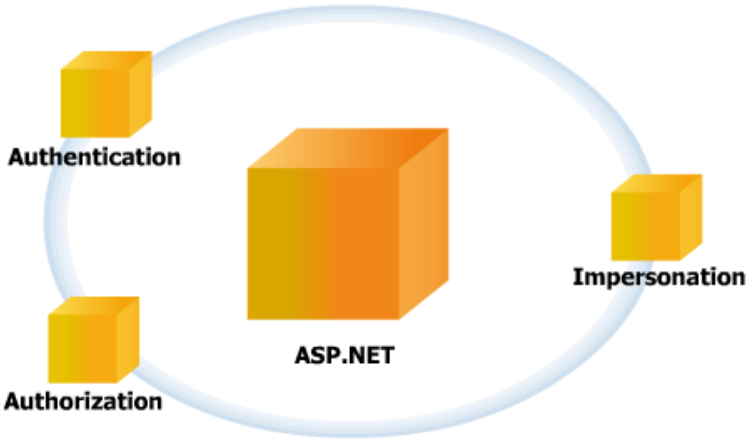
Chương 10: Bảo Mật Ứng Dụng Web

Kết thúc chương này các bạn có thể :

- *Trình bày được các loại bảo mật trong ASP.Net*
- *Mô tả được Form-Base Authentication*
- *Thực thi được bảo mật sử dụng Form-Base 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 :

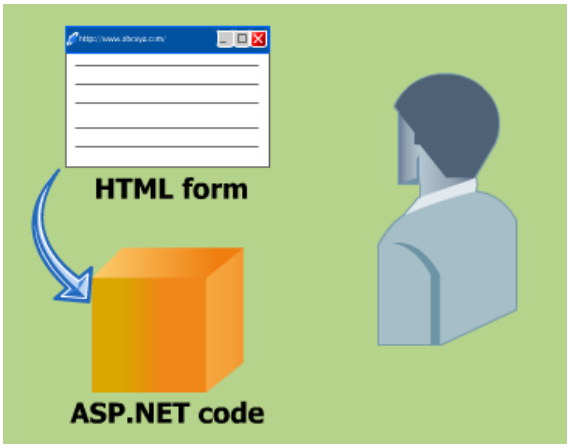


Hình 10.1 Ba tính năng bảo mật cho các ứng dụng ASP.NET

- **Authentication** : được dùng để xác nhận định dannh 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 này trong cơ sở dữ liệu.
- **Authorization :** chỉ những người dùng có sự định danh hợp lệ mới có thể truy cập đền 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.
- **Impersionation :** 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.

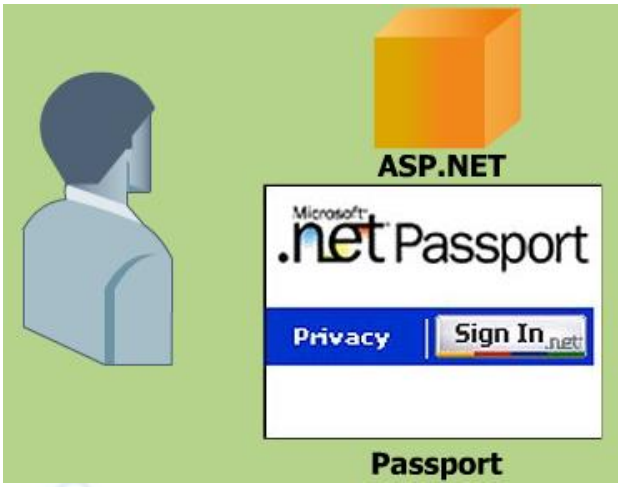
Form-Based Authentication

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.



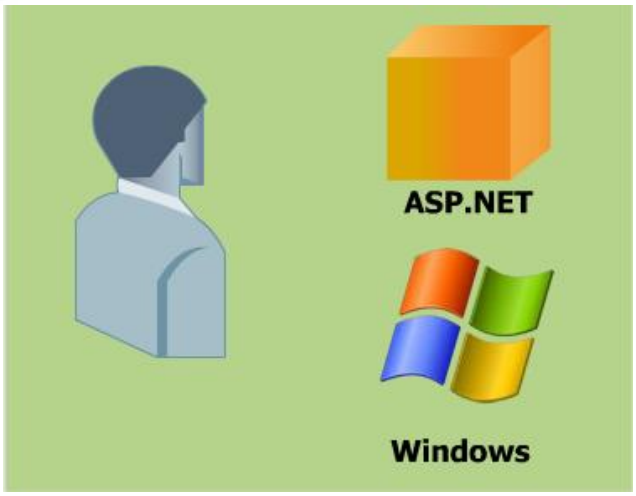
Passport Authentication

Người dùng được xác nhận bằng các 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.



