BÀI 12. QUẢN LÝ TRẠNG THÁI

Nội dung bài học:

1. Quản lý trạng thái phía Client

Trong quản lý trang phía máy khách, thông tin được lưu trữ trên hệ thống máy khách. Thông tin này sẽ truyền qua lại với mọi request và response đến và đi từ máy chủ.

*Ưu điểm:

Tiết kiệm bộ nhớ máy chủ. Chúng giải phóng máy chủ khỏi gánh nặng lưu giữ thông tin liên quan đến trạng thái.

*Nhược điểm:

Cần nhiều băng thông hơn vì lượng dữ liệu đáng kể được truyền qua lại. Do đó, trang web tải chậm.

Vấn đề chính là nó tạo ra vấn đề bảo mật cho thông tin nhạy cảm như mật khẩu, số thẻ tín dụng, v.v.

ASP.NET cung cấp các các phương pháp bảo toàn trạng thái trang web phía máy khách như sau:

- Hidden Field
- View State (Không hỗ trợ trong MVC)
- Cookies
- Control State (Không hỗ trợ trong MVC)
- Query Strings
- View Data (chỉ có trong MVC)
- View Bag (chỉ có trong MVC)
- Temp Data (chỉ có trong MVC)

1.1. Hidden field

Hidden field được sử dụng để lưu trữ một lượng nhỏ dữ liệu trên hệ thống máy khách. Đây là cách thích hợp hơn khi giá trị của một biến được thay đổi thường xuyên. Hạn chế duy nhất đối với hidden field là nó sẽ giữ thông tin khi bài đăng HTTP đang được thực hiện. Nó sẽ không hoạt động với HTTP get. Ví dụ để lưu trữ trong hidden field trong trang xem MVC như sau:

Mvc4DemoProject.Models.User

```
<div class=”display-field”>
    @Html.HiddenFor(m  =>  m.Id)
</div>
```

1.2. Cookies

Cookie là một tệp văn bản nhỏ được tạo bởi máy chủ và được trình duyệt lưu trữ trên đĩa cứng của máy khách.

Nó không sử dụng bộ nhớ máy chủ. Nói chung cookie được sử dụng để xác định người dùng. Khi người dùng gửi một yêu cầu đến máy chủ, máy chủ sẽ tạo một cookie và đính kèm tiêu đề và gửi lại cho người dùng cùng với phản hồi.

Trình duyệt chấp nhận cookie và lưu trữ nó trên máy khách vĩnh viễn hoặc tạm thời. Lần tiếp theo người dùng đưa ra request cho cùng một trang web, trình duyệt sẽ kiểm tra sự tồn tại của cookie cho trang web đó trong thư mục. Nếu cookie tồn tại, nó sẽ gửi một yêu cầu với cùng một cookie, nếu không request đó sẽ được coi là một request mới. Có 2 loại cookie.

- Để lấy ra 1 cookie:

```
HttpCookie cookie = HttpContext.Request.Cookies.Get("cookie_name");
```

- Để kiểm tra có cookie này hay không:

```
HttpContext.Request.Cookies["cookie_name"] != null
```

- Để lưu trữ cookie:

```
HttpCookie cookie = new HttpCookie("cookie_name");
HttpContext.Response.Cookies.Remove("cookie_name");
HttpContext.Response.SetCookie(cookie )
```

Persistence : cookie được lưu trữ vĩnh viễn cho đến hết thời gian đặt.

Non-Persistence: cookie không được lưu trữ vĩnh viễn trên hệ thống của người dùng. Khi người dùng đóng trình duyệt, cookie sẽ bị xóa.

```
public ActionResult NonPersistenceCookie()
{
   HttpCookie cookie = new HttpCookie(“MyCookie”);
   cookie.Value = “Hello Cookie! CreatedOn: “ + DateTime.Now.ToShortTimeString();
   this.ControllerContext.HttpContext.Response.Cookies.Add(cookie);
   return RedirectToAction(“Index”, “Home”);
}
 public ActionResult PersistenceCookie()
{
   if(this.ControllerContext.HttpContext.Request.Cookies.AllKeys.Contains(“MyCookie”))
   {
     HttpCookie cookie
=  this.ControllerContext.HttpContext.Request.Cookies[“MyCookie”];
     cookie.Expires = DateTime.Now.AddSeconds(10);
     this.ControllerContext.HttpContext.Response.Cookies.Add(cookie);
   }
   return RedirectToAction(“Index”, “Home”);
}
```

Lưu ý: Số lượng cookie được phép thay đổi tùy theo trình duyệt. Hầu hết các trình duyệt cho phép 20 cookie trên mỗi máy chủ trong thư mục đĩa cứng của máy khách và kích thước của cookie không quá 4 KB dữ liệu.

