



# LẬP TRÌNH WEBSITE (ASP.NET MVC 5)

## CHƯƠNG 6:

# SINH GIAO DIỆN VÀ CHIA SẺ DỮ LIỆU

- ✓ Giảng Viên: ThS. Dương Thành Phết
- ✓ Email: [phetcm@gmail.com](mailto:phetcm@gmail.com)
- ✓ Website: <http://www.thayphet.net>
- ✓ Mobile: 0918158670

# NỘI DUNG

---

1. Razor
2. MVC Helper
3. Sinh giao diện theo Model
4. Custom Helper
5. ViewBag, Model
6. Session
7. Application
8. Cookie
9. Global.asax

# 1. RAZOR

---

## 1.1. Razor là gì?

- ✓ Razor là ngôn ngữ ngắn gọn, rõ ràng và hữu ích cho phép tạo ra các giao diện ứng dụng ASP.NET MVC
- ✓ Trong khối lệnh `@{...}` là mã C# hoặc VB.NET trộn HTML

```
<!-- Khối lệnh đơn -->
```

```
@{ var message = "Hello World"; }
```

```
<!-- Biểu thức nội tuyến -->
```

```
<p>Giá trị của message là: @message</p>
```

```
<!-- Khối nhiều dòng mã lệnh -->
```

```
@{
```

```
    var greeting = "Welcome to our site!";
```

```
    var weekDay = DateTime.Now.DayOfWeek;
```

```
    var greetingMessage = greeting + " Today is: " + weekDay;
```

```
}
```

```
<p>Lời chào là: @greetingMessage</p>
```

# 1. RAZOR

## 1.2. Cú pháp:

Khởi mã	<pre>@{     int x = 123;     string y = "because." }</pre>
Biểu thức (đã mã hóa HTML)	<pre>&lt;span&gt;@model.Message&lt;/span&gt;</pre>
Biểu thức (chưa mã hóa HTML)	<pre>&lt;span&gt;@Html.Raw(model.Message)&lt;/span&gt;</pre>
Kết hợp text và HTML	<pre>@foreach (var item in items) {     &lt;span&gt;@item.Prop&lt;/span&gt; }</pre>
Trộn code và text	<pre>@if (foo) {     &lt;text&gt;Plain Text&lt;/text&gt; }</pre>
Trộn code và text	<pre>@if (foo) {     @:Plain Text is @bar }</pre>

# 1. RAZOR

## 1.2. Cú pháp:

Khởi using	<code>@ using (Html.BeginForm()) {     &lt;input type="text" value="input here"&gt; }</code>
Địa chỉ email	Hi philha@example.com
Biểu thức (tường minh)	<code>&lt;span&gt;ISBN@(isbnNumber)&lt;/span&gt;</code>
Mã hóa ký hiệu @	<code>&lt;span&gt;In Razor, you use the @@foo to display the value of foo&lt;/span&gt;</code>
Chú thích phía server	<code>@* This is a server side multiline comment *@</code>
Trộn biểu thức và text	Hello @title. @name.

## 2. MVC HELPER - SINH GIAO DIỆN

---

### 2.1. Helper là gì?

- ❑ Helper là các thành phần sinh giao diện web phù hợp buộc dữ liệu với model để duy trì thông tin trên các thành phần đó.
- ❑ Đơn giản việc viết mã sinh giao diện
- ❑ Helper được chia làm 1 số nhóm
  - ✂ Liên kết
  - ✂ Form
  - ✂ Sinh giao diện từ model
  - ✂ Kiểm lỗi

**@Html.TextBox()**  
**@Html.ActionLink()**  
**@Html.Format()**

## 2. MVC HELPER - SINH GIAO DIỆN

### 2.2. HyperLink Helpers:

- @Html.ActionLink() được sử dụng để **sinh liên kết**

```
@Html.ActionLink("Giới thiệu", "About" )  
<a href="/Home/About">Giới thiệu</a>
```

- @Html.ActionLink() nhận một số tham số:

- ✂ **linkText** – nhãn của liên kết
- ✂ **actionName** – tên action
- ✂ **routeValues** – tập các giá trị truyền đến action.
- ✂ **controllerName** – tên controller
- ✂ **htmlAttributes** – tập thuộc tính HTML của thẻ <a>

- Ví dụ:

```
@Html.ActionLink("Edit Record", "Edit", new {Id=3})  
<a href="/Store/Edit/3">Edit Record</a>
```

