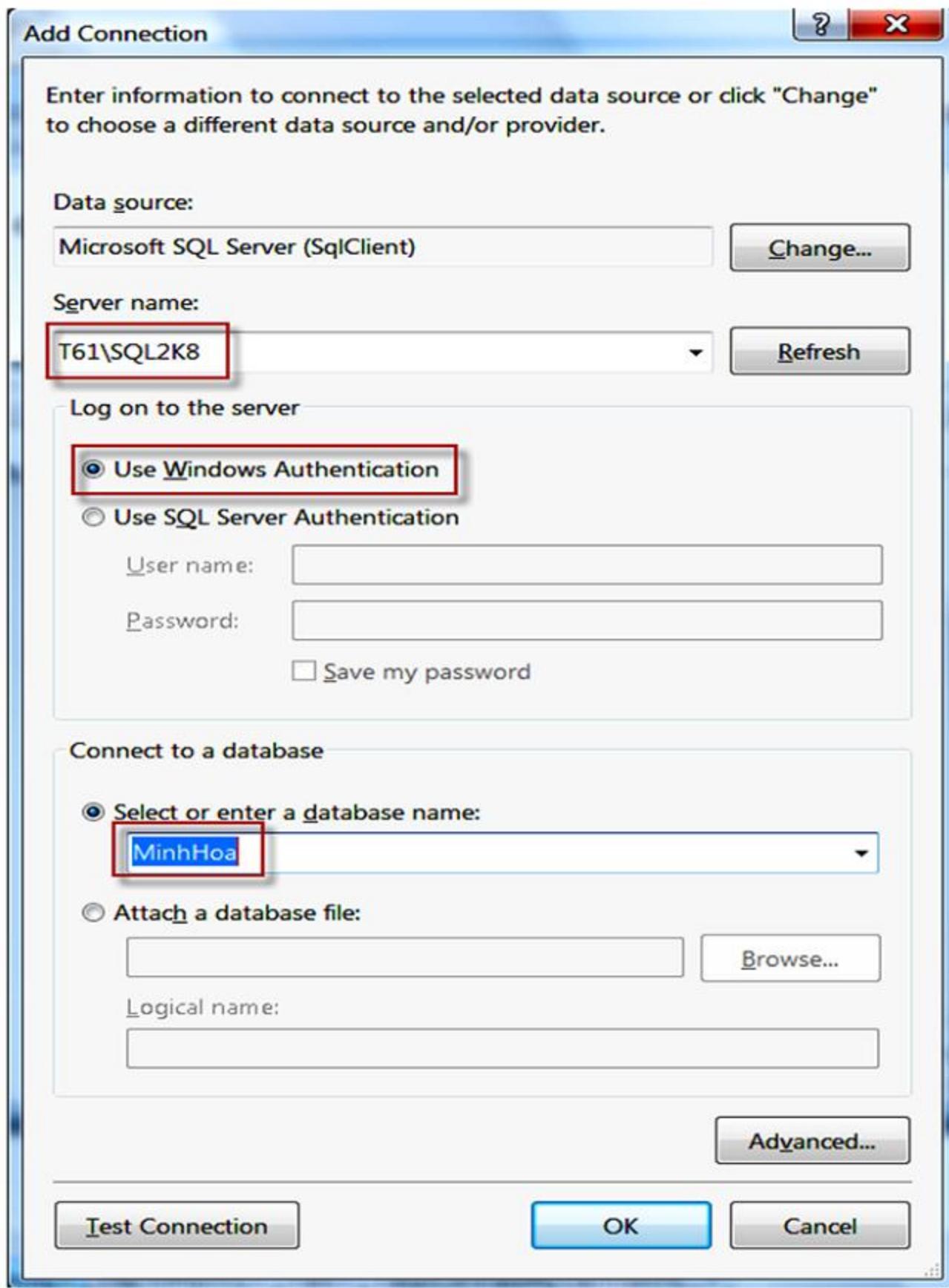
Mã loại mặt hàng :	<input type="text"/>																																			
Tên loại mặt hàng :	<input type="text"/>																																			
Xuất xứ	<input type="button" value="Thêm vào danh sách"/>																																			
<table border="1"><thead><tr><th>Mã loại mặt hàng</th><th>Tên loại mặt hàng</th><th>Xuất xứ</th><th></th><th></th></tr></thead><tbody><tr><td>Databound</td><td>Databound</td><td>Databound</td><td>Cập nhật</td><td>Xóa</td></tr><tr><td>Databound</td><td>Databound</td><td>Databound</td><td>Cập nhật</td><td>Xóa</td></tr></tbody></table>		Mã loại mặt hàng	Tên loại mặt hàng	Xuất xứ			Databound	Databound	Databound	Cập nhật	Xóa	Databound	Databound	Databound	Cập nhật	Xóa	Databound	Databound	Databound	Cập nhật	Xóa	Databound	Databound	Databound	Cập nhật	Xóa	Databound	Databound	Databound	Cập nhật	Xóa	Databound	Databound	Databound	Cập nhật	Xóa
Mã loại mặt hàng	Tên loại mặt hàng	Xuất xứ																																		
Databound	Databound	Databound	Cập nhật	Xóa																																
Databound	Databound	Databound	Cập nhật	Xóa																																
Databound	Databound	Databound	Cập nhật	Xóa																																
Databound	Databound	Databound	Cập nhật	Xóa																																
Databound	Databound	Databound	Cập nhật	Xóa																																
Databound	Databound	Databound	Cập nhật	Xóa																																
1 2																																				

Bước 2. Tạo DataConnection để kết nối với cơ sở dữ liệu *MinhHoa*

Trên thanh công cụ *Server Explorer | Data Connection*, nhấn phải chuột chọn *Add New Connection*.

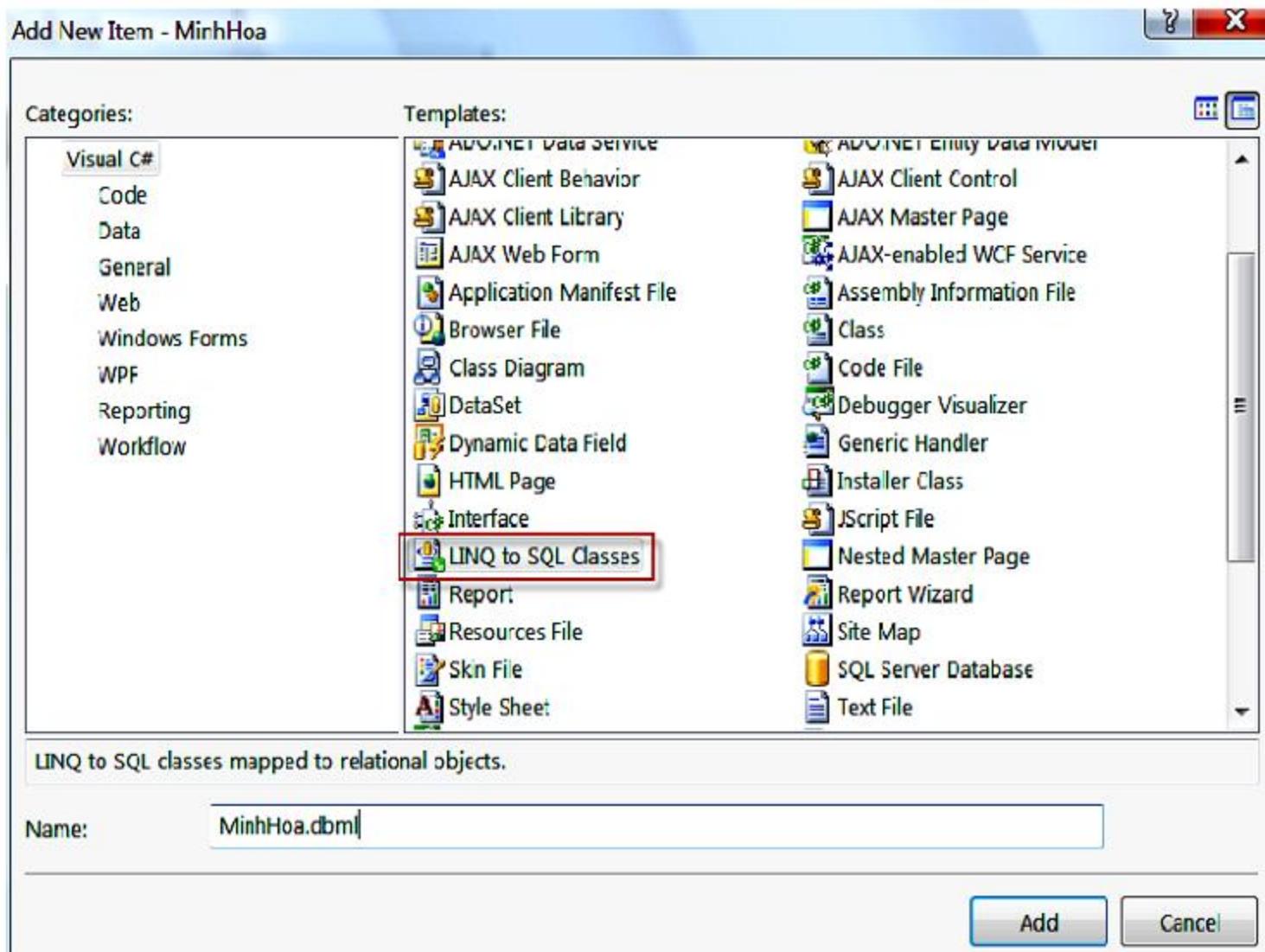


Thực hiện các bước như hình sau và nhấn nút OK để hoàn tất tạo *DataConnection*

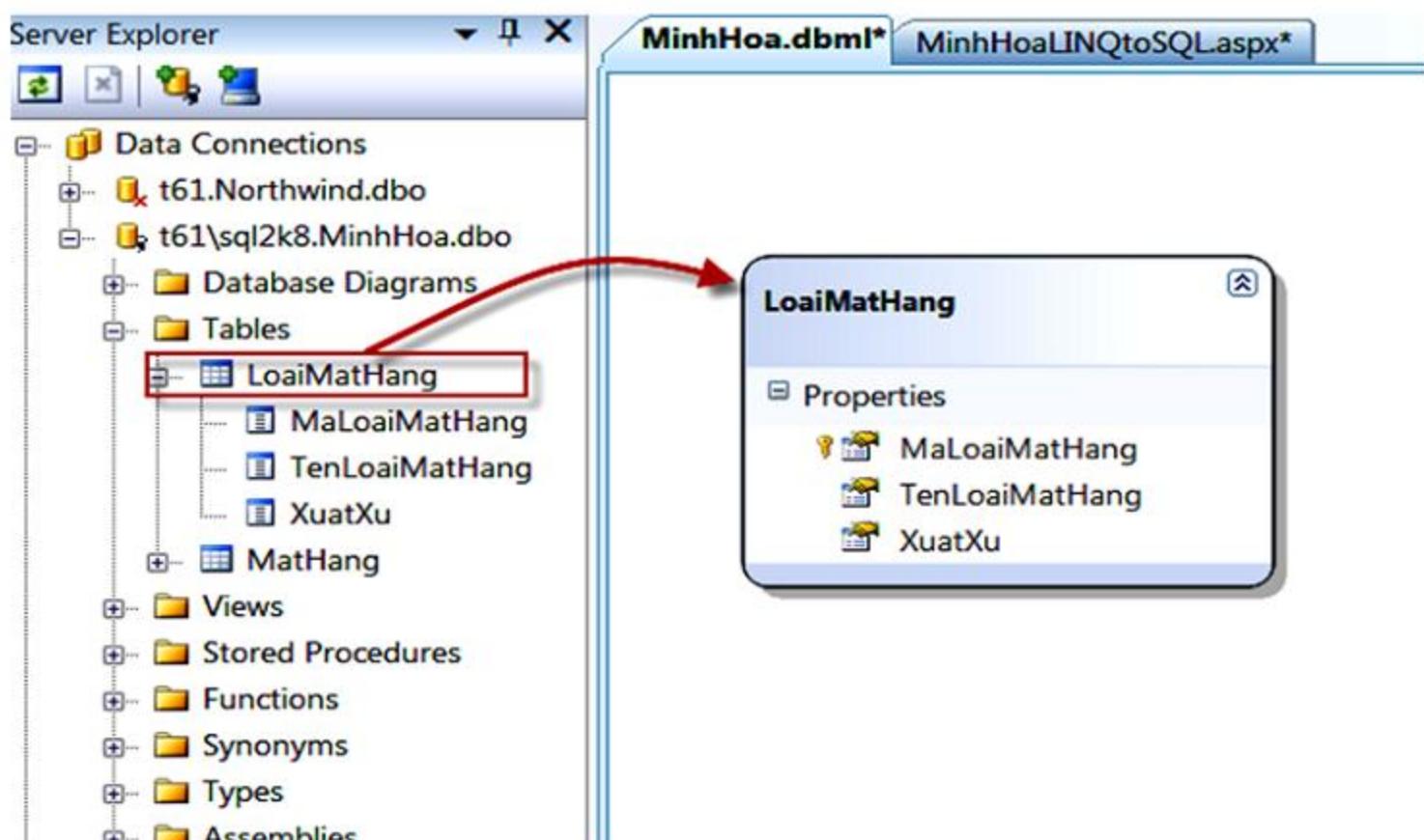


Bước 3. Tạo Data Context sử dụng “LINQ to SQL” để kết nối với bảng LoaiMatHang của cơ sở dữ liệu *MinhHoa*

Vào menu *Project | Add New Items*, chọn **LINQ to SQL Classes** như hình sau:



Trên *Server Explorer* chọn bảng LoaiMatHang kéo thả vào MinhHoa.dbml



Bước 4. Viết lệnh xử lý sự kiện.

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace MinhHoa
{
    public partial class MinhHoaLINQtoSQL: System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                LayDanhSachLoaiMatHang();
            }
        }

        public void LayDanhSachLoaiMatHang()
        {
            MinhHoaDataContext MinhHoaDC = new MinhHoaDataContext();
            gvLoaiMatHang.DataKeyNames = new string[] {
                "MaLoaiMatHang" };
            gvLoaiMatHang.DataSource = MinhHoaDC.LoaiMatHangs;
            gvLoaiMatHang.DataBind();
        }

        protected void btnThem_Click(object sender, EventArgs e)
        {
            MinhHoaDataContext MinhHoaDC = new MinhHoaDataContext();
            LoaiMatHang objLoaiMatHang = new LoaiMatHang {
                MaLoaiMatHang = int.Parse(txtMaLoaiMatHang.Text),
                TenLoaiMatHang = txtTenLoaiMatHang.Text,
                XuatXu = txtXuatXu.Text
            };
            MinhHoaDC.LoaiMatHangs.InsertOnSubmit(objLoaiMatHang);
            MinhHoaDC.SubmitChanges();
            LayDanhSachLoaiMatHang();
        }

        protected void gvLoaiMatHang_PageIndexChanging(object sender,
```

```

        GridViewPageEventArgs e)
    {
        gvLoaiMatHang.PageIndex = e.NewPageIndex;
        LayDanhSachLoaiMatHang();
    }
    protected void gvLoaiMatHang_RowUpdating(object sender,
        GridViewUpdateEventArgs e)
    {
        //Lay các giá trị thay đổi từ người dùng
        TextBox txtTenLoaiMatHang =
        gvLoaiMatHang.Rows[e.RowIndex].Cells[1].Controls[0] as
        TextBox;
        TextBox txtXuatXu =
        gvLoaiMatHang.Rows[e.RowIndex].Cells[2].Controls[0] as
        TextBox;
        int MaLoaiMatHang =
            (int)gvLoaiMatHang.DataKeys[e.RowIndex].Value;
        MinhHoaDataContext MinhHoaDC = new MinhHoaDataContext();
        LoaiMatHang lmh = MinhHoaDC.LoaiMatHangs.SingleOrDefault
            (loaihang => loaihang.MaLoaiMatHang == MaLoaiMatHang);
        lmh.MaLoaiMatHang = MaLoaiMatHang;
        lmh.TenLoaiMatHang = txtTenLoaiMatHang.Text;
        lmh.XuatXu = txtXuatXu.Text;
        MinhHoaDC.SubmitChanges();
        gvLoaiMatHang.EditIndex = -1;
        LayDanhSachLoaiMatHang();
    }
    protected void gvLoaiMatHang_RowDeleting(object sender,
        GridViewDeleteEventArgs e)
    {
        int MaLoaiMatHang =
        (int)gvLoaiMatHang.DataKeys[e.RowIndex].Value;
        MinhHoaDataContext MinhHoaDC = new MinhHoaDataContext();
        LoaiMatHang lmh = MinhHoaDC.LoaiMatHangs.SingleOrDefault
            (loaihang => loaihang.MaLoaiMatHang == MaLoaiMatHang);
        MinhHoaDC.LoaiMatHangs.DeleteOnSubmit(lmh);
    }

```

```

        MinhHoaDC.SubmitChanges();
        LayDanhSachLoaiMatHang();
    }

    protected void gvLoaiMatHang_RowEditing(object sender,
    GridViewEditEventArgs e)
    {
        gvLoaiMatHang.EditIndex = e.NewEditIndex;
        LayDanhSachLoaiMatHang();
    }