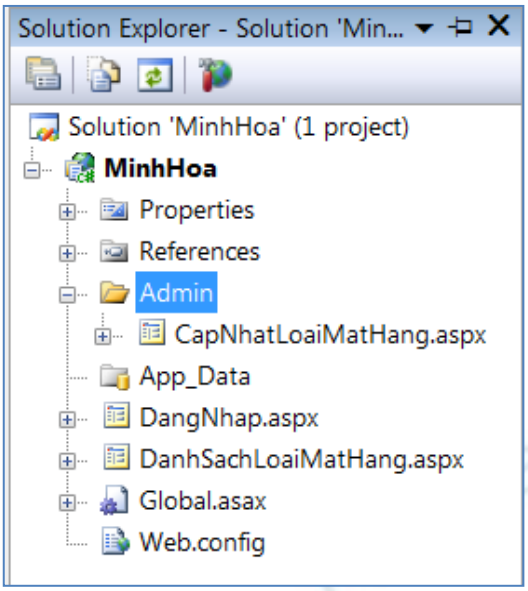
➢ Có 4 tùy chọn được thiết lập trên IIS :

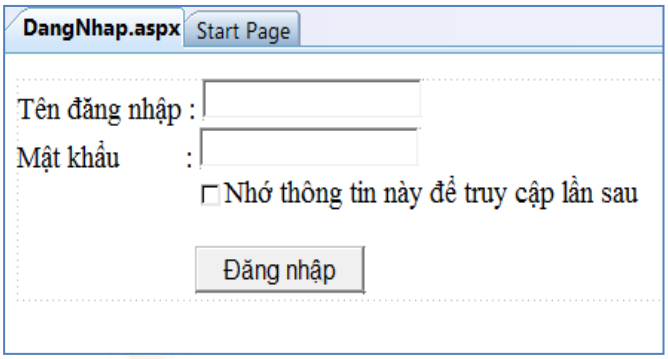
- **Anonymous Authentication** : cho phép bất kỳ người dùng truy cập đến ứng dụng ASP.Net
- **Basic Authentication** : yêu cầu sử dụng tên người dùng và mật khẩu của Windows để kết nối đến ứng dụng. Tuy nhiên mật khẩu được truyền trong dạng text đơn giản , do đó loại xác nhận này không được an toàn .
- **Digest Authentication** : tương tự như Basic Authentication. Tuy nhiên mật khẩu được hashed sau đó mới được truyền đi (gia tăng sự an toàn).
- **Integrated Windows Authentication** : sử dụng Kerberos hoặc giao thức change/response để xác nhận người dùng.

10.2 Thí dụ minh họa

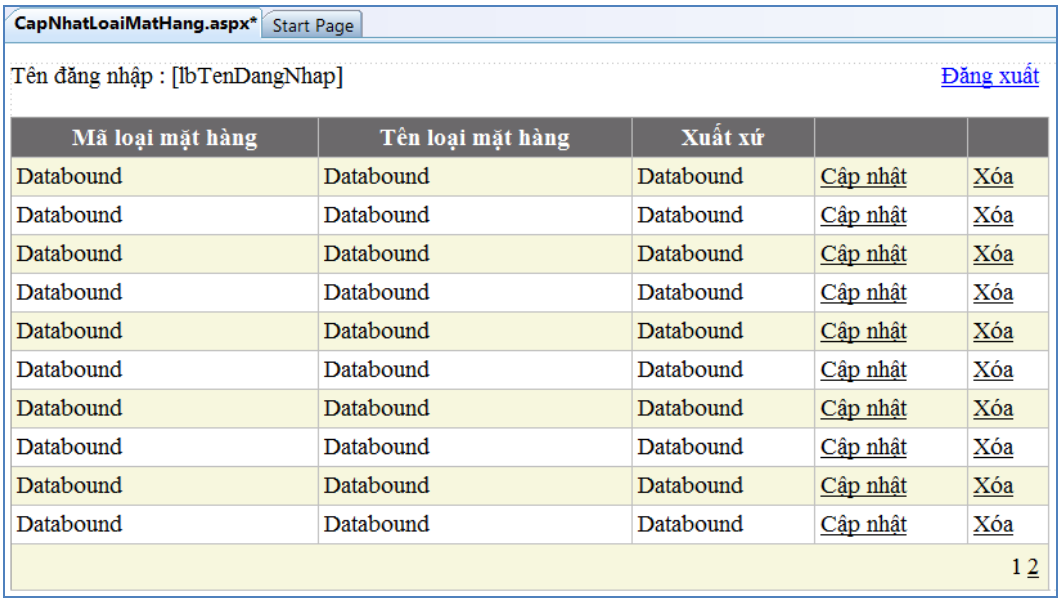
Thí dụ : minh họa sử **dụng Form-based Authentication** trong ứng dụng Web theo yêu cầu : người dùng truy cập vào các trang trong thư mục **Admin** bắt buộc phải đăng nhập , nếu không đăng nhập người dùng chỉ có xem danh sách các loại mặt hàng bằng cách truy cập vào trang DanhSachLoaiMatHang.aspx

Bước 1 : Xây dựng 1 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 10.2Giao diện trang DangNhap.aspx