1.3. Query Strings

Một Query String (chuỗi truy vấn) là một biến chuỗi được nối vào cuối URL của trang. Nó có thể được sử dụng để gửi dữ liệu qua các trang. Nó lưu trữ thông tin trong một cặp khóa / giá trị. Dấu “?” chữ ký được sử dụng để nối khóa và giá trị vào URL trang. Trong ứng dụng MVC, chúng ta có thể chuyển các giá trị query string cùng với một id tham số tuyến đường như sau:

```
http://MyDomain/product/Edit/1?name=Mobile.
```

File action có dạng như sau:

```
public ActionResult Edit(int id,string name)
{
    //To Do
    return View();
```

```
}
```

Lưu ý: Hầu hết các trình duyệt đặt giới hạn 255 ký tự cho độ dài URL. Chúng ta nên mã hóa các giá trị truy vấn.

4. ViewData :

ViewData là một đối tượng từ điển có dẫn xuất từ class ViewDataDictionary. Nó sẽ có thể được truy cập được bằng cách sử dụng các chuỗi dưới dạng key/value như sau:

```
public ActionResult Index()
{
  ViewData[“hasPermission”] = true;
  return View();
}
```

ViewData chỉ nằm trong request hiện tại từ controller đến view tương ứng. Nếu có sự chuyển hướng thì giá trị của nó sẽ trở thành null. Nó yêu cầu có chuyển kiểu khi nhận dữ liệu trong view:

```
@{
   bool hasPermission = (bool)ViewData[“hasPermission”];
}
```

1.4. ViewBag

Đối tượng ViewBag là một thuộc tính động được bao bọc xung quanh đối tượng ViewData. Thuộc tính động là một tính năng mới trong Ngôn ngữ động ASP.NET. Chúng ta đơn giản là chỉ cần đặt một thuộc tính động trong cho ViewBag trong controller.

```
public ActionResult Index()
{
  ViewBag.HasPermission = true;
  return View();
}
```

ViewBag cũng chỉ tồn tại trong request hiện tại từ controller tới view. Nếu có chuyển hướng thì giá trị của nó sẽ trở thành null. ViewBag không yêu cầu chuyển kiểu để nhận dữ liệu trong view.

```
@if(ViewBag.HasPermission)
{
  //do somethings….
}
```

* **Lưu ý**: Cả ViewData và ViewBag gần như tương tự nhau và giúp duy trì dữ liệu. Chúng cung cấp một cách để giao tiếp giữa controller và view.

1.5. TempData

- TempData là một đối tượng từ điển lưu trữ dữ liệu dưới dạng cặp khóa / giá trị và có dẫn xuất từ class TempDataDictionary. TempData Giúp duy trì dữ liệu khi chúng ta di chuyển từ bộ controller này tới controller khác hoặc từ action này tới action khác.
- Nói cách khác, nó giúp duy trì dữ liệu giữa các lần chuyển hướng.
- RedirectToAction không ảnh hưởng đến TempData cho đến khi TempData được đọc. Sau khi TempData được đọc, các giá trị của nó sẽ bị mất.

```
public class ViewController : Controller
{
  public ActionResult Index()
  {
    TempData[“MyData”] = “Hello! Wecome”;
    return RedirectToAction(“About”);
  }

  public ActionResult About()
  {
    return RedirectToAction(“Test1”);
  }

  public ActionResult Test1()
  {
    string Str = Convert.ToString(TempData[“MyData”]);
    TempData.Keep(); // Keep TempData
    return RedirectToAction(“Test2”);
  }

  public ActionResult Test2()
  {
    string Str = Convert.ToString(TempData[“MyData”]); //OutPut
    return View();
  }
}
```

*__Lưu ý:__ Phương thức TempData.Keep () được sử dụng để lưu trữ dữ liệu trong controller đến thời điểm chúng ta mong muốn.

2. Quản lý trạng thái phía Server

Trong quản lý trạng thái phía máy chủ, chúng lưu trữ tất cả thông tin trong bộ nhớ máy chủ.

Ưu điểm:

Ưu điểm chính của việc quản lý trạng thái kiểu này là nó bảo mật thông tin nhạy cảm và bí mật của người dùng.

Nhược điểm:

Nhược điểm của của việc quản lý trạng thái phía server là sử dụng nhiều bộ nhớ máy chủ hơn.