    protected void gvLoaiMatHang_RowCancelingEdit(object sender,
    GridViewCancelEditEventArgs e)
    {
        gvLoaiMatHang.EditIndex = -1;
        LayDanhSachLoaiMatHang();
    }
}

```

Bước 5. Thi hành ứng dụng và thực các chức năng tương ứng. Kết quả như hình sau.

Mã loại mặt hàng	Tên loại mặt hàng	Xuất xứ	Cập nhật	Xóa
7	Loại D	Trung Quốc	Cập nhật	Xóa
8	Loại E	Trung Quốc	Cập nhật	Xóa
9	Loại F	Trung Quốc	Cập nhật	Xóa
10	Loại A1	Việt Nam	Cập nhật	Xóa
11	Loại B1	Việt Nam	Cập nhật	Xóa
12	Loại B3	Thái Lan	Cập nhật	Xóa

Chương 13

LẬP TRÌNH WEBSERVICES

Kết thúc chương này các bạn có thể:

- *Trình bày được các khái niệm của WebService*
- *Mô tả được các thành phần trong WebService*
- *Xây dựng được ứng dụng WebService*

13.1. WEB SERVICES

Một dịch vụ web (web services) là một hệ thống phần mềm được xây dựng để hỗ trợ khả năng tương tác giữa các máy tính trên mạng (theo Wide Web Consortium). Nó cung cấp một giao tiếp được mô tả theo một định dạng chung thường gọi là ngôn ngữ mô tả dịch vụ web (Web Service Description Language – WSDL). Các hệ thống khác thực hiện tương tác với dịch vụ web thông qua giao thức SOAP (Simple Object Access Protocol). Đây là giao thức giúp trao đổi thông tin sử dụng HTTP kết hợp với việc sử dụng đặc tả XML cùng với một số chuẩn khác.

Như vậy, mục đích chính của việc phát triển của dịch vụ web là cho phép giao tiếp và trao đổi các chức năng, thông tin, dữ liệu giữa các ứng dụng một cách dễ dàng mà không cần quan tâm đến môi trường phát triển, ngôn ngữ lập trình bởi tất cả đã được quy về một định dạng chung. Bản chất của web service là một tập hợp các đối tượng, các phương thức được thực thi và công bố trên mạng để có thể được triệu gọi từ xa thông qua các ứng dụng khác.

Để xem xét kỹ hơn về web service, chúng ta sẽ tìm hiểu sự ra đời và vai trò của web service.

13.1.1. Sự ra đời của web services

Web service là sự kế thừa từ các công nghệ phân tán trước đó như CORBA, DCOM và RMI.

CORBA (Common Object Request Broker Architecture) là một giải pháp dựa trên các chuẩn mở do tổ chức OMG (Object Management Group) đưa ra. Điểm mạnh của CORBA là các ứng dụng trên máy client và trên máy chủ có thể được viết bằng các ngôn ngữ lập trình khác nhau nhờ sử dụng một ngôn ngữ định nghĩa giao tiếp (Interface Definition

Language – IDL). Tuy nhiên, việc xây dựng ứng dụng phân tán sử dụng CORBA rất phức tạp và có rất ít ngôn ngữ lập trình được CORBA hỗ trợ.

DCOM (Distributed Component Object Model) được đề xuất bởi Microsoft, giúp các thành phần phần mềm (software component) giao tiếp với nhau trên môi trường phân tán. Một DCOM server sẽ công bố các phương thức, các đối tượng cho các máy khách bằng cách hỗ trợ đa giao tiếp (multiple interfaces). Giao thức được sử dụng cho việc giao tiếp giữa các ứng dụng là Object Remote Procedure Call (ORPC). DCOM chủ yếu được phát triển trên các hệ điều hành Windows và chủ yếu được hỗ trợ phát triển bởi Microsoft.

RMI (Remote Method Invocation – RMI) cho phép xây dựng các ứng dụng phân tán dựa trên công nghệ Java. Một đối tượng viết bằng Java có thể gọi đến một đối tượng từ xa mà nó tham chiếu đến. RMI sử dụng giao thức JRMP (Java Remote Method Protocol). Để xây dựng một ứng dụng phân tán sử dụng RMI đòi hỏi người lập trình phải có kiến thức, kinh nghiệm về lập trình với ngôn ngữ Java và kỹ thuật phân tán. Mặc khác, hạn chế lớn nhất của RMI là chỉ hỗ trợ đối với các ứng dụng Java **Error! Reference source not found.**

Bên cạnh những hạn chế của các công nghệ nói trên, một vấn đề phát sinh ở đây là khả năng tương tác giữa các ứng dụng được xây dựng trên các công nghệ khác nhau. Chúng ta không thể giao tiếp với một server sử dụng công nghệ DCOM từ một ứng dụng sử dụng công nghệ RMI bởi chúng sử dụng hai giao thức khác nhau. Vì vậy, web service ra đời là một sự phát triển có vai trò quan trọng trong lĩnh vực phân tán trên internet.

13.1.2. Vai trò và đặc điểm của web services

Web service ra đời mở ra một hướng mới cho việc phát triển các ứng dụng trên internet. Ngày nay, dịch vụ web được sử dụng trong nhiều lĩnh vực phát triển phần mềm khác nhau như các dịch vụ cung cấp giá cổ phiếu, hỗ trợ thanh toán trực tuyến, chuyển đổi ngoại tệ, tìm kiếm thông tin liên quan đến sản phẩm, đặc biệt là các dịch vụ web của các website bán hàng trực tuyến như Amazon, eBay. Web service đóng vai trò quan trọng trong việc phát triển các ứng dụng thương mại điện tử (e-commerce). Dùng dịch vụ web để kết hợp các khả năng đơn lẻ chạy trên các hệ thống khác nhau thành một hệ thống tích hợp.

13.2. KIẾN TRÚC VÀ CÁC THÀNH PHẦN WEB SERVICES

Công nghệ web service không phải là một công nghệ mới hoàn toàn, mà nó ra đời dựa trên sự kết hợp các nền tảng công nghệ sẵn có trước đó.

Nó là sự tích hợp các ứng dụng dựa trên web sử dụng các chuẩn mở như XML, SOAP, WSDL, UDDI. Trong đó, XML được sử dụng để mô tả dữ liệu, SOAP đóng vai trò giao thức truyền tải dữ liệu, WSDL mô tả cho dịch vụ web và UDDI liệt kê danh sách các dịch vụ web đang hoạt động.

13.2.1. XML – Extensible Markup Language

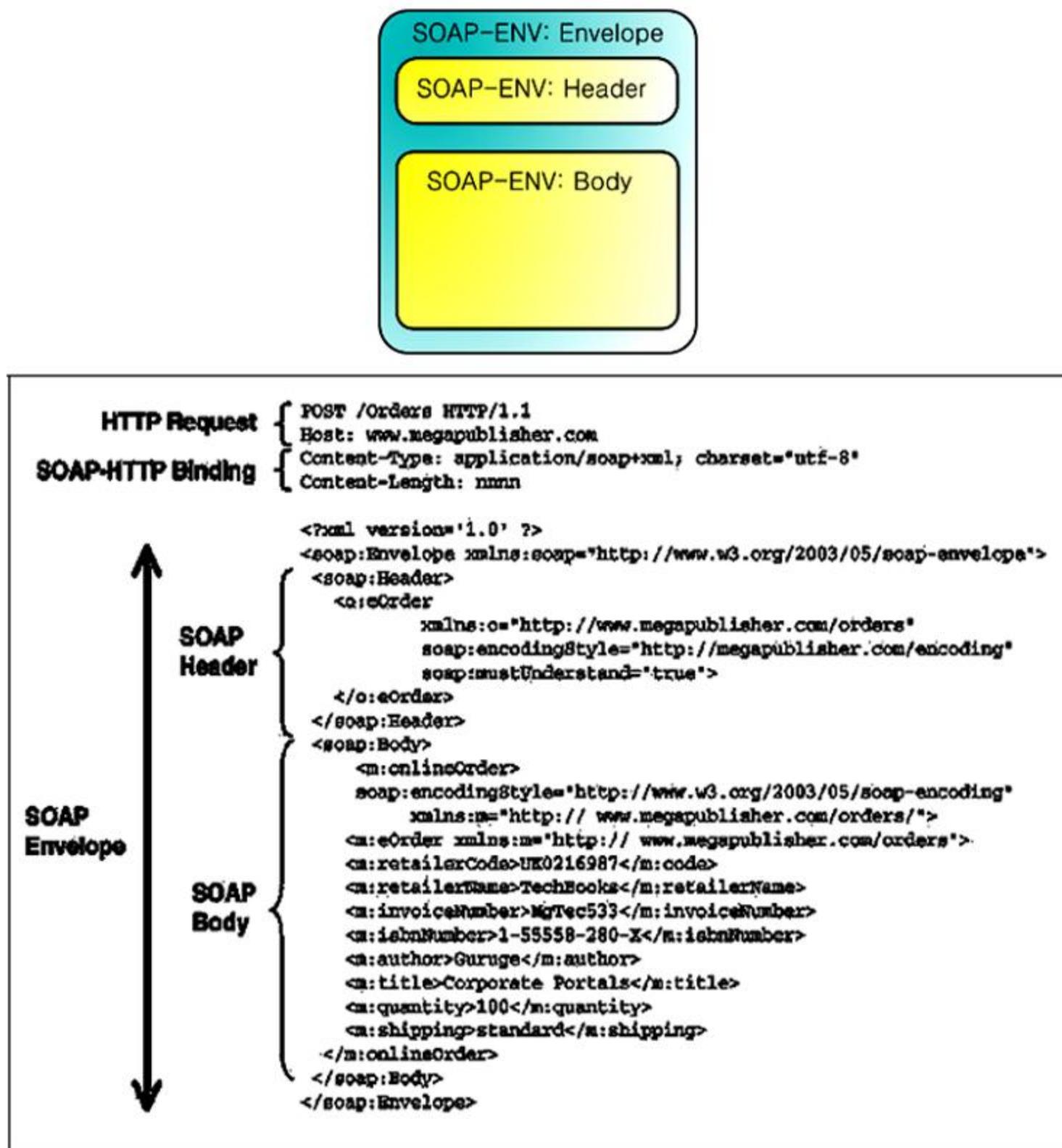
XML do W3C đề ra và được phát triển từ SGML. XML là một ngôn ngữ đánh dấu mở rộng với cấu trúc do người dùng định nghĩa. Về hình thức, XML có cú pháp tương tự HTML, nhưng không tuân theo một đặc tả quy ước như HTML. Người sử dụng hay các chương trình có thể quy ước định dạng các thẻ XML, ngoài ra không chứa bất cứ thông tin nào khác về cách sử dụng hay hiển thị những thông tin ấy.

Web service là sự kết hợp của nhiều thành phần khác nhau, và nó hỗ trợ tương tác giữa các hệ thống được cài đặt trên các môi trường khác nhau. Do đó, cần sử dụng một dạng tài liệu có thể giúp giải quyết vấn đề tương thích và XML hoàn toàn phù hợp với yêu cầu trên. Nó đã trở thành nền tảng cho việc xây dựng các web service. XML có hai vai trò chính:

- Trao đổi dữ liệu trong hệ thống sử dụng web service.
- Mô tả các giao thức sử dụng trong dịch vụ web.

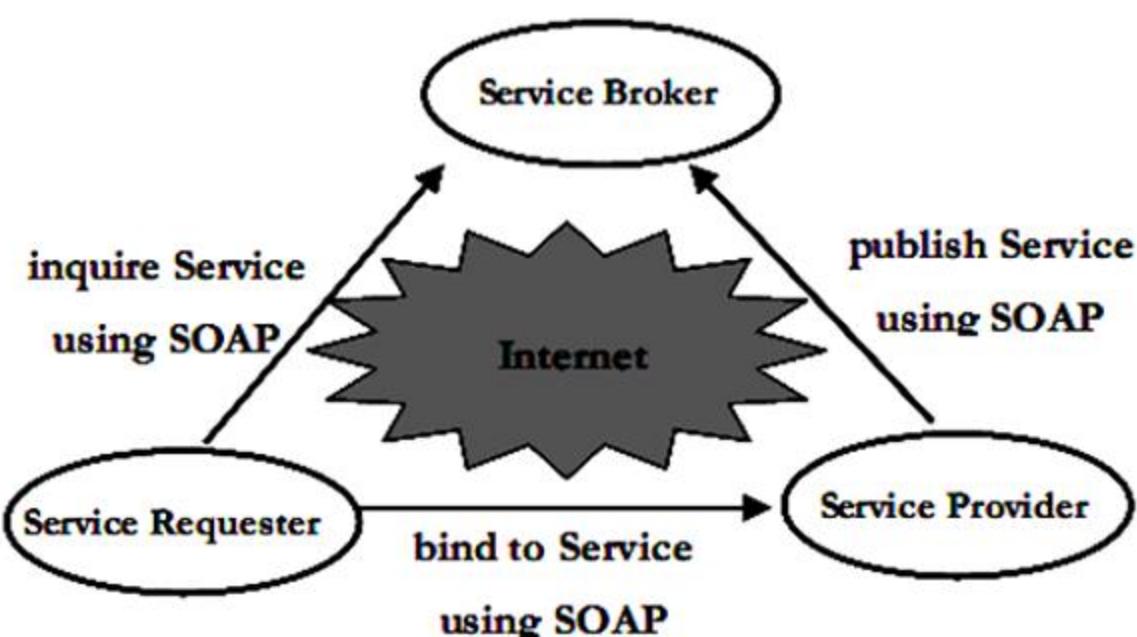
13.2.2. SOAP – Simple Object Access Protocol

SOAP (Simple Object Access Protocol) là giao thức dùng để truy xuất thông tin từ web service thông qua một dạng thông điệp chung. SOAP được Microsoft đề xuất vào năm 1998. Hiện nay, nó thuộc quyền quản lý và cải tiến bởi tổ chức W3C. SOAP là một giao thức dựa trên nền tảng XML, mô tả cách định dạng, đóng gói thông tin của các thông điệp và trao đổi chúng thông qua mạng mà không phụ thuộc vào bất kỳ ngôn ngữ hay môi trường thực thi nào. Đơn vị trao đổi thông tin cơ bản của giao thức SOAP là thông điệp SOAP (SOAP Message). Mỗi thông điệp SOAP sẽ được chỉ định bởi một thẻ root **<Envelope>** chứa hai thành phần là SOAP Header và SOAP Body. SOA Header chứa các thông tin cần thiết cho việc thực hiện chuyển thông điệp hay cơ chế định danh, bảo mật. SOAP Body chứa dữ liệu ứng dụng Cấu trúc của một thông điệp SOAP như hình sau:



Hình 13.1: Cấu trúc của một thông điệp SOAP

Hình sau sẽ mô tả cách mà SOAP được sử dụng trong web services..



Hình 13.2: Sử dụng SOAP trong Web services

13.2.3. WSDL – Web Services Description Language

WSDL (Web Services Description Language) là một dạng tài liệu dựa trên cú pháp XML để mô tả các dịch vụ web. Lúc đầu nó được Microsoft, IBM và Ariba đề xuất, nhưng hiện nay được quản lý bởi tổ chức W3C. Một tài liệu WSDL sẽ cung cấp tài liệu cho các hệ thống phân tán như mô tả chức năng của một web service, cách thức tương tác, các thông điệp tương ứng cho các tác vụ request hay response. Sau đây là cấu trúc cơ bản của một tài liệu

Một tài liệu WSDL bao gồm hai thành phần chính: Phần trừu tượng (abstract definitions), và phần hiện thực (concrete definitions). Phần trừu tượng bao gồm các thông tin được chứa các thẻ types, message, operation và port types. Phần hiện thực chứa thông tin trong các thẻ bindings và ports. Mỗi thành phần sẽ có một tham chiếu đến một thành phần khác được mô tả như sau:

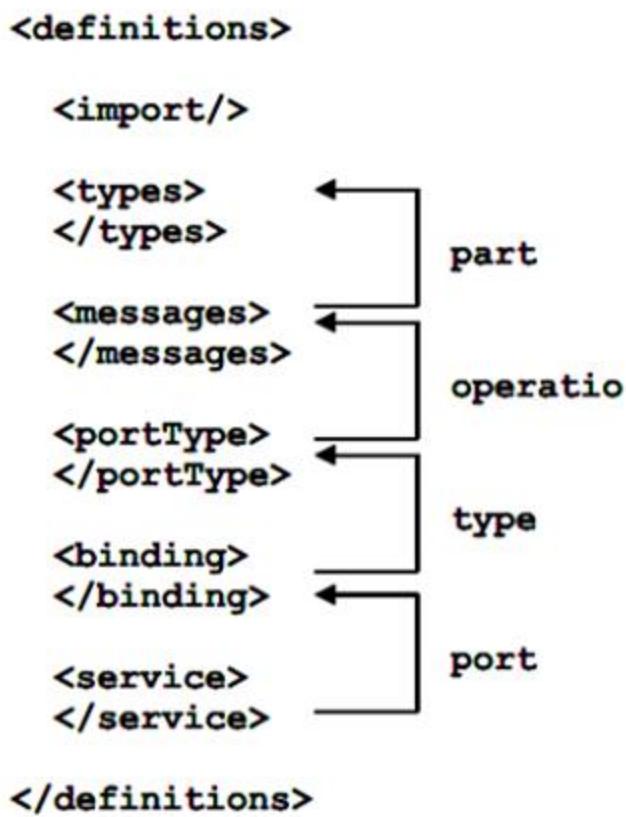
```

<!-- WSDL definition structure -->
<definitions name="MathService"
             targetNamespace="http://example.org/math/"
             xmlns="http://schemas.xmlsoap.org/wsdl/">
    <!-- abstract definitions -->
    <types> ...
    <message> ...
    <portType> ...

    <!-- concrete definitions -->
    <binding> ...
    <service> ...
</definitions>

```

Hình 13.3: WSDL



Hình 13.4: Các thành phần của WSDL

Mỗi thành phần có một chức năng riêng, cụ thể như sau:

- **types**: chỉ định kiểu dữ liệu cho các thông điệp gửi và nhận
- **messages**: là một thành phần trừu tượng mô tả cách thức giao tiếp giữa client và server
- **porttypes**: mô tả ánh xạ giữa các thông điệp – được mô tả trong phần tử messages – và các phương thức (operations)
- **binding**: xác định giao thức nào được sử dụng khi giao tiếp với dịch vụ web. Định nghĩa kiểu binding (RPC/Document) và giao thức vận chuyển. binding cũng định nghĩa các operations
- **Port**: chỉ định địa chỉ hoặc điểm kết nối đến web service, nó thường là một chuỗi URL đơn giản

13.2.4. UDDI – Universal Description, Discovery, and Integration

UDDI (Universal Description, Discovery, and Integration) được đề xuất bởi Microsoft, IBM và Ariba vào năm 2000. Ngày nay, UDDI thuộc quyền quản lý và phát triển bởi tổ chức OASIS (Organization for the Advancement of Structured Information Standards). Nó được xây dựng nhằm mục đích cung cấp khả năng cho phép công bố, tổng hợp và tìm kiếm các dịch vụ web.

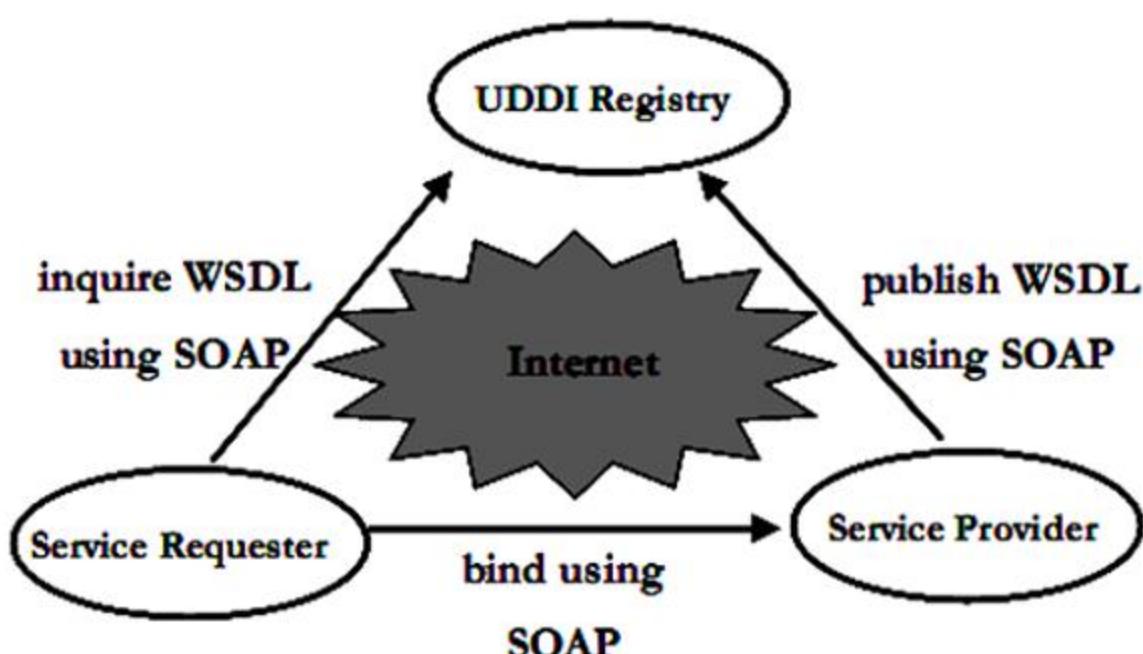
UDDI đưa ra một tập các hàm API được chia làm hai phần: Inquiry API (dùng để tìm kiếm và truy xuất) và Publisher's API (công bố các web services).

Thông tin tổ chức trong UDDI được chia thành ba phần:

- **White pages**: liệt kê thông tin của các nhà cung cấp dịch vụ web bao gồm địa chỉ, thông tin liên lạc, và định danh
- **Yellow pages**: phân loại dịch vụ theo tổ chức hay nhóm dịch vụ hoặc địa điểm đặt các dịch vụ
- **Green pages**: cung cấp thông tin về các dịch vụ web được, về cách thức truy xuất các web services đó

13.2.5. Kết luận

Thông qua các phần trên, chúng ta có một cái nhìn toàn cảnh về Web Service, các kỹ thuật cốt lõi của việc áp dụng các ứng dụng phân tán dựa vào SOAP, WSDL và UDDI:



Các nhà cung cấp Web Service sẽ mô tả Web Service của mình trong một tài liệu WSDL và công bố thông qua việc đăng ký UDDI sử dụng Publisher's API (dựa trên nền tảng SOAP)

Một service requester sử dụng UDDI Inquiry API để tìm kiếm các service provider tương ứng với yêu cầu bên trong hệ thống đăng ký UDDI. Nếu có một dịch vụ nào đó được tìm thấy, việc làm tiếp theo là dựa vào <tModel> để tham chiếu đến tài liệu WSDL tương ứng.

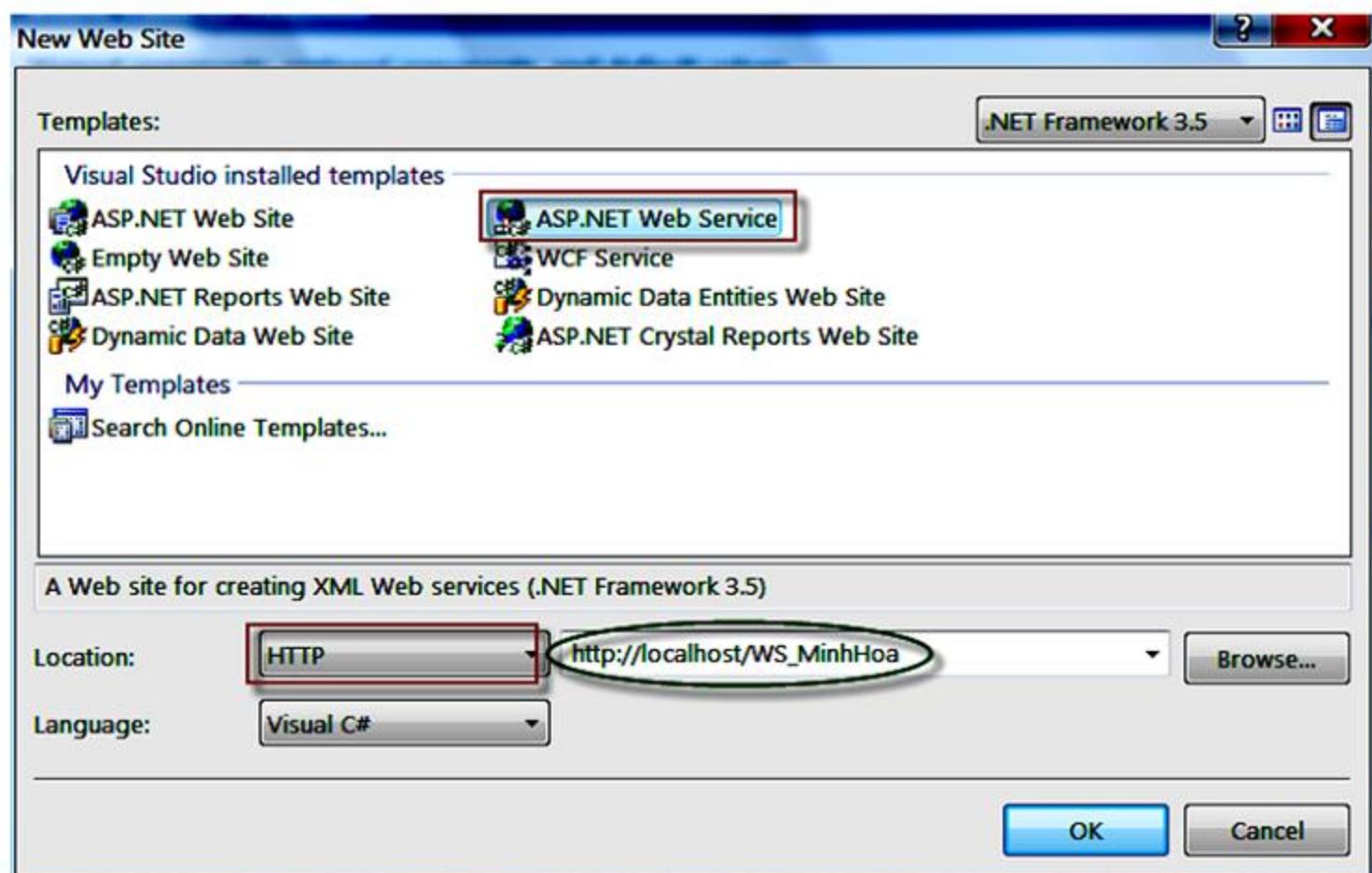
Một SOAP request sẽ được tạo ra tương ứng với Web Service được tìm thấy.

Sau cùng SOAP request sẽ được gửi đến service provider, và provider xử lý trả về.

13.3. XÂY DỰNG ỨNG DỤNG WEB SERVICE

Bước 1: Tạo ứng dụng WebService

Từ menu *File | New | New WebSite...*, cửa sổ *New Web Site* xuất hiện và thiết lập như hình dưới đây để tạo ứng dụng Web Service tên WS_MinhHoa sử dụng Web Server IIS.



Bước 2: Sau khi WS_MinhHoa được tạo, các bạn viết lệnh trong tập tin App_Code/Service.cs như sau.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class Service : System.Web.Services.WebService
{
    public Service () {
    }
    [WebMethod]
    public string LayGioHienHanh() {
```

```

        return DateTime.Now.ToString("hh:mm:ss");
    }
}

```

Trong webservice này chúng ta khai báo một phương thức tên LayGioHienHanh dùng để lấy giờ hiện hành phía server.

Bước 3: Nhấn Ctrl+F5 thi hành ứng dụng, giao diện thi hành sau đây liệt kê danh sách phương thức web hiện có trong Web Service.

The following operations are supported. For a formal definition, please review the [Service Description](#).

- [LayGioHienHanh](#)

This web service is using <http://tempuri.org/> as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. <http://tempuri.org/> is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.

Your XML Web service should be identified by a namespace that you control. For example, you can use your company's Internet domain name as part of the namespace. Although many XML Web service namespaces look like URLs, they need not point to actual resources on the Web. (XML Web service namespaces are URIs.)

Nhập vào liên kết LayGioHienHanh để gọi phương thức này.

Click [here](#) for a complete list of operations.

LayGioHienHanh

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

SOAP 1.1

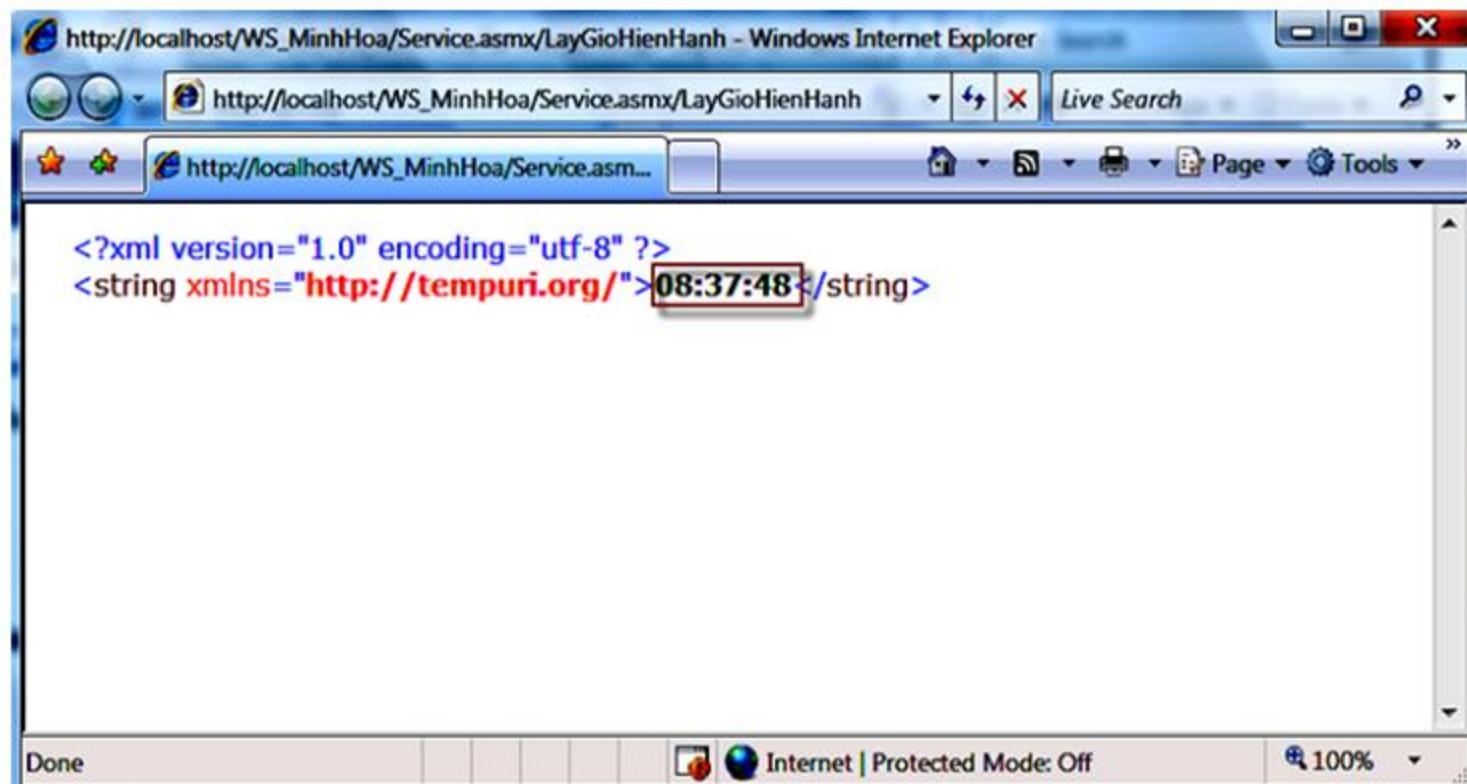
The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```