Hình 10.3 Giao diện trang CapNhatLoaiMatHang.aspx

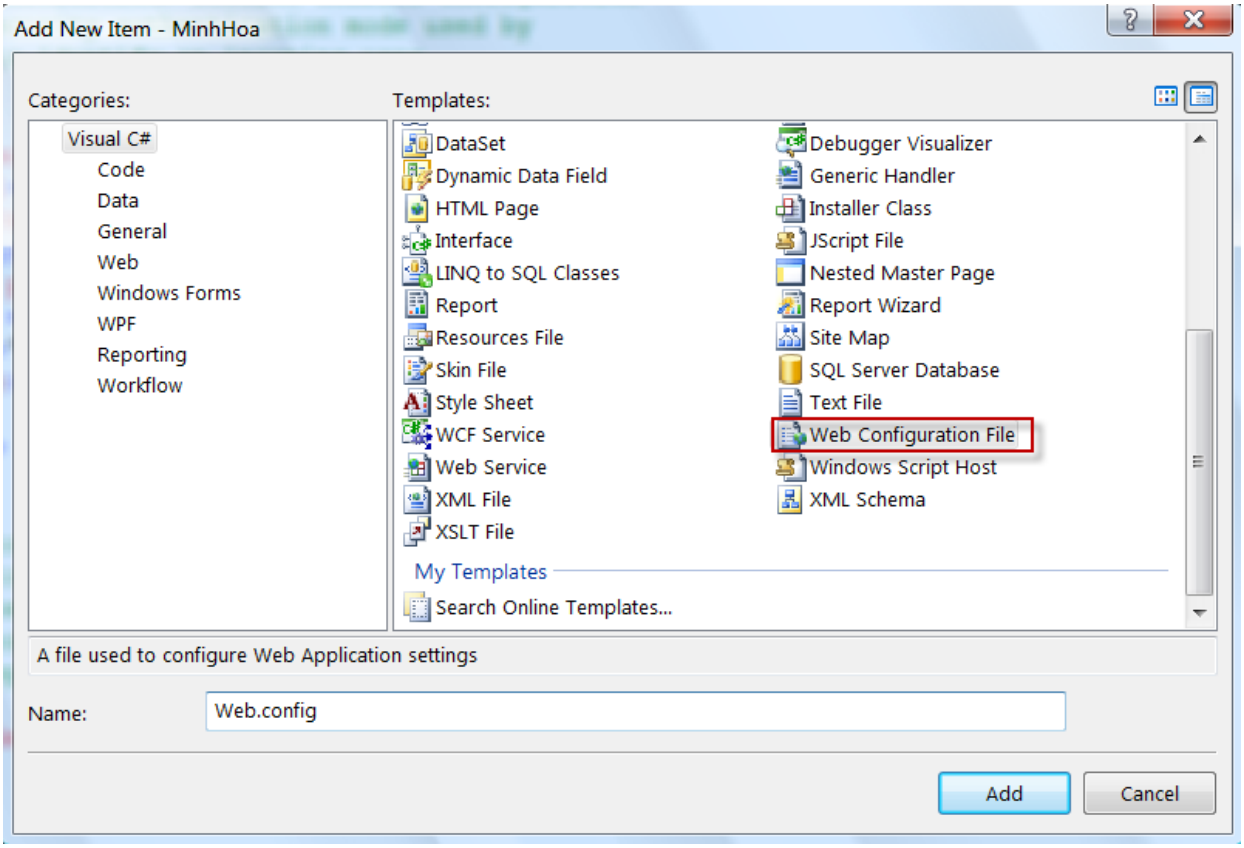
Bước 2 : Cấu hình phần **<authentication>....</authentication>** trong tập tin Web.config như sau :

```
<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 2 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 1 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 :

DangNhap.aspx.cs* | Start Page

MinhHoa.DangNhap | btnDangNhap_Click(object sender,

```
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 hiện Page_Load và sự kiện cho link button Đăng Xuất như hình sau :

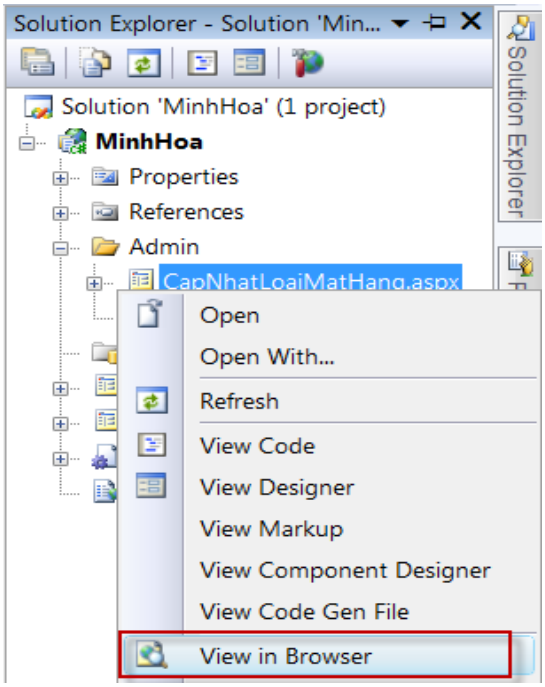
CapNhatLoaiMatHang.aspx.cs* | Start Page

