



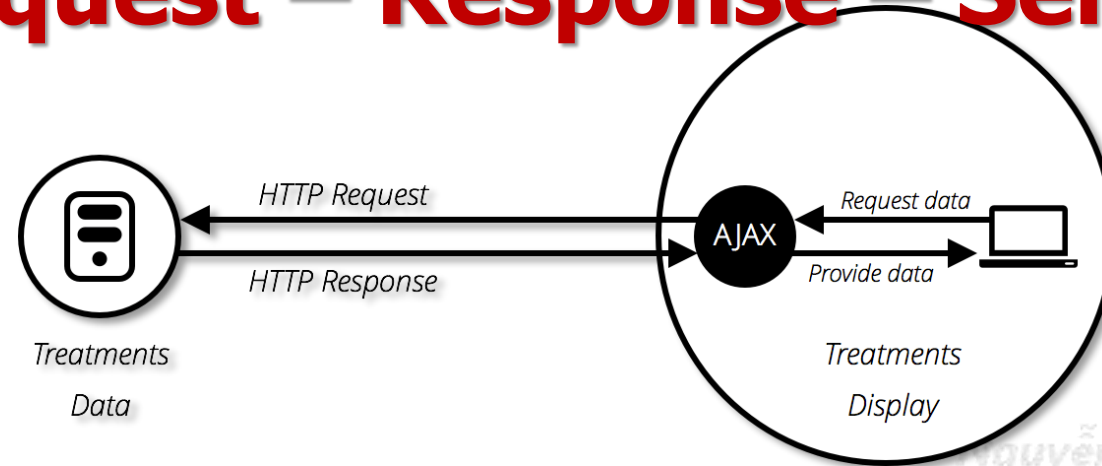
# Lưu trữ và Truy vết

ASP.NET programming with MVC design pattern

Lecture: Nguyễn Mai Huy

[*Week: 8*]

# Đối tượng ngầm định (**Request – Response – Server**)



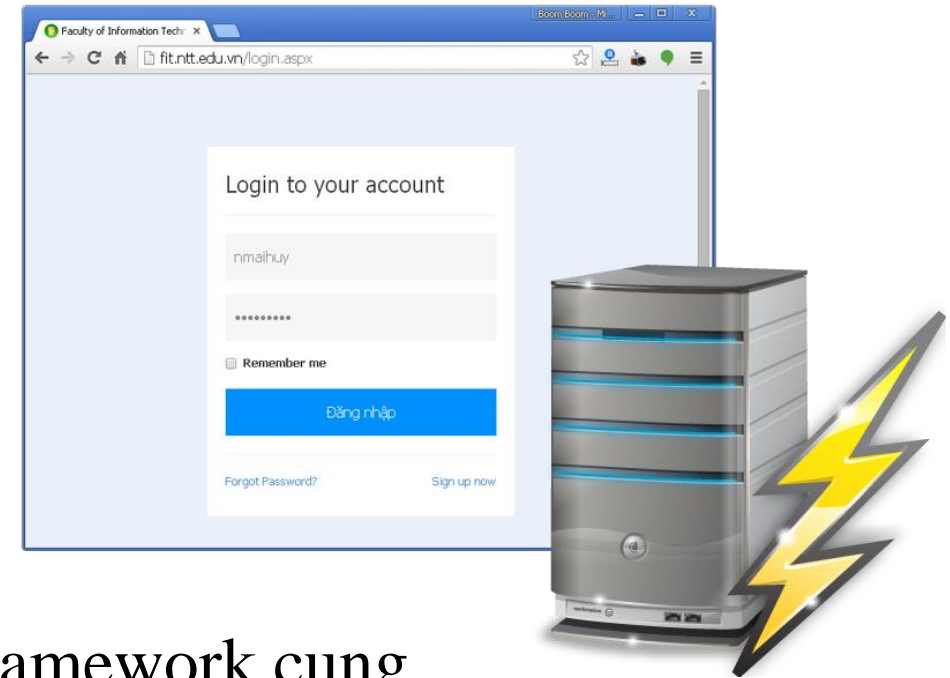
Nguyễn Mai Huy - [nmaihuy@bodua.com](mailto:nmaihuy@bodua.com)

# Tại sao lại cần **Implicit object**?

- Trong quá trình hoạt động của 1 ứng dụng web được xây dựng bởi ASP.NET, các thông tin có liên quan đến quá trình hoạt động của ứng dụng như:
  - Thông tin yêu cầu từ phía người dùng (Client machine)
  - Các đối tượng lưu trữ thông tin về quá trình làm việc trong hệ thống (Session)
  - Các biến toàn cục lưu trữ trạng thái của ứng dụng, ... (*application object*)
  - ...
- Tất cả đều được lưu trữ trong những đối tượng quy ước và xây dựng sẵn trên nền tảng của .NET Framework, người lập trình hoàn toàn có thể khai thác dựa trên các thuộc tính, phương thức của những đối tượng này, chúng bao gồm:
  - ❖ **Request** Object
  - ❖ **Response** Object
  - ❖ **Server** Object

# Request object - **HttpRequestBase** class

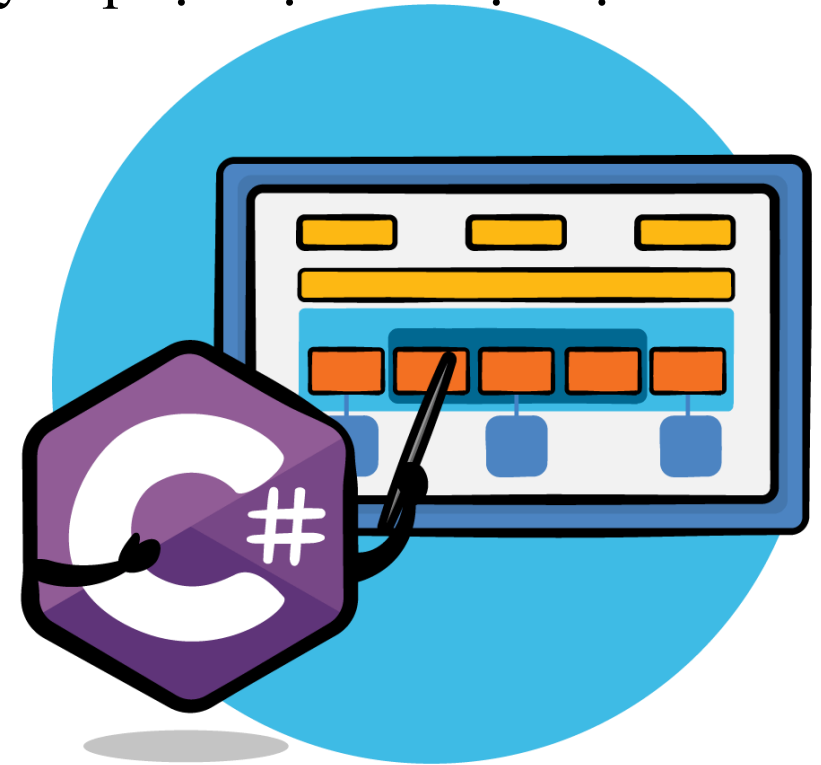
- Như đã giới thiệu trong bài trước, **Request** chính là sự tương tác giữa trình duyệt chạy trên máy tính truy cập với máy chủ web thông qua 1 trang web nào đó thuộc chương trình, URL được gửi đi từ trình duyệt sẽ đại diện cho các yêu cầu gửi về máy chủ
- Để quản lý các tương tác từ phía Client, .NET Framework cung cấp các đối tượng **Request** thuộc lớp **HttpRequestBase** nhằm giúp chương trình web có thể khai thác các thông tin cần thiết từ phía máy truy cập: URL, Browser, IP address, Parameter, ....