POST /WS_MinhHoa/Service.asmx HTTP/1.1
Host: localhost

```

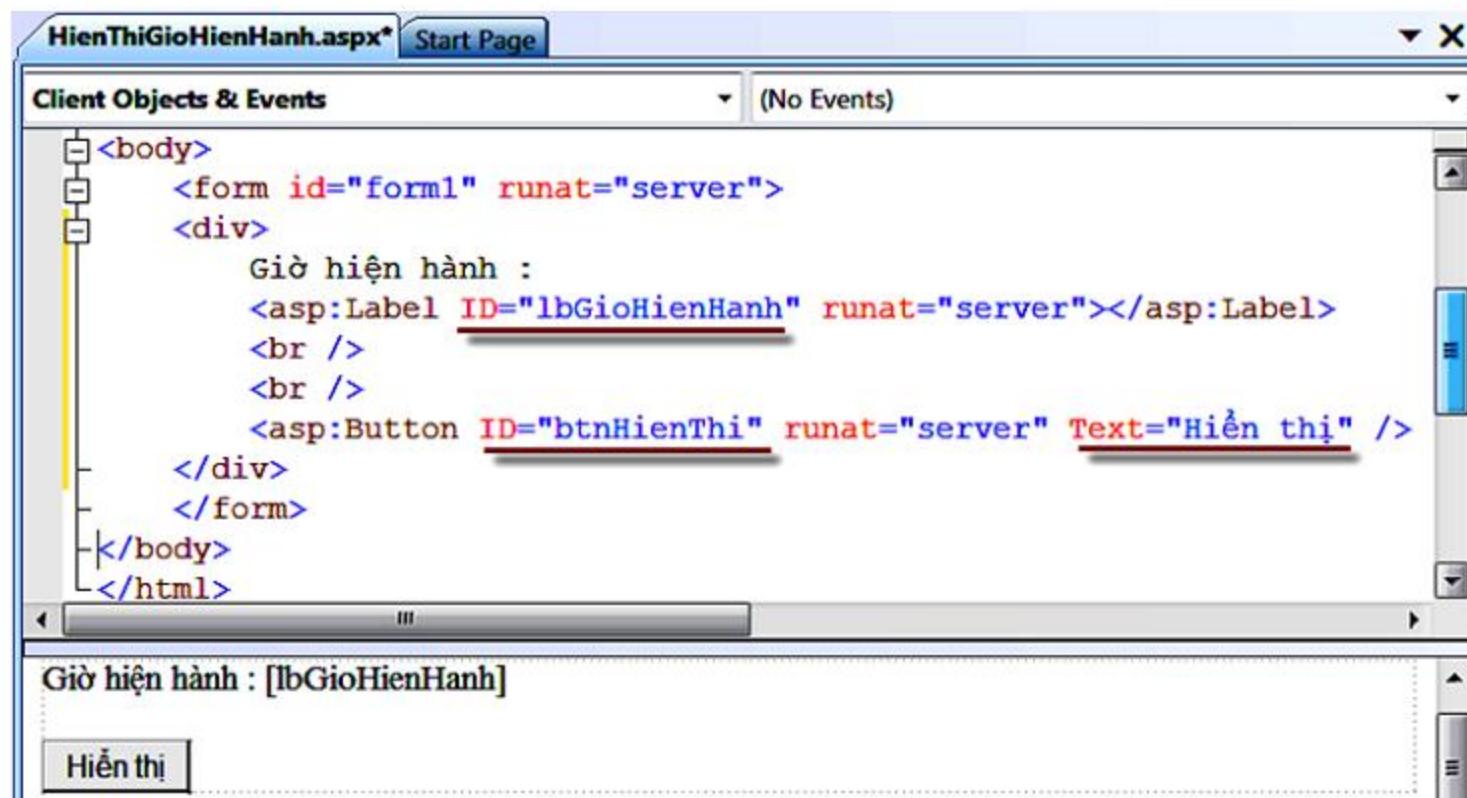
Nhấn nút **Invoke** để thực thi phương thức LayGioHienHanh, kết quả như màn hình sau.



Kết quả trả về từ phương thức LayGioHienHanh

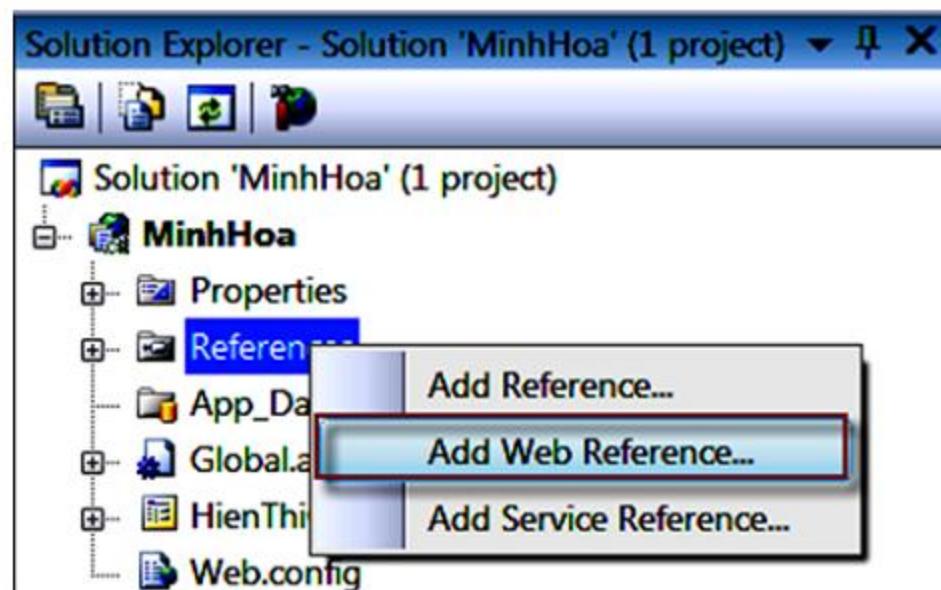
Bước 4: Tạo ứng dụng Web Consumer, các bạn có thể sử dụng bất kỳ ứng dụng nào để gọi đến Web Service (ASP.NET, Window Form, Console,...). Trong ví dụ này, chúng ta sử dụng ASP.NET.

Tạo ứng dụng ASP.NET, gồm một trang tên LayGioHienHanh.aspx giao diện như sau:

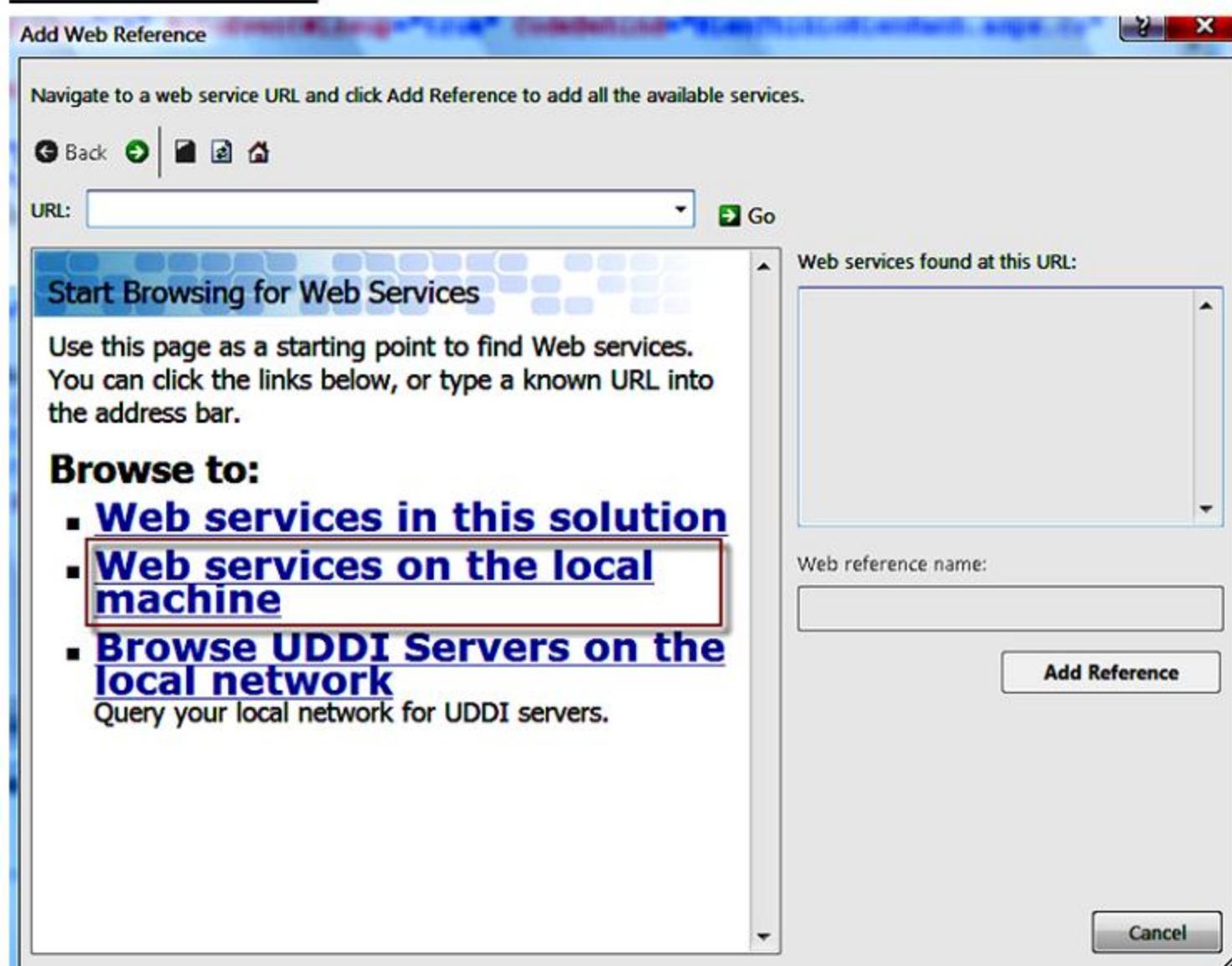


Bước 5: Thiết lập tham chiếu đến ứng dụng web service WS_MinHoa.

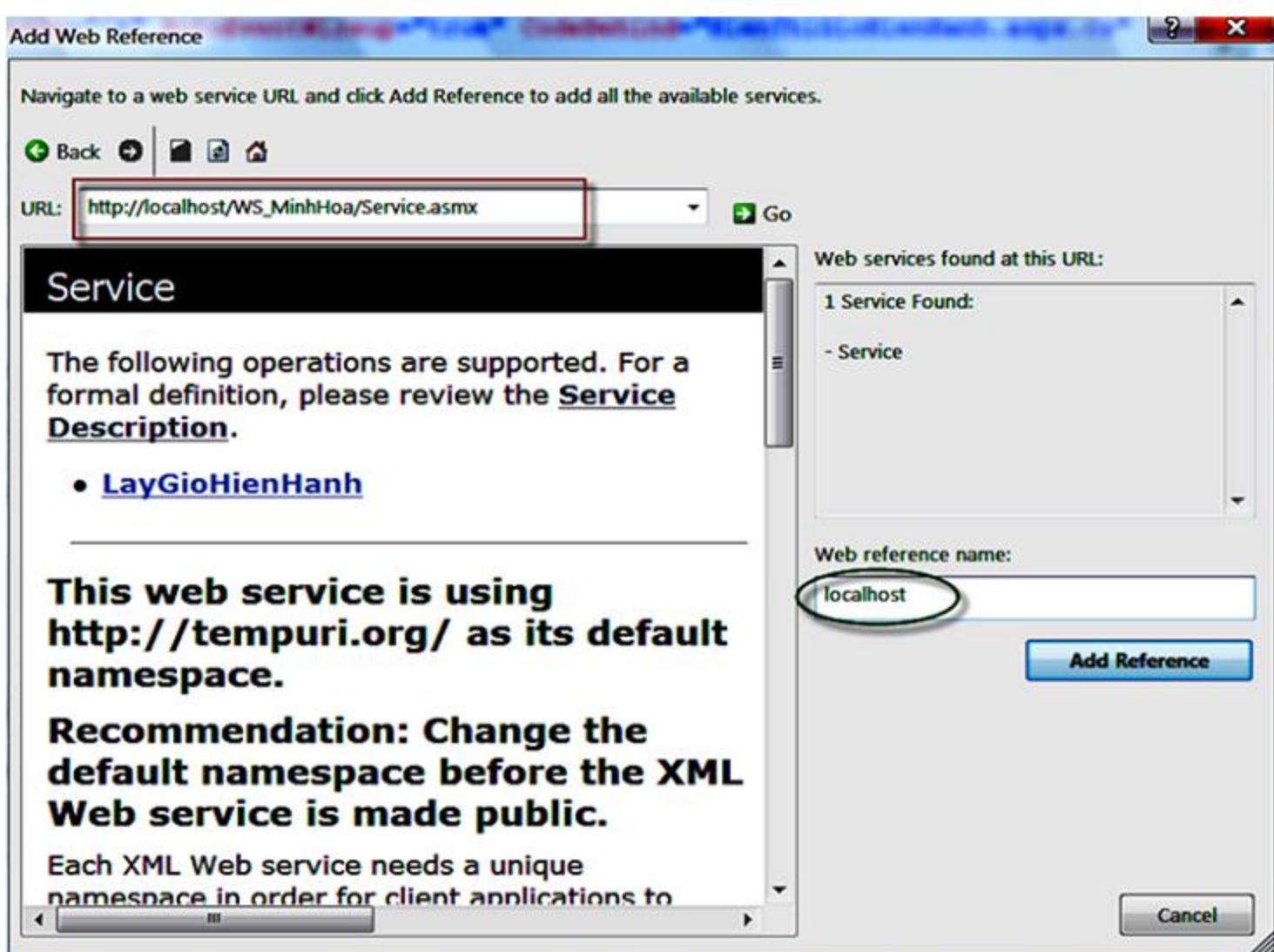
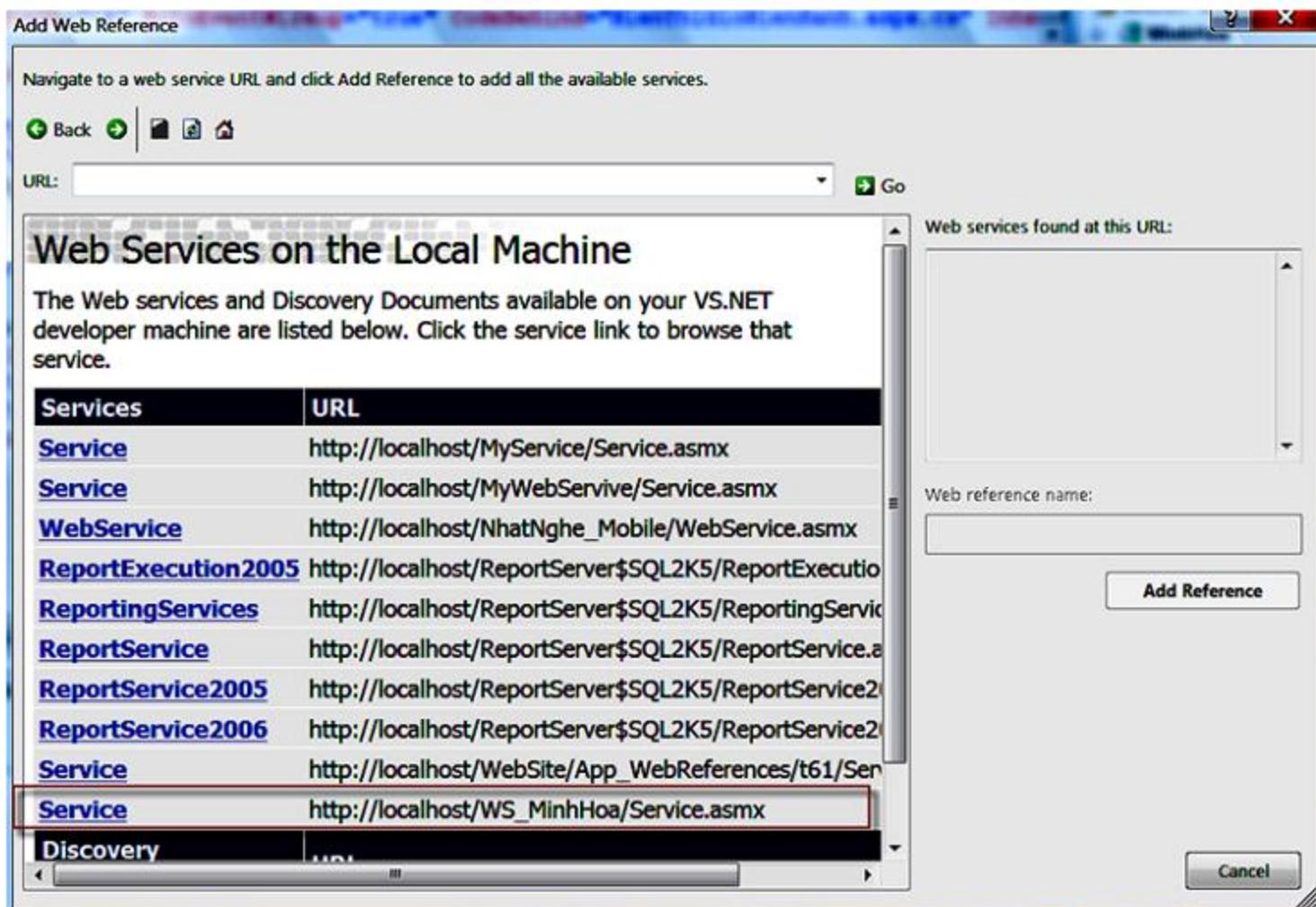
Từ ứng dụng ASP.NET, nhập chọn *References*, nhập phải chuột chọn *Add Web Reference...*



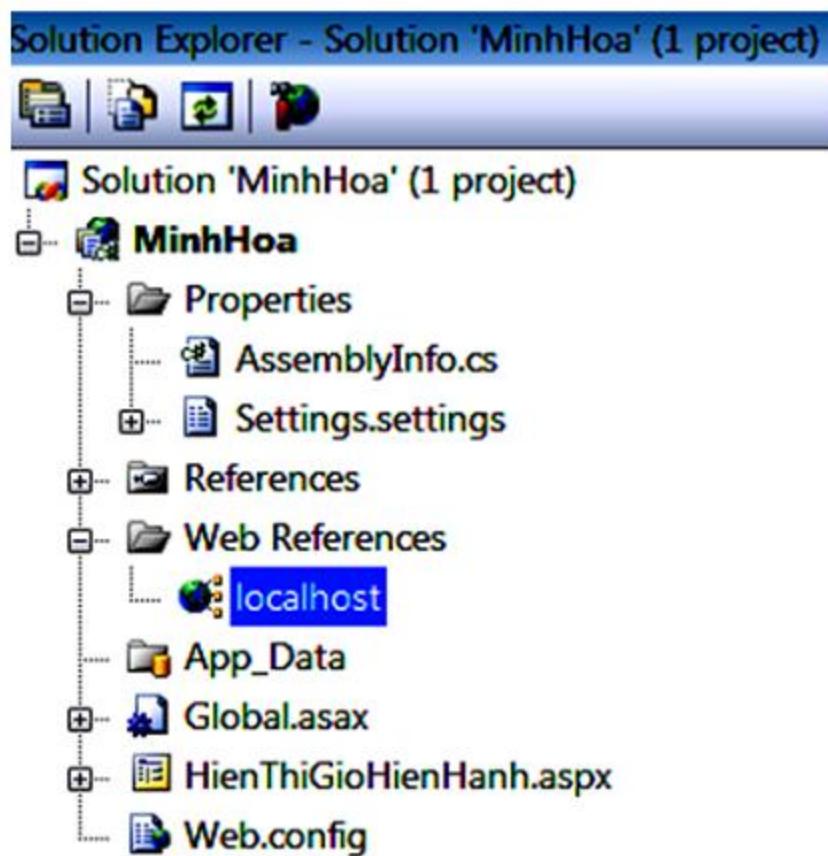
Trên cửa sổ *Add Web Reference*, nhấp vào liên kết **Web services on the local machine**



Sau đó chọn **service** như hình sau.



Nhấn *Add Reference* để thêm tham chiếu vào ứng dụng



Bước 6. Viết lệnh xử lý sự kiện trang LayGioHienHanh.aspx

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace MinhHoa
{
    public partial class HienThiGioHienHanh : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }

        protected void btnHienThi_Click(object sender, EventArgs e)
        {
            //khai báo đối tượng Service
            localhost.Service s = new localhost.Service();
            //gọi phương thức
        }
    }
}
```

```
lbGioHienHanh.Text = s.LayGioHienHanh();  
}  
}  
}
```

Bước 7. Nhấn Ctrl+F5 để thi hành trang LayGioHienHanh.aspx, sau đó nhấn nút **Hiển thị** để xem kết quả.



Chương 14

XÂY DỰNG WEBSITE BÁN HÀNG

14.1. THƯƠNG MẠI ĐIỆN TỬ

14.1.1. Khái niệm

14.1.1.1. Theo nghĩa hẹp

Thương mại điện tử (E-Commerce hay E-Business) là hình thái hoạt động thương mại trong việc mua bán hàng hóa và dịch vụ bằng phương tiện điện tử, nhất là qua internet và các mạng liên thông khác.

Theo Tổ chức thương mại thế giới (WTO): “Thương mại điện tử bao gồm việc sản xuất, quảng cáo, bán hàng và phân phối sản phẩm được mua bán và thanh toán trên mạng internet, nhưng được giao nhận một cách hữu hình, cả các sản phẩm giao nhận cũng như những thông tin số hóa thông qua mạng internet”.

14.1.1.2. Theo nghĩa rộng

Thương mại điện tử là các giao dịch tài chính và thương mại bằng phương tiện điện tử như: trao đổi dữ liệu điện tử, chuyển tiền điện tử và các hoạt động như gửi/rút tiền bằng thẻ tín dụng.

Theo Luật mẫu về thương mại điện tử của Ủy ban Liên hợp quốc về Luật Thương mại quốc tế (UNCITRAL): “Thuật ngữ thương mại-Commerce- cần được diễn giải theo nghĩa rộng để bao quát các vấn đề phát sinh từ mọi quan hệ mang tính chất thương mại dù có hay không có hợp đồng. Các quan hệ mang tính thương mại bao gồm các dịch vụ sau đây: bất cứ giao dịch nào về cung cấp hoặc trao đổi hàng hóa, dịch vụ; thỏa thuận phân phối; đại diện hoặc đại lý thương mại, ủy thác hoa hồng (factoring), cho thuê dài hạn (leasing); xây dựng các công trình, tư vấn, kỹ thuật công trình (engineering); đầu tư, cấp vốn, ngân hàng, bảo hiểm; thỏa thuận khai thác hoặc tò nhượng, liên doanh và các hình thức khác về hợp tác công nghiệp hoặc kinh doanh; chuyên chở hàng hóa hay hành khách bằng đường biển, đường hàng không, đường sắt hoặc đường bộ”.

Như vậy, có thể thấy rằng phạm vi của Thương mại điện tử rất rộng, bao quát hầu hết các lĩnh vực hoạt động kinh tế. Trong đó, hoạt động mua bán hàng hóa và dịch vụ chỉ là một phạm vi rất nhỏ trong Thương mại điện tử.

Ngày nay, người ta hiểu khái niệm Thương mại điện tử thông thường là tất cả các phương pháp tiến hành kinh doanh và các quy trình

quản trị thông qua các kênh điện tử, mà trong đó internet hay ít nhất là các kỹ thuật và giao thức được sử dụng trong internet đóng một vai trò cơ bản, và công nghệ thông tin được coi là điều kiện tiên quyết.

14.1.2. Đặc trưng

So với các hoạt động thương mại truyền thống, Thương mại điện tử có một số điểm khác biệt cơ bản sau:

- a. Các bên tiến hành giao dịch trong Thương mại điện tử không tiếp xúc trực tiếp với nhau và không đòi hỏi phải biết nhau từ trước.
- b. Các giao dịch thương mại truyền thống được thực hiện với sự tồn tại của khái niệm biên giới quốc gia, còn Thương mại điện tử được thực hiện trong một thị trường không có biên giới (thị trường thống nhất toàn cầu). Thương mại điện tử tác động trực tiếp tới môi trường cạnh tranh toàn cầu.
- c. Trong hoạt động giao dịch Thương mại điện tử đều có sự tham gia của ít nhất ba chủ thể, trong đó có một bên không thể thiếu được là người cung cấp dịch vụ mạng, các cơ quan chứng thực.
- d. Đối với thương mại truyền thống thì mạng lưới thông tin chỉ là phương tiện để trao đổi dữ liệu, còn với Thương mại điện tử thì mạng lưới thông tin chính là thị trường.