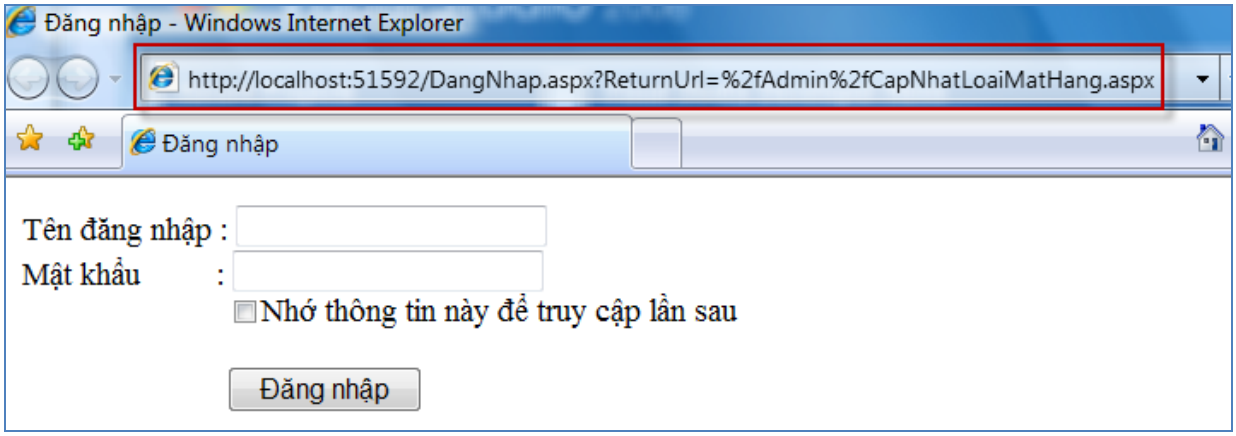
MinhHoa.CapNhatLoaiMatHang | CapNhatThongTinLoaiMatHang

```
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 tất cả các thông tin đã lưu và trở về trang đăng nhập
            FormsAuthentication.SignOut();
            FormsAuthentication.RedirectToLoginPage();

        }
```

Hình 10.5 Phần mã lệnh trang CapNhatLoaiMatHang.aspx

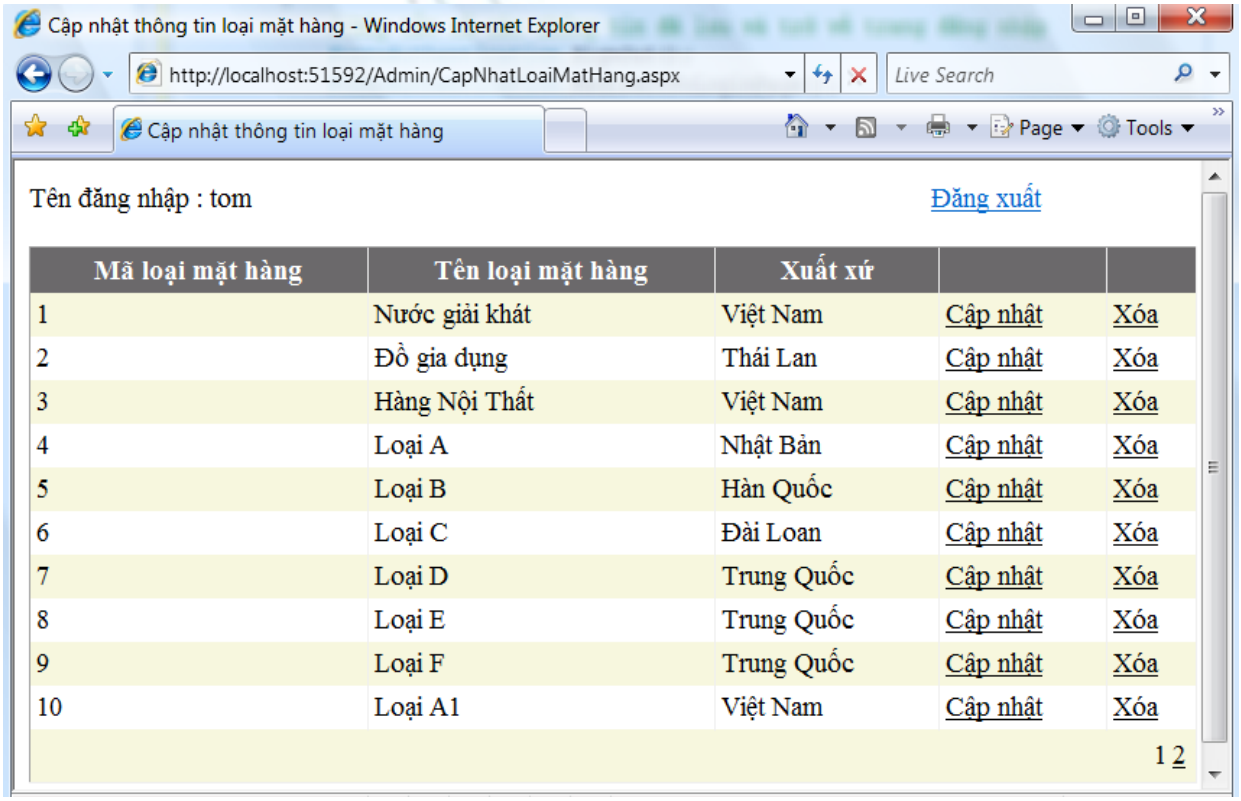
Bước 5. Trong cửa sổ Solution Explorer , chọn trang *CapNhatLoaiMatHang.aspx* nhấn phải chuột | View in Brower để thi hành trang web.Khi người dùng truy cập vào trang này mà chưa đăng nhận thì ASP.Net tự động chuyển qua trang DangNhap.aspx Kết quả sẽ xuất hiện như sau.





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 .