Nguyễn Mai Huy - nmai Huy@bodu a.com

# Thuộc tính :: **HttpRequestBase** class

- **HttpRequestBase** là 1 lớp được cung cấp sẵn trong thư viện (namespace) **System.web** thuộc .NET Framework, các thuộc tính của lớp này chủ yếu phục vụ cho việc đọc thông tin từ phía Client.
- Một số các thuộc tính tiêu biểu
  - ✓ `ApplicationPath`
  - ✓ `Browser`
  - ✓ `ContentLength`
  - ✓ `FilePath`
  - ✓ `QueryString`
  - ✓ `RequestType`





# Thuộc tính :: **HttpRequestBase** class

- ✓ **ApplicationPath**: trả về “đường dẫn ảo” (Virtual path) của ứng dụng web
- ✓ **Browser**: Được dùng để truy xuất thông tin về trình duyệt mà người dùng đang sử dụng để truy cập ứng dụng. VD: Tên trình duyệt, phiên bản, nền tảng của trình duyệt (Webkit, Mozilla, ...)
- ✓ **ContentLength**: Trả về giá trị số mô tả cho kích thước của dữ liệu được gửi về từ client
- ✓ **FilePath**: Tương tự như ApplicationPath, tuy nhiên bao gồm cả tên trang aspx cần truy cập
- ✓ **QueryString**: Thuộc tính này trả về giá trị tượng trưng cho tham số của Request (*Request parameter*) với các cặp giá trị theo kiểu “**name=value**”, trong đó ngăn cách giữa các cặp tham số là dấu “&”
- ✓ **RequestType**: Đây là thuộc tính mô tả cho phương thức truyền dữ liệu của form dựa trên HTTP: “**get**” hoặc “**post**”

# Minh hoạ đọc thông tin của trình duyệt

```
/// <summary>
/// Sự kiện minh hoạ khai thác thông tin máy truy cập dựa trên Request Object
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
protected void aaa_Click(object sender, EventArgs e)
{
    string duongDan = Request.ApplicationPath;
    string phuongThuc = Request.RequestType;
    //-- Khai báo đối tượng browser để nhận thông tin trình duyệt của máy truy cập
    HttpBrowserCapabilities browser = Request.Browser;
    string ttTrinhDuyet = browser.Browser + " " + browser.Version;
    //---- In thông báo thông qua Label: ketQua
    this.ketQua.Text = "Đường dẫn truy cập ứng dụng: " + duongDan + "<br/> " +
        "Phương thức: " + phuongThuc + "<br/> " +
        "Trình duyệt của bạn: " + ttTrinhDuyet;
}
```

# Minh họa đọc thông tin từ **Request**

```
//-----  
//--- Implicit object -- [Request - Response - Server] -----  
//-----  
//--- Request object :: HttpRequestBase class -----  
string os = Request.Browser.Platform;           //--- Đọc tên của hệ điều hành -----  
string trìnhDuyet = Request.Browser.Browser;    //--- Đọc tên trình duyệt -----  
string tdVersion = Request.Browser.Version;     //--- Đọc phiên bản của trình duyệt ----  
HttpCookieCollection ck = Request.Cookies;      //--- Đọc Cookies từ máy truy cập ----  
string requestMethod = Request.HttpMethod;      //--- Đọc phương thức của Request ----  
int ts = Request.QueryString.Count;             //--- Đọc số lượng tham số gửi về ----  
string ip =Request.UserHostAddress;              //--- Đọc địa chỉ IP của máy truy cập -  
string hName = Request.UserHostName;            //--- Đọc tên miền của máy truy cập ---  
  
//-----
```

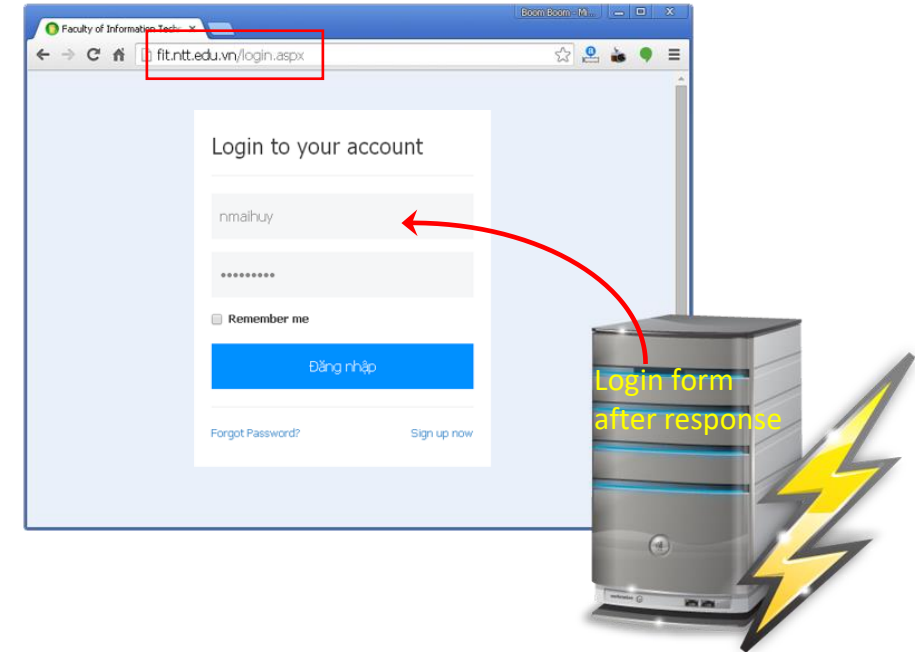


# Phương thức :: **HttpRequestBase** class

- Một số phương thức tiêu biểu của HttpRequest thường được sử dụng bao gồm
  - ✓ **SaveAs()**: Lưu lại nội dung của Request và ghi thành tập tin trên đĩa
  - ✓ **MapPath()**: Trả về chuỗi giá trị mô tả cho “*đường dẫn vật lý*” của trang aspx ở trên đĩa dựa vào “*đường dẫn ảo*” thông qua request

# Response Object - **HttpServletResponse**

- **Response** là đối tượng tự động được tạo ra tương ứng với mỗi **Request** của người dùng, để thực hiện tương tác giữa máy chủ web với trình duyệt trên máy truy cập, tuy nhiên nhiệm vụ của **Response** là chuyển thông tin từ máy chủ web xuống cho client (*Khác với **Request** là chuyển thông tin từ Client về cho máy chủ*)
- ASP.NET cung cấp cho chúng ta các đối tượng Response thuộc lớp **HttpServletResponse** thuộc .NET Framework với các thuộc tính tiêu biểu như:
  - ❖ BufferOutput
  - ❖ CharSet
  - ❖ ContentEncoding
  - ❖ ContentType
  - ❖ Cookies
  - ❖ IsClientConnected



Nguyễn Mai Huy - nmai.huy@bodu.com

# Thuộc tính :: **HttpResponseBodyBase** class

- ✓ **BufferOutput**: Quy định việc có sử dụng bộ đệm để chuyển thông tin từ máy chủ web xuống client hay không
- ✓ **Charset**: Chỉ ra định dạng đối với dữ liệu cần thể hiện trên trình duyệt (Unicode, ISO-8859-1, ...)
- ✓ **ContentType**: Mô tả loại dữ liệu sử dụng đối với nội dung gửi xuống client
- ✓ **Cookies**: Thuộc tính này cho phép tương tác với cookies tại máy truy cập (Lưu thông tin vào cookies của client)
- ✓ **IsClientConnected**: Trả về trạng thái kết nối của Client trong quá trình đang chuyển dữ liệu từ máy chủ web xuống

# Phương thức :: **HttpResponseBase** class

- ✓ **Clear()**: Xoá toàn bộ nội dung trong bộ đệm
- ✓ **Close()**: Đóng kết nối với client
- ✓ **End()**: Gửi toàn bộ dữ liệu hiện có trong buffer xuống client
- ✓ **Redirect()**: Chuyển thi hành tới 1 trang khác thông qua URL được chỉ định
- ✓ **Write()**: Ghi thông tin xuống Client