14.1.3. Phân loại

Trong thương mại điện tử có ba chủ thể tham gia: Doanh nghiệp (B) giữ vai trò động lực phát triển thương mại điện tử, Người tiêu dùng (C) giữ vai trò quyết định sự thành công của thương mại điện tử và Chính phủ (G) giữ vai trò định hướng, điều tiết và quản lý. Theo tính chất của người tham gia, thương mại điện tử bao gồm:

- Người tiêu dùng:
 - C2C (Consumer to Consumer): Người tiêu dùng với người tiêu dùng
 - C2B (Consumer to Business): Người tiêu dùng với doanh nghiệp
 - C2G (Consumer to Government): Người tiêu dùng với Chính phủ
- Doanh nghiệp:
 - B2C (Business to Consumer): Doanh nghiệp với người tiêu dùng
 - B2B (Business to Business): Doanh nghiệp với doanh nghiệp
 - B2G (Business to Government): Doanh nghiệp với Chính phủ

- B2E (Business to Employee): Doanh nghiệp với nhân viên.
- Chính phủ:
 - G2C (Government to Consumer): Chính phủ với người tiêu dùng
 - G2B (Government to Business): Chính phủ với doanh nghiệp
 - G2G (Government to Government): Chính phủ với Chính phủ

Trong đó, B2B và B2C là hai loại hình giao dịch thương mại điện tử quan trọng nhất.

- **B2B (Business to Business)**: là việc thực hiện các giao dịch giữa các doanh nghiệp với nhau trên mạng. Các bên tham gia giao dịch gồm: người trung gian trực tuyến, người mua và người bán. Các loại giao dịch gồm: mua ngay theo yêu cầu khi giá cả thích hợp và mua theo hợp đồng dài hạn, dựa trên đàm phán cá nhân giữa người mua và người bán. Các loại giao dịch cơ bản:
 - Bên bán: (một bên bán - nhiều bên mua) là mô hình dựa trên công nghệ web trong đó một công ty bán cho nhiều công ty mua. Có ba phương pháp bán trực tiếp trong mô hình này: bán từ Catalog điện tử, bán qua quá trình đấu giá, bán theo hợp đồng cung ứng dài hạn đã thỏa thuận trước.
 - Bên mua: một bên mua – nhiều bên bán
 - Sàn giao dịch: nhiều bên bán – nhiều bên mua
 - Thương mại điện tử phối hợp: các đối tác phối hợp nhau ngay trong quá trình thiết kế, chế tạo sản phẩm
- **B2C (Business to Consumer)**: đây là mô hình bán lẻ trực tiếp đến người tiêu dùng. Trong thương mại điện tử, bán lẻ điện tử có thể từ nhà sản xuất hoặc từ một cửa hàng thông qua kênh phân phối. Mô hình kinh doanh bán lẻ có thể phân loại theo quy mô các loại hàng hóa, theo phạm vi địa lý (tỉnh, thành phố, khu vực) hoặc theo kênh bán (bán trực tiếp hoặc qua kênh phân phối). Một số hình thức của cửa hàng bán lẻ trên mạng:
 - Brick-and-mortar: là loại cửa hàng bán lẻ kiểu truyền thống, không sử dụng internet
 - Click-and-mortar: là loại cửa hàng bán lẻ truyền thống nhưng có kênh bán hàng qua mạng
 - Cửa hàng ảo: là cửa hàng bán lẻ hoàn toàn trên mạng mà không sử dụng kênh bán truyền thống

14.1.4. Các hình thức hoạt động

a. Thư điện tử (email)

Các doanh nghiệp, cửa hàng kinh doanh,... sử dụng thư điện tử để gửi thư cho nhau hoặc gửi thư quảng bá sản phẩm một cách trực tuyến thông qua mạng, gọi là thư điện tử (electronic mail, viết tắt là e-mail). Thông tin trong thư điện tử không phải tuân theo một cấu trúc định trước nào.

b. Thanh toán điện tử

- ✓ Thanh toán điện tử (electronic payment) là việc thanh toán tiền thông qua thư điện tử (electronic message) như trả lương bằng cách chuyển tiền trực tiếp vào tài khoản, trả tiền mua hàng bằng thẻ tín dụng,... Ngày nay, với sự phát triển của TMĐT, thanh toán điện tử đã mở rộng sang các lĩnh vực mới.
- ✓ Trao đổi dữ liệu điện tử tài chính (Financial Electronic Data Interchange, gọi tắt là FEDI): chuyên phục vụ cho việc thanh toán điện tử giữa các công ty giao dịch với nhau bằng điện tử.
- ✓ Tiền lẻ điện tử (Internet Cash): là tiền mặt được mua từ nơi phát hành (ngân hàng hoặc một tổ chức tín dụng nào đó), sau đó được chuyển đổi tự do sang các đồng tiền khác thông qua internet, áp dụng trong cả phạm vi một nước cũng như giữa các quốc gia.
- ✓ Ví điện tử (Electronic Purse): là nơi để tiền mặt Internet, chủ yếu là thẻ thông minh (smart card), còn gọi là thẻ giữ tiền (stored value card), tiền được trả cho bất kỳ ai đọc được thẻ đó.
- ✓ Giao dịch điện tử của ngân hàng (Digital Banking).

c. Trao đổi dữ liệu điện tử

- ✓ Trao đổi dữ liệu điện tử (Electronic Data Interchange, viết tắt là EDI) là việc trao đổi các dữ liệu dưới dạng “có cấu trúc” (structured form) từ máy tính điện tử này sang máy tính điện tử khác, giữa các công ty hoặc đơn vị đã thỏa thuận buôn bán với nhau.
- ✓ Ngày nay, EDI chủ yếu được thực hiện thông qua mạng Internet. Để phục vụ cho buôn bán giữa các doanh nghiệp thuận lợi hơn với chi phí truyền thông không quá tốn kém, người ta đã xây dựng một kiểu mạng mới gọi là “mạng riêng ảo” (Virtual Private Network), là mạng riêng dạng Intranet của một doanh nghiệp nhưng được thiết lập dựa trên chuẩn trang Web và truyền thông qua mạng Internet.
- ✓ Công việc trao đổi EDI trong TMĐT thường gồm các nội dung sau:
 - Giao dịch kết nối

- Đặt hàng
- Giao dịch gửi hàng
- Thanh toán

d. Truyền dung liệu (Content)

Dung liệu (Content) là nội dung của hàng hóa số, giá trị của nó không phải trong vật mang tin mà nằm trong bản thân nội dung của nó. Hàng hóa số có thể giao qua mạng, như: tin tức, nhạc, phim, các chương trình truyền hình, các chương trình phần mềm, vé xem phim, vé máy bay,...

e. Mua bán hàng hóa hữu hình

Đến nay, danh sách các hàng hóa bán lẻ qua mạng đã mở rộng, từ hoa, quần áo đến ô tô và xuất hiện một loại hoạt động gọi là “mua hàng điện tử” (electronic shopping) hay “mua hàng trên mạng”; ở một số nước, internet bắt đầu trở thành công cụ để cạnh tranh bán lẻ hàng hữu hình (Retail of tangible goods). Tận dụng tính năng đa phương tiện (multimedia) của môi trường Web, người bán hàng xây dựng trên mạng các “cửa hàng ảo” (virtual shop), gọi là ảo bởi vì, cửa hàng có thật nhưng ta chỉ xem toàn bộ quang cảnh cửa hàng và các hàng hóa chứa trong đó trên từng màn hình.

14.1.5. Lợi ích

Sau đây là một số lợi ích mà thương mại điện tử mang lại:

- Thu thập được nhiều thông tin: giúp doanh nghiệp thu thập thông tin phong phú về kinh tế thị trường, nhờ đó có thể xây dựng chiến lược sản xuất và kinh doanh thích hợp với xu thế phát triển của thị trường trong nước và quốc tế.
- Giảm chi phí sản xuất: trước hết là chi phí văn phòng, chi phí tìm kiếm chuyển giao tài liệu và chi phí in.
- Giảm chi phí bán hàng, tiếp thị và giao dịch: bằng phương tiện internet/web một nhân viên bán hàng có thể giao dịch được với rất nhiều khách hàng, giúp giảm chi phí và thời gian giao dịch.
- Xây dựng quan hệ với đối tác: tạo điều kiện cho việc thiết lập và củng cố mối quan hệ giữa các thành viên tham gia vào quá trình thương mại như người tiêu thụ, doanh nghiệp, cơ quan chính phủ,... Mọi người có thể giao tiếp trực tiếp và liên tục với nhau, có cảm giác như không có khoảng cách về địa lý và thời gian nữa.
- Tạo điều kiện sớm tiếp cận với kinh tế tri thức: trước hết, thương mại điện tử sẽ kích thích sự phát triển của công nghệ thông tin, tạo cơ sở cho phát triển kinh tế tri thức.

14.1.6. Kết Luận

Phần này giới thiệu một số lý thuyết về thương mại điện tử như khái niệm, đặc trưng, phân loại, các hình thức hoạt động và lợi ích.

14.2. KHẢO SÁT HIỆN TRẠNG

14.2.1. Đánh giá một số website thương mại điện tử tại Việt Nam và trên thế giới

Để có cái nhìn tổng quát về thương mại điện tử, chúng tôi đã khảo sát một số website thương mại điện tử nổi tiếng ở Việt Nam và trên thế giới.

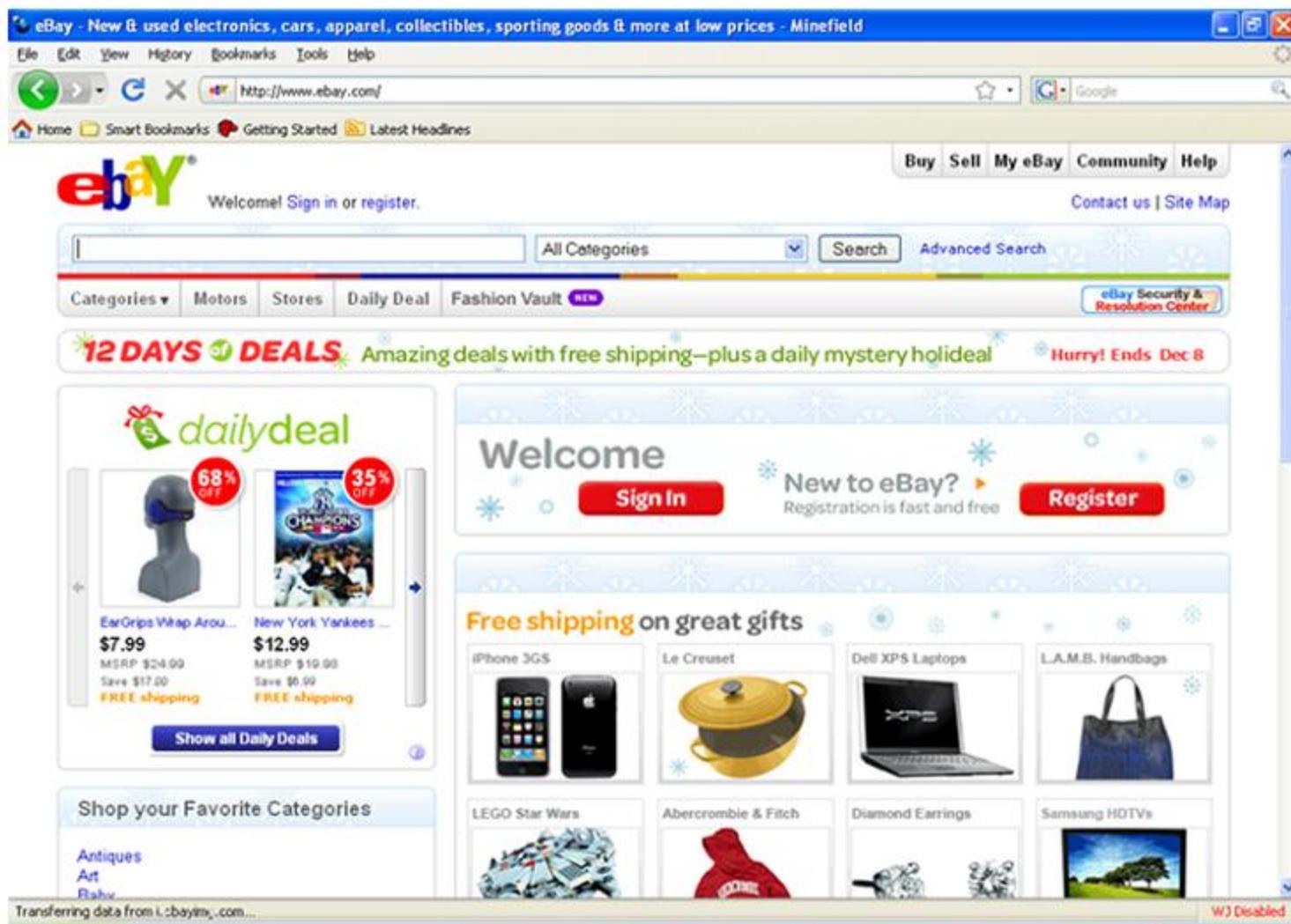
Thế giới

Thương mại điện tử ở nước ngoài nhất là các nước phát triển đã có từ lâu và hiện nay đã phát triển vượt bậc. Do điều kiện cơ sở vật chất thuận lợi, hầu hết mọi người dân đều sử dụng internet và việc gửi tiền ở ngân hàng là rất phổ biến, nên thương mại điện tử được xem như là sự lựa chọn hàng đầu của mọi người.Thêm vào đó, thương mại điện tử mang lại nhiều lợi ích thiết thực cho người sử dụng như tiết kiệm thời gian, tiết kiệm tiền bạc và có các phương thức thanh toán rất dễ dàng, thuận lợi và nhanh chóng. Từ đó, sự tin tưởng của người dùng vào việc lựa chọn thương mại điện tử ngày càng tăng và ngày càng thu hút mọi người tham gia. Do đó, lợi nhuận mà thương mại điện tử mang lại cho các nước phát triển là không nhỏ.

Một số website thương mại điện tử nổi tiếng ở nước ngoài:

[1] <http://www.ebay.com>

Trang web này không chỉ cho phép người dùng mua sản phẩm trực tiếp mà còn cho họ đưa sản phẩm lên để bán cho các khách hàng khác. Ebay cho phép khách hàng lựa chọn phương thức thanh toán sao cho tiện lợi với mình nhất. Một số phương thức thanh toán mà Ebay hỗ trợ là: thanh toán thông qua PayPal, ProPay, Paymate, Moneybookers và Credit Card hoặc Debit Card. Trong đó, phương thức thanh toán chủ yếu dùng hầu hết cho người bán và người mua đó là thanh toán qua PayPal. Khách hàng có thể thanh toán ở bất kỳ nơi nào trên thế giới.



Hình 14.1: Website www.ebay.com

[2] <http://www.amazon.com>

Trang web Amazon có một điểm mới là cho phép khách hàng có thẻ bán sản phẩm của mình trên chính trang web thương mại điện tử của khách hàng (WebStore with Amazon). Amazon cũng hỗ trợ nhiều phương thức thanh toán cho khách hàng, sao cho có thể mang lại sự tiện lợi nhất. Một số phương thức thanh toán mà Amazon hỗ trợ là: thanh toán trực tiếp từ tài khoản ngân hàng của khách hàng, Amazon.com Store Card, Gift Cards, Credit Card và Check Card như Visa, MasterCard/EuroCard, Discover Network, American Express, JCB, Amazon Credit Account.



Hình 14.2: Website www.amazon.com

Việt Nam

Nhìn chung thương mại điện tử vẫn còn trong giai đoạn thử thách và có tiềm năng phát triển rất lớn. Ở Việt Nam, thương mại điện tử dường như vẫn còn mới đối với mọi người. Hiện nay, thương mại điện tử đang dần dần khẳng định vị thế và lợi ích của mình, thu hút ngày càng nhiều khách hàng tham gia mua hàng trực tuyến. Nhưng do tâm lý người dân chưa quen với việc có một tài khoản ở ngân hàng, cũng như chỉ một số ngân hàng có sự hỗ trợ thanh toán trực tuyến, nên việc giao dịch thương mại điện tử thông qua ngân hàng vẫn còn hạn chế. Các website thương mại điện tử có những phương thức thanh toán riêng cho mình, nhằm hỗ trợ tối đa cho các khách hàng, và giúp các khách hàng có thể thực hiện phương thức thanh toán đơn giản, tiện dụng và nhanh nhất.

Một số website thương mại điện tử tiêu biểu:

[1]. <http://ebay.chodientu.vn>

Ở website này, khách hàng có thể lựa chọn các khu vực được hỗ trợ, đó là khu vực khách hàng có thể mua sản phẩm, khu vực khách hàng có thể bán sản phẩm của mình và cuối cùng là khu vực mua đặc biệt, khi mà sản phẩm khách hàng cần mua không có trên website. Ngoài ra, khách hàng còn có thể lựa chọn mua hàng bằng hình thức đấu giá hoặc

mua ngay. Trang web này có hai phương thức thanh toán được hỗ trợ cho khách hàng, đó là thông qua:

- Tài khoản tại Ngân lượng (<http://www.nganluong.vn>).
- Tài khoản tại một số ngân hàng có hỗ trợ thanh toán trực tuyến như Ngân hàng Đông Á, Ngân hàng Đầu tư và phát triển Việt Nam, Ngân hàng Ngoại thương Việt Nam,...



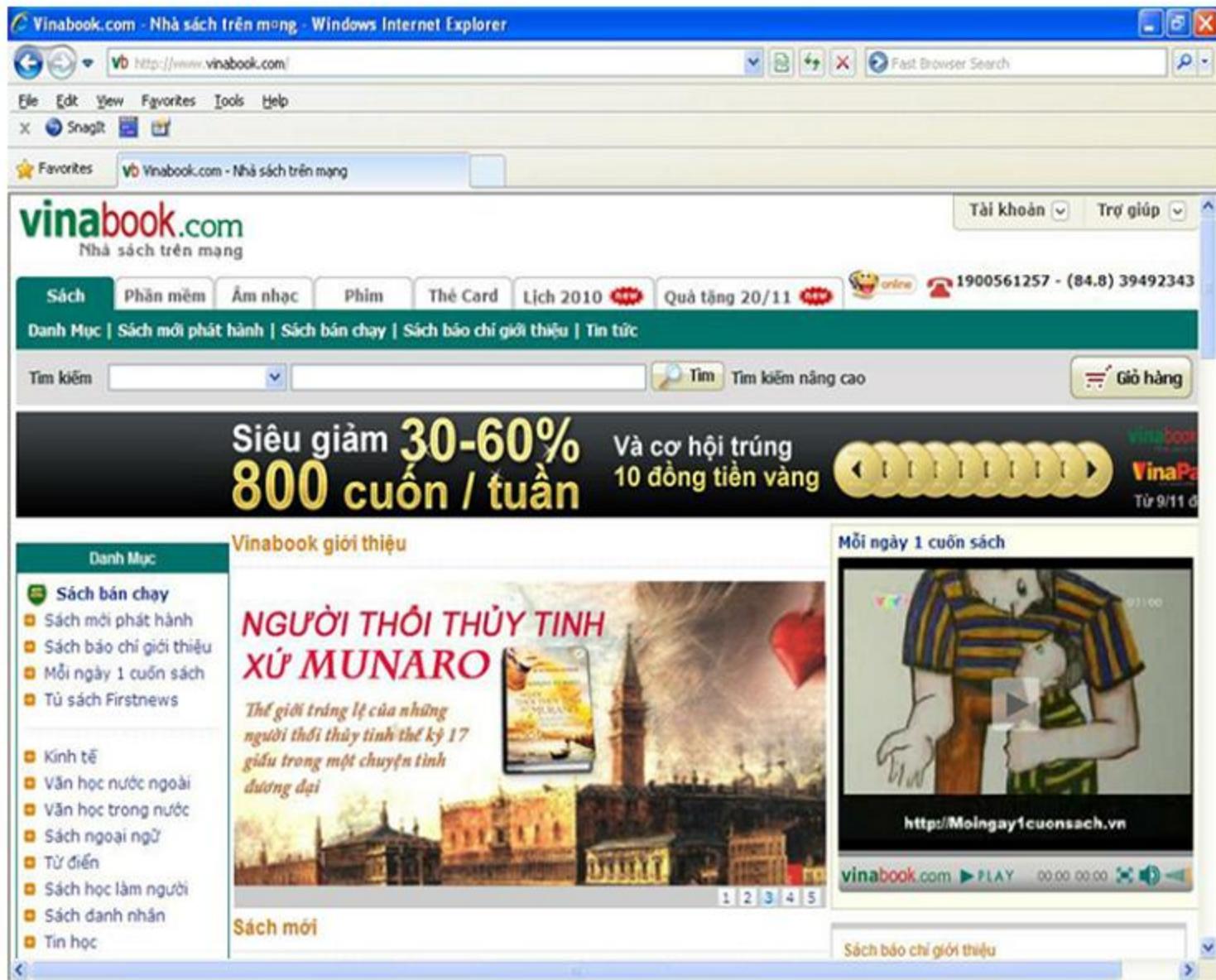
Hình 14.3: Website ebay.chodientu.vn

[2]. <http://www.vinabook.com>

Ở website này, chỉ có thể hỗ trợ khách hàng mua sản phẩm mà không thể bán sản phẩm mình có thông qua trang web cho các khách hàng khác. Bù lại, trang web có hỗ trợ nhiều phương thức thanh toán, có thể thanh toán trong nước cũng như thanh toán quốc tế, giúp khách hàng có thể lựa chọn phương thức thanh toán phù hợp với mình nhất.

- Thanh toán trong nước
 - Thanh toán trực tiếp khi nhận hàng
 - Chuyển tiền qua bưu điện

- Thanh toán bằng thẻ ATM
- Thanh toán bằng ví điện tử Payoo
- Thanh toán bằng Vinapay: ví điện tử Vcash
- Thẻ trả trước Vinabook
- Thanh toán quốc tế
 - Chuyển tiền Western Union
 - Thanh toán bằng Credit Card
 - Chuyển khoản ngân hàng



Hình 14.4: Website www.vinabook.com

14.2.2. Các yêu cầu nghiệp vụ của website

Xây dựng ứng dụng sử dụng công nghệ web service với các yêu cầu sau:

- Xây dựng website bán sách qua mạng
- Tìm kiếm và so sánh sách của website với các sản phẩm tương tự trên các website khác (amazon, ebay, ...)