ASP.NET cung cấp các các phương pháp bảo toàn trạng thái trang web phía máy chủ sau:

- Session state
- Application state
- Profile Properties
- Cache

Ngoài các phương pháp quản lý trạng thái trên, một số phương pháp không được ứng dụng ASP.NET MVC hỗ trợ như view state, control, v.v. Và một số phương pháp chỉ áp dụng trong ứng dụng MVC.

2.1. Session

- Trong ASP.NET MVC, Session quản lý để lưu trữ và truy xuất các giá trị cho người dùng khi người dùng điều hướng giữa các chế độ view. Nói chung, session được sử dụng để lưu trữ thông tin của người dùng, để xác định duy nhất một người dùng.
- Máy chủ duy trì trạng thái thông tin người dùng bằng cách sử dụng ID của session. Khi người dùng đưa ra request mà không có ID session, ASP.NET tạo ID session và gửi nó với mỗi request và response cho người dùng đó.
- Trạng thái session được lưu trữ trong cặp khóa/giá trị.

```
// Model
namespace MvcSessionApp.Models
{
  public class UserAccount
  {
    public string UserName { get; set; }
    public string UserType { get; set; }
    public string Email { get; set; }
  }
}
//Controller
namespace MvcSessionApp.Controllers
{
  public class UserAccountController : Controller
  {
    //
    // GET: /UserAccount/
    public ActionResult Index()
    {
      UserAccount ucObj = new UserAccount();
      ucObj.UserName = “Biswa”;
      ucObj.UserType = “Admin”;
```

```
      ucObj.Email = "biswa@session.com";
      Session["AdminUser"] = ucObj;
      return View();
    }
  }
}
//View
@model MvcSessionApp.Models.UserAccount
@{
   ViewBag.Title = "Index";
}
@{
   var user = (MvcSessionApp.Models.UserAccount)Session["AdminUser"];
}
<h2>Index</h2>
<fieldset>
  <legend>UserAccount</legend>
  <div class="display-label">
    @Html.DisplayNameFor(model => model.UserName) : @user.UserName
  </div>
  <div class="display-label">
    @Html.DisplayNameFor(model => model.UserType) : @user.UserType
  </div>
  <div class="display-label">
    @Html.DisplayNameFor(model => model.Email) : @user.Email
  </div>
</fieldset>
```

2.2. Profile properties

- ASP.NET cung cấp các thuộc tính profile (file hồ sơ), cho phép chúng ta lưu trữ dữ liệu tùy chỉnh của người dùng theo cách thức thuận tiện của người dùng.
- Nó tương tự như quản lý trạng thái session, ngoại trừ dữ liệu profile không bị mất khi session của người dùng hết hạn.
- Đặc điểm của profile là được sử dụng để lưu trữ các thuộc tính cố định và được liên kết với một người dùng.
- Để sử dụng thuộc tính của profile, trước tiên chúng ta cần bật cấu hình bằng cách sửa đổi tệp cấu hình Web.Config.
- Trong đó chúng ta cần chỉ định một nhà cung cấp profile, là class phía dưới thực hiện các nhiệm vụ cấp thấp là lưu trữ và truy xuất dữ liệu profile.
- Chúng ta có thể lưu trữ dữ liệu profile người dùng trong máy chủ SQL.

```
<profile defaultProvider="DefaultProfileProvider">
   <providers>
       <add name="DefaultProfileProvider"
type="System.Web.Providers.DefaultProfileProvider,
System.Web.Providers, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" connectionStringName= "DefaultConnection"
 applicationName="/" />
   </providers>
</profile>
```

2.3. Caching

- Bộ nhớ đệm cung cấp cách lưu trữ dữ liệu được truy cập thường xuyên và sử dụng lại dữ liệu đó.
- Là một cách hiệu quả để cải thiện hiệu suất của ứng dụng web. Bộ đệm đầu ra cho phép chúng ta lưu vào bộ đệm nội dung được trả về bởi một action của controller.
- Bằng cách đó, nội dung giống nhau sẽ không cần phải tạo ra mỗi khi cùng một action của controller được gọi.
- Bộ lọc OutputCache cho phép lưu vào bộ nhớ cache dữ liệu là output của một phương thức action.
- Theo mặc định, bộ lọc (filer) thuộc tính này lưu dữ liệu vào bộ nhớ cache cho đến 60 giây. Sau 60 giây, Asp.Net MVC sẽ thực thi lại phương thức action và lưu lại output vào bộ nhớ cache.
- Chúng ta có thể kích hoạt bộ nhớ đệm output cho một phương thức action hoặc controller bằng cách thêm thuộc tính [OutputCache] như hình sau:

```
[OutputCache(Duration=10, VaryByParam="none")]
public ActionResult Index()
{
   return View();
}
```

3. Demo sử dụng Session

Viết code mỗi lần click vào link 'Refresh' giá trị sẽ tăng lên 1. Sử dụng session để bảo toàn giá trị của biến.