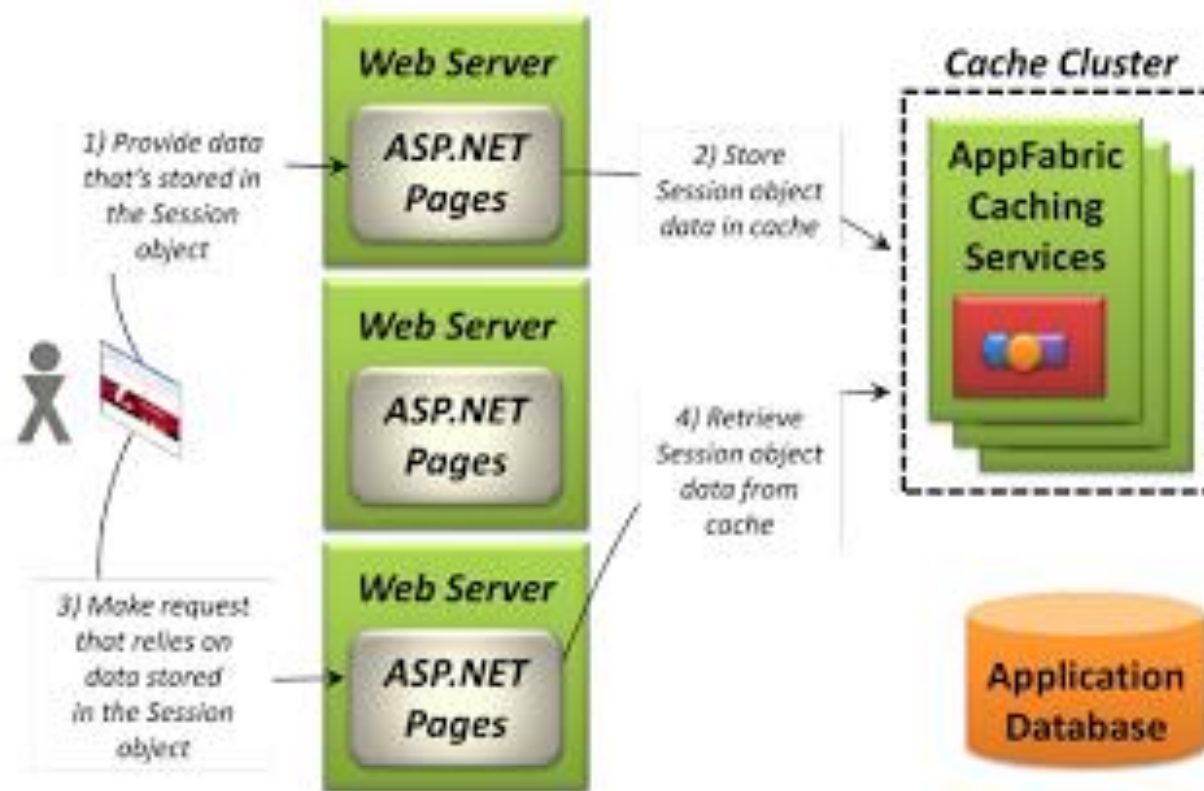
# Minh họa **Response**

```
//-----  
//--- Implicit object -- [Request - Response - Server] -----  
//-----  
//--- HttpCookie :: Ghi dữ liệu trên trình duyệt của khách truy cập -----  
HttpCookie dlGhi = new HttpCookie("DaXem");  
dlGhi.Name = "SpDaXem";  
dlGhi.Value = "Ma_SP_CanGhi";  
//--- Quy định Cookies sẽ hết hạn sau 10 ngày [Tính từ thời điểm hiện tại] -----  
dlGhi.Expires = DateTime.Now.AddDays(10);  
//--- Response object :: HttpResponseBase class -----  
Response.SetCookie(dlGhi);  
//--- Chuyển truy cập về đường dẫn gốc [Home page] của site -----  
Response.Redirect("/");  
  
//-----  
//--- [Request - Response - Server] -----
```



# Server Object - **HttpServerUtilityBase**

**Server** object là đối tượng của lớp **HttpServerUtilityBase** thuộc .NET Framework. **Server** object được trang bị sẵn các phương thức phục vụ cho việc kiểm soát, điều khiển giữa các trang trong ứng dụng web, phân tích HTML gửi về, ...



vi Huy - [nmaihuy@boduac.com](mailto:nmaihuy@boduac.com)

# Các thuộc tính của **HttpServerUtilityBase**

- ✓ **MachineName**: thuộc tính mô tả thông tin của máy chủ web hiện hành
- ✓ **ScriptTimeout**: Dùng để truy xuất thời gian chờ tối đa khi chạy script (*Time out*)

# Phương thức :: **HttpServerUtilityBase** class

- ✓ **Execute()**: Phương thức cho phép gọi thi hành mã lệnh phía Server của 1 trang web.
- ✓ **HtmlEncode()**: Dùng để chuyển đổi chuỗi yêu cầu thành mã HTML
- ✓ **MapPath()**: Trả về “*đường dẫn vật lý*” (*Đường dẫn lưu trữ thực tế*) của trang ASP ở trên đĩa
- ✓ **UrlEncode()**: URL chỉ có thể được gửi qua Internet bằng cách sử dụng bộ ký tự ASCII. Mã hóa URL thay thế các ký tự ASCII không an toàn bằng "%" theo sau là hai chữ số thập lục phân.

URL không được chứa khoảng trắng. Mã hóa URL thường thay thế khoảng trắng bằng dấu cộng (+) hoặc bằng "%20".


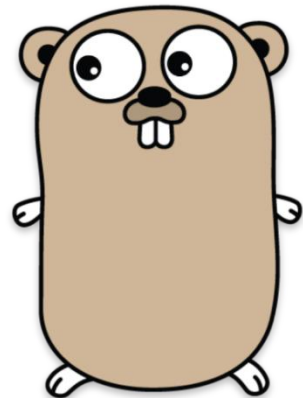
Nguyễn Mai Huy - [nmaihuy@bodu.com](mailto:nmaihuy@bodu.com)

# Minh họa **Server** object

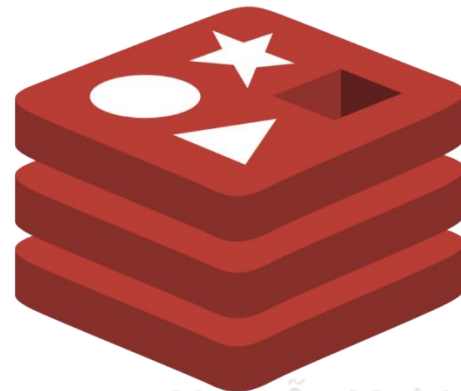
```
//-----  
//--- Implicit object -- [Request - Response - Server] -----  
//-----  
//--- Server object :: HttpServerUtilityBase -----  
string phyPath = Server.MapPath("~/");    //--- Đọc đường dẫn vật lý của site -----  
string name = Server.MachineName;        //--- Đọc tên của máy chủ -----  
Server.Execute("Path");                  //--- Phương thức gọi thi hành ỨD -----
```

# Lưu trữ và Truy vết

(Application – Session – Cookies)



14b2c5cb-d379-40f3-b19e-c8aee093daf  
2d324eeb-4e4a-4235-8bdc-f18f03d2bceb  
86e7b69a-d7b8-4285-9d48-22aa5d6e4c2a  
ad5fa9eb-fce4-4f59-9eb1-1b76da12594e



Nguyễn Mai Huy - [nmaihuy@bodua.com](mailto:nmaihuy@bodua.com)



# Application objects

- Application object là đối tượng phục vụ cho việc *lưu trữ dữ liệu xuyên suốt ứng dụng web*. Các đối tượng dữ liệu được lưu trữ ở dạng này có thể tồn tại trong bộ nhớ trên máy chủ từ khi ứng dụng web bắt đầu thi hành cho đến khi ngừng hoạt động. Có thể nói thời gian tồn tại của Application object được xem là dài nhất, tương ứng với vòng đời của 1 ứng dụng web
- Về cơ bản, Application object là những thể hiện của lớp **HttpApplicationState** class được định nghĩa sẵn trong .NET Framework tại thư viện **System.Web**. Đối tượng của lớp này cho phép lưu trữ dữ liệu khá đa dạng, từ các kiểu dữ liệu cơ bản (*Primitive data type*) như: **int, short, long, float, ...** cho đến dữ liệu ở dạng tham chiếu (*References type*) **string, object, ...** kết cả các kiểu dữ liệu do người dùng tự định nghĩa (*HangHoa, nhanVien, SinhVien, TaiKhoan, ...*)