- Thực hiện thanh toán trực tuyến ngay trên website

Từ các yêu cầu trên, web site phải đảm bảo các nghiệp vụ chủ yếu sau:

Bảng 14.1: Các nghiệp vụ của Website

Nghiệp vụ	Mô tả
Đăng ký tài khoản	Cho phép khách hàng đăng ký tài thành viên, tiện cho những khách hàng thường xuyên mua sản phẩm trên website
Tìm kiếm sản phẩm có trong website	Người dùng có thể tìm kiếm sản phẩm mình cần trên website
Tìm kiếm sản phẩm ở website khác	Chức năng này cho phép người dùng tìm sản phẩm trên các website khác Amazon, Ebay... Để người dùng có thể tham khảo, mua hàng từ các website đó
Xem thông tin sản phẩm	Liệt kê một số thông tin liên quan đến sản phẩm
Thanh toán trực tuyến	Cho phép người dùng thanh toán trực tuyến thông qua dịch vụ thanh toán của Paypal,...
Tạo giỏ hàng	Người dùng sẽ tạo cho mình một giỏ hàng để lưu trữ sản phẩm mình cần mua
Cập nhật, thêm, xóa sản phẩm trong giỏ hàng	Cho phép người dùng thêm, xóa sản phẩm cần mua từ giỏ hàng
So sánh sản phẩm	So sánh sản phẩm đang xem với các sản phẩm tương tự trên amazon và ebay (nếu có)
Thanh toán trực tuyến	Thanh toán thông qua dịch vụ của PayPal
Chuyển đổi tiền tệ	Chuyển đổi qua lại giá của sản phẩm giữa các loại tiền tệ thông dụng
Quản lý user	Admin quản lý tài khoản người dùng
Quản lý tài khoản	Người dùng có thể thay đổi một số thông tin đã đăng ký
Thêm, xóa, sửa sản phẩm của website	Cập nhật sản phẩm của website
Quản lý các đơn đặt hàng	Xem chi tiết, xác nhận, hoặc hủy bỏ các đơn đặt hàng

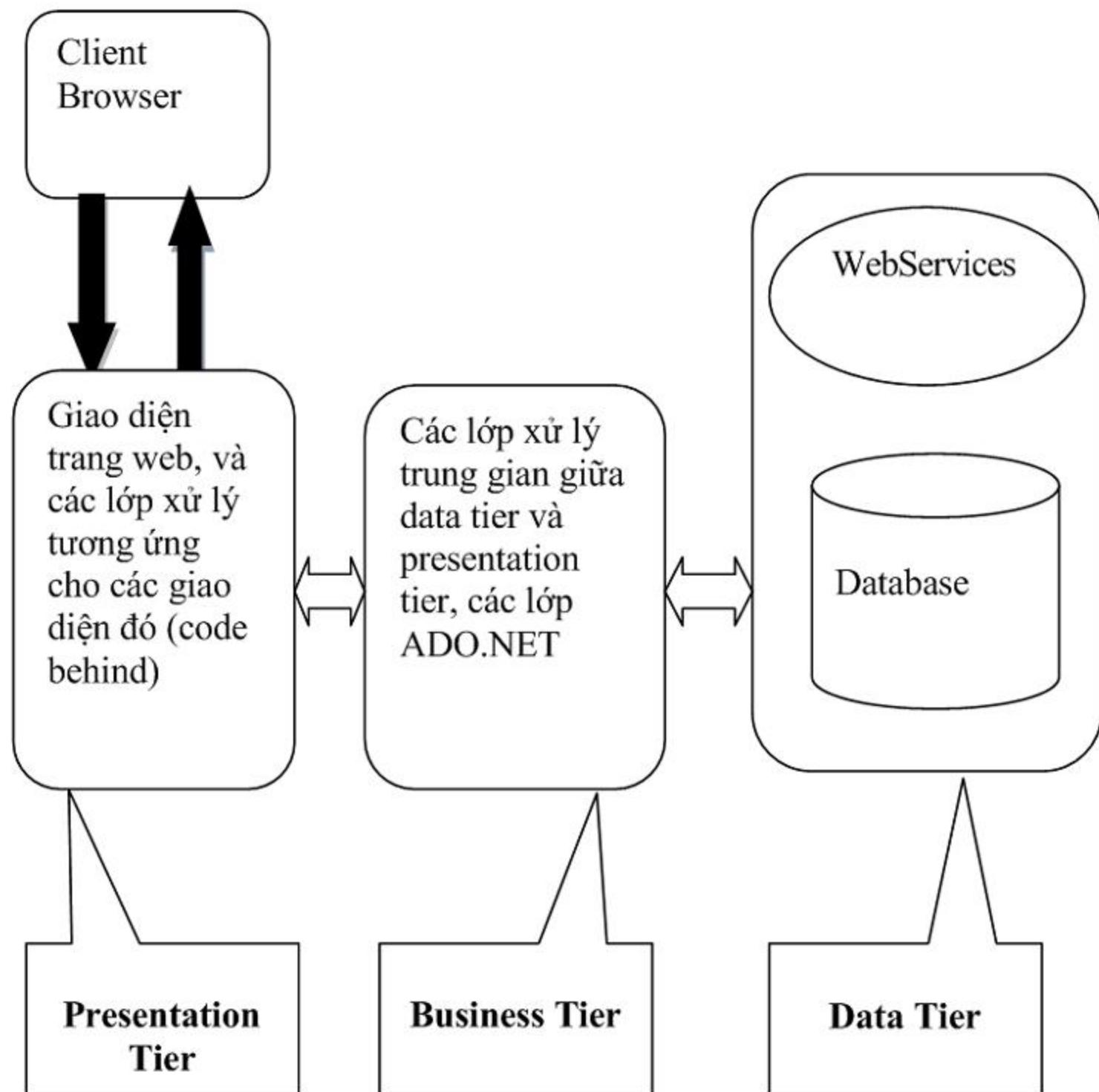
14.3. KẾT LUẬN

Chương này trình bày văn tắt hiện trạng một số website thương mại điện tử ở Việt Nam và trên thế giới, qua đó xây dựng các yêu cầu nghiệp vụ phù hợp cho website.

Chương 15

THIẾT KẾ WEBSITE BÁN HÀNG – MỨC DỮ LIỆU

Kiến trúc xây dựng website bán sách qua mạng có áp dụng webservices.



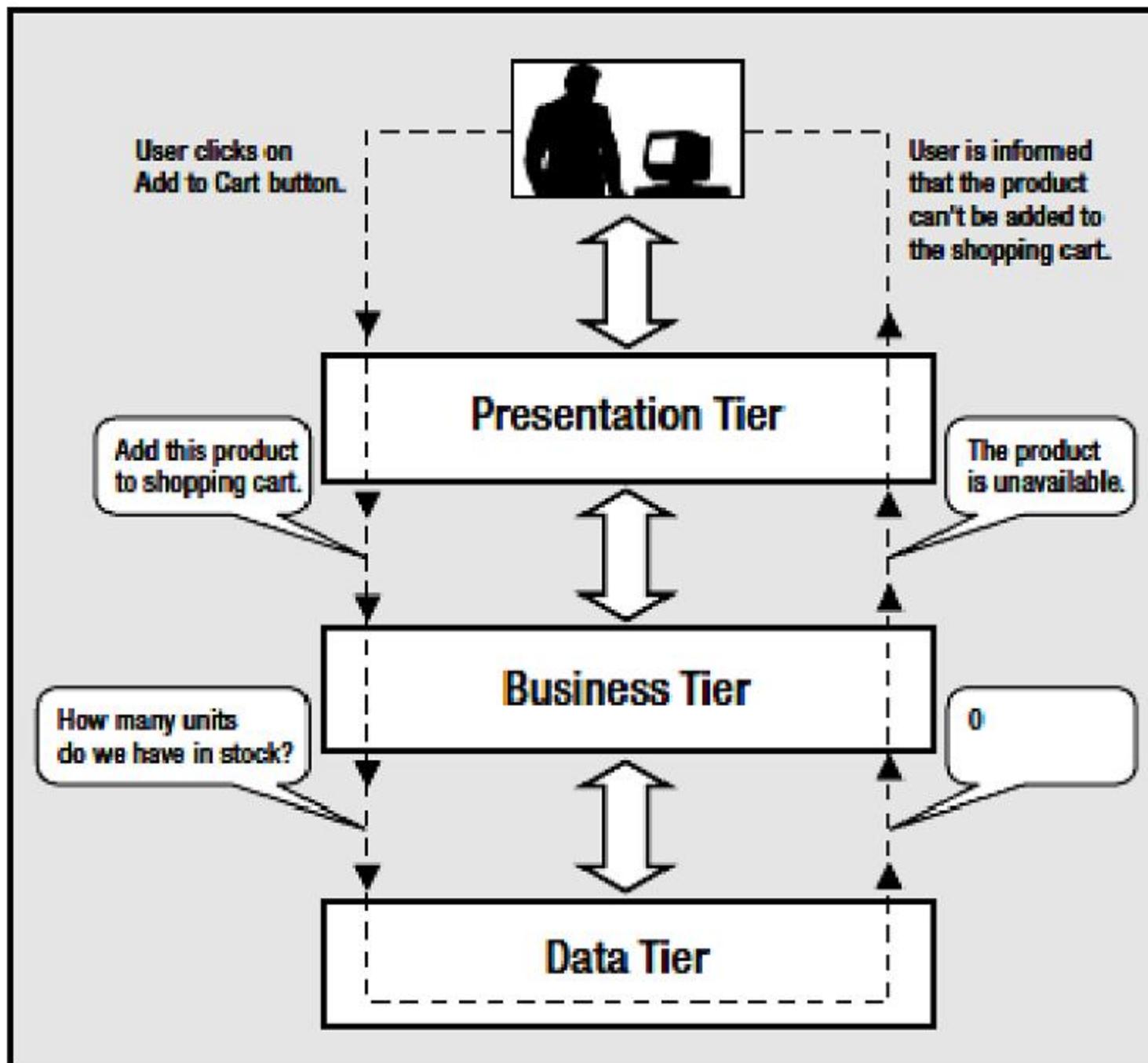
Website sử dụng mô hình ba lớp (three-tier) để xây dựng

Presentation Tier (mức giao diện): chứa các thành phần giao diện người dùng (UI) của website, bao gồm các thành phần quản lý tương tác giữa người dùng và mức xử lý (business).

Bussiness Tier (mức xử lý): nhận yêu cầu từ mức giao diện và trả về kết quả phụ thuộc vào quá trình xử lý. Hầu hết các kết quả xử lý sự kiện xảy ra trên mức giao diện đều được thông qua mức xử lý này, ngoại trừ một số những sự kiện validation). Ở mức này, cần gọi đến mức thứ ba là mức dữ liệu để đáp trả lại các yêu cầu từ mức giao diện.

Data Tier (mức dữ liệu): đóng vai trò lưu trữ dữ liệu và gửi chúng đến mức xử lý mỗi khi được yêu cầu.

Sau đây là hình ảnh diễn tả cho mô hình ba lớp này:



Để thực hiện mô hình ba lớp trên, website đã xây dựng tương ứng với từng chức năng như sau:

STT	Chức năng	Presentation Tier .aspx, .aspx.cs, .ascx, .ascx.cs)	Business Tier .cs, ADO.NET)	Data Tier (Webservices, database)
1	Đăng ký tài khoản	Register.aspx Register	LoginHelper	Database
2	Đăng nhập	Login.aspx Login	LoginHelper	Database
3	Thay đổi thông tin tài khoản	AccountManager.aspx AccountManager	LoginHelper	Database
4	Thay đổi mật khẩu	AccountManager.aspx AccountManager	LoginHelper	Database
5	Tìm kiếm sản phẩm trên website	Search.aspx Search	CatalogAccess	Database
6	Tìm kiếm sản phẩm trên một website khác	AmazonSearch.aspx AmazonSearch eBaySearch.aspx BaySearch	AmazonAccess eBayAccess	WebServices
7	Xem thông tin sản phẩm	Catalog.aspx Catalog Product.aspx Product	CalalogAccess	Database
8	Tìm sản phẩm tương tự trên các website khác với sản phẩm đang xem	Product.aspx Product	AmazonAccess eBayAccess	WebSercices
9	Chuyển đổi giữa một số loại tiền thông dụng	ProductList.ascx ProductList	CurrencyConverter	CurrencyWeb services
10	Thanh toán	Checkout.aspx Checkout	paypalfunctions	PayPal Webservices (Express Checkout)

Chúng ta sẽ đi vào chi tiết ba mức để xây dựng nên website:

MỨC DỮ LIỆU (Data Tier)

Ở mức này, chúng ta sẽ tổ chức dữ liệu cho website, bao gồm hai phần chính: Các bảng dữ liệu và các StoreProcedure.

a. Các bảng dữ liệu

- Website quản lý sản phẩm theo ba cấp: gian hàng, loại mặt hàng và sản phẩm. Trong đó, mỗi gian hàng sẽ có nhiều mặt hàng khác nhau như vậy quan hệ giữa hai bảng tblGianHang và tblLoaiMatHang là 1-∞. Giữa sản phẩm và loại mặt hàng sẽ có quan hệ là ∞-∞ vì sản phẩm này có thể vừa là loại mặt hàng này lại vừa là loại mặt hàng kia. Do đó, cần tạo thêm một bảng tblSanPham_MatHang thể hiện mối quan hệ này là hợp lý.

- Ngoài sản phẩm, chúng ta cần phải quản lý thông tin người dùng để tiện trong việc mua hàng, đặt hàng trên website. Website sẽ lưu trữ các thông tin cơ bản của người dùng (tài khoản, mật khẩu, email, địa chỉ liên hệ, địa chỉ nhận hàng,...)

- Một phần không kém phần quan trọng cần được lưu trữ là đơn đặt hàng. Một đơn đặt hàng có thể có nhiều sản phẩm, và sản phẩm được đặt hàng phải được chứa trong danh mục sản phẩm mà website hiện có. Do đó, mối quan hệ giữa các bảng này được mô hình quan hệ trên là hoàn toàn phù hợp.

- Ngoài ra, do website có sử dụng webservices, cụ thể là tích hợp các webservices của eBay, Amazon, Paypal ... nên một phần dữ liệu sẽ ở dạng tích hợp. Nghĩa là, chúng ta sẽ lấy dữ liệu từ các nhà cung cấp dịch vụ web.

Ở trên đã đưa ra mô hình quan hệ, cũng như một số chú thích về cách tổ chức dữ liệu của website. Sau đây là mô tả chi tiết cho từng bảng dữ liệu.

tblLoaiMatHang

- MaMatHang: mã của mặt hàng, có kiểu dữ liệu là kiểu int và không cho phép NULL, đây là khóa chính của bảng.
- TenMatHang: tên của mặt hàng, có kiểu dữ liệu là kiểu nvarchar(50) và không cho phép NULL.
- MaGianHang: đây là mã gian hàng, có kiểu dữ liệu là kiểu int và không cho phép NULL.
- MoTaMatHang: mô tả chi tiết cho mặt hàng, có kiểu dữ liệu là kiểu nvarchar(1000) và cho phép NULL.

tblGianHang

- MaGianHang: mã của gian hàng, có kiểu dữ liệu là kiểu int và không cho phép NULL, đây là khóa chính của bảng.
- TenGianHang: tên của gian hàng, có kiểu dữ liệu là kiểu nvarchar(50) và không cho phép NULL.
- MoTaGianHang: mô tả chi tiết cho gian hàng, có kiểu dữ liệu nvarchar(1000) và có thể NULL.

tblSanPham MatHang

- MaSanPham: mã của sản phẩm, có kiểu dữ liệu là kiểu int và không cho phép NULL.
- MaMatHang: mã của mặt hàng, có kiểu dữ liệu là kiểu int và không cho phép NULL.

Khóa chính của bảng là MaSanPham, MaMatHang.

tblSanPham

- MaSanPham: mã của sản phẩm, có kiểu dữ liệu là kiểu int và không cho phép NULL, đây là khóa chính của bảng.
- TenSanPham: tên của sản phẩm, có kiểu dữ liệu là kiểu nvarchar(1000) và không cho phép NULL.
- MoTaSanPham: mô tả chi tiết cho sản phẩm, có kiểu dữ liệu là kiểu nvarchar(MAX) và không cho phép NULL.
- Gia: giá tiền của sản phẩm, có kiểu dữ liệu là kiểu money và không cho phép NULL.
- Thumbnail: đường dẫn hình ảnh nhỏ của sản phẩm, có kiểu dữ liệu là kiểu nvarchar(50) và có thể NULL.
- Image: đường dẫn hình ảnh bình thường của sản phẩm, có kiểu dữ liệu là kiểu nvarchar(50) và có thể NULL.
- PromoFront:
- PromoDept:
- TacGia: tác giả của sản phẩm, có kiểu dữ liệu là kiểu nvarchar(1000) và không cho phép NULL.
- NXB: tên nhà xuất bản của sản phẩm, có kiểu dữ liệu là kiểu nvarchar(1000) và không cho phép NULL.

- NgonNgu: tên loại ngôn ngữ, có kiểu dữ liệu là nvarchar(50) và không cho phép NULL.
- Solanmua: số lần mua của sản phẩm, có kiểu dữ liệu là int và không cho phép NULL.
- NgayThem: ngày thêm sản phẩm vào cơ sở dữ liệu, có kiểu dữ liệu là datetime và có thể NULL.
- TenTimKiem: tên sản phẩm dùng để tìm kiếm, có kiểu dữ liệu là nvarchar(500) và có thể NULL.

tblChiTietDonDatHang

- MaDonDatHang: mã của đơn đặt hàng, có kiểu dữ liệu là uniqueidentifier và không cho phép NULL. Đây là khóa chính của bảng.
- MaSanPham: mã của sản phẩm, có kiểu dữ liệu là int và không cho phép NULL. Đây cũng là khóa chính của sản phẩm.
- TenSanPham: tên của sản phẩm, có kiểu dữ liệu là nvarchar(50) và không cho phép NULL.
- SoLuong: số lượng sản phẩm được mua, có kiểu dữ liệu là int và không cho phép NULL.
- DonGia: đơn giá của một sản phẩm, có kiểu dữ liệu là money và không cho phép NULL.
- TongCong: tổng giá tiền của các sản phẩm được mua, có kiểu dữ liệu là money và có thể NULL.

tblDonDatHang

- MaDonDatHang: mã của đơn đặt hàng, có kiểu dữ liệu là uniqueidentifier và không cho phép NULL. Đây là khóa chính của bảng.
- NgayTao: ngày tạo đơn đặt hàng, có kiểu dữ liệu là smalldatetime và không cho phép NULL.
- NgayGiao: ngày giao sản phẩm, có kiểu dữ liệu là smalldatetime và có thể NULL.
- XacNhan: xác nhận đơn đặt hàng, có kiểu dữ liệu là bit và không cho phép NULL.
- HoanTat: hoàn tất đơn đặt hàng, có kiểu dữ liệu là bit và không cho phép NULL.

- Huy: hủy đơn đặt hàng, có kiểu dữ liệu là bit và không cho phép NULL.
- GhiChu: ghi chú cho đơn đặt hàng, có kiểu dữ liệu là nvarchar(1000) và cho phép NULL.
- TenKhachHang: tên của khách hàng đặt mua sản phẩm, có kiểu dữ liệu là nvarchar(256) và không cho phép NULL.
- EmailKhachHang: email của khách hàng, có kiểu dữ liệu là nvarchar(256) và không cho phép NULL.
- TenNguoiTT: tên của người thanh toán, có kiểu dữ liệu là nvarchar(256) và không cho phép NULL.
- DiaChiTT1: địa chỉ thứ nhất của người thanh toán, có kiểu dữ liệu là nvarchar(256) và không cho phép NULL.
- DiaChiTT2: địa chỉ thứ hai của người thanh toán, có kiểu dữ liệu là nvarchar(256) và cho phép NULL.
- DienThoaiTT: điện thoại của người thanh toán, có kiểu dữ liệu là nvarchar(128) và không cho phép NULL.
- TenNguoiNH: tên của người nhận hàng, có kiểu dữ liệu là nvarchar(256) và không cho phép NULL.
- DiaChiNH1: địa chỉ thứ nhất của người nhận, có kiểu dữ liệu là nvarchar(256) và không cho phép NULL.
- DiaChiNH2: địa chỉ thứ hai của người nhận, có kiểu dữ liệu là nvarchar(256) và cho phép NULL.
- DienThoaiNH: điện thoại của người nhận, có kiểu dữ liệu là nvarchar(128) và không cho phép NULL.

tblHoSoTaiKhoan

- MaTaiKhoan: mã tài khoản của khách hàng, có kiểu dữ liệu là uniqueidentifier và không cho phép NULL. Đây là khóa chính của bảng.
- CreditCard: thẻ thanh toán của khách hàng, có kiểu dữ liệu là ntext và cho phép NULL.
- TenNguoiTT: tên của người thanh toán, có kiểu dữ liệu là nvarchar(50) và không cho phép NULL.
- DiaChiTT1: địa chỉ thứ nhất của người thanh toán, có kiểu dữ liệu là nvarchar(256) và không cho phép NULL.

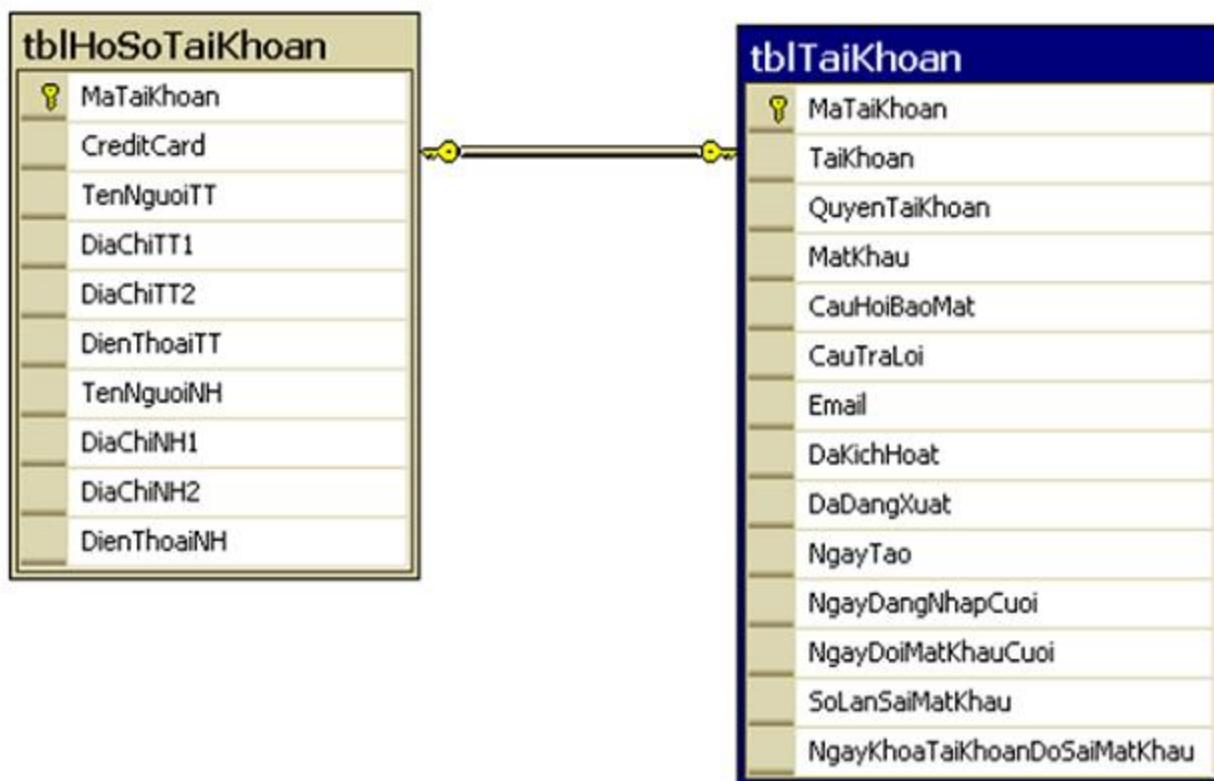
- DiaChiTT2: địa chỉ thứ hai của người thanh toán, có kiểu dữ liệu là nvarchar(256) và cho phép NULL.
- DienThoaiTT: điện thoại của người thanh toán, có kiểu dữ liệu là nvarchar(128) và không cho phép NULL.
- TenNguoiNH: tên của người nhận hàng, có kiểu dữ liệu là nvarchar(256) và không cho phép NULL.
- DiaChiNH1: địa chỉ thứ nhất của người nhận, có kiểu dữ liệu là nvarchar(256) và không cho phép NULL.
- DiaChiNH2: địa chỉ thứ hai của người nhận, có kiểu dữ liệu là nvarchar(256) và cho phép NULL.
- DienThoaiNH: điện thoại của người nhận, có kiểu dữ liệu là nvarchar(128) và không cho phép NULL.

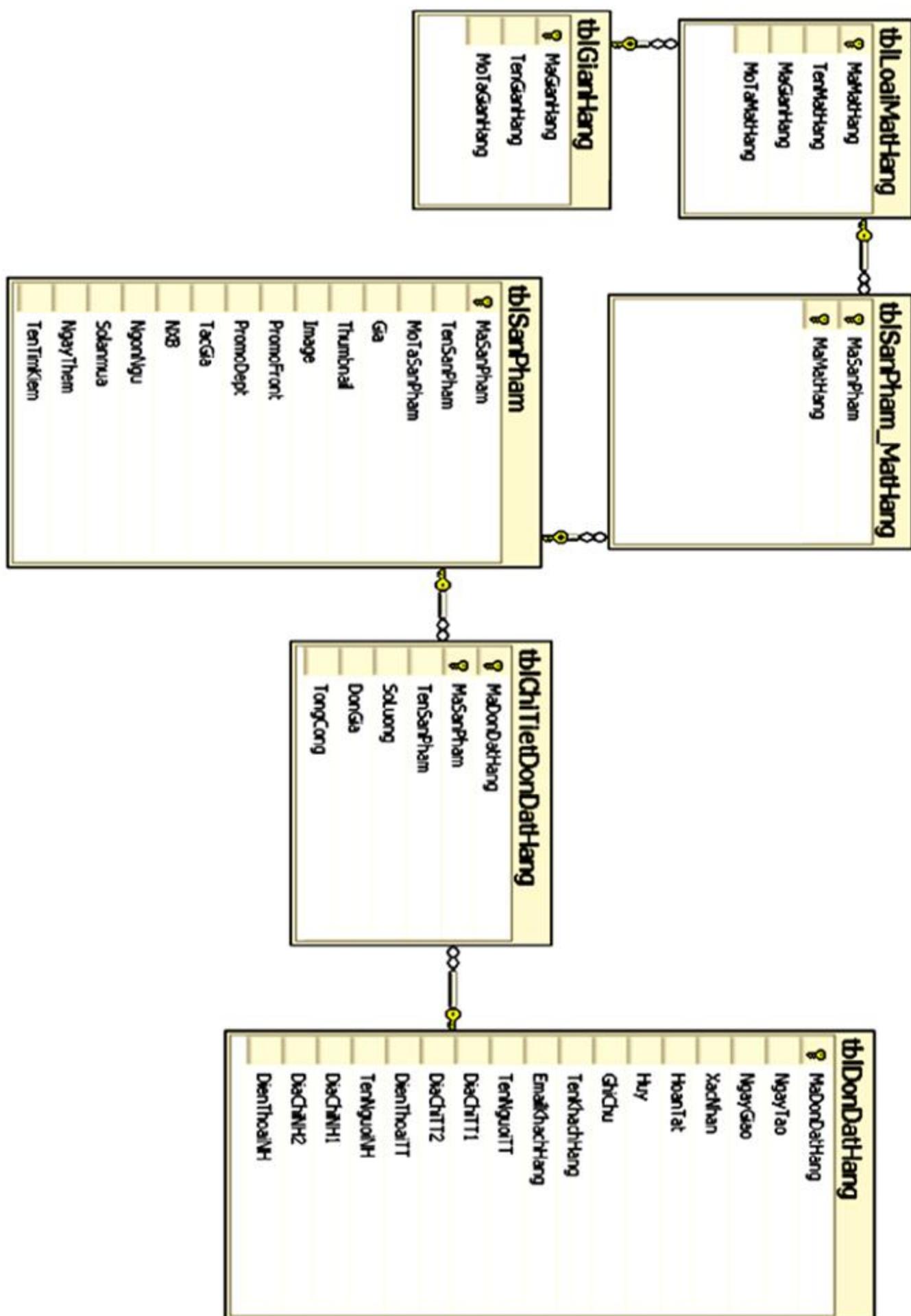
tblTaiKhoan

- MaTaiKhoan: mã tài khoản của khách hàng, có kiểu dữ liệu là uniqueidentifier và không cho phép NULL. Đây là khóa chính của bảng.
- TaiKhoan: tên tài khoản của khách hàng, có kiểu dữ liệu là nvarchar(50) và không cho phép NULL.
- QuyenTaiKhoan: quyền của khách hàng, có kiểu dữ liệu là int và không cho phép NULL.
- MatKhau: mật khẩu của tài khoản, có kiểu dữ liệu là nvarchar(128) và không cho phép NULL.
- CauHoiBaoMat: câu hỏi bảo mật của tài khoản, có kiểu dữ liệu là nvarchar(256) và có thể NULL.
- CauTraLoi: câu trả lời cho câu hỏi bảo mật, có kiểu dữ liệu là nvarchar(128) và có thể NULL.
- Email: email kích hoạt tài khoản, có kiểu dữ liệu là nvarchar(256) và không cho phép NULL.
- DaKichHoat: tài khoản đã kích hoạt, có kiểu dữ liệu là bit và không cho phép NULL.
- DaDangXuat: tài khoản đã đăng xuất, có kiểu dữ liệu là bit và không cho phép NULL.
- NgayTao: ngày tài khoản được tạo, có kiểu dữ liệu là datetime và không cho phép NULL.