- Liên kết chứa ảnh

```
<a href="@Url.Action("Delete")">  
  </a>
```

## 2. MVC HELPER - SINH GIAO DIỆN

### 2.3. Form Field Helpers:

- ❑ Duy trì dữ liệu các trường form

Helper	HTML
@Html.BeginForm()	Sinh thẻ <form> bắt đầu
@Html.EndForm()	Sinh thẻ </form> kết thúc
@Html.CheckBox()	Sinh thẻ <input type="checkbox" >
@Html.Hidden()	Sinh thẻ <input type="hidden" >
@Html.Password()	Sinh thẻ <input type="password" >
@Html.RadioButton()	Sinh thẻ <input type="radio" >
@Html.TextArea()	Sinh thẻ <textarea></textarea>
@Html.TextBox()	Sinh thẻ <input type="text" >
@Html.DropDownList()	Sinh thẻ <select><option></select>
@Html.ListBox()	Sinh thẻ <select multiple><option></select>



## 2. MVC HELPER - SINH GIAO DIỆN

### 2.3. Form Field Helpers:

Full Name	<code>@{Html.BeginForm("Action", "Controller");}</code>
<input type="text"/>	<code>&lt;div&gt;Full Name&lt;/div&gt;</code>
	<code>@Html.TextBox("FullName")</code>
Password	<code>&lt;div&gt;Password&lt;/div&gt;</code>
<input type="password"/>	<code>@Html.Password("Password")</code>
Photo	<code>&lt;div&gt;Photo&lt;/div&gt;</code>
<input type="button" value="Chọn tệp"/> Không có tệp	<code>&lt;input name="Photo" type="file" /&gt;</code>
Married Status	<code>&lt;div&gt;Married Status&lt;/div&gt;</code>
<input checked="" type="checkbox"/> Single	<code>&lt;label&gt;@Html.CheckBox("Status") Single&lt;/label&gt;</code>
Gender	<code>&lt;div&gt;Gender&lt;/div&gt;</code>
<input checked="" type="radio"/> Male <input type="radio"/> Female	<code>&lt;label&gt;@Html.RadioButton("Gender", true) Male&lt;/label&gt;</code>
	<code>&lt;label&gt;@Html.RadioButton("Gender", false) Female&lt;/label&gt;</code>
Description	<code>&lt;div&gt;Description&lt;/div&gt;</code>
<input type="text"/>	<code>@Html.TextArea("Description")</code>
	<code>@Html.Hidden("Active")</code>
	<code>&lt;hr /&gt;</code>
<input type="button" value="Submit"/>	<code>&lt;input type="submit" value="Submit" /&gt;</code>
	<code>@{Html.EndForm();}</code>

## 2. MVC HELPER - SINH GIAO DIỆN

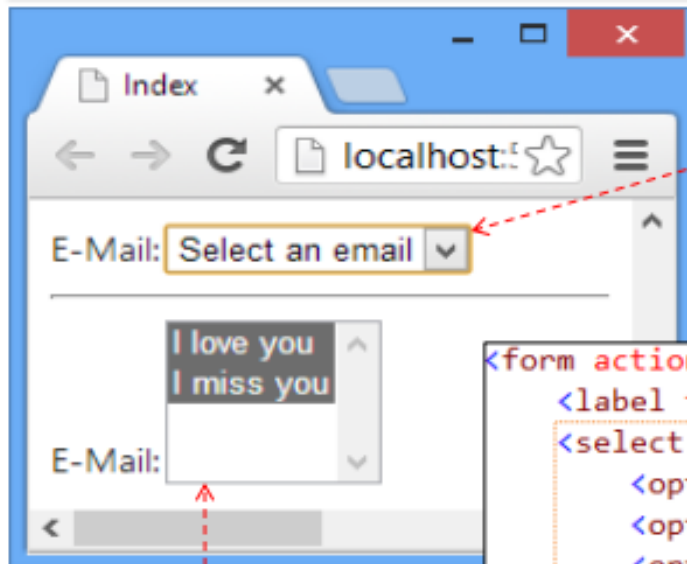
### 2.4. DropDownList & ListBox

```
List<Mail> Mails = new List<Mail>{  
    new Mail {  
        To = "sender1@gmail.com",  
        Subject = "I love you"  
    },  
    new Mail {  
        To = "sender2@gmail.com",  
        Subject = "I miss you"  
    }  
};  
ViewBag.Mails = new SelectList(Mails, "To", "Subject");
```

```
@using (Html.BeginForm()){  
    @Html.Label("Mails", "E-Mail:");  
    @Html.DropDownList("Mails", "Select an email")  
    <hr />  
    @Html.Label("Mails", "E-Mail:");  
    @Html.ListBox("Mails")  
}
```