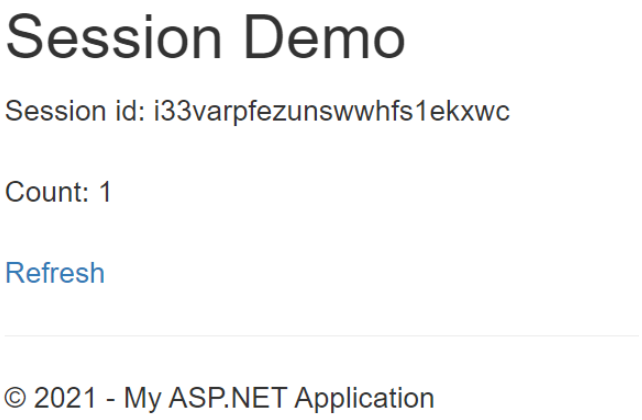
Tạo Project web mvc mới, tạo file controller SessionDemo có nội dung sau:

```
0 references
public class SessionDemoController : Controller
{
    // GET: SessionDemo
    0 references
    public ActionResult Index()
    {
        if (Session["count"] == null)
        {
            Session["count"] = 0;
        }
        int c = (int)Session["count"];
        c++;
        Session["count"] = c;

        return View();
    }
}
```

Tạo view cho method Index:

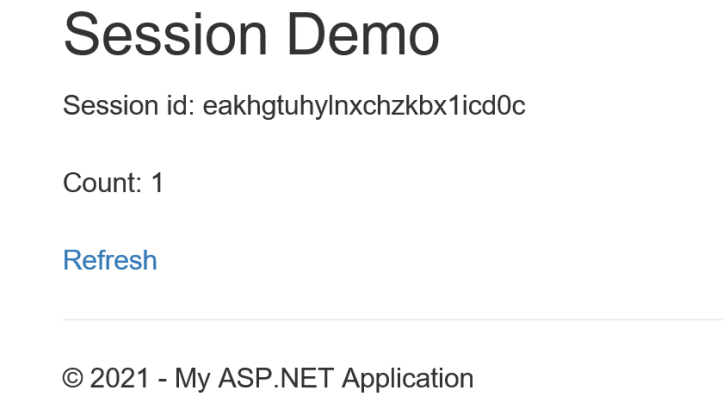
```
@{
    ViewBag.Title = "Index";
}

<h2>Session Demo</h2>

Session id: @Session.SessionID <br /> <br />

Count: @Session["count"]
<br /> <br />

@Html.ActionLink("Refresh","Index","SessionDemo")
```

Kết quả nhận được:



Mỗi khi click vào link Refresh, giá trị của Count sẽ tăng lên 1.

Mở trình duyệt IE và paste đường link vừa thực hiện, chúng ta thấy một session mới được tạo ra với id khác, và Count lại bắt đầu từ giá trị 1.



4. Demo sử dụng Application

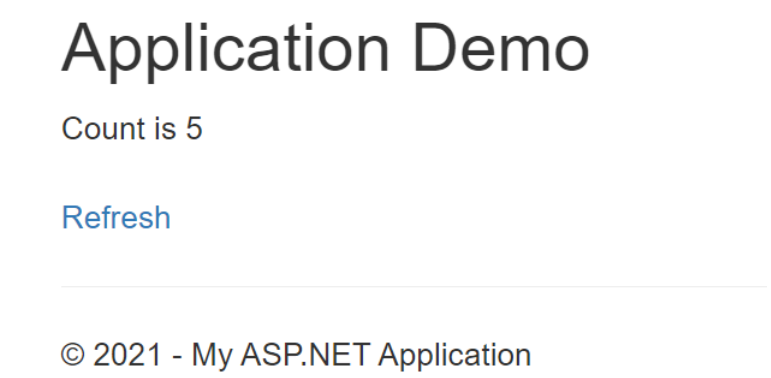
Tạo controller ApplicationDemo và viết code như sau:

```
0 references
public class ApplicationDemoController : Controller
{
    // GET: ApplicationDemo
    0 references
    public ActionResult Index()
    {
        if (HttpContext.Application["count"] == null)
        {
            HttpContext.Application["count"] = 0;
        }
        int c = (int)HttpContext.Application["count"];
        c++;
        HttpContext.Application["count"] = c;

        ViewBag.app = c;

        return View();
    }
}
```