- NgayDangNhapCuoi: ngày tài khoản đăng nhập lần cuối, có kiểu dữ liệu là datetime và cho phép NULL.
- NgayDoiMatKhauCuoi: ngày tài khoản đổi mật khẩu, có kiểu dữ liệu là datetime và cho phép NULL.
- SoLanSaiMatKhau: số lần sai mật khẩu khi tài khoản đăng nhập, có kiểu dữ liệu là int và không cho phép NULL.
- NgayKhoaTaiKhoanDoSaiMatKhau: ngày tài khoản bị khóa do đăng nhập sai mật khẩu vượt quá số lần cho phép, có kiểu dữ liệu là datetime và không cho phép NULL.

Sơ đồ quan niệm





Mỗi liên hệ giữa các bảng

b. Store Procedure

Store Procedure (thủ tục lưu trữ): là các câu lệnh SQL được thực thi trực tiếp trên máy chứa SQL Database. Store Procedure cũng giống như các hàm hoặc các thủ tục trong các ngôn ngữ lập trình, có nghĩa là nó có thể nhận vào các tham số và trả về kết quả cho chương trình gọi nó.

Việc dùng Store Procedure giúp cho việc thực thi các câu lệnh nhanh hơn nhiều so với việc thực thi từ Server Code, nghĩa là nó sẽ không phải biên dịch lại mà chạy trực tiếp mã máy đã được biên dịch. Đồng thời làm giảm đáng kể lưu lượng lưu thông qua mạng bởi nó sẽ không phải tạo lại kết nối.

Sau đây là một số Store Procedure quan trọng sử dụng cho website:

Tên	LayCacGianHang	
Chức năng	Lấy danh sách các gian hàng của website	
Mô tả	Đây là Store Procedure đơn giản hỗ trợ xây dựng menu website	
Tham số (Không có)		
Tên tham số	Kiểu	Ý nghĩa

Tên	ThêmGianHang	
Chức năng	Thêm vào một gian hàng cho website	
Mô tả	Vì khóa của gian hàng là kiểu int tự tăng, nên chỉ cần thêm vào hai tham số cho gian hàng	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
@TenGianHang	nvarchar(50)	Tên của gian hàng
@MoTaGianHang	nvarchar(50)	Mô tả về gian hàng

Tên	XoaGianHang	
Chức năng	Xóa một gian hàng cho website	
Mô tả		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
@MaGianHang	Int	Khóa của gian hàng

Tên	ThemMatHang	
Chức năng	Thêm một loại mặt hàng vào trong một gian hàng	
Mô tả		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
@MaGianHang	Int	Khóa của gian hàng
@TenMatHang	nvarchar(50)	Tên của loại mặt hàng
@MoTaMatHang	Nvarchar(1000)	Mô tả về loại mặt hàng

Tên	ThemSanPham	
Chức năng	Thêm một sản phẩm vào website	
Mô tả		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
@MaMatHang	Int	Khóa của gian hàng
@TenSanPham	nvarchar(1000)	Tên của sản phẩm
@MoTaSanPham	nvarchar(max)	Mô tả về sản phẩm
@Gia	Money	Giá của sản phẩm
@Thumbnail	Nvarchar(50)	Lưu đường dẫn của file thumbnail sản phẩm
@Image	Nvarchar(50)	Lưu đường dẫn của file hình sản phẩm
@PromoFront	Bit	Đánh dấu sản phẩm có được hiển thị ngay trang chủ hay không
@PromoDept	Bit	Đánh dấu sản phẩm có được hiển thị tại trang đầu của mỗi gian hàng hay không
@TacGia	Nvarchar(1000)	Tên tác giả
@NXB	Nvarchar(1000)	Tên nhà xuất bản
@NgonNgu	Nvarchar(50)	Ngôn ngữ được viết trên sản phẩm
@NgayThem	Datetime	Ngày thêm sản phẩm vào website

Tên	XoaSanPham	
Chức năng	Xóa một sản phẩm trong website	
Mô tả		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
@MaSanPham	Int	Khóa của sản phẩm

Tên	LayCacLoaiMatHangTrongGianHang	
Chức năng	Lấy danh sách các loại mặt hàng phụ thuộc vào gian hàng được chọn	
Mô tả	Danh sách trả về được dùng để xây dựng sub menu	
Tham số (Không có)		
Tên tham số	Kiểu	Ý nghĩa
@MaGianHang	Int	Dựa vào mã gian hàng này để lấy danh sách các loại mặt hàng tương ứng có trong gian hàng đó

Tên	LaySanPhamTheoLoaiMatHang	
Chức năng	Lấy danh sách các sản phẩm theo loại mặt hàng được chọn	
Mô tả	Danh sách sản phẩm trả về sẽ được hiển thị mỗi khi người dùng chọn một loại mặt hàng nào đó. Thủ tục này còn có tác dụng hỗ trợ phân trang bằng cách tính toán thông qua các dữ liệu đầu vào, và trả về danh sách các sản phẩm phù hợp.	
Tham số (Không có)		

Tên tham số	Kiểu	Ý nghĩa
@MaMatHang	Int	Dựa vào mã mặt hàng này để lấy danh sách các sản phẩm tương ứng
@DoDaiMoTaSanPham	Int	Quy định độ dài mô tả sản phẩm khi được hiển thị lên website

@SoTrang	Int	Tham số này hỗ trợ cho chức năng phân trang, trang cần điều hướng tới sẽ được truyền vào thông qua tham số này
@SoSanPhamTrenTrang	Int	Quy định số sản phẩm có trên một trang
@SoLuongSanPham	Int (Output)	Tổng số lượng sản phẩm có trong loại mặt hàng được chỉ định

Tên	LaySanPhamChoTrangChinhTheoGianHang	
Chức năng	Lấy danh sách các sản phẩm sẽ được hiển thị lên trang chính của mỗi gian hàng khi được chọn	
Mô tả	Danh sách sản phẩm trả về sẽ được hiển thị mỗi khi người dùng chọn một loại gian hàng (menu) nào đó. Thủ tục này còn có tác dụng hỗ trợ phân trang bằng cách tính toán thông qua các dữ liệu đầu vào, và trả về danh sách các sản phẩm phù hợp.	
Tham số (Không có)		
Tên tham số	Kiểu	Ý nghĩa
@MaMatHang	Int	Dựa vào mã mặt hàng này để lấy danh sách các sản phẩm tương ứng
@DoDaiMoTaSanPham	Int	Quy định độ dài mô tả sản phẩm khi được hiển thị lên website
@SoTrang	Int	Tham số này hỗ trợ cho chức năng phân trang, trang cần điều hướng tới sẽ được truyền vào thông qua tham số này
@SoSanPhamTrenTrang	Int	Quy định số sản phẩm có trên một trang
@SoLuongSanPham	Int (Output)	Tổng số lượng sản phẩm có trong loại mặt hàng được chỉ định

Tên	LaySanPhamChoTrangChinh	
Chức năng	Lấy danh sách các sản phẩm sẽ được hiển thị lên trang chủ của website	
Mô tả	Danh sách sản phẩm trả về sẽ được hiển thị mỗi khi người dùng vào trang chủ của website. Thủ tục này còn có tác dụng hỗ trợ phân trang bằng cách tính toán thông qua các dữ liệu đầu vào và trả về danh sách các sản phẩm phù hợp	
Tham số (Không có)		
Tên tham số	Kiểu	Ý nghĩa
@MaMatHang	Int	Dựa vào mã mặt hàng này để lấy danh sách các sản phẩm tương ứng
@DoDaiMoTaSanPham	Int	Quy định độ dài mô tả sản phẩm khi được hiển thị lên website
@SoTrang	Int	Tham số này hỗ trợ cho chức năng phân trang, trang cần điều hướng tới sẽ được truyền vào thông qua tham số này
@SoSanPhamTrenTrang	Int	Quy định số sản phẩm có trên một trang
@SoLuongSanPham	Int (Output)	Tổng số lượng sản phẩm có trong loại mặt hàng được chỉ định

Ở trên là một số Store Procedure được dùng vào việc lấy danh sách các gian hàng, mặt hàng cũng như sản phẩm cho website tùy vào từng trường hợp điều hướng. Ngoài ra, còn một số thủ tục khác có tác dụng và tham số tương tự: LaySanPhamMoiNhat, LaySanPhamBanChay,...

Các thủ tục liên quan đến tài khoản người dùng

Tên	TaoTaiKhoan	
Chức năng	Tạo một tài khoản người dùng trên website	
Mô tả		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
@MaTaiKhoan	Uniqueidentifier	Khóa

@TaiKhoan	Nvarchar(50)	Tài khoản để đăng nhập
@MatKhau	Nvarchar(128)	Mật khẩu
@Email	Nvarchar(256)	Địa chỉ email của khách hàng
@TenNguoiTT	Nvarchar(50)	Tên người thanh toán
@DiaChiTT1	Nvarchar(256)	Địa chỉ thanh toán 1
@DiaChiTT2	Nvarchar(256)	Địa chỉ thanh toán 2
@DienThoaiTT	Nvarchar(128)	Số điện thoại liên lạc thanh toán
@TenNguoiNH1	Nvarchar(50)	Tên người nhận hàng
@DiaChiNH1	Nvarchar(256)	Địa chỉ nhận hàng 1
@DiaChiNH2	Nvarchar(256)	Địa chỉ nhận hàng 2
@DienThoaiNH	Nvarchar(128)	Số điện thoại liên lạc nhận hàng

Tên	LayThongTinTaiKhoan	
Chức năng	Lấy tất cả thông tin liên quan đến tài khoản được chỉ định	
Mô tả		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
@TaiKhoan	Nvarchar(50)	Tài khoản của người dùng

Các Store Procedure liên quan đến đơn hàng: Website chỉ cho phép người dùng có tài khoản mới được thanh toán giờ hàng. Trường hợp người dùng chưa có tài khoản muốn thanh toán thì sẽ tạo tài khoản theo từng bước thanh toán.

Tên	ThemDonDatHang	
Chức năng	Thêm một đơn hàng khi khách hàng bắt đầu quy trình thanh toán	
Mô tả		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
@MaDonDatHang	Uniqueidentifier	Khóa của đơn đặt hàng

@TaiKhoan	Nvarchar(50)	Tài khoản người đặt hàng
@NgayGiao	Smalldatetime	Ngày giao hàng
@LoaiVanChuyen	Int	(trong ngày, 3 ngày, 1 tuần ...)
@LoaiTien	Nvarchar(50)	Loại tiền sử dụng lúc thanh toán
@TyGia	Float	Tỷ giá lúc thanh toán

Tên	ThemSanPhamVaoChiTietDonDatHang	
Chức năng	Thêm một sản phẩm vào chi tiết đơn đặt hàng	
Mô tả	Một đơn đặt hàng có thể có nhiều sản phẩm, do đó thủ tục này sẽ đóng vai trò thêm vào từng sản phẩm tương ứng với đơn hàng đang được xử lý	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
@MaDonDatHang	Uniqueidentifier	Khóa của đơn đặt hàng
@MaSanPham	Int	Mã sản phẩm được thêm vào đơn đặt hàng
@TenSanPham	Nvarchar(50)	Tên sản phẩm
@SoLuong	Int	Số lượng sản phẩm được mua
@DonGia	Money	Đơn giá sản phẩm

Tên	LayThongTinDonDatHang	
Chức năng	Lấy các thông tin cần thiết của một đơn đặt hàng bao gồm tổng tiền của đơn hàng đó	
Mô tả		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
@MaDonDatHang	Uniqueidentifier	Khóa của đơn đặt hàng

Tên	LaySanPhamTrongDonDatHang	
Chức năng	Lấy danh sách các sản phẩm trong đơn đặt hàng (mã sản phẩm, tên sản phẩm, số lượng, đơn giá)	
Mô tả		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
@MaDonDatHang	Uniqueidentifier	Khóa của đơn đặt hàng

Tên	XoaDonDatHang	
Chức năng	Xóa đơn đặt hàng	
Mô tả		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
@MaDonDatHang	Uniqueidentifier	Khóa của đơn đặt hàng

Tên	XacNhanDonDatHang	
Chức năng	Đánh dấu xác nhận đơn hàng đã được duyệt	
Mô tả		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
@MaDonDatHang	Uniqueidentifier	Khóa của đơn đặt hàng

Tên	HoanTatDonDatHang	
Chức năng	Xác nhận đơn hàng đã hoàn tất	
Mô tả		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
@MaDonDatHang	Uniqueidentifier	Khóa của đơn đặt hàng

Tên	HuyDonDatHang	
Chức năng	Hủy đơn hàng	
Mô tả		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
@MaDonDatHang	Uniqueidentifier	Khóa của đơn đặt hàng

Tên	LayCacDonHangTrongKhoangThoiGian	
Chức năng	Lấy danh sách các đơn hàng trong một khoảng thời gian chỉ định	
Mô tả		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
@NgayBatDau	Smalldatetime	Thời gian bắt đầu
@NgayKetThuc	Smalldatetime	Thời gian kết thúc

Ngoài các thủ tục lưu trữ được liệt kê ở trên, website còn sử dụng một số store procedure khác, xin tham khảo chi tiết trong phần cơ sở dữ liệu của website.

Chương 16

THIẾT KẾ WEBSITE BÁN HÀNG

– MỨC XỬ LÝ (BUSSINESS TIER)

Ở mức dữ liệu, chúng ta đã xây dựng được cơ sở dữ liệu và các store procedure cho website bán sách qua mạng. Trong mức xử lý này, chúng ta sẽ xây dựng các lớp C#, thực hiện việc lấy dữ liệu, xử lý dữ liệu từ database sau đó trả kết quả về cho mức trên (Presentation Tier) mỗi khi được yêu cầu.

Các lớp C# chính được sử dụng:

Tên lớp	File (.cs)	Chức năng
GenericDataAccess	GenericDataAccess.cs	Có tác dụng thực thi các chức năng cơ bản sẽ được sử dụng lại nhiều lần mỗi khi cần truy xuất đến database.
OnlineStoreConfiguration	OnlineStoreConfiguration.cs	Lớp này giúp lấy được những thông số cấu hình của website được lưu trong file web.config
Utilities	Utilities.cs	Chứa các hàm xử lý thông thường như gửi email, log lỗi và nhiều chức năng khác sẽ được sử dụng nhiều trong website
CatalogAccess	CatalogAccess.cs	Đây là lớp quan trọng, chứa tất cả các phương thức tương tác với sản phẩm, như lấy danh sách các sản phẩm, cập nhật thông tin một sản phẩm ...v.v
LinkBuilder	LinkBuilder.cs	Có tác dụng tạo ra các đường link, hỗ trợ trong việc điều hướng các trang trong website
LoginHelper	LoginHelper.cs	Lớp này được xây dựng để quản lý người dùng, truy xuất, chỉnh sửa thông tin... người dùng. Quản lý vấn đề đăng nhập và đăng xuất của user

ShoppingCartAccess	ShoppingCartAccess.cs	Chứa các chức năng liên quan đến giỏ hàng, cập nhật, thêm, xóa, sửa giỏ hàng,...
OrdersAccess	OrdersAccess.cs	Chứa các phương thức thao tác với đơn đặt hàng trong database (thêm, cập nhật, xóa,...)
AmazonAccess	AmazonAccess.cs	Lớp này cung cấp các phương thức thao tác với webservices của Amazon
eBayAccess	eBayAccess.cs	Lớp này cung cấp các phương thức thao tác với webservices của eBay

Các lớp trên được cài đặt theo ngôn ngữ C# chi tiết chức năng, cài đặt như sau:

1. **OnlineStoreConfigurations:** Lớp tĩnh (static class) dùng để lấy cấu hình website từ file web.config.

```

using System.Configuration;
using System;
public static class OnlineStoreConfigurations
{
    private static string dbConnectionString;
    private static string dbProviderName;
    private readonly static int productsPerPage;
    private readonly static int productDescriptionLength;
    private readonly static string siteName;

    static OnlineStoreConfigurations()
    {
        dbConnectionString =
ConfigurationManager.ConnectionStrings["OnlineStoreConnection"].ConnectionString;
        dbProviderName =
ConfigurationManager.ConnectionStrings["OnlineStoreConnection"].ProviderName;
        productsPerPage =
System.Int32.Parse(ConfigurationManager.AppSettings["ProductsPerPage"]);
        productDescriptionLength =
System.Int32.Parse(ConfigurationManager.AppSettings["ProductDescriptionLength"]);
    }
}

```

```
siteName = ConfigurationManager.AppSettings["SiteName"];
}

public static string DbConnectionString
{
    get
    {
        return dbConnectionString;
    }
}

public static string DbProviderName
{
    get
    {
        return dbProviderName;
    }
}

public static string MailServer
{
    get
    {
        return ConfigurationManager.AppSettings["MailServer"];
    }
}

public static string MailUsername
{
    get
    {
        return ConfigurationManager.AppSettings["MailUsername"];
    }
}

public static string MailPassword
{
    get
```

```
        {
            return ConfigurationManager.AppSettings["MailPassword"];
        }
    }

    public static string MailFrom
    {
        get
        {
            return ConfigurationManager.AppSettings["MailFrom"];
        }
    }

    public static bool EnableErrorLogEmail
    {
        get
        {
            return
bool.Parse(ConfigurationManager.AppSettings["EnableErrorLogEmail"]);
        }
    }

    public static string ErrorLogEmail
    {
        get
        {
            return ConfigurationManager.AppSettings["ErrorLogEmail"];
        }
    }

    public static int ProductsPerPage
    {
        get
        {
            return productsPerPage;
        }
    }

    public static int ProductDescriptionLength
```

```
{  
    get  
    {  
        return productDescriptionLength;  
    }  
}  
  
public static string SiteName  
{  
    get  
    {  
        return siteName;  
    }  
}  
  
public static int CartPersistDays  
{  
    get  
    {  
        return Int32.Parse(ConfigurationManager.AppSettings["CartPersistDays"]);  
    }  
}  
  
public static string PaypalUrl  
{  
    get  
    {  
        return ConfigurationManager.AppSettings["PaypalUrl"];  
    }  
}  
  
public static string PaypalEmail  
{  
    get  
    {  
        return ConfigurationManager.AppSettings["PaypalEmail"];  
    }  
}
```

```

public static string PaypalCurrency
{
    get
    {
        return ConfigurationManager.AppSettings["PaypalCurrency"];
    }
}

public static string PaypalReturnUrl
{
    get
    {
        return ConfigurationManager.AppSettings["PaypalReturnUrl"];
    }
}

public static string PaypalCancelUrl
{
    get
    {
        return ConfigurationManager.AppSettings["PaypalCancelUrl"];
    }
}

```

Đây là một lớp đơn giản, bao gồm các Properties phục vụ cho việc lấy các thông số cấu hình cho website bán sách.

2. GenericAccess: Có tác dụng thực thi các chức năng cơ bản sẽ được sử dụng lại nhiều lần mỗi khi cần truy xuất đến database

using System;

using System.Data;

using System.Data.Common;

```

/// <summary>
/// Summary description for GenericDataAccess
/// </summary>
public static class GenericDataAccess
{

```

```
static GenericDataAccess()
{
    //
    // TODO: Add constructor logic here
    //
}

public static DataTable ExecuteSelectCommand(DbCommand command)
{
    DataTable table;
    try
    {
        command.Connection.Open();
        DbDataReader reader = command.ExecuteReader();
        table = new DataTable();
        table.Load(reader);
        reader.Close();
    }
    catch (Exception ex)
    {
        Utilities.LogError(ex);
        throw;
    }
    finally
    {
        command.Connection.Close();
    }
    return table;
}

public static DbCommand CreateCommand()
{
    string dataProviderName = OnlineStoreConfigurations.DbProviderName;
    string connectionString = OnlineStoreConfigurations.DbConnectionString;
    DbProviderFactory factory = DbProviderFactories.GetFactory(dataProviderName);
    DbConnection conn = factory.CreateConnection();
    conn.ConnectionString = connectionString;
}
```

```
        DbCommand comm = conn.CreateCommand();
        comm.CommandType = CommandType.StoredProcedure;
        return comm;
    }

    public static int ExecuteNonQuery(DbCommand command)
    {
        int affectedRows = -1;
        try
        {
            command.Connection.Open();
            affectedRows = command.ExecuteNonQuery();
        }
        catch (Exception e)
        {
            Utilities.LogError(e);
            throw;
        }
        finally
        {
            command.Connection.Close();
        }
        return affectedRows;
    }

    public static string ExecuteScalar(DbCommand command)
    {
        string value = "";
        try
        {
            command.Connection.Open();
            value = command.ExecuteScalar().ToString();
        }
        catch (Exception ex)
        {
            Utilities.LogError(ex);
            throw;
        }
    }
}
```

```

        }
        finally
        {
            command.Connection.Close();
        }
        return value;
    }
}

```

Chức năng chính của lớp này là tạo nên những đối tượng ADO.NET (DbConnection, DbCommand,... sử dụng trong namespace System.Data.Common) nhằm mục đích tránh sử dụng các lớp truy xuất dữ liệu cụ thể nào như SqlConnection, SqlCommand hoặc thay vào đó ứng dụng của chúng ta sẽ quyết định provider nào được sử dụng ngay tại lúc chạy (runtime) dựa vào chuỗi kết nối được cung cấp (connection string).

Phương thức quan trọng quyết định ưu điểm nói trên là CreateCommand()

```
public static DbCommand CreateCommand()
```

```

{
    string dataProviderName = OnlineStoreConfigurations.DbProviderName;
    string connectionString = OnlineStoreConfigurations.DbConnectionString;
    DbProviderFactory factory = DbProviderFactories.GetFactory(dataProviderName);
    DbConnection conn = factory.CreateConnection();
    conn.ConnectionString = connectionString;
    DbCommand comm = conn.CreateCommand();
    comm.CommandType = CommandType.StoredProcedure;
    return comm;
}
```

Đầu tiên, chúng ta sẽ tạo một đối tượng DbProviderFactory để lấy provider cụ thể. Sau đó tạo một đối tượng DbConnection (tương ứng với SqlConnection nếu provider là System.Data.SqlClient) thông qua factory method của đối tượng DbProvider vừa được tạo ra. Cuối cùng là tạo ra đối tượng DbCommand (tương tự như SqlCommand)

Các phương thức còn lại của lớp này: ExecuteSelectCommand, ExecuteNonQuery, ExecuteScalar sẽ nhận một tham số truyền vào là một đối tượng DbCommand và thực thi chức năng tương ứng.