## 2. MVC HELPER - SINH GIAO DIỆN

### 2.5. Sinh mã HTML



**Html.DropDownList**

**Html.ListBox**

```
<form action="/" method="post">
  <label for="Mails">E-Mail:</label>
  <select id="Mails" name="Mails">
    <option value="">Select an email</option>
    <option value="sender1@gmail.com">I love you</option>
    <option value="sender2@gmail.com">I miss you</option>
  </select>

  <hr />

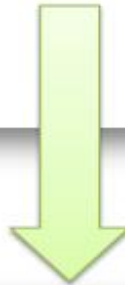
  <label for="Mails">E-Mail:</label>
  <select id="Mails" multiple="multiple" name="Mails">
    <option value="sender1@gmail.com">I love you</option>
    <option value="sender2@gmail.com">I miss you</option>
  </select>
</form>
```

## 2. MVC HELPER - SINH GIAO DIỆN

---

### 2.6. Sinh Form

```
@using (Html.BeginForm("Register", "Member")) {  
    ... nội dung form ...  
}
```



```
<form action="/Member/Register" method="post">  
    ... nội dung ...  
</form>
```

## 2. MVC HELPER - SINH GIAO DIỆN

### 2.7. Helper định dạng

Helper	Mô tả
@Html.FormatValue (value, format)	Định dạng một giá trị số, chuỗi hoặc thời gian
@String.Format(format, value1, value2...)	Định dạng nhiều giá trị hỗn hợp
@Html.Raw (html)	Giải mã chuỗi đã mã hóa HTML

- Số bình thường: 12345.8765
- Phân nhóm: 12,345.877
- Tiền tệ: \$12,345.88
- Phần trăm: 72.00 %

- Ngày bình thường: 5/27/2014 9:26:09 PM
- Định dạng D: Tuesday, May 27, 2014
- Định dạng ISO: 2014-05-27
- Định dạng English: 05/27/2014
- Định dạng 24 giờ: 21:26:09
- Định dạng 12 giờ: 09:26:09 PM


- Có mã hóa HTML: <strong>Hello</strong>
- Không mã hóa HTML : **Hello**

## 2. MVC HELPER - SINH GIAO DIỆN

### 2.7.1 Helper định dạng số

Ký hiệu	Mô tả
{0:C}	Currency – tiền tệ theo ngôn ngữ
{0:P}	Percent – số phần trăm
{0:#,###.##0}	Number – số phân nhóm và 3 số lẻ

```
@{  
    var number1 = 12345.8765;  
    var number2 = 0.72;  
}  
<ul>  
    <li>Số bình thường: @number1</li>  
    <li>Phân nhóm: @Html.FormatValue(number1, "{0:#,###.##0}")</li>  
    <li>Tiền tệ: @Html.FormatValue(number1, "{0:c}")</li>  
    <li>Phần trăm: @Html.FormatValue(number2, "{0:p}")</li>  
</ul>
```

- 
- Số bình thường: 12345.8765
  - Phân nhóm: 12,345.877
  - Tiền tệ: \$12,345.88
  - Phần trăm: 72.00 %



## 2. MVC HELPER - SINH GIAO DIỆN

### 2.7.2 Helper định dạng thời gian

Ký hiệu	Mô tả
{0:D}	Date – theo ngôn ngữ được chọn
{0:MMMM-dd-yyyy hh:mm:ss tt}	<ul style="list-style-type: none"><li>✓ M,MM,MMM,MMMM: tháng 1, 2 ký tự số, 3 ký tự viết tắt, tên tháng đầy đủ</li><li>✓ d,dd: ngày 1, 2 ký tự</li><li>✓ yy,yyyy: năm 2, 4 ký tự số</li><li>✓ H, HH, h, hh: 1,2 ký tự giờ 24 hoặc 12 giờ mỗi ngày</li><li>✓ m,mm: 1,2 ký tự số phút</li><li>✓ s,ss: 1,2 ký tự số giây</li><li>✓ tt: 2 ký tự sáng/chiều</li></ul>