Hình 10.7 Trang cập nhập 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

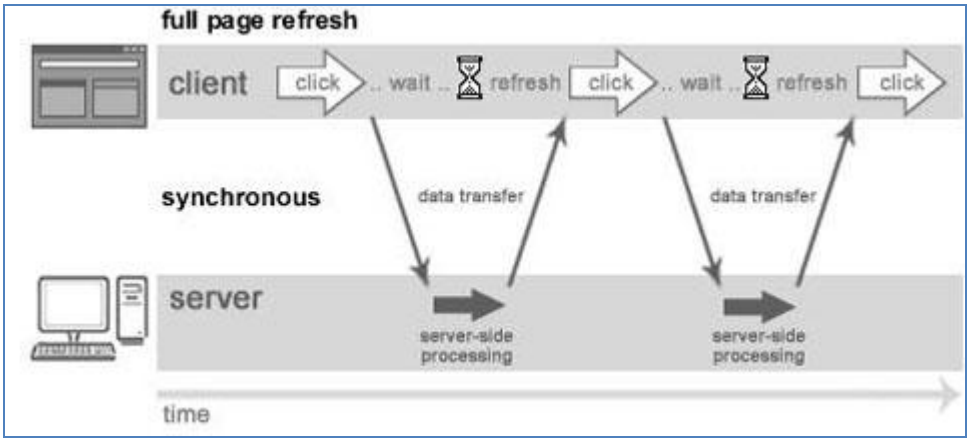
AJAX là một công nghệ cho phép lập trình bất đồng bộ trong ứng dụng Web. Thông thường người dùng muốn thay đổi thông tin từ trang Web bằng cách nhấp vào các nút lệnh(button) hay các liên kết (link) để submit yêu cầu về Web Server để thay đổi nội dung trang Web (postback). Như vậy toàn bộ trang Web phải 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 trực tiếp 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 (Asynchonous), 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 sử 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 kĩ 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 gởi 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 .

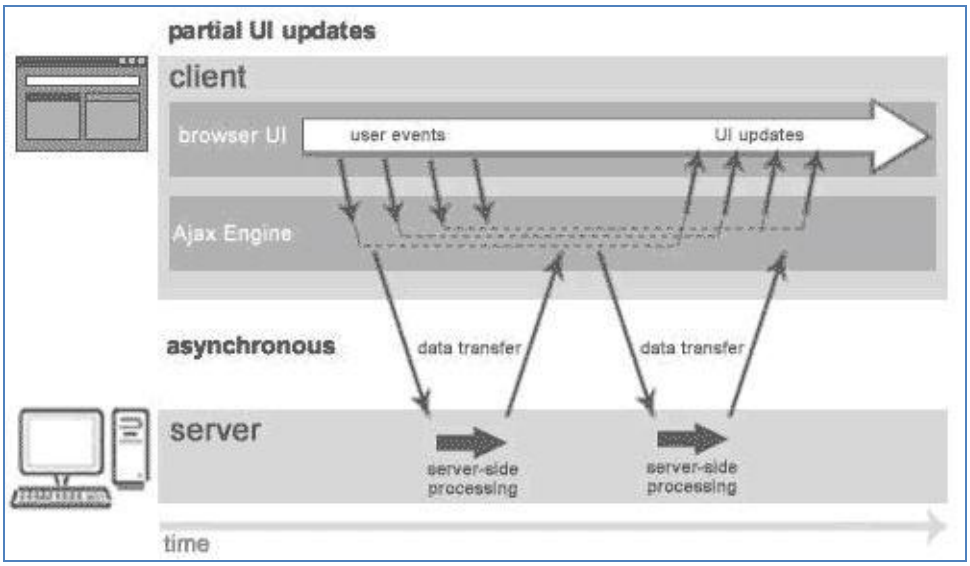
Với cách lập trình Web trước đây (còn gọi là Web 1.0) thì khi người dùng cần cập nhật thông tin (click vào một một Button nào đó), thì yêu cầu thay đổi thông tin sẽ được gửi từ phía Client về Server dưới dạng HTTP request , toàn bộ trang web sẽ được gửi chứ không riêng gì một vài thông tin cần thay đổi (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). Qui 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 1 trang web cổ điện là : Click → waiting .. → refresh …→….Do đó cho dù yêu cầu cập nhật một lượng thong tin nhỏ thì trang web cũng phải load lại, do đó cách 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 .



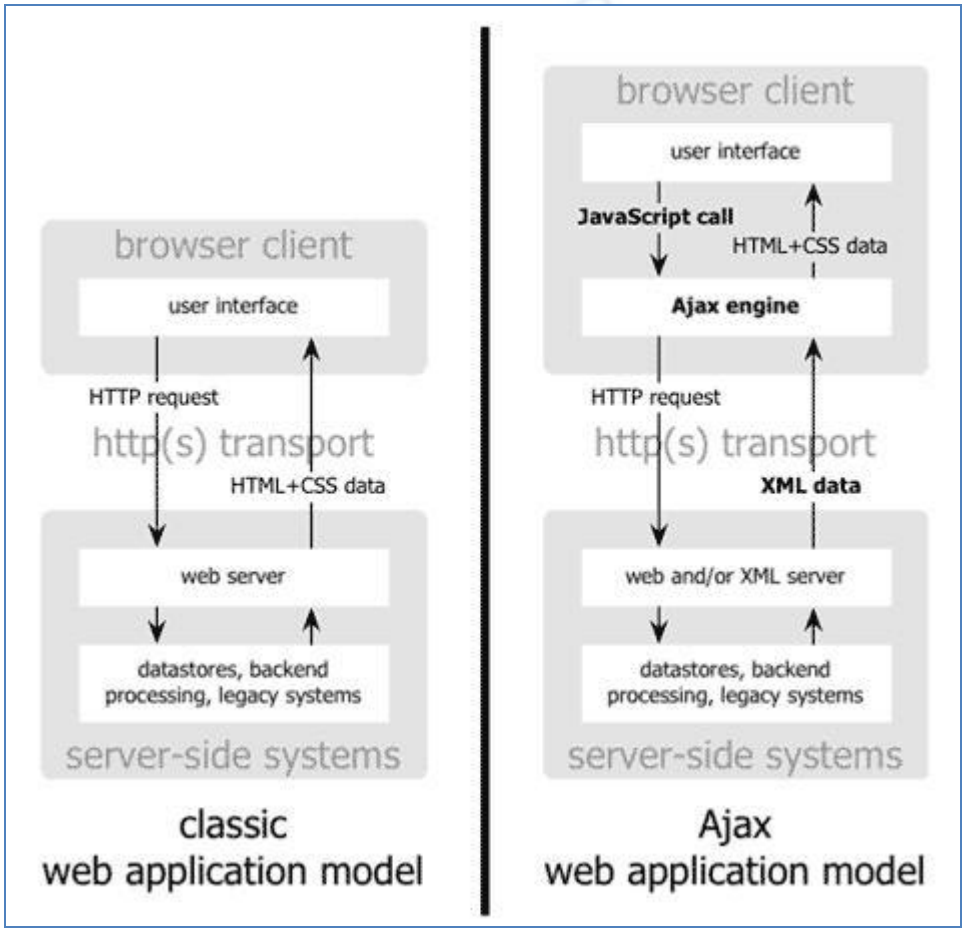
Hình 11.2 Cách thức hoạt động trang Web có Ajax.