# Properties :: **HttpApplicationState** class

- ❖ **AllKeys**: cho phép truy xuất danh sách “tên” của các đối tượng Application thuộc ứng dụng hiện hành
- ❖ **Count**: Trả về giá trị số tương ứng với số lượng application object đã được tạo trong ứng dụng
- ❖ **Item**: truy xuất đến các Application object dựa trên chỉ số  
(*Tương tự như mảng hay Collection*)

VD: Ta cần lấy biến “soNguoi” thuộc kiểu **long** từ application, có thể truy xuất như sau

```
long n = (long)Application[“soNguoi”];
```

# Methods :: **HttpApplicationState** class

Một số phương thức đặc thù của Application phục vụ cho việc thêm, xoá các biến dữ liệu đã có

- ❖ **Add()** : Thêm mới 1 Application object vào ứng dụng hiện hành
- ❖ **Clear()**: Xoá bỏ toàn bộ các đối tượng có trong Application object
- ❖ **Remove()**: Xoá 1 application object thông qua tên gọi
- ❖ **RemoveAll()**: Tương tự như Clear()
- ❖ **RemoveAt()**: Xoá 1 application object dựa vào chỉ số (*Vị trí lưu trữ trong danh sách*)

# Lock and Unlock methods

- ❑ Trong một ứng dụng web, Application object được xem như tài nguyên dùng chung cho mọi người dùng của ứng dụng, có thể truy xuất từ bất kỳ trang nào (*Access by any request*). Điều này đôi khi lại không tránh khỏi những xung đột liên quan đến quyền sử dụng chung tài nguyên của ứng dụng web.
- ❑ Ví dụ: Trên thực tế, trong một công ty bán vé máy bay có nhiều đại lý đặt ở nhiều nơi, nhiều vị trí khác nhau (*trên thế giới*). Vấn đề xung đột về sử dụng tài nguyên thường thấy có nguy cơ xảy ra đối với vé đã bán (*Chỗ ngồi*), tức là cùng một thời điểm thì từ nhiều đại lý khác nhau bán cho khách hàng vé của cùng 1 chỗ ngồi.
- ❑ Để giải quyết vấn đề trên, **HttpApplicationState** cung cấp 2 phương thức **Lock()** giúp cho việc ngăn ngừa sử dụng 1 Application object nào đó khi đối tượng này đang có người dùng và **Unlock()** để trả về trạng thái bình thường sau khi đã dùng xong



**Lock()** method



Nguyễn Mai Huy **Unlock()** method

# Cookies Object

**Cookies** là những thông tin ở dạng văn bản được gửi từ máy chủ web xuống và lưu trữ trên máy tính của khách truy cập.

**Cookies** rất hữu ích khi ứng dụng web cần lưu vết người dùng (*Tracking*) khi làm việc trên ứng dụng trong một thời gian dài (*Từng ngày này, qua ngày khác*)

Một số ứng dụng của **Cookies** phổ biến với những trang web bán hàng trực tuyến có thể kể đến như: *Ghi nhận về loại sản phẩm mà người dùng quan tâm; Ghi nhớ mật khẩu đăng nhập; Thống kê sở thích, thói quen của người dùng khi truy cập vào website, ...*

Tuy nhiên việc lạm dụng **Cookies** lại có nguy cơ xâm phạm đến quyền riêng tư của người dùng Internet và điều này có thể dẫn đến “*phạm luật*”



Nguyễn Mai Huy - [nmaihuy@boduca.com](mailto:nmaihuy@boduca.com)



# Ghi cookies

- Tạo cookies và ghi xống máy tính của người dùng có thể được thực hiện bằng một trong các cách sau thông qua đối tượng **Response**:

- Sử dụng **HttpCookies** class

**HttpCookie** lsMua = new **HttpCookie**("lichSu");

lsMua.**Value** = maHang.**ToString**();

lsMua.**Expires** = DateTime.Now.**AddDays**(100);

**Response.Cookies.Add**(lsMua);

- Gán dữ liệu & ghi trực tiếp thông qua **Response** object

**Response.Cookies**["lichSu"].**Value** = maHang.**ToString**();

**Response.Cookies**["lichSu"].**Expires** = DateTime.Now.**AddDays**(100);

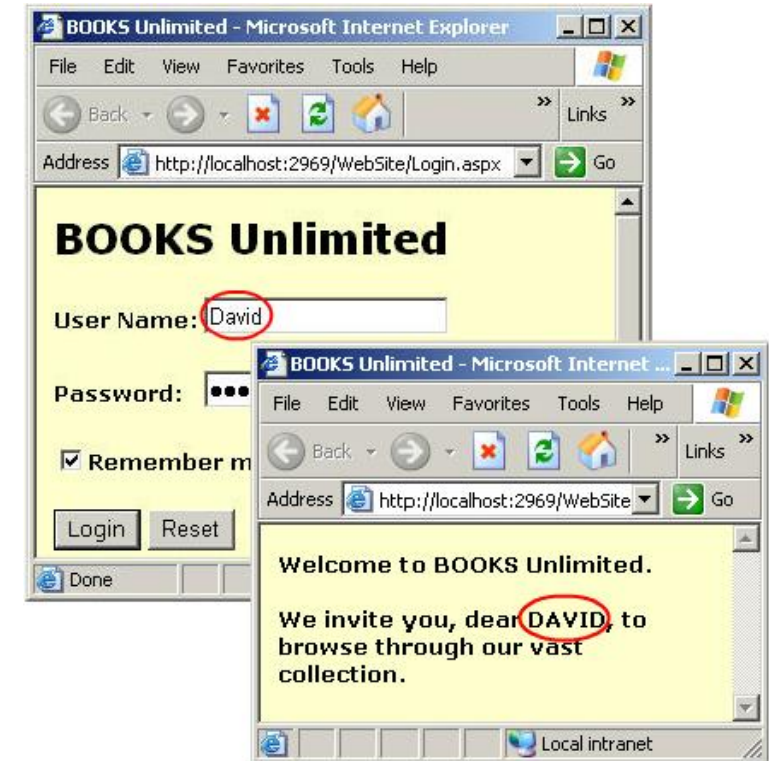
- Chứa thông tin nhiều thành phần

**Response.Cookies**["lichSu"]["maHang"] = "A312";

**Response.Cookies**["lichSu"]["soLuong"] = "5";

**Response.Cookies**["lichSu"]["quaTang"] = "Gấu bông";

**Response.Cookies**["lichSu"].**Expires** = DateTime.Now.**AddDays**(100);



# Reading **cookies**

- Ngược lại với việc tạo, ghi cookies xuống client, để đọc thông tin từ Cookies của máy truy cập, ta sử dụng đối tượng **Request**

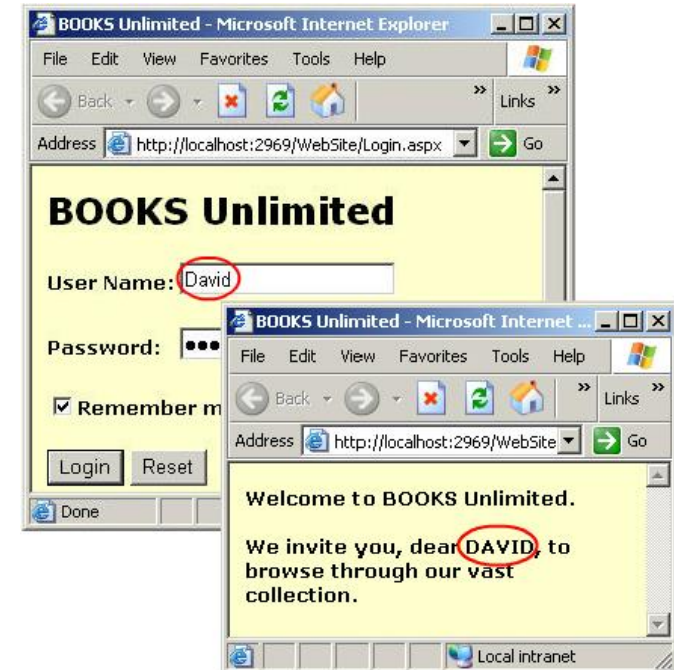
**string** lsMua = **Request.Cookies**['lichSu'].value;