- Ngày bình thường: 5/27/2014 9:26:09 PM
- Định dạng D: Tuesday, May 27, 2014
- Định dạng ISO: 2014-05-27
- Định dạng English: 05/27/2014
- Định dạng 24 giờ: 21:26:09
- Định dạng 12 giờ: 09:26:09 PM

```
@{  
    var now = DateTime.Now;  
}  
<ul>  
    <li>Ngày bình thường: @now</li>  
    <li>Định dạng D: @Html.FormatValue(now, "{0:D}")</li>  
    <li>Định dạng ISO: @Html.FormatValue(now, "{0:yyyy-MM-dd}")</li>  
    <li>Định dạng English: @Html.FormatValue(now, "{0:MM/dd/yyyy}")</li>  
    <li>Định dạng 24 giờ: @Html.FormatValue(now, "{0:HH:mm:ss}")</li>  
    <li>Định dạng 12 giờ: @Html.FormatValue(now, "{0:hh:mm:ss tt}")</li>  
</ul>
```

## 2. MVC HELPER - SINH GIAO DIỆN

### 2.8. Mã hóa HTML

- Có mã hóa HTML: `<strong>Hello</strong>`
- Không mã hóa HTML : **Hello**

```
@{  
    var chuoi = "<strong>Hello</strong>";  
}  
<ul>  
    <li>Có mã hóa HTML: @chuoi</li>  
    <li>Không mã hóa HTML : @Html.Raw(chuoi)</li>  
</ul>
```



## 3. SINH GIAO DIỆN THEO MODEL

### 3.1. Sinh giao diện động

- ❑ Dựa vào các đặc điểm của thuộc tính trong lớp model để sinh ra giao diện người dùng.
  - ✎ Sinh các control tường minh
  - ✎ Sinh các control ngầm định

```
public class Student
{
    [DisplayName("Mã sinh viên")]
    public String Id { get; set; }
    [DisplayName("Mật khẩu")]
    public String Password { get; set; }
    [DisplayName("Họ và tên")]
    public String FullName { get; set; }
    [DisplayName("Giới tính")]
    public bool Gender { get; set; }
    [DisplayName("Ngày sinh")]
    public DateTime Birthday { get; set; }
    [DisplayName("Ghi chú")]
    public String Notes { get; set; }
}
```



**Đăng ký thành viên**

Mã sinh viên	<input type="text"/>
Mật khẩu	<input type="password"/>
Họ và tên	<input type="text" value="Nguyễn Nghiệm"/>
Giới tính	<input checked="" type="radio"/> Nam <input type="radio"/> Nữ
Ngày sinh	<input type="text" value="9/1/1971 12:00:00 AM"/>
Ghi chú	<input type="text"/>
<input type="button" value="Register"/>	

## 3. SINH GIAO DIỆN THEO MODEL

### 3.2. Sinh giao trường minh

❑ Chỉ định loại control đối với các thuộc tính

Helper	Mô tả
@Html.TextBoxFor (m => m.Id)	<input type="text" name="Id" id="Id">
@Html.PasswordFor (m => m.Pwd)	<input type="password" name="Pwd" id="Pwd">
@Html.TextAreaFor (m => m.Notes)	<textarea name="Notes" id="Notes"></textarea>
@Html.CheckBoxFor (m => m.Status)	<input type="checkbox" name="Status" id="Status">
@Html.RadioButtonFor (m => m.Gender)	<input type="radio" name="Gender" id="Gender">
@Html.HiddenFor (m=> m.Name)	<input type="hidden" name="Name" id="Name">
@Html.DropDownListFor (m=> m.Blood)	<select id="Blood" name="Blood">...</select>
@Html.ListBoxFor (m=> m.Jobs)	<select id="Jobs" name="Jobs" multiple>...</select>
@Html.LabelFor (m=> m.Name)	<label for="Name"> Name </label>