Việc sử dụng lớp này như thế nào sẽ được đề cập đến trong các phần sau:

3. CatalogAccess

Như đã trình bày ở trên, chúng ta đã tạo một lớp GenericAccess có tác dụng tạo ra các đối tượng DbCommand và thực thi các chức năng tùy thuộc vào đối số đầu vào. Ở lớp CatalogAccess này, sẽ mô tả cách sử dụng các chức năng đã được cài đặt trong lớp GenericAccess thông qua một vài phương thức sau đây. Để biết được cách cài đặt các phương thức khác, xin tham khảo source code.

```
public static DataTable GetProductsInCategory(string categoryId, string pageNumber,
out int howManyPages)

{
    DbCommand comm = GenericDataAccess.CreateCommand();
    comm.CommandText = "LaySanPhamTheoLoaiMatHang";

    DbParameter param = comm.CreateParameter();
    param.ParameterName = "@MaMatHang";
    param.Value = categoryId;
    param.DbType = DbType.Int32;
    comm.Parameters.Add(param);

    param = comm.CreateParameter();
    param.ParameterName = "@DoDaiMoTaSanPham";
    param.Value = OnlineStoreConfigurations.ProductDescriptionLength;
    param.DbType = DbType.Int32;
    comm.Parameters.Add(param);

    param = comm.CreateParameter();
    param.ParameterName = "@SoTrang";

    param.Value = pageNumber;
    param.DbType = DbType.Int32;
    comm.Parameters.Add(param);

    param = comm.CreateParameter();
    param.ParameterName = "@SoSanPhamTrenTrang";
    param.Value = OnlineStoreConfigurations.ProductsPerPage;
    param.DbType = DbType.Int32;
    comm.Parameters.Add(param);
}
```

```

param = comm.CreateParameter();
param.ParameterName = "@SoLuongSanPham";
param.Direction = ParameterDirection.Output;
param.DbType = DbType.Int32;
comm.Parameters.Add(param);

DataTable table = GenericDataAccess.ExecuteSelectCommand(comm);
int howManyProducts =
Int32.Parse(comm.Parameters["@SoLuongSanPham"].Value.ToString());
howManyPages = (int)Math.Ceiling((double)howManyProducts /
(double)OnlineStoreConfigurations.ProductsPerPage);
return table;
}

```

Phương thức này đảm nhận công việc lấy về danh sách sản phẩm nằm trong một loại mặt hàng cụ thể được truyền vào thông qua tham số categoryId, số trang được lấy pageNumber, và một tham số output cho biết tổng số trang sản phẩm có trong loại mặt hàng.

Chúng ta sẽ tạo một đối tượng DbCommand thông qua phương thức CreateCommand() của lớp GenericAccess vừa được tạo ở trên. Tiếp theo đó cần tạo các tham số SqlParameter tương ứng với Store Procedure tương ứng. Trong phương thức này sẽ gọi thực thi thủ tục LaySanPhamTheoLoaiMatHang đã được xây dựng trong phần data tier. Mỗi tham số sẽ truyền vào tên tham số, kiểu giá trị, kiểu tham số input (mặc định) hoặc output.

Danh sách các phương thức cơ bản có trong lớp CatalogAccess:

Tên	GetDePartments()	
Kiểu trả về	DataTable	
Chức năng	Lấy về danh sách các gian hàng	
Store Procedure sử dụng	LayCacGianHang	
Tham số (không có)		
Tên tham số	Kiểu	Ý nghĩa

Tên	GetDepartmentDetails()	
Kiểu trả về	DepartmentDetails	

Chức năng	Lấy chi tiết một gian hàng cụ thể dựa vào mã gian hàng	
Store Procedure sử dụng	LayChiTietGianHang	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
departmentId	string	Mã gian hàng

Tên	GetCategoryDetails()	
Kiểu trả về	CategoryDetails	
Chức năng	Lấy chi tiết một loại mặt hàng cụ thể dựa vào mã loại mặt hàng	
Store Procedure sử dụng	LayChiTietGianHang	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
categoryId	string	Mã loại mặt hàng

Tên	GetProductDetails()	
Kiểu trả về	ProductDetails	
Chức năng	Lấy chi tiết một sản phẩm cụ thể dựa vào mã sản phẩm	
Store Procedure sử dụng	LayChiTietSanPham	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
productId	string	Mã gian hàng

Tên	GetCategoriesInDepartment()	
Kiểu trả về	DataTable	
Chức năng	Lấy danh sách loại mặt hàng trong một gian hàng cụ thể	

Store Procedure sử dụng	LayCacLoaiMatHangTrongGianHang	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
departmentId	string	Mã gian hàng

Tên	GetCategoriesOnFrontPromo()	
Kiểu trả về	DataTable	
Chức năng	Lấy danh sách sản phẩm hiển thị trên trang chủ	
Store Procedure sử dụng	LaySanPhamChoTrangChinh	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
pageNumber	string	Số trang cần điều hướng đến
howManyPages (out)	Int	Số trang được trả về

Tên	GetProductsInCategory ()	
Kiểu trả về	DataTable	
Chức năng	Lấy danh sách sản phẩm thuộc vào mã loại mặt hàng nào đó	
Store Procedure sử dụng	LaySanPhamTheoLoaiMatHang	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
categoryId	string	Mã loại mặt hàng
pageNumber	string	Số trang cần điều hướng tới
howManyPages (out)	Int	Tổng số trang có được

Các phương thức sử dụng cho việc quản lý sản phẩm website.

Tên	UpdateDepartment()	
Kiểu trả về	Bool	
Chức năng	Cập nhật thông tin của một gian hàng	
Store Procedure sử dụng	CapNhatGianHang	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
id	string	Mã gian hàng
name	string	Tên gian hàng
description	string	Mô tả về gian hàng

Tên	DeleteDepartment()	
Kiểu trả về	Bool	
Chức năng	Xóa một gian hàng	
Store Procedure sử dụng	XoaGianHang	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
id	string	Mã gian hàng

Tên	AddDepartment()	
Kiểu trả về	Bool	
Chức năng	Thêm một gian hàng mới	
Store Procedure sử dụng	ThemGianHang	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
name	string	Tên gian hàng
description	string	Mô tả về gian hàng

Tên	CreateCategory()	
Kiểu trả về	Bool	
Chức năng	Thêm một loại mặt hàng mới	
Store Procedure sử dụng	ThemMatHang	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
departmentId	string	Mã gian hàng
name	string	Tên loại mặt hàng
description	string	Mô tả về loại mặt hàng

Tên	UpdateCategory()	
Kiểu trả về	Bool	
Chức năng	Cập nhật thông tin một loại mặt hàng	
Store Procedure sử dụng	CapNhatMatHang	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
Id	string	Mã loại mặt hàng
name	string	Tên loại mặt hàng
description	string	Mô tả về loại mặt hàng

Tên	DeleteCategory()	
Kiểu trả về	Bool	
Chức năng	Xóa một loại mặt hàng	
Store Procedure sử dụng	XoaMatHang	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
id	string	Mã loại mặt hàng

Tên	CreateProduct()	
Kiểu trả về	Bool	
Chức năng	Tạo một sản phẩm mới	
Store Procedure sử dụng	ThemSanPham	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
maMatHang	string	Mã loại mặt hàng
tenSanPham	string	Tên sản phẩm
tacgia	string	Tác giả
nxb	string	Nhà xuất bản
moTaSanPham	string	Mô tả về sản phẩm
gia	string	Giá của sản phẩm
thumbnail	string	Đường dẫn chứa file thumbnail
image	string	Đường dẫn chứa file hình ảnh sản phẩm
promoFront	string	(“True”, hoặc “False) chỉ định sản phẩm có được hiển thị trên trang chủ hay không
promoDept	string	(“True”, hoặc “False) chỉ định sản phẩm có được hiển thị trên đầu mỗi gian hàng hay không
ngaythem	string	Ngày thêm sản phẩm

Tên	UpdateProduct()
Kiểu trả về	Bool
Chức năng	Cập nhật thông tin một sản phẩm
Store Procedure sử dụng	CapNhatSanPham

Tham số

Tên tham số	Kiểu	Ý nghĩa
maSanPham	string	Mã sản phẩm
tenSanPham	string	Tên sản phẩm
tacgia	string	Tác giả
nxb	string	Nhà xuất bản
moTaSanPham	string	Mô tả về sản phẩm
gia	string	Giá của sản phẩm
thumbnail	string	Đường dẫn chứa file thumbnail
image	string	Đường dẫn chứa file hình ảnh sản phẩm
promoFront	string	(“True”, hoặc “False) chỉ định sản phẩm có được hiển thị trên trang chủ hay không
promoDept	string	(“True”, hoặc “False) chỉ định sản phẩm có được hiển thị trên đầu mỗi gian hàng hay không
ngaythem	string	Ngày thêm sản phẩm

Tên	DeleteProduct()
Kiểu trả về	Bool
Chức năng	Xóa một sản phẩm
Store Procedure sử dụng	XoaSanPham

Tham số

Tên tham số	Kiểu	Ý nghĩa
maSanPham	string	Mã sản phẩm cần xóa

Tên	AssignProductToCategory()	
Kiểu trả về	Bool	
Chức năng	Gán một sản phẩm vào một loại mặt hàng	
Store Procedure sử dụng	ChiDinhLoaiMatHangChoSanPham	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
maSanPham	string	Mã sản phẩm
maMatHang	string	Mã loại mặt hàng

Tên	MoveProductToCategory()	
Kiểu trả về	Bool	
Chức năng	Đổi loại mặt hàng cho sản phẩm	
Store Procedure sử dụng	DoiLoaiMatHangChoSanPham	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
maSanPham	string	Mã sản phẩm
maMatHangCu	string	Mã loại mặt hàng cũ
maMatHangMoi	string	Mã loại mặt hàng mới

Tên	RemoveProductFromCategory()	
Kiểu trả về	Bool	
Chức năng	Xóa sản phẩm khỏi loại mặt hàng	
Store Procedure sử dụng	XoaSanPhamTrongLoaiMatHang	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
maSanPham	string	Mã sản phẩm
maMatHang	string	Mã loại mặt hàng

4. LoginHelper

Tác dụng của lớp này dùng để xử lý các sự kiện xảy ra liên quan đến user như đăng nhập, tạo tài khoản, chỉnh sửa thông tin tài khoản,...

Sau đây xin được đề cập đến một số phương thức cơ bản được xây dựng trong lớp này:

Tên	Login	
Kiểu trả về	Bool	
Chức năng	Được sử dụng mỗi khi user đăng nhập vào hệ thống từ giao diện web. Trả về True nếu đăng nhập thành công và False nếu thất bại	
Store Procedure sử dụng		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
acc	string	Tài khoản
pass	string	Mật khẩu
authorityNo	string (out)	Quyền người dung (user hay admin)

Tên	CheckAvailbleAccount	
Kiểu trả về	Bool	
Chức năng	Kiểm tra tài khoản có thể được sử dụng hay không trong quá trình đăng ký một tài khoản mới	
Store Procedure sử dụng		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
acc	string	Tài khoản

Tên	CheckAvailableEmail	
Kiểu trả về	Bool	
Chức năng	Kiểm tra email có thể được sử dụng hay không trong quá trình đăng ký một tài khoản mới	
Store Procedure sử dụng		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
email	string	Địa chỉ email

Tên	CreateAccount	
Kiểu trả về	Bool	
Chức năng	Tạo tài khoản mới, trả về True nếu thành công và False nếu thất bại	
Store Procedure sử dụng	TaoTaiKhoan	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
acc	string	Tài khoản
pass	String	Mật khẩu
Email	String	Địa chỉ email
Billingname	String	Tên người thanh toán
Billingaddress1	String	Địa chỉ thanh toán 1
Billingaddress2	String	Địa chỉ thanh toán 2
Billingphone	String	Số điện thoại thanh toán
Shippingname	String	Tên người nhận hàng
Shippingaddress1	String	Địa chỉ người nhận hàng 1
Shippingaddress2	String	Địa chỉ người nhận hàng 2
Shippingphone	String	Điện thoại nhận hàng

Tên	GetAccountDetails	
Kiểu trả về	AccountDetails	
Chức năng	Lấy tất cả thông tin liên quan đến user	
Store Procedure sử dụng	LayThongTinTaiKhoan	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
acc	string	Tài khoản

5. ShoppingCartAccess

Lớp này thực hiện các chức năng như tạo giỏ hàng, thêm xóa sửa và cập nhật giỏ hàng. Bởi vì, giỏ hàng được xây dựng bằng Session nên hầu hết các phương thức trong lớp này sẽ không thực các Store procedure mà chỉ lưu trên đối tượng Session “BookStoreCart”

Tên	GetShoppingCart	
Kiểu trả về	DataTable	
Chức năng	Lấy danh sách các sản phẩm trong giỏ hàng, nếu giỏ hàng chưa có thì sẽ tạo giỏ hàng mới.	
Store Procedure sử dụng		
Tham số (không có)		
Tên tham số	Kiểu	Ý nghĩa

Tên	AddToShoppingCart	
Kiểu trả về	Bool	
Chức năng	Thêm một sản phẩm vào trong giỏ hàng	
Store Procedure sử dụng		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
MaSanPham	String	Mã sản phẩm

Tên	UpdateItemInShoppingCart	
Kiểu trả về	Bool	
Chức năng	Cập nhật số lượng cho sản phẩm tương ứng có trong giỏ hàng	
Store Procedure sử dụng		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
productID	String	Mã sản phẩm
quanity	String	Số lượng

Tên	DeleteItemInShoppingCart	
Kiểu trả về	Bool	
Chức năng	Xóa một sản phẩm ra khỏi giỏ hàng	
Store Procedure sử dụng		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
productID	String	Mã sản phẩm

Tên	CreateOrder	
Kiểu trả về	Bool	
Chức năng	Tạo một đơn hàng với giỏ hàng tương ứng	
Store Procedure sử dụng	ThemDonDatHang	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
Ordered	String	Mã đơn hàng
Account	String	Tài khoản người mua
shippingType	String	Loại vận chuyển
Currency	String	Loại tiền tệ
Rate	Double	Tỷ giá hiện tại

Tên	AddProductInOrder	
Kiểu trả về	Int	
Chức năng	Cập nhật số lượng cho sản phẩm tương ứng có trong giỏ hàng	
Store Procedure sử dụng	ThemSanPhamVaoChiTietDonDatHang	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
ordered	String	Mã đơn đặt hàng
productId	String	Mã sản phẩm
productName	String	Tên sản phẩm
quanity	String	Số lượng
unitPrice	String	Đơn giá

6. OrdersAccess

Lớp này sẽ thực hiện các chức năng quản lý các đơn đặt hàng, lấy thông tin của một đơn đặt hàng, đánh dấu đơn hàng đã đã hoàn tất, đã được duyệt, hoặc xóa đơn hàng,...

Tên	GetInfo	
Kiểu trả về	OrderInfo	
Chức năng	Lấy thông tin cơ bản của một đơn hàng	
Store Procedure sử dụng	LayThongTinDonDatHang	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
Ordered	String	Mã đơn đặt hàng

Tên	GetDetails	
Kiểu trả về	LaySanPhamTrongDonDatHang	
Chức năng	Lấy thông tin các sản phẩm trong đơn đặt hàng	
Store Procedure sử dụng	LaySanPhamTrongDonDatHang	

Tham số		
Tên tham số	Kiểu	Ý nghĩa
Ordered	String	Mã đơn đặt hàng

Tên	Update	
Kiểu trả về	Void	
Chức năng	Cập nhật thông tin cho một đơn đặt hàng	
Store Procedure sử dụng	CapNhatDonDatHang	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
orderInfo	OrderInfo	Một đối tượng chứa các thông tin đơn đặt hàng

Tên	MarkVerified	
Kiểu trả về	Void	
Chức năng	Đánh dấu đơn hàng đã được xác nhận	
Store Procedure sử dụng	XacNhanDonDatHang	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
ordered	String	Mã đơn đặt hàng

Tên	MarkCompleted	
Kiểu trả về	Void	
Chức năng	Đánh dấu đơn hàng đã hoàn tất	
Store Procedure sử dụng	HoanTatDonDatHang	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
ordered	String	Mã đơn đặt hàng

Tên	MarkCanceled	
Kiểu trả về	Void	
Chức năng	Đánh dấu đơn hàng đã bị hủy	
Store Procedure sử dụng	HuyDonDatHang	
Tham số		
Tên tham số	Kiểu	Ý nghĩa
ordered	String	Mã đơn đặt hàng

7. AmazonAccess, eBayAccess

Chức năng chính của hai lớp này là thực hiện giao tiếp với web services được Amazon, eBay cung cấp. Trong mỗi lớp sẽ có một số biến và cách thiết lập tham số riêng, nhưng đều có phương thức lấy các sản phẩm từ các web services này.

Tên	GetAmazonProducts	
Kiểu trả về	DataTable	
Chức năng	Lấy về danh sách sản phẩm từ Amazon thông qua web services	
Store Procedure sử dụng		
Tham số		
Tên tham số	Kiểu	Ý nghĩa
Keywords	String	Từ khóa để tìm về các sản phẩm tương ứng trên Amazon
Page	String	Số trang mà chúng ta muốn lấy về
howmanypage	Int (out)	Lấy ra số trang tìm được trên Amazon với từ khóa được cung cấp

Tên	GeteBayProducts	
Kiểu trả về	DataTable	
Chức năng	Lấy về danh sách sản phẩm từ eBay thông qua web services	
Store Procedure sử dụng		

Tham số		
Tên tham số	Kiểu	Ý nghĩa
Keywords	String	Từ khóa để tìm về các sản phẩm tương ứng trên Amazon
Page	String	Số trang mà chúng ta muốn lấy về
howmanypage	Int (out)	Lấy ra số trang tìm được trên Amazon với từ khóa được cung cấp

Đến đây chúng ta gần như hoàn thành việc xây dựng mức thứ hai trong mô hình ba lớp xây dựng website bán sách qua mạng. Phần tiếp theo chúng ta sẽ xây dựng các thành phần thuộc lớp thứ ba Presentation Tier bao gồm các Master Page, User Control, Web Page,...

Chương 17

THIẾT KẾ WEBSITE BÁN HÀNG

– MỨC TRÌNH DIỄN (PRESENTATION TIER)

Ở mức này, chúng ta sẽ xây dựng các Master Page, User Control và Web Page. Dưới đây là một số thành phần cơ bản của website.

Master Page	BookStore.master	Quy định layout chính cho các trang mà người dùng xem
	Admin.master	Quy định layout cho các trang cho người dùng có quyền admin vào quản lý website
User Control	AmazonProductList.ascx	
	AmazonSearch.ascx	
	CartSummary.ascx	
	CartSummary2.ascx	
	CategoriesList.ascx	
	DepartmentsList.ascx	
	eBayProductList.ascx	
	eBaySearch.ascx	
	Menu.ascx	
	Pager.ascx	
	ProductList.ascx	
	SearchBook.ascx	
Web page	UserInfo.ascx	
	AccountManager.aspx	
	AdminDepartment.aspx	
	AdminOrderDetails.aspx	
	AdminOrder.aspx	

	AdminProductDetails.aspx	
	AdminProducts.aspx	
	Catalog.aspx	
	Checkout.aspx	
	CheckoutInfo.aspx	
	ConfirmCheckoutInfo.aspx	
	Default.aspx	
	Login.aspx	
	ShoppingCart.aspx	

Trong phần này, chúng ta không đi vào xây dựng chi tiết từng webpage mà chỉ trình bày bầy bố cục các Master Page và một số User Control

Master Page

	Logo, banner
	SearchBook.ascx
UserInfo.ascx	
Menu.ascx	ContentPlaceHolder
CartSummary.ascx	
AmazonSearch.ascx	
eBaySearch.ascx	
	Info

Admin.master

UserInfo.ascx	Logo, banner
ContentPlaceHolder	

Pager.ascx

1. Thêm một Web User Control mới có tên là Pager vào thư mục UserControls (chọn with a code behind file).
2. Ở phần Source View, thiết kế như đoạn mã sau:

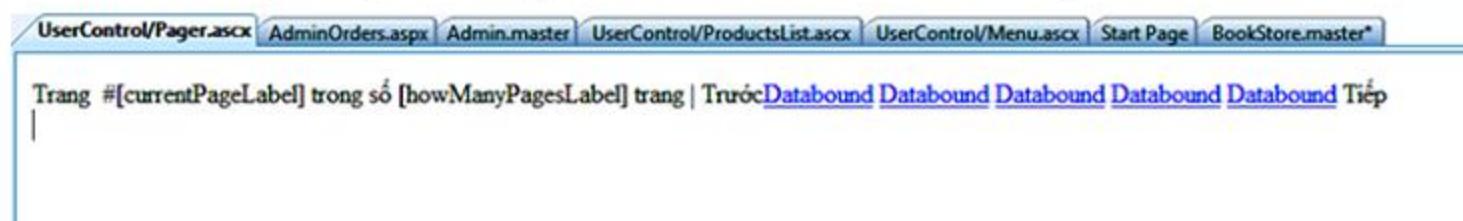
```

<%@ Control Language="C#" AutoEventWireup="true" CodeFile="Pager.ascx.cs"
Inherits="UserControl_Pager" %>

<p>
    Trang &nbsp; #<asp:Label ID="currentPageLabel" runat="server" />
    &nbsp;trong số
    <asp:Label ID="howManyPagesLabel" runat="server" /> &nbsp;trang
    |
    <asp:HyperLink ID="previousLink" Runat="server">Trước</asp:HyperLink>
    <asp:Repeater ID="pagesRepeater" runat="server">
        <ItemTemplate>
            <asp:HyperLink ID="hyperlink" runat="server" Text='<%# Eval("Page") %>' NavigateUrl='<%# Eval("Url") %>'>
        </ItemTemplate>
    </asp:Repeater>
    <asp:HyperLink ID="nextLink" Runat="server">Tiếp</asp:HyperLink>
</p>

```

3. Chuyển qua Design View, control của chúng ta sẽ như sau:



4. Ở file code behind điều chỉnh phần code như sau:

using System;

```

public struct PageUrl
{
    private string page;
    private string url;
    // Page and Url property definitions
    public string Page
    {
        get
    }
}

```

```
        return page;
    }

}

public string Url
{
    get
    {
        return url;
    }
}

// constructor

public PageUrl(string page, string url)
{
    this.page = page;
    this.url = url;
}

public partial class UserControl_Pager: System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    private const int half = 5;

    public void Show(int currentPage, int howManyPages, string firstPageUrl, string
pageUrlFormat, bool showPages)
    {

        if (howManyPages > 1)
        {

            this.Visible = true;

            currentPageLabel.Text = currentPage.ToString();
        }
    }
}
```

```
howManyPagesLabel.Text = howManyPages.ToString();

if (currentPage == 1)
{
    previousLink.Enabled = false;
}
else
{
    previousLink.NavigateUrl = (currentPage == 2) ?
        firstPageUrl: String.Format(pageUrlFormat, currentPage - 1);
}

if (currentPage == howManyPages)
{
    nextLink.Enabled = false;
}
else
{
    nextLink.NavigateUrl = String.Format(pageUrlFormat, currentPage + 1);
}

if (showPages)
{
    PageUrl[] pages = new PageUrl[half*2];
    if (currentPage > half)
    {
        int i = currentPage - half;
        pages[0] = new PageUrl(i.ToString(), string.Format(pageUrlFormat, i));
        int count = 1;
        i++;
        while ((i < currentPage + half) && (i <= howManyPages))
        {
            pages[count] = new PageUrl(i.ToString(), string.Format(pageUrlFormat, i));
            count++;
            i++;
        }
    }
}
```

```

        pages[count] = new PageUrl(i.ToString(), string.Format(pageUrlFormat,
i));
        count++; i++;
    }
    pages[5] = new PageUrl((currentPage).ToString(), "");
}
else
{
    pages[0] = new PageUrl("1",firstPageUrl);
    int i = 2;
    while ((i <= half + currentPage - 1) && (i <= howManyPages))
    {
        pages[i-1] = new PageUrl(i.ToString(), string.Format(pageUrlFormat,
i));
        i++;
    }
    pages[currentPage - 1] = new PageUrl((currentPage).ToString(), "");
}
pagesRepeater.DataSource = pages;
pagesRepeater.DataBind();
}
}
}
}