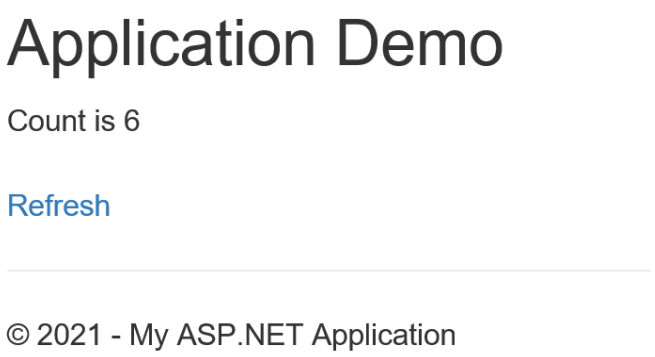
Tạo view cho action Index:

```
@{
    ViewBag.Title = "Index";
}

<h2>Application Demo </h2>

Count is @ViewBag.app
<br /> <br />

@Html.ActionLink("Refresh","Index","ApplicationDemo")
```

Chạy page view và click vài lần vào link Refresh:



Copy link và paste sang trình duyệt IE:



Giá trị Count được tăng lên 6.

Như vậy biến application có tác dụng trong phạm vi ứng dụng.

5. Demo login vào hệ thống

Tạo controller LoginDemo có nội dung sau:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Proj_StateManagement.Controllers
{
    public class LoginDemoController : Controller
    {
        // GET: LoginDemo
        public ActionResult Index()
        {
            return View();
        }

        [HttpPost]
        public ActionResult CheckLogin(string username, string password)
        {
            if (username == "admin" && password == "123456")
            {
```

```
                Session["user"] = username;
                return RedirectToAction("Index", "Home");
            }

            if (username == "")
                ViewBag.msg = "Username cannot be empty.";
            else
                ViewBag.msg = "Invalid username or password.";

            return View("Index");
        }

        public ActionResult Logout()
        {
            Session.Remove("user");
            return View("Index");
        }
    }

}
```

Tạo file view.cshtml có nội dung:

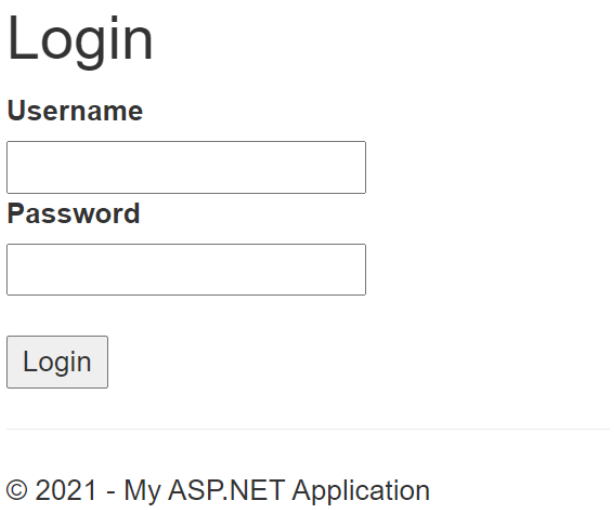
```
@{
    ViewBag.Title = "Index";
}
<h2>Login</h2>
@using (Html.BeginForm("CheckLogin", "LoginDemo"))
{
    <div>
        @Html.Label("Username")
    </div>
    <div>
        @Html.TextBox("username")
    </div>
    <div>
        @Html.Label("Password")
    </div>
    <div>
        @Html.Password("password")
    </div>
    <br />
    <input type="submit" value="Login" />
    <br />
    @ViewBag.msg
}
```

Mở file Views/Home/Index.cshtml, gõ vào nội dung sau:

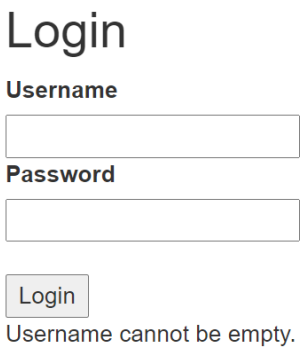
```
@{
    ViewBag.Title = "Home Page";
}
 <div>
     <h2>Home Page</h2>
     @if (Session["user"] == null)
     {
```

```
            <p>@Html.ActionLink("Login please","Index","LoginDemo")</p>
        }
        else
        {
            <br />
            <h3 style="color:blue">Welcome @Session["user"]</h3>
            @Html.Label("Today is")<text> </text>@DateTime.Now.DayOfWeek
            <br/>
            <br />
            @Html.ActionLink("Logout","Logout","LoginDemo")
        }