- Đọc cookies nhiều thành phần

**string** mh = **Request.Cookies**["lichSu"]["maHang"];

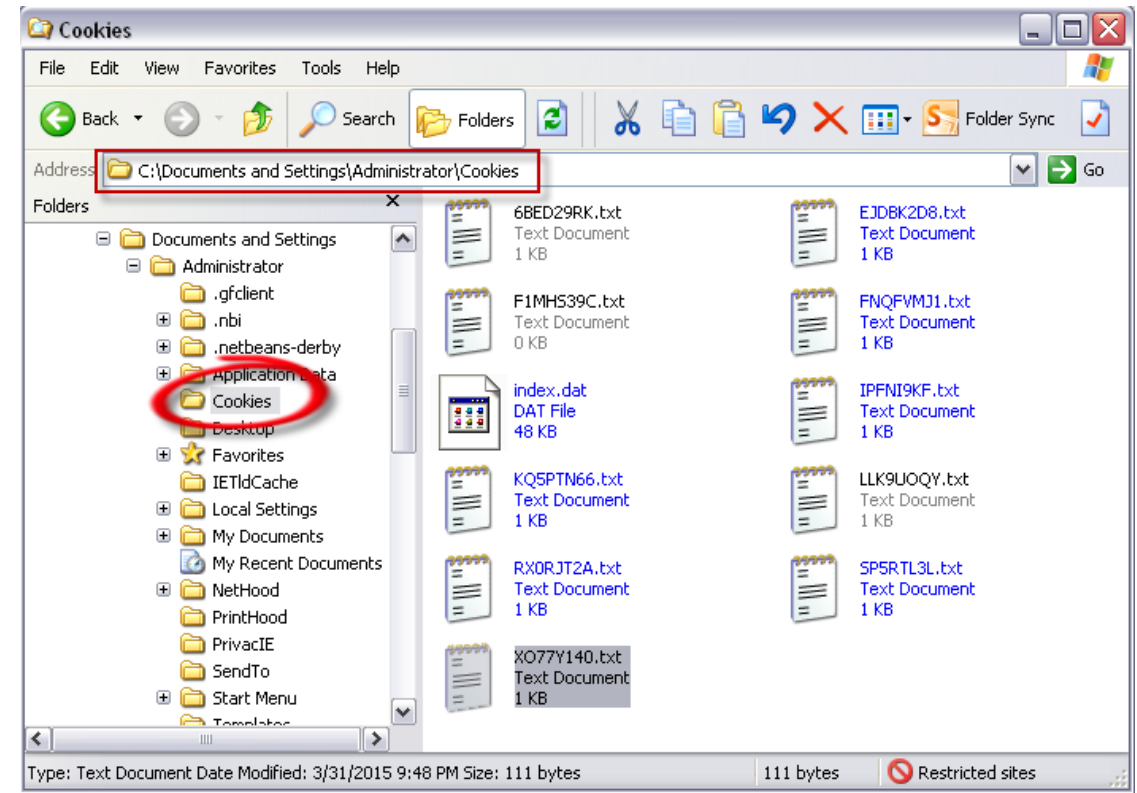
**string** sl = **Request.Cookies**["lichSu"]["soLuong"];

**string** qt = **Request.Cookies**["lichSu"]["quaTang"];



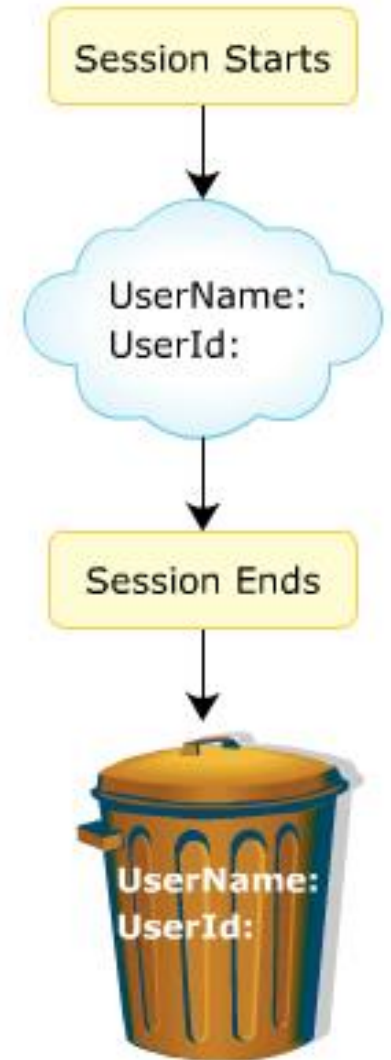
# Storing **cookies**

- Thông tin trong cookies có thời hạn tùy thuộc vào thuộc tính Expires và được lưu trữ trên đĩa cứng của máy truy cập (*Máy tính của người dùng*) tại thư mục cookies, dưới dạng các file Text (\*.txt).
- Thông thường Cookies thường dùng cho mục đích theo dõi thói quen của người dùng khi duyệt web



# Session object

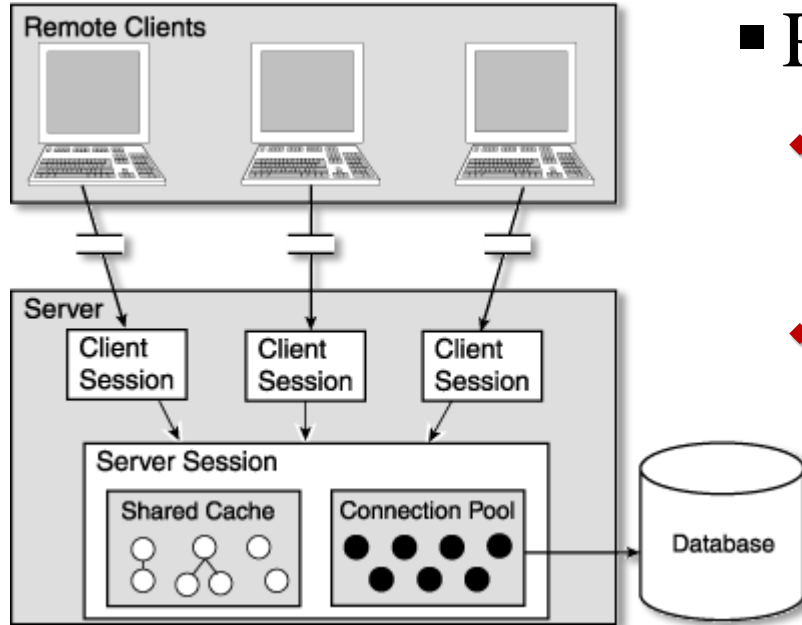
- Khi truy cập và làm việc trên ứng dụng web, người dùng có thể làm việc với đồng thời nhiều trang chức năng khác nhau trong ứng dụng. Làm sao để ghi nhận kết quả làm việc của họ, nhất là đôi khi kết quả làm việc khi làm việc trên trang chức năng này lại được xem là dữ liệu đầu vào cho một chức năng thuộc về trang khác
- VD: Người dùng đặt hàng trên các trang sản phẩm khác nhau (*sanPham.aspx*), tuy nhiên khi tính tiền thì thông tin về các sản phẩm mà họ đã chọn phải được thể hiện trong trang thanh toán (*datHang.aspx*)
- Dĩ nhiên, chúng ta có thể sử dụng application object để lưu trữ thông tin phục vụ cho mục đích này; nhưng nếu làm như vậy thì sẽ rất lãng phí bộ nhớ tại máy chủ, hơn nữa cũng cần phải có cơ chế để xoá, huỷ thông tin người dùng sau khi họ kết thúc quá trình làm việc của mình, điều này sẽ dẫn đến giải thuật phức tạp hơn.
- Giải pháp đưa ra để phục vụ cho việc lưu trữ các thông tin cho nhu cầu trên đối với một ứng dụng web chính là Session object.