```

ProductsList.ascx

1. Thêm một Web User Control vào thư mục UserControls, đặt tên là ProductsList
2. Mở Design View và kéo thả một Pager từ Solution Explore vừa được xây dựng ở trên vào ProductsList control.
3. Tiếp tục kéo vào một đối tượng DataList vào ProductsList.
4. Tiếp tục kéo vào một Pager khác vào cuối của ProductsList.
5. Ở phần Source sẽ được chỉnh sửa lại như sau:

```

<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="ProductsList.ascx.cs" Inherits="UserControl_ProductsList" %>
<%@ Register src="Pager.ascx" tagname="Pager" tagprefix="uc1" %>
<%@ Register Assembly="AjaxControlToolkit" Namespace="AjaxControlToolkit"
TagPrefix="cc1" %>

<uc1:Pager ID="topPager" runat="server" Visible="False" />
<script src="jsModalProgress.js" type="text/javascript"></script>
<script type="text/javascript" language="javascript">
    var ModalProgress = '<%= ModalProgress.ClientID %>';
</script>
<table style="width:100%">
    <tr>
        <td style="text-align:right">
            <span style="font-style:oblique;font-size:14px">Chọn loại tiền: </span>
            <asp:DropDownList ID="ddlCurrency" runat="server"
DataSourcesID="SqlDataSource1" AutoPostBack="true"
    DataTextField="QuocGia" DataValueField="DonVi"
    onselectedindexchanged="ddlCurrency_SelectedIndexChanged" Font-
Size="14px">
            </asp:DropDownList>
            <asp:SqlDataSource ID="SqlDataSource1" runat="server"
                ConnectionString="<%$ ConnectionStrings:OnlineStoreConnection %>" 
                SelectCommand="SELECT DonVi, QuocGia, ThongDung FROM tblTienTe
ORDER BY ThongDung DESC, DonVi">
            </asp:SqlDataSource>
            <asp:HiddenField ID="hdRate" runat="server" />
        </td>
    </tr>
    <tr>
        <td>
            <asp:Label ID="lblError" runat="server" ForeColor="Red"></asp:Label>
        </td>
    </tr>
</table>
<asp:Panel ID="panelUpdateProgress" runat="server">

```

```

<asp:UpdateProgress ID="UpdateProg1" DisplayAfter="0" runat="server" >
    <ProgressTemplate>
        <div style="position: relative; top: 2%; text-align: center; left: 4px; height: 70px; width: 150px;">
            
            <br />
            <span style="color:White">Vui lòng chờ....</span>
        </div>
    </ProgressTemplate>
</asp:UpdateProgress>
</asp:Panel>
<cc1:ModalPopupExtender ID="ModalProgress" runat="server"
TargetControlID="panelUpdateProgress"
BackgroundCssClass="modalBackground" PopupControlID="panelUpdateProgress" />
<asp:DataList ID="list" runat="server" RepeatColumns="3"
DataKeyField="MaSanPham" onitemcommand="list_ItemCommand"
onitemdatabound="list_ItemDataBound">
    <ItemTemplate>

        <table style="width:100%;">
            <tr>
                <td style="width:100%; padding-left:10px">
                    <span style="font-size:12px; font-weight:bold; margin-left:20px">
                        <br style="line-height:10px" />
                        <asp:HyperLink ID="hplnkName" Height="30px" runat="server"
NavigateUrl='<%# LinkBuilder.ToProduct(Eval("MaSanPham").ToString())%>'
Text='<%# HttpUtility.HtmlEncode(Eval("TenSanPham").ToString())
%>'></asp:HyperLink>
                    </span>
                </td>
            </tr>
            <tr>
                <td style="padding-left:10px; border-left-color:Green; border-left-
style:solid; border-left-width:1px">
                    <table>
                        <tr>

```

```

<td style="width:5px;">

</td>
<td>
    <br style="line-height:10px">
    <a href="<%# LinkBuilder.ToProduct(Eval("MaSanPham").ToString()) %>">
        " border="0"
        alt="<%# HttpUtility.HtmlEncode(Eval("TenSanPham").ToString())%>"></a><br><br
        style="line-height:9px">
    </a>
</td>
<td>
    <br><br style="line-height:9px"><%# HttpUtility.HtmlEncode(Eval("MoTaSanPham").ToString()) %>
</td>
</tr>
<tr>
    <td>
        </td>
        <td>
            <asp:LinkButton ID="btnAddToCart" runat="server" Text="Thêm
vào giỏ" Width="100px"></asp:LinkButton>
        </td>
    </td>
</tr>
<tr>
    <td style="padding-left:10px; border-left-color:Green; border-left-
style:solid; border-left-width:1px">
        <table>

```

```

<tr>
    <td>
        <asp:HiddenField ID="hd" runat="server"
Value='<%#Eval("Gia") %>' />
        <span class="productSpecialPrice"><asp:Label ID="lblPrice"
runat = "server"
Text='<%#FormatCurrency(Double.Parse(Eval("Gia").ToString()))%>'></asp:Label>
</span>

        <br style="line-height:3px">
        <br style="line-height:6px">
    </td>
    <td>

        <br />
        <br style="line-height:6px">
    </td>
</tr>
<tr>
    <td>

        </td>
    </td>
</tr>
</table>
</td>
</tr>
<tr style="width:80%">
    <td colspan="6" style="background:url(images/1_bg1.gif); width:1px;
height:1px"> </td>
</tr>
</table>

</ItemTemplate>
</asp:DataList>

```

```

<script language="javascript" type="text/javascript">
    var lblprice,ddlist,hidden;
    function Select1_onchange(seft,lblPrice,hd) {
        lblprice = lblPrice;
        ddlist = seft;
        hidden = hd;

        CurrencyService.GetCurrency(lblPrice=document.getElementById(hd).value,seft.value,
        Complete,Error);
    }
    function Error(result)
    {
        ddlist.value = "USD";
        alert("Vui lòng thử lại sau");
    }
    function Complete(result)
    {
        document.getElementById(lblprice).innerHTML= result;
    }
</script>
<uc1:Pager ID="bottomPager" runat="server" Visible="False" />

```

6. chỉnh sửa lại file ProductsList.ascx.cs:

```

using System;
using System.Collections;
using System.Web.UI.WebControls;
using System.Globalization;
using CurrencyWS;
public class MyCompareClass: IComparer
{
    int IComparer.Compare(Object x, Object y)
    {
        return (string.Compare(x.ToString(), y.ToString()));
    }
}
public partial class UserControl_ProductsList: System.Web.UI.UserControl

```

```

{
private void InitializeComponent()
{
}
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        InitCurrencyRate();
        PopulateControls();
    }
}
private void InitCurrencyRate()
{
    if (Session["BookStoreCurrency"] == null)
    {
        hdRate.Value = "1";
        ddlCurrency.SelectedIndex = 0;
        Session["BookStoreCurrency"] = "0,USD";
    }
    else
    {
        string[] CurrencyInfo = (Session["BookStoreCurrency"].ToString()).Split(',');
        ddlCurrency.SelectedIndex = Int32.Parse(CurrencyInfo[0]);
        CurrencyConvertor currencyConvertor = new CurrencyConvertor();
        if (CurrencyInfo[1] != "USD")
            try
            {
                hdRate.Value = (currencyConvertor.ConversionRate(Currency.USD,
(Currency)Enum.Parse(typeof(Currency), CurrencyInfo[1])).ToString());
            }
            catch
            {
                ddlCurrency.SelectedIndex = 0;
            }
    }
}

```

```

        hdRate.Value = "1";
    }
    else
        hdRate.Value = "0";
    }
}

private void PopulateControls()
{
    string departmentId = Request.QueryString["DepartmentID"];
    string categoryId = Request.QueryString["CategoryID"];
    string page = Request.QueryString["Page"];
    string searchString = Request.QueryString["Search"];
    string dpId = Request.QueryString["dpid"];
    if (page == null) page = "1";
    int howManyPages = 1;
    string firstPageUrl = "";
    string pagerFormat = "";
    string currentpage = Request.AppRelativeCurrentExecutionFilePath;
    if (String.Compare(currentpage, "~/NewestProducts.aspx", true) == 0)
    {
        DateTime from = new DateTime(2009, 10, 1, 12, 0, 0);
        list.DataSource = CatalogAccess.GetNewestProducts(from,
            DateTime.Now, OnlineStoreConfigurations.ProductDescriptionLength, Int32.Parse(page),
            OnlineStoreConfigurations.ProductsPerPage, out howManyPages);
        list.DataBind();
        firstPageUrl = LinkBuilder.ToNewestProducts("1");
        pagerFormat = LinkBuilder.ToNewestProducts("{0}");
    }
    else
        if (String.Compare(currentpage, "~/BestSellerProducts.aspx", true) == 0)
    {
        list.DataSource =
CatalogAccess.GetBestSellerProducts(OnlineStoreConfigurations.ProductDescriptionL
ength, Int32.Parse(page), OnlineStoreConfigurations.ProductsPerPage, out
howManyPages);
        list.DataBind();
    }
}

```

```

firstPageUrl = LinkBuilder.ToBestSellerProducts("1");
pagerFormat = LinkBuilder.ToBestSellerProducts("{0}");
}

else
if (searchString != null)
{
    string allWords = Request.QueryString["AllWords"];

    list.DataSource = CatalogAccess.Search(searchString, dpId,
Int32.Parse(page), OnlineStoreConfigurations.ProductsPerPage,out howManyPages);
    list.DataBind();
    // Display pager
    firstPageUrl = LinkBuilder.ToSearch(searchString, dpId, "1");
    pagerFormat = LinkBuilder.ToSearch(searchString, dpId, "{0}");
    if (list.Items.Count == 0)
        lblError.Text = "Không có sản phẩm nào";
    }
    else
if (categoryId != null)
{
    list.DataSource = CatalogAccess.GetProductsInCategory(categoryId,
page, out howManyPages);
    list.DataBind();
    firstPageUrl = LinkBuilder.ToCategory(departmentId, categoryId, "1");
    pagerFormat = LinkBuilder.ToCategory(departmentId, categoryId,
"{0}");
    }
    else
if (departmentId != null)
{
    list.DataSource =
CatalogAccess.GetProductsOnDeptPromo(departmentId, page, out howManyPages);
    list.DataBind();
}

```

```

        firstPageUrl = LinkBuilder.ToDepartment(departmentId, "1");
        pagerFormat = LinkBuilder.ToDepartment(departmentId, "{0}");
    }
    else
    {
        list.DataSource = CatalogAccess.GetProductsOnFrontPromo(page, out
howManyPages);
        list.DataBind();
        howManyPages = 1;
    }
    topPager.Show(int.Parse(page), howManyPages, firstPageUrl, pagerFormat, true);
    bottomPager.Show(int.Parse(page), howManyPages, firstPageUrl, pagerFormat,
true);
}
protected void list_ItemCommand(object source,
System.Web.UI.WebControls.DataListCommandEvent Args e)
{
    DataListItem items = e.Item;
    string productId = list.DataKeys[e.Item.ItemIndex].ToString();
    ShoppingCartAccess.AddToShoppingCart(productId);
    Response.Redirect(Request.RawUrl);
}
public string FormatCurrency(double price)
{
    NumberFormatInfo nfi = new CultureInfo("en-US", false).NumberFormat;
    return price.ToString("N",nfi);
}
protected void list_SelectedIndexChanged(object sender, EventArgs e)
{
}
protected void list_ItemDataBound(object sender, DataListItemEventArgs e)
{
    Label lblPrice = e.Item.FindControl("lblPrice") as Label;
    HiddenField hd = e.Item.FindControl("hd") as HiddenField;
}

```

```

NumberFormatInfo nfi = new CultureInfo("en-US", false).NumberFormat;
double dprice = Double.Parse(hd.Value) * Double.Parse(hdRate.Value);

    lblPrice.Text = dprice.ToString("N", nfi) + " " +
(Session["BookStoreCurrency"].ToString()).Split(',')[1];
//}

}

protected void ddlCurrency_SelectedIndexChanged(object sender, EventArgs e)
{

if (ddlCurrency.SelectedValue != "USD")
{
    try
    {
        CurrencyConvertor currencyConvertor = new CurrencyConvertor();
        hdRate.Value = (currencyConvertor.ConversionRate(Currency.USD,
(Currency)Enum.Parse(typeof(Currency), ddlCurrency.SelectedValue))).ToString();
    }
    catch
    {
        hdRate.Value = "1";
        ddlCurrency.SelectedIndex = 0;
        lblError.Text = "Có lỗi trong quá trình chuyển đổi, vui lòng thử lại sau!";
    }
}
else
{
    hdRate.Value = "1";
}

Session["BookStoreCurrency"] =
ddlCurrency.SelectedIndex.ToString() + "," + ddlCurrency.SelectedValue;
PopulateControls();
Response.Redirect(Request.RawUrl);
}

}

```

Mỗi khi ProductsList được load lại sẽ căn cứ vào query string để lấy được danh sách sản phẩm tương ứng. Như vậy, khi muốn load danh sách sản phẩm cho một trang nào đó trên website thì ta chỉ cần sử dụng ProductsList User control này.

Sau đây là một số giao diện chính của website:

1. Trang chủ

Ở trang này, người dùng có thể xem các loại sách, đăng nhập, đăng ký, tìm kiếm



2. Trang đăng ký

Cho phép người dùng đăng ký một tài khoản để có thể sử dụng đầy đủ các chức năng của trang web.

Thông tin tài khoản	
Tên tài khoản:	<input type="text"/>
Mật khẩu:	<input type="password"/>
Xác nhận mật khẩu:	<input type="password"/>
E-mail:	<input type="text"/>
Các thông tin cần thiết khi người dùng đăng ký tài khoản	
<input type="checkbox"/> Thông tin thanh toán và nhận hàng là như nhau	
Thông tin thanh toán	
Tên người thanh toán :	<input type="text"/>
Địa chỉ:	<input type="text"/>
Địa chỉ 2:	<input type="text"/>
Điện thoại:	<input type="text"/>
Thông tin nhận hàng	
Tên người nhận hàng :	<input type="text"/>
Địa chỉ:	<input type="text"/>
Địa chỉ 2:	<input type="text"/>
Điện thoại:	<input type="text"/>
<input type="button" value="Đăng ký"/>	

3. Trang đăng nhập

Người dùng phải nhập chính xác thông tin tài khoản và mật khẩu.

ĐĂNG NHẬP VÀO HỆ THỐNG

Tài khoản:

Mật khẩu:

Nhớ tài khoản này.

Nếu người dùng đăng nhập thành công.

Chào, tôi ban

ĐĂNG XUẤT

Trang chủ
Quản lý tài khoản

CÁC THỂ LOẠI SÁCH:

KHOA HỌC
KINH TẾ - QUẢN LÝ
NGHỆ THUẬT
TIN HỌC - CNTT

Chào mừng tài
khoản đã đăng
nhập thành công

Trang quản lý tài
khoản của người dùng

Chọn loại tiền: U.S Dollar [USD]

Thêm vào giỏ

Thông tin về sách
như tên sách, hình
ảnh, giá tiền...

The diagram illustrates the user interface flow. It starts with a 'ĐĂNG NHẬP VÀO HỆ THỐNG' (Login) form. If successful, it leads to a 'Chào mừng tài khoản đã đăng nhập thành công' (Welcome to your account) message. This message points to a 'Trang quản lý tài khoản của người dùng' (User Account Management Page). On this page, there are three rows of book cards, each with a red 'X' icon and a 'Thêm vào giỏ' (Add to cart) link. A sidebar on the left lists book categories: Khoa học, Kinh tế - Quản lý, Nghệ thuật, and Tin học - CNTT. A note at the bottom left provides information about books, such as name, image, price, etc.

4. Trang các loại sách

- Sách Khoa học

The screenshot shows a search results page for 'CÁC THỂ LOẠI SÁCH' (Types of Books). The left sidebar displays categories: KHOA HỌC (Hoa học, Thiên văn học, Toán học, Y học), KINH TẾ - QUẢN LÝ, NGHỆ THUẬT, and TIN HỌC - CNTT. A box highlights 'Các thể loại sách con' (Sub-categories of book types). The main content area shows a grid of 12 book thumbnails, each with a red 'X' icon in the top right corner and a 'Thêm vào giỏ' (Add to cart) link below it. A dropdown menu at the top right shows 'Chọn loại tiền: U.S Dollar [USD]'. A callout box points to one of the 'Thêm vào giỏ' links with the text 'Thêm sản phẩm vào giỏ hàng' (Add product to shopping cart). Navigation links at the bottom indicate this is page 1 of 44.

5. Trang chi tiết sản phẩm

Xin chào!

Bạn chưa đăng nhập.
[Đăng nhập](#)
hoặc
[Đăng ký](#)

CÁC THỂ LOẠI SÁCH:

- KHOA HỌC
- KINH TẾ - QUẢN LÝ
- NGHỆ THUẬT
- TIN HỌC - CNTT

Tìm sản phẩm tương tự trên Amazon

Sản phẩm tương tự trên Amazon

Tìm sản phẩm tương tự trên Ebay

Sản phẩm tương tự trên ebay

The C# Programming Language - Third Edition

Special Annotated Edition for C# 3.0

The C# Programming Language
Third Edition

Tác giả: [Anders Hejlsberg](#)
Nhà xuất bản: [Apress](#)
Ngôn ngữ: [Tiếng Anh](#)
Giá: 26.99 USD

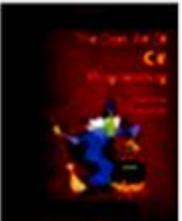
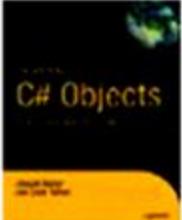
Chọn loại tiền: [U.S Dollar](#)

Chuyển đổi tiền tệ

Các loại tiền tệ có thể chuyển đổi:



Những sản phẩm tương tự trên Amazon

Sản phẩm tương tự trên Amazon		
Trước 1 2 3 4 5 Tiếp		
The C# Programming Language (3rd Edition)  \$44.99	Addison-Wesley.(CHMI-ACQ Guidelines for Improving ...  \$9.95	Programming Microsoft® Visual C#® 2005: The Langua ...  \$49.99
Programming Microsoft® Visual C#® 2008: The Langua ...  \$59.99	The Dark Art of C# Programming: .Net Core Language ...  \$19.95	The C# 3.0 Programming Language 
Programming Microsoft Visual C# 2005: The Language  \$49.99	Programming Microsoft Visual C# 2008: The Language  \$49.99	C# Programming: From Problem Analysis to Program D ...  \$124.95
Beginning C# Objects: From Concepts to Code  \$49.99		

Những sản phẩm tương tự trên Ebay

Sản phẩm tương tự trên ebay		
Trước 1 2 3 4 5 Tiếp		
NEW Essentials of C Programming Language - Ackerman ...  \$5.99	The C Programming Language by Brian W. Kernighan (...  \$9.99	The C Programming Language by Brian W. Kernighan (...  \$34.99
C++ Programming Language by Bjarne Stroustrup  \$32.50	The C++ Programming Language by Bjarne Stroustrup ...  \$0.99	The C++ Programming Language, Bjarne Stroustrup, G ...  \$7.01

6. Trang quản lý tài khoản user

Thay đổi thông tin tài khoản

THAY ĐỔI MẬT KHẨU

Bạn có thể thay đổi mật khẩu tại đây

Tài khoản : tphon

[Thay đổi](#)

THÔNG TIN THANH TOÁN

Bạn có thể thay đổi các thông tin thanh toán tại đây

Tên người thanh toán cũ : Cao Thiện Quang

[Thay đổi](#)

Địa chỉ cũ : 23 đường số 4, F. Trường Thọ, Q. Thủ Đức, TP Hồ Chí Minh

Địa chỉ 2 cũ:

Điện thoại cũ: 0903327267

THÔNG TIN NHẬN HÀNG

Bạn có thể thay đổi các thông tin nhận hàng tại đây

Tên người nhận hàng cũ : Cao Thiện Quang

[Thay đổi](#)

Địa chỉ cũ : 23 đường số 4, F. Trường Thọ, Q. Thủ Đức, TP Hồ Chí Minh

Địa chỉ 2 cũ:

Điện thoại cũ: 0903327267

7. Trang giỏ hàng

GIỎ HÀNG CỦA BẠN

Chọn loại tiền: U.S Dollar [USD]

Ánh	Tên sách	Đơn giá	Số Lượng	Thành tiền	
	The C# Programming Language - Third Edition	26.99	1	26.99	<input type="button" value="Cập nhật"/> <input type="button" value="Xóa"/>
	Pro ASP.NET 3.5 in VB 2008	5.99	1	5.99	<input type="button" value="Cập nhật"/> <input type="button" value="Xóa"/>

Tổng cộng: 32.98 USD

8. Trang đặt hàng

Điền đầy đủ các thông tin dưới đây để tiếp tục thanh toán

THÔNG TIN THANH TOÁN

Tên người thanh toán: *
 Địa chỉ: *
 Địa chỉ 2:
 Điện thoại: *

THÔNG TIN NHẬN HÀNG

Thông tin thanh toán và nhận hàng là như nhau
 Tên người nhận hàng: *
 Địa chỉ: *
 Địa chỉ 2:
 Điện thoại: *

chi tiết giỏ hàng U.S Dollar [USD]

Số lượng: 1 Đơn giá: 26.99 USD

Số lượng: 1 Đơn giá: 5.99 USD

Chi phí vận chuyển:

0.00 USD (3-7 ngày)
 5.00 USD (2 ngày)
 12.00 USD (Hỗm sau)

Tổng cộng: 32.98 USD

PayPal Specials | Advanced Search | Contact Us | Create an account
 Books Store © 2009 | Privacy Policy

9. Trang quản lý

Chào, admin

Đăng xuất

Trang chủ

Quản lý sản phẩm

Quản lý đơn đặt hàng

Các thực đơn chức năng

QUẢN LÝ CÁC LOẠI SÁCH

Loại sách	Mô tả	Xem các mục sách	Chỉnh sửa	Xóa
Khoa học		Xem các mục sách	Chỉnh sửa	Xóa
Kinh tế - Quản lý		Xem các mục sách	Chỉnh sửa	Xóa
Nghệ thuật		Xem các mục sách	Chỉnh sửa	Xóa
Tin học - CNTT		Xem các mục sách	Chỉnh sửa	Xóa

Thêm loại sách mới:

Tên loại sách:

Mô tả về loại sách này:

Thêm loại sách mới

Thêm loại sách

PayPal

Specials | Advanced Search | Contact Us | Create an account
Books Store © 2009 | Privacy Policy

Quản lý đơn đặt hàng

Chào, admin

Đăng xuất

Trang chủ

Quản lý sản phẩm

Quản lý đơn đặt hàng

QUẢN LÝ ĐƠN ĐẶT HÀNG

20 đơn đặt hàng gần đây nhất Xem

Các đơn đặt hàng từ ngày [] đến ngày [] Xem

Các đơn đặt hàng chưa được xác nhận và chưa hủy Xem

Các đơn đặt hàng đã được xác nhận nhưng chưa hoàn tất Xem

TÀI LIỆU THAM KHẢO

1. Giáo trình ngôn ngữ C#
2. Apress - Pro C# 2008 and the NET 3.5 Platform Fourth Edition
3. Kỹ thuật Lập trình C#
4. Apress - Pro C# 2008 and the NET 3.5 Platform Fourth Edition
5. Apress.Pro.LINQ.Language.Integrated.Query.in.C.Sharp.2008
6. Apress.Pro ASP.NET 3.5 in C# 2008
7. Lập trình ASP.NET- Tập 5. Quyển 3. Phạm Hữu Khang

GIÁO TRÌNH

LẬP TRÌNH WEB VỚI ASP.NET

Nguyễn Minh Đạo

NHÀ XUẤT BẢN

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

Khu Phố 6, Phường Linh Trung, Quận Thủ Đức, TPHCM

Số 3, Công trường Quốc tế, Quận 3, TP Hồ Chí Minh

ĐT: 38239171 – 38225227 - 38239172

Fax: 38239172 - Email: vnuhp@vnuhcm.edu.vn

PHÒNG PHÁT HÀNH NHÀ XUẤT BẢN

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

Số 3 Công trường Quốc tế - Quận 3 – TPHCM

ĐT: 38239170 – 0982920509 – 0913943466

Fax: 38239172 – Website: www.nxbdhqghcm.edu.vn

Chịu trách nhiệm xuất bản:

NGUYỄN HOÀNG DŨNG

Chịu trách nhiệm nội dung:

NGUYỄN HOÀNG DŨNG

Tổ chức bản thảo và chịu trách nhiệm về tác quyền

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TPHCM

Biên tập:

HOÀNG KHẮC THỦY

Sửa bản in:

MINH NHẬT

Trình bày bìa

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TPHCM

Mã số ISBN: 978-604-73-1691-5

Số lượng 300 cuốn; khổ 16 x 24cm.

Số đăng ký kế hoạch xuất bản: 126-2013/CXB/166-07/ĐHQGTPHCM.

Quyết định xuất bản số: 133 ngày 01/07/2014 của NXB ĐHQGTPHCM.

In tại Công ty TNHH In và Bao bì Hưng Phú.

Nộp lưu chiểu quý III năm 2014.



TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH

www.hcmute.edu.vn

ISBN: 978-604-73-1691-5

9 786047 316915