    </div>
```

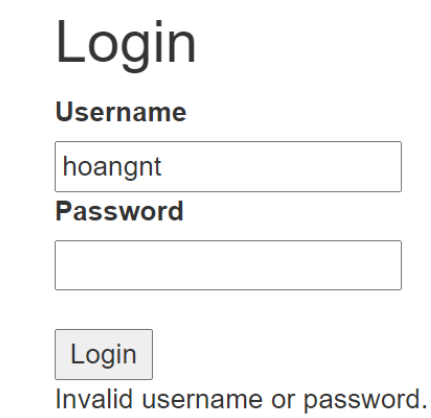
Chạy page view:



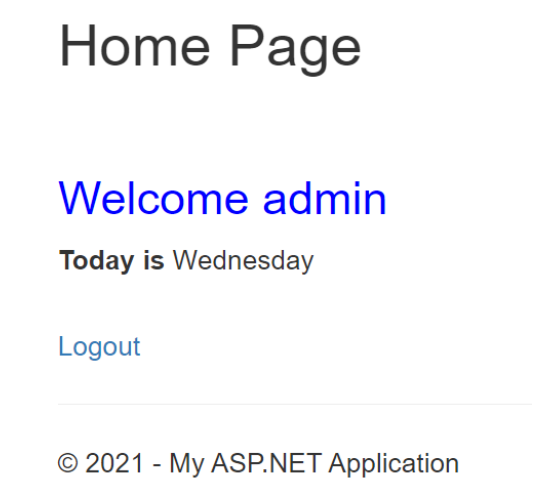
Nếu không nhập đủ dữ liệu:



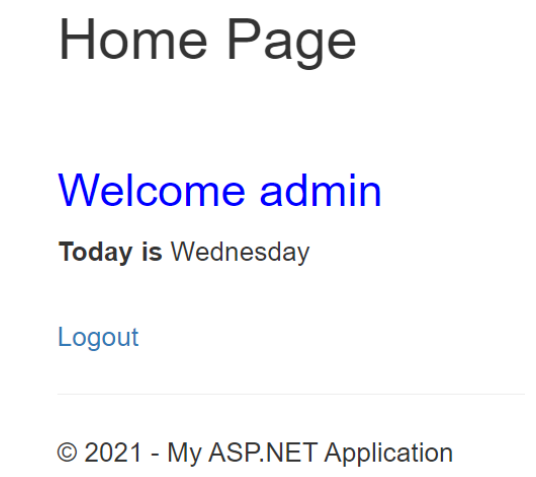
Nếu nhập sai:



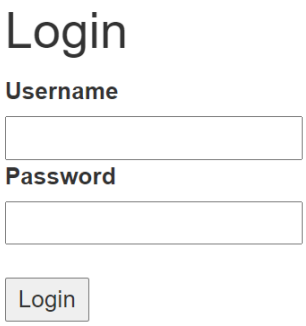
Nếu nhập đúng username và password:



Copy sang tag mới, kết quả nhận được vẫn là:



Khi click vào link Logout tại 1 page thì xuất hiện cửa sổ login:



Và quay về page kia, ấn refresh, trạng thái page cũng sẽ thay đổi theo:

Nếu bắt đầu từ trang Views/Home/index.cshtml thì sẽ không hiện thông tin trong trang này mà sẽ nhận được thông báo như hình trên. Chỉ khi đăng nhập đúng thì mới hiển thị thông tin trong trang index.cshtml này.

6. Hướng dẫn bài tập Mua bán hàng online.

Trước hết chúng ta tạo class **Sanpham** như sau:

```
public class Sanpham
   {
        public string masp { get; set; }
        public string tensp { get; set; }

        public string hinhanh { get; set; }
        public int giatien { get; set; }
        public Sanpham()
        {
        }
        public Sanpham(string masp)
        {
            this.masp = masp;
        }

        public override bool Equals(object obj)
        {
            Sanpham s = (Sanpham)obj;
            return (this.masp == s.masp);
        }
        public Sanpham(string masp, string tensp, string hinhanh, int giatien)
        {
            this.masp = masp;
```

```
            this.tensp = tensp;
            this.hinhanh = hinhanh;
            this.giatien = giatien;
        }
    }
```

Lưu ý: Viết lại phương thức Equals để so sánh 2 đối tượng sản phẩm là bằng nhau khi mã của chúng giống nhau.

Sau đó tạo tiếp class **SanphamMua**, class này chỉ có 2 thuộc tính là masp (mã sản phẩm) và soluong (số lượng). Class này sẽ được dùng để ghi nhớ mã của sản phẩm và số lượng của sản phẩm khi người dùng chọn mua.

Khi lập đơn hàng hoặc cho khách hàng xem chi tiết sản phẩm họ đã chọn thì chúng ta sẽ đối chiếu mã sản phẩm mua này với mã sản phẩm đã có ở trên để lấy ra được tên sản phẩm, hình ảnh và giá tiền.