Nguyễn Mai Huy -



# Session properties - method



Cũng giống như **Application** object, Session object có thể lưu bất cứ dữ liệu thuộc loại nào từ **Primitive data type** cho đến **reference type**, tuy nhiên, thời gian duy trì các đối tượng thuộc dạng này tối đa chỉ tương ứng với thời gian làm việc của người dùng trên hệ thống (*Tính từ thời điểm truy cập cho đến khi tắt trình duyệt*). **Session** object là những thể hiện của **HttpSessionStateBase** class

## ■ Properties

- ❖ **Count**: Xác định số lượng **Session** Object đang sử dụng trong ứng dụng
- ❖ **IsNewSession**: Trả về giá trị logic tượng trưng cho trạng thái của **Session** object hiện hành. Nếu **Session** mới được tạo thì trả về giá trị true, ngược lại là False
- ❖ **Keys**: Danh sách tên của các **Session** Object hiện có trong hệ thống
- ❖ **SessionID**: Trả về chuỗi giá trị được xem là “định danh” (*Identify*) của **Session** object hiện hành

Nguyễn Mai Huy - [nmai.huy@boduca.com](mailto:nmai.huy@boduca.com)



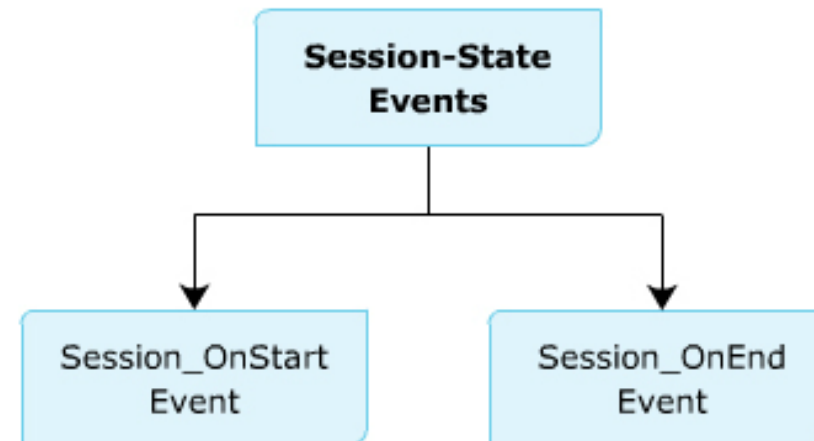
# Session properties - method

Một số phương thức tiêu biểu cho phép bạn Thêm, xoá **Session** object mang ý nghĩa như sau

- **Add**: Thêm, tạo mới 1 **Session** object để lưu trữ thông tin theo phiên làm việc của người dùng trong ứng dụng
- **Clear**: Xoá toàn bộ các **Session** object hiện có
- **Remove**: Xoá 1 **session** object dựa vào tên của nó
- **RemoveAt**: Xoá 1 **Session** object bằng cách chỉ ra vị trí (*Chỉ số*)
- **RemoveAll**: Tương tự như Clear, dùng để xoá tất cả các **Session** object hiện có trong hệ thống
- **Abandon**: Phương thức này dùng để thực hiện kết thúc phiên làm việc của người dùng một cách chủ động. Khi phương thức này được gọi, các sự kiện **Session\_End**, **Session\_start**, cùng các thông số như **SessionID** sẽ tự động được “*làm mới*” lại (*Refresh*)

# Session-State events

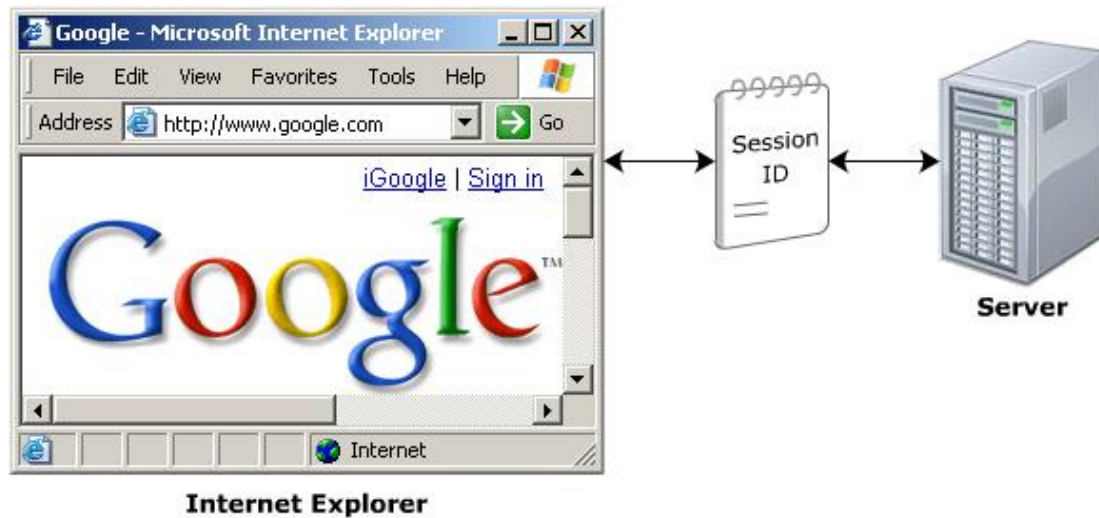
- Có hai sự kiện quan trọng liên quan đến **Session** object
  - **Session\_Start**: sự kiện này xảy ra khi **session** bắt đầu được khởi tạo vào thời điểm người dùng truy xuất và hệ thống
  - **Session\_End**: Đây là sự kiện xảy ra khi người dùng thoát khỏi hệ thống hoặc phiên làm việc của người dùng rơi vào trạng thái **Session** Time-out
- Cả 2 sự kiện trên đều được lưu trữ trong tập tin quản lý biến môi trường của ASP.NET và lập trình viên có thể viết mã lệnh thi hành cho các sự kiện này trong tập tin có tên **Global.asax**



Nguyễn Mai Huy - [nmaihuy@boduca.com](mailto:nmaihuy@boduca.com)

# Session Identifiers

- ❖ Mỗi phiên làm việc của người dùng đều được lưu trữ và quản lý tại Web Server, mỗi Session đều được xác định duy nhất và có định danh riêng mà chúng ta đã biết thông qua thuộc tính **SessionID** đã đề cập ở trên.