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 2 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 gởi request đến Server).
  - Viết hàm gởi yêu cầu (request) đến Server.

- Viết hàm xử lý thông tin sau khi gởi request đến Server. Có thể là thông tin lỗi trả về (nếu việc gởi 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 dung để gởi 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 2 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 gởi 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 XMLHttpRquest để thực hiện gởi 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 kì 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 gởi request, Thí dụ như : id=tên …Mặc 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 gởi 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 gởi 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 4 là request đã được gởi đến Server .

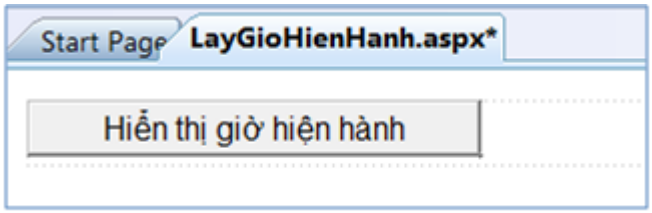
Kế đến ta kiểm tra xem Server có response lại hay chưa với thuộc tính readyState, nếu bằng 200 tức là Server đã response lại hòan tất .

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 1 trang LayGioHienHanh.aspx có giao diện như sau :



Bước 2 . Tạo 1 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"));
        }
    }
}
```

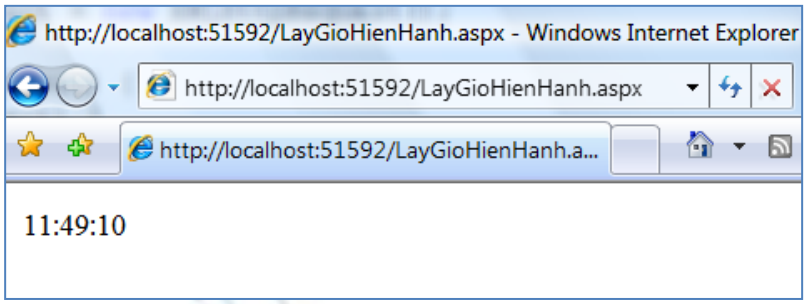
Hình 11.4 Phần mã lệnh trang LayGioHienHanh.aspx

Bước 3. Mở trang LayGioHienHanh.aspx , trong phần mã HTML viết các lệnh javascript như hình sau

```
<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>
```

```
<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** .



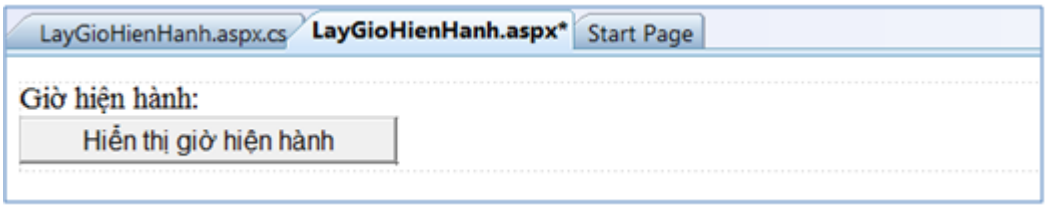
Hình 11.5 Kết quả thực thi.