```
public class SanphamMua
    {
        public string masp { get; set; }
        public int soluong { get; set; }
        public override bool Equals(object obj)
        {
            SanphamMua spm = (SanphamMua)obj;
            return (spm.masp.Equals(this.masp));

        }
    }
```

Lưu ý: Viết lại phương thức Equals để so sánh 2 đối tượng sản phẩm mua là bằng nhau khi mã của chúng giống nhau.

*Tạo **MuaBanController** và viết các phương thức như dưới đây:

```
public class MuaBanController : Controller
    {
        // GET: Sanpham
        public ActionResult Index()
        {
            List<Sanpham> ds = new List<Sanpham>();
            ds.Add(new Sanpham("sp1", "Iphone 12 pro 512", "x.jpg", 1000));
            ds.Add(new Sanpham("sp2", "Vsmart Aren", "y.jpg", 400));
            ds.Add(new Sanpham("sp3", "Realme", "z.jpg", 350));
            Session["hanghoa"] = ds;
            return View(ds);
        }
        public ActionResult Chonmua(SanphamMua spm)
        {
            List<SanphamMua> dsmua = (List<SanphamMua>)Session["giohang"];
            if (dsmua == null)
            {
```

```
                dsmua = new List<SanphamMua>();
            }
            if (dsmua.Contains(spm))
            {
                int index = dsmua.IndexOf(spm);
                dsmua[index].soluong++;
            }
            else
            {
                spm.soluong = 1;
                dsmua.Add(spm);
            }
            Session["giohang"] = dsmua;
            return View();
        }
        public ActionResult XemGiohang()
        {
            List<SanphamMua> dsmua = (List<SanphamMua>)Session["giohang"];
            return View(dsmua);
        }

        public ActionResult XoaSanpham(string masp)
        {
            List<SanphamMua> dsmua = (List<SanphamMua>)Session["giohang"];
            SanphamMua s = new SanphamMua();
            s.masp = masp;
            int index = dsmua.IndexOf(s);
            s = dsmua[index];
            dsmua.Remove(s);
            Session["giohang"] = dsmua;
            return View("Chonmua");
        }
        public ActionResult Datmua()
        {
            List<SanphamMua> dsm = (List<SanphamMua>)Session["giohang"];
            Session.Remove("giohang");
            return View(dsm);
        }
    }
```

Tạo các view tương ứng với các ActionResult ở trên.

*View **Index**.cshtml:

```
@model IEnumerable<First_MVC.Models.Sanpham>

@{
    ViewBag.Title = "Index";
}

<h2>Cửa hàng online</h2>
<h4>Quý khách chọn mua các sản phẩm sau đây: </h4>
<table class="table-bordered" width="400">
    <tr style="background-color:#ffffcc">
```

```
        <td align="center">Mã sản phẩm</td>
        <td>Tên sản phẩm</td>
        <td>Hình ảnh</td>
        <td align="center">Giá tiền</td>
        <td>Mua hàng</td>
    </tr>
    @foreach (var item in Model)
    {
        <tr>
           <td align="center">@item.masp</td>
            <td>@item.tensp </td>
            <td align="center"><img src="~/Content/Images/@item.hinhanh"
width="80" /></td>
            <td align="center">@item.giatien </td>
            <td align="center">
                @Html.ActionLink("Chon mua", "Chonmua", "MuaBan",
          new First_MVC.Models.Sanpham { masp = item.masp}, null)
            </td>

        </tr>
    }
</table>
<br />
<br />
@Html.ActionLink("Xem giỏ hàng","XemGioHang","MuaBan")
```

*View **Chonmua**.cshtml

```
@model First_MVC.Models.SanphamMua
@{
    ViewBag.Title = "Chọn mua";
}

<h2>Chọn mua</h2>
@{
    List<First_MVC.Models.SanphamMua> dsmua =
(List<First_MVC.Models.SanphamMua>)Session["giohang"];
    List<First_MVC.Models.Sanpham> ds =
(List<First_MVC.Models.Sanpham>)Session["hanghoa"];
}
    <table class="table-bordered">
        <tr style="background-color:#ffffcc">
            <th align="center">Mã sản phẩm</th>
            <th>Tên sản phẩm</th>
            <th>Hình Ảnh</th>
            <th>Số lượng</th>
            <th>Giá tiền</th>
            <th>Xóa sản phẩm</th>
        </tr>