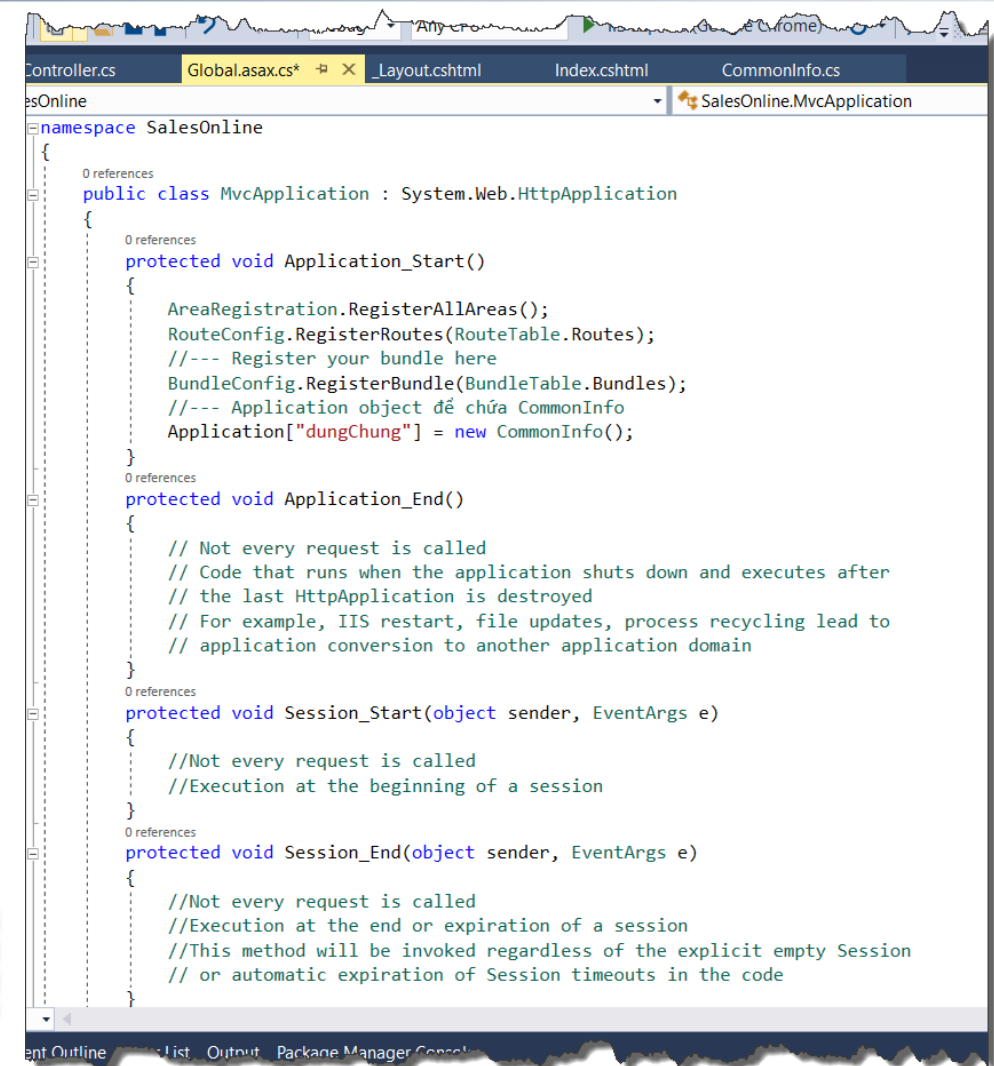
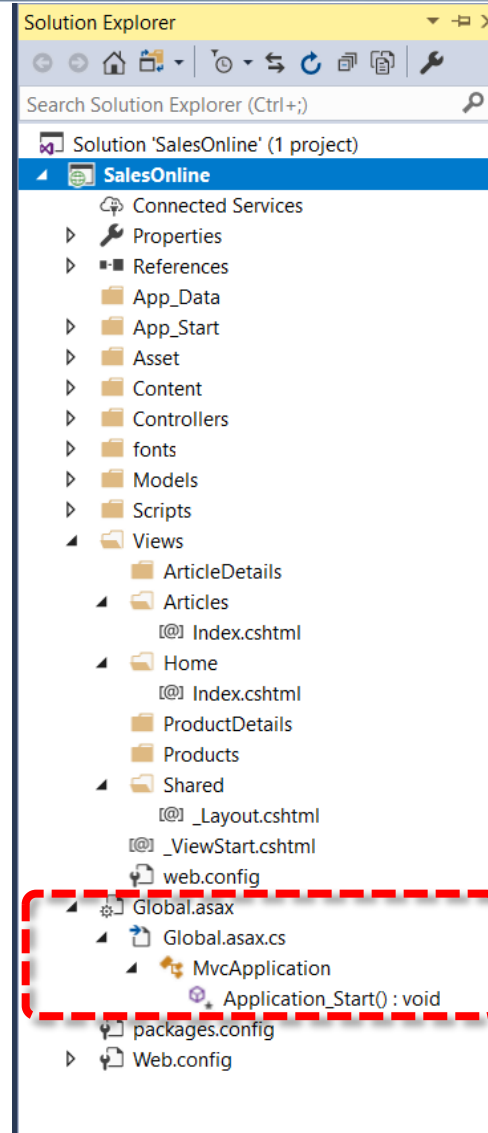
- ❖ Giá trị này sẽ tự động được khởi tạo vào thời điểm người dùng truy cập hệ thống hoặc sẽ tự động được cập lại sau khi phương thức **Abandon** (Nếu có) được gọi thi hành trong hệ thống

Nguyễn Mai Huy - nmaihuy@bodua.com

# Global.asax file

Là Module dùng để lưu trữ và quản lý mã nguồn cho việc xử lý các sự kiện có liên quan đến Application Object và Session object như

- ❖ **Application\_Start**
- ❖ **Application\_End**
- ❖ **Session\_Start**
- ❖ **Session\_End**



Nguyễn Mai Huy - nmai.huy@bodu.com

# Remember what ?!

- ❖ Đối tượng ngầm định [Request – Response - Server]
- ❖ Phương thức và thuộc tính của : Request – Response – Server
- ❖ Application – Session – Cookies
- ❖ Các sự kiện đặc biệt: Application Event – Session Event



# References


- Julia Lerman, “**Programming Entity Framework**”, O’Reilly Media Inc, 2010
- Freeman, Adam, “**Pro ASP.NET MVC 5**”, apress , 2013
- William Penberthy, “**BEGINNING ASP.NET for Visual Studio® 2015**”, John Wiley & Sons Inc, 2016
- **Entity Framework**  
[https://msdn.microsoft.com/en-us/library/aa937723\(v=vs.113\).aspx](https://msdn.microsoft.com/en-us/library/aa937723(v=vs.113).aspx)

# Facebook Group


facebook.com/groups/1508092902561925/

Facebook


Quản lý nhóm


 **Web programming [ASP.NET]**   
Nhóm Riêng tư

← Nếu muốn, bạn có thể chuyển sang Facebook.com thông thường trong 48 giờ để quản lý nhóm.  
[Chuyển ngay](#)


 Trang chủ

Công cụ quản trị

-  Yêu cầu tham gia
-  Tự động phê duyệt thành viên



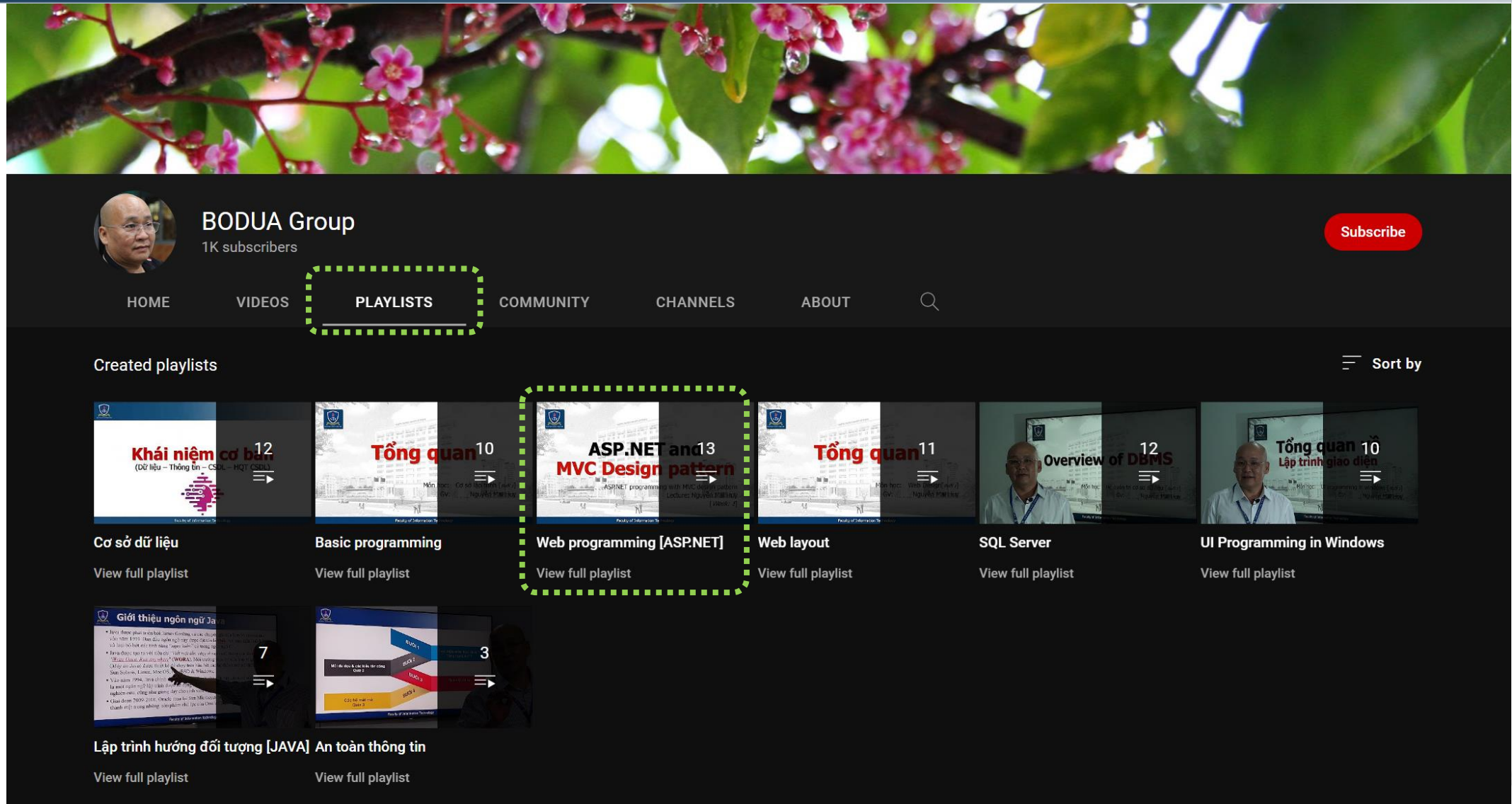
**Web programming [ASP.NET]**  
Nhóm Riêng tư · 378 thành viên