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 2 trang ASP.Net và viết mã javascript rất nhiều . ASP CallBack hổ trợ xây dựng ứng dụng ASP.Net Ajax một cách đơn giản hơn (chỉ viết trong 1 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>
```

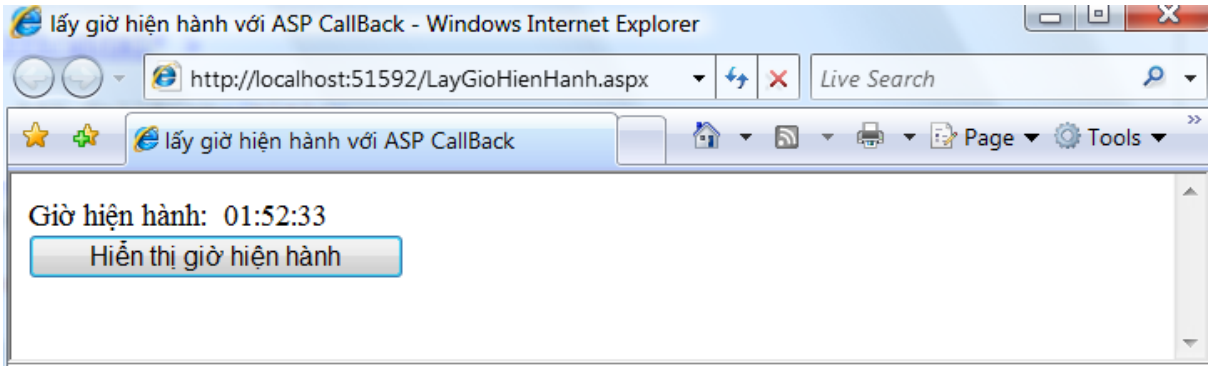
Bước 3 : Viết lệnh cho phía Sever trong tập tin LayGioHienHanh.aspx .cs

LayGioHienHanh.aspx.cs* | Start Page

MinhHoa.LayGioHienHanh | RaiseCallbackEvent(String eventArgument)

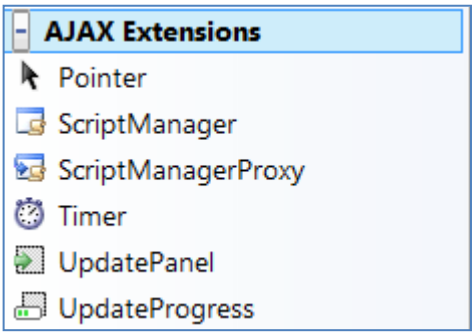
```
namespace MinhHoa
{
    public partial class LayGioHienHanh: System.Web.UI.Page,ICallbackEventHandler
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            //Đăng ký các lệnh CallBack
            String cbReference =
            Page.ClientScript.GetCallbackEventReference(this,
            "", "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 .



11.3 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.

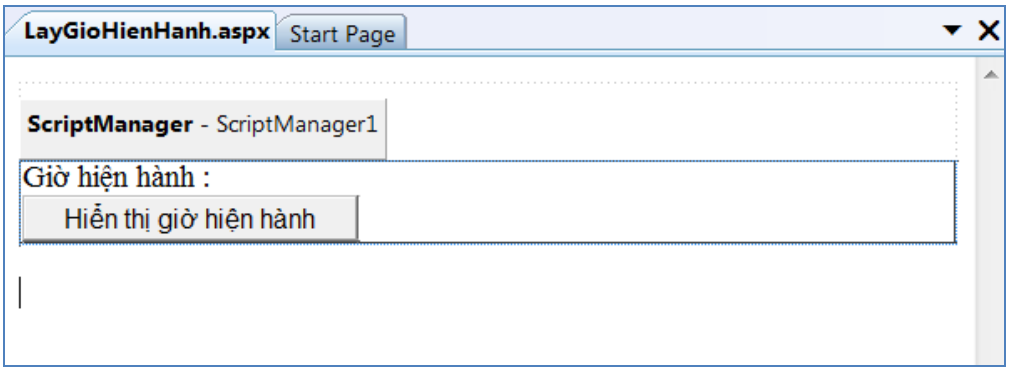


Ajax control gồm các control sau đây :

- **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

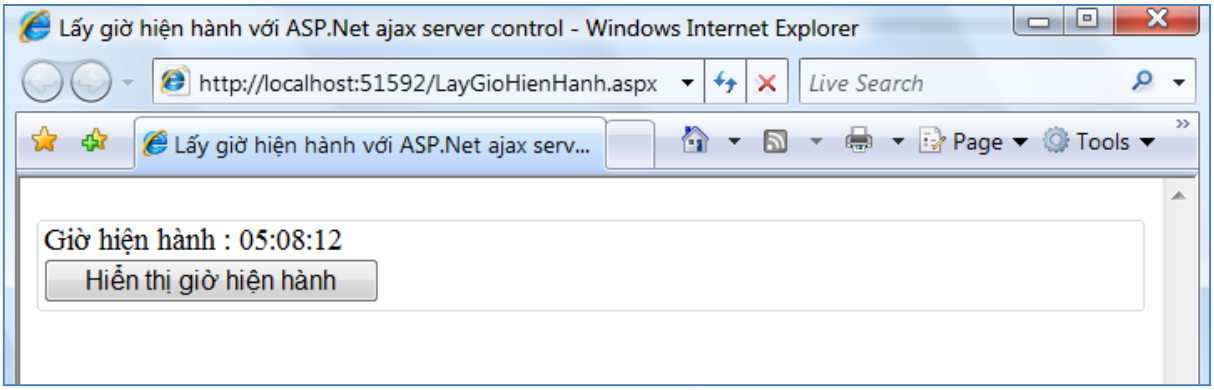
```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Lấy giờ hiện hành với ASP.Net ajax server control</title>
</head>
<body>
    <form id="form1" runat="server">
    <div style="padding-top: 10px">
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <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");
    }
}
```