        @foreach (var item in dsmua)
        {
            First_MVC.Models.Sanpham s = new First_MVC.Models.Sanpham(item.masp);
```

```
            int vitri = ds.IndexOf(s);  //tìm sản phẩm mua trong danh sách hàng
hóa
            s = ds[vitri];
            <tr>
                <td align="center">@s.masp  </td>
                <td align="center"><img src="~/Content/Images/@s.hinhanh"
width="80" /></td>
                <td align="left">@s.tensp  </td>
                <td align="center">@item.soluong  </td>
                <td align="center">@s.giatien</td>

                <td align="center"><a href="@Url.Action("XoaSanpham", "MuaBan",
new { masp = item.masp })">Xóa</a></td>
            </tr>

        }
    </table>
<br />
@Html.ActionLink("Mua tiếp", "Index", "MuaBan")
<br /><br />
@Html.ActionLink("Đặt hàng", "Datmua", "MuaBan")
```

*View XemGioHang.cshtml

```
@model IEnumerable<First_MVC.Models.SanphamMua>

@{
    List<First_MVC.Models.Sanpham> ds =
(List<First_MVC.Models.Sanpham>)Session["hanghoa"];
}
@if (Model != null)
{
    <h2>Bạn đã chọn mua</h2>
    <table class="table-bordered">
        <tr style="background-color:#ffffcc">
            <th align="center">Mã sản phẩm</th>
            <th>Tên sản phẩm</th>
            <th>Hình Ảnh</th>
            <th>Số lượng</th>
            <th>Giá tiền</th>
            <th>Xóa sản phẩm</th>
        </tr>

        @foreach (var item in Model)
        {
            First_MVC.Models.Sanpham s = new First_MVC.Models.Sanpham(item.masp);
            int vitri = ds.IndexOf(s);  //tìm sản phẩm mua trong danh sách hàng
hóa
            s = ds[vitri];
            <tr>
                <td align="center">@s.masp  </td>
                <td align="center"><img src="~/Content/Images/@s.hinhanh"
width="80" /></td>
                <td align="left">@s.tensp  </td>
```

```
                <td align="center">@item.soluong  </td>
                <td align="center">@s.giatien</td>

                <td><a href="@Url.Action("XoaSanpham", "MuaBan", new { masp =
item.masp })">Xóa</a></td>
            </tr>

        }
    </table>
}
else
{
    <h3>Bạn chưa chọn mua sản phẩm nào</h3>
}

<br />
@Html.ActionLink("Mua tiếp", "Index", "MuaBan")
<br /><br />
@Html.ActionLink("Đặt hàng", "Datmua", "MuaBan")
```

*View **Datmua**.cshtml

```
@model IEnumerable<First_MVC.Models.SanphamMua>
@{
    ViewBag.Title = "Đặt hàng";
}
<h2>Bạn đã đặt mua</h2>
 @{
     List<First_MVC.Models.Sanpham> ds =
(List<First_MVC.Models.Sanpham>)Session["hanghoa"];
     }
<table width="500" class="table-bordered">
    <tr style="background-color:#ffffcc">
        <td>Mã sản phẩm</td>
        <td>Tên hàng</td>
        <td>Hình Ảnh</td>
        <td>Số lượng</td>
        <td>Giá tiền</td>
        <td>Thành tiền</td>
    </tr>
    @{
        var tongsl = 0;
        var thanhtien = 0;
        var tongtien = 0;
    }

    @foreach (var item in Model)
    {
        First_MVC.Models.Sanpham s = new First_MVC.Models.Sanpham(item.masp);
        int vitri = ds.IndexOf(s);  //tìm sản phẩm mua trong danh sách hàng hóa
        s = ds[vitri];
        tongsl += @item.soluong;
        thanhtien = @item.soluong * s.giatien;
```

```
        tongtien += thanhtien;
    <tr>
        <td align="center">@s.masp  </td>
        <td align="center"><img src="~/Content/Images/@s.hinhanh" width="80"
/></td>
        <td align="left">@s.tensp  </td>
        <td align="center">@item.soluong  </td>
        <td align="center">@s.giatien</td>
        <td align="center">@thanhtien</td>
    </tr>
    }
    </table>
<br />
<b>Số lượng các sản phẩm quý khách đã mua là: @tongsl</b>
<br />
<b>Tổng tiền quý khách phải trả là: @tongtien</b>
<br />
<br />
@Html.ActionLink("Mua tiếp đơn hàng khác", "Index", "MuaBan")
```