 [+ Mời](#)

[Chỉnh sửa](#)

Nguyễn Mai Huy - [nmai-huy@boduac.com](mailto:nmai-huy@boduac.com)

# Youtube chanel :: Lecture



The image shows a screenshot of the BODUA Group YouTube channel page. The channel has 1K subscribers. The 'PLAYLISTS' tab is selected and highlighted with a green dashed border. Below the tab, there are several video thumbnails, each representing a playlist. The thumbnails are arranged in two rows. The first row includes: 'Khái niệm cơ bản' (12 videos), 'Tổng quan' (10 videos), 'ASP.NET and MVC Design pattern' (13 videos), 'Tổng quan' (11 videos), 'Overview of DBMS' (12 videos), and 'Tổng quan Lập trình giao diện' (10 videos). The second row includes: 'Giới thiệu ngôn ngữ Java' (7 videos), 'Lập trình hướng đối tượng [JAVA]' (3 videos), and 'An toàn thông tin' (3 videos). Each thumbnail has a 'View full playlist' link below it. The 'ASP.NET and MVC Design pattern' thumbnail is also highlighted with a green dashed border. The background of the channel page features a close-up image of pink cherry blossoms.

**BODUA Group**  
1K subscribers

Subscribe

HOME VIDEOS **PLAYLISTS** COMMUNITY CHANNELS ABOUT

Created playlists Sort by

**Khái niệm cơ bản** 12  
(Dữ liệu - Thông tin - CSOL - HQT CSOL)  
View full playlist

**Tổng quan** 10  
View full playlist

**ASP.NET and MVC Design pattern** 13  
View full playlist

**Tổng quan** 11  
View full playlist

**Overview of DBMS** 12  
View full playlist

**Tổng quan Lập trình giao diện** 10  
View full playlist

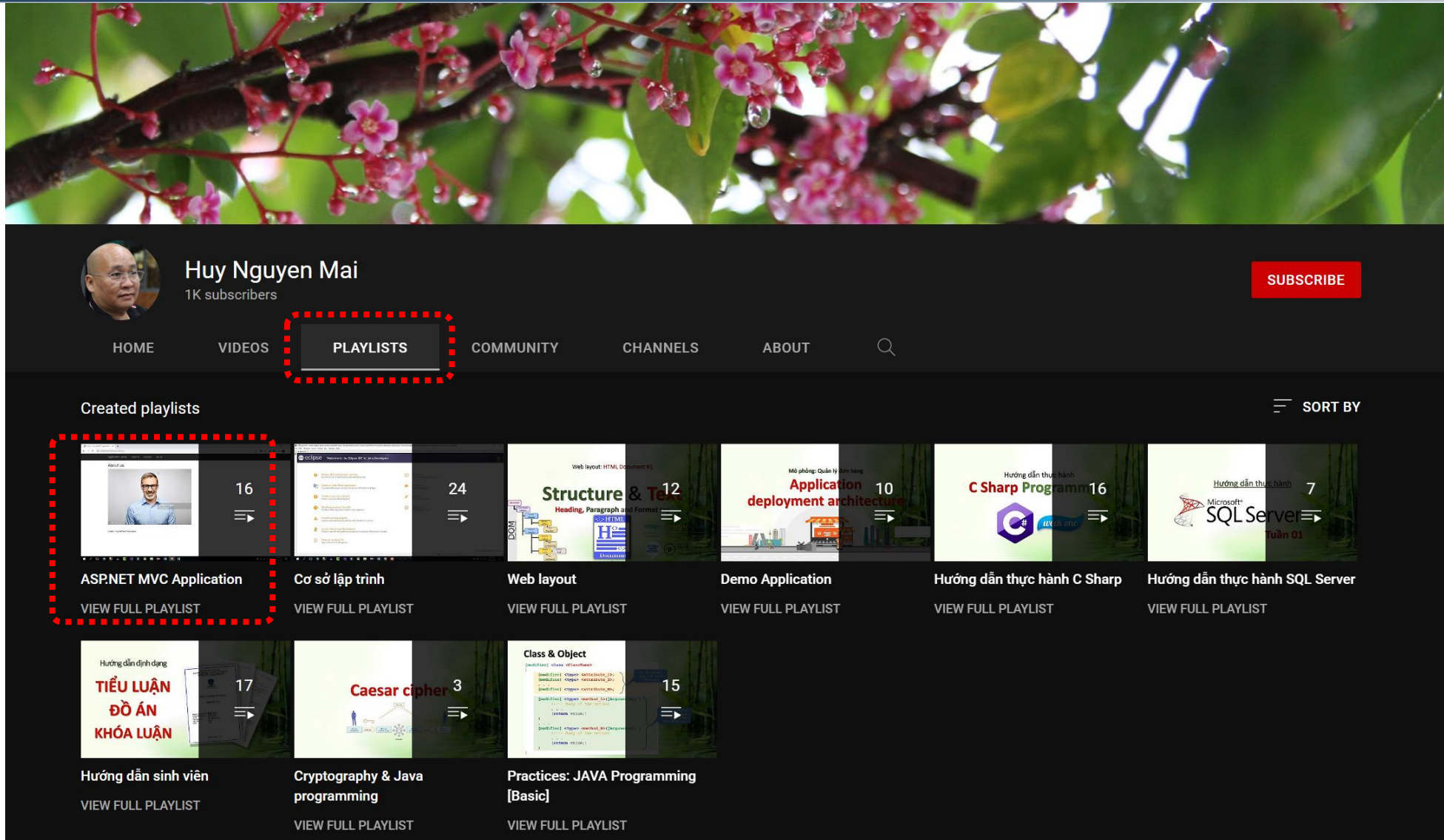
**Giới thiệu ngôn ngữ Java** 7  
View full playlist

**Lập trình hướng đối tượng [JAVA]** 3  
View full playlist

**An toàn thông tin** 3  
View full playlist



# Youtube chanel :: Practice



The image shows a YouTube channel page for "Huy Nguyen Mai" with 1K subscribers. The channel is highlighted with a red dashed border. The page displays a list of created playlists, each with a thumbnail, title, and a "VIEW FULL PLAYLIST" button. The playlists are:

- ASP.NET MVC Application (16 videos)
- Cơ sở lập trình (24 videos)
- Web layout (12 videos)
- Demo Application (10 videos)
- Hướng dẫn thực hành C Sharp (16 videos)
- Hướng dẫn thực hành SQL Server (7 videos)
- Hướng dẫn sinh viên (17 videos)
- Cryptography & Java programming (3 videos)
- Practices: JAVA Programming [Basic] (15 videos)

The "PLAYLISTS" tab is selected, and the "SORT BY" option is visible. The background of the channel banner shows pink cherry blossoms.



**Working hard,  
Success in the future !**

*Nguyễn Mai Huy - [nmaihuy@bodua.com](mailto:nmaihuy@bodua.com)*