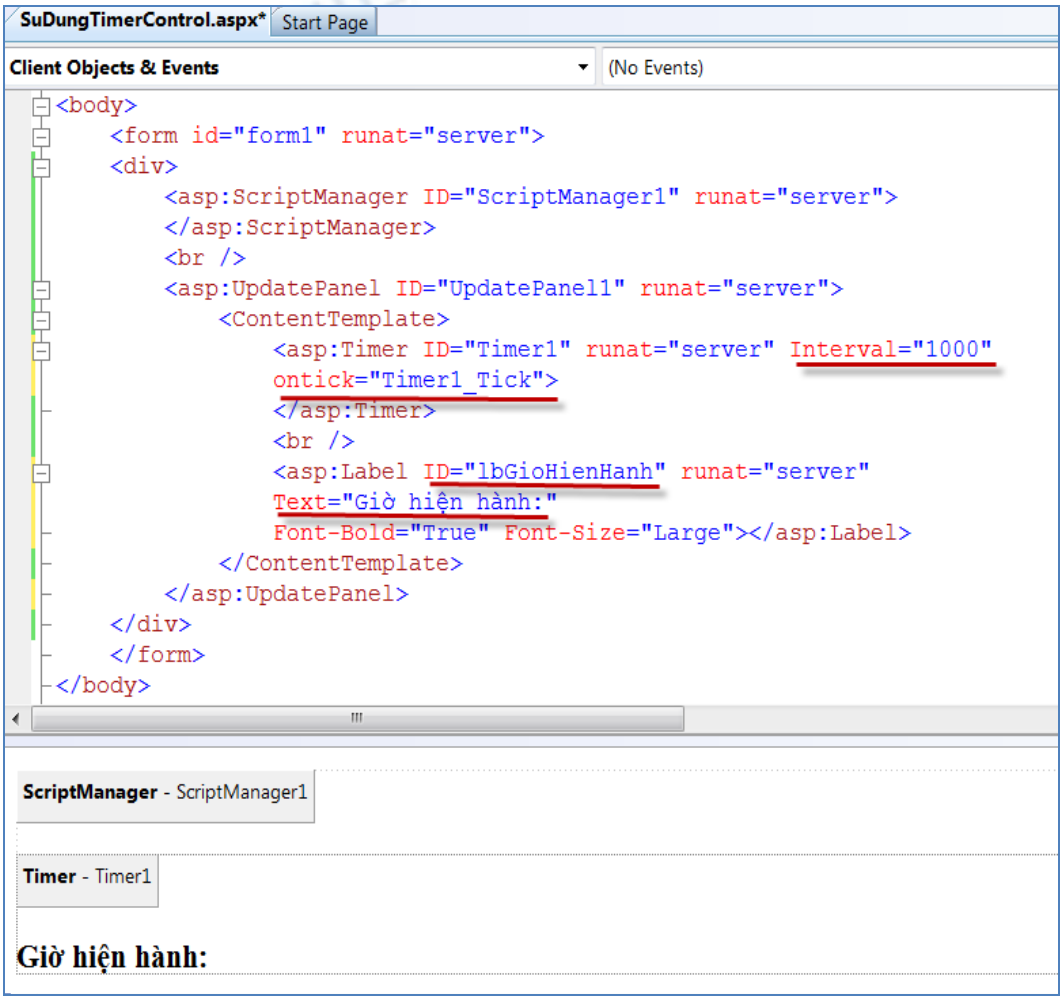
Bước 4 .Nhấn **Ctrl+F5** hoặc **View in Browser** để thi hành . Nhấn nút **Hiển thị giờ hiện hành**

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.

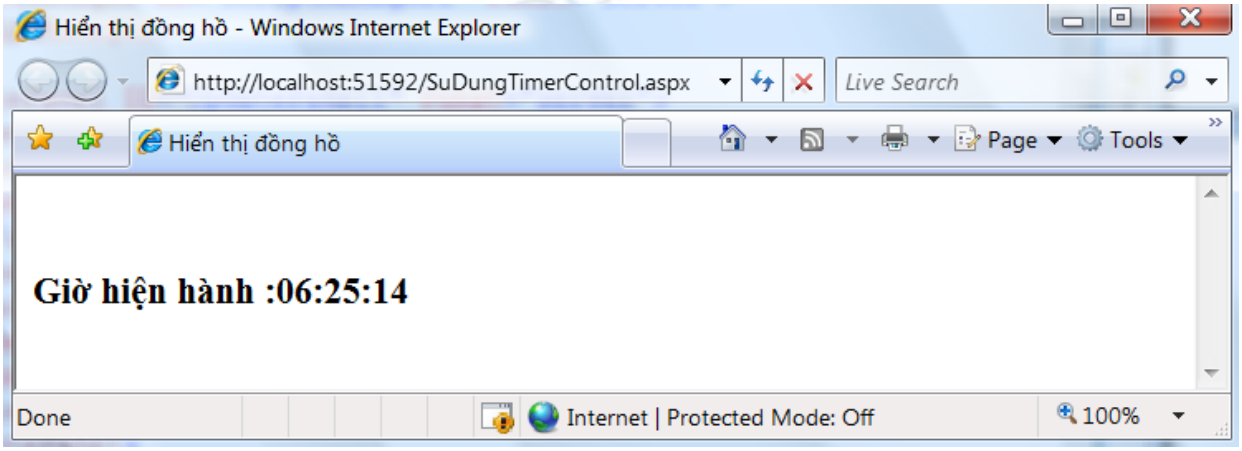
Bước 1: Thiết kế giao diện



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 :