# 3. SINH GIAO DIỆN THEO MODEL

## 3.3. View

```
@model Mvc5CodeDemo.Models.Student
<h2>Đăng ký thành viên</h2>
@using (Html.BeginForm())
{
    <table><tr>
        <td>@Html.LabelFor(m => m.Id)</td>
        <td>@Html.TextBoxFor(m => m.Id)</td>
    </tr><tr>
        <td>@Html.LabelFor(m => m.Password)</td>
        <td>@Html.PasswordFor(m => m.Password)</td>
    </tr><tr>
        <td>@Html.LabelFor(m => m.FullName)</td>
        <td>@Html.TextBoxFor(m => m.FullName)</td>
    </tr><tr>
        <td>@Html.LabelFor(m => m.Gender)</td>
        <td>
            <label>@Html.RadioButtonFor(m => m.Gender, true) Nam</label>
            <label>@Html.RadioButtonFor(m => m.Gender, false) Nữ</label>
        </td>
    </tr><tr>
        <td>@Html.LabelFor(m => m.Birthday)</td>
        <td>@Html.TextBoxFor(m => m.Birthday)</td>
    </tr><tr>
        <td>@Html.LabelFor(m => m.Notes)</td>
        <td>@Html.TextAreaFor(m => m.Notes)</td>
    </tr><tr>
        <td>&nbsp;</td>
        <td><input type="submit" value="Register" /></td>
    </tr></table>
}
```

Kiểu của Model

Sinh <label for="Id">Mã  
sinh viên</label>

Sinh <input type="text" name="Id"  
id="Id"> từ thuộc tính Id của Model

## 3. SINH GIAO DIỆN THEO MODEL

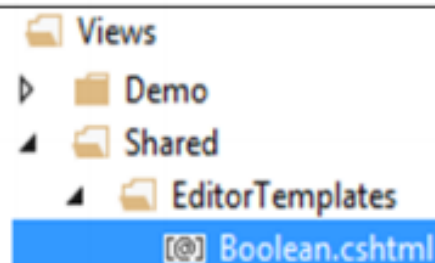
### 3.4. Sinh giao diện ngầm định

- ❑ Tự sinh loại control phù hợp với đặc điểm của thuộc tính của lớp model.

Helper	Mô tả
Html.EditorFor(m=>m.Property)	Sinh 1 control cho 1 thuộc tính.
Html.EditorForModel()	Sinh toàn form theo các thuộc tính của Model
Html.Editor(object)	Sinh toàn form theo các thuộc tính của Object đặt trong ViewBag

Bổ sung thêm template để hiển thị giới tính dạng RadioButtonList bằng cách thêm Boolean.cshtml vào thư mục Views/Shared/EditorTemplates

```
@model Boolean
<label>@Html.RadioButton("Gender", true, @Model == true)
Nam</label>
<label>@Html.RadioButton("Gender", false, @Model == false)
Nữ</label>
```



## 3. SINH GIAO DIỆN THEO MODEL

### 3.4. Sinh giao diện ngầm định

```
public class Student
{
    [DisplayName("Mã sinh viên")]
    public String Id { get; set; }
    [DisplayName("Mật khẩu"), DataType(DataType.Password)]
    public String Password { get; set; }
    [DisplayName("Họ và tên")]
    public String FullName { get; set; }
    [DisplayName("Giới tính")]
    public bool Gender { get; set; }
    [DisplayName("Ngày sinh")]
    public DateTime Birthday { get; set; }
    [DisplayName("Ghi chú"), DataType(DataType.MultilineText)]
    public String Notes { get; set; }
}
```

```
@model Mvc5CodeDemo.Models.Student
@using (Html.BeginForm())
{
    @Html.EditorForModel()
    <input type="submit" value="Register" />
}
```

Mã sinh viên

Mật khẩu

Họ và tên

Giới tính

☒ Nam ☐ Nữ

Ngày sinh

9/1/1971 12:00:00 AM

Ghi chú

Register

## 3. SINH GIAO DIỆN THEO MODEL

### 3.5. Kiểu Control

DataType	Mô tả
DataType.CreditCard	Chỉ cho phép nhập số thẻ tín dụng
DataType.Currency	Hiển thị và tiếp nhận dạng tiền tệ theo địa phương được chọn
DataType.Date	Hiển thị và tiếp nhận dạng ngày theo địa phương được chọn
DataType.DateTime	Hiển thị và tiếp nhận dạng ngày và giờ theo địa phương được chọn
DataType.Duration	Sinh slider trên thiết bị di động
DataType.EmailAddress	Chỉ cho phép nhập email
DataType.Html	Cho phép nhập mã html
DataType.ImageUrl	Chỉ cho phép nhập địa chỉ ảnh
DataType.MultilineText	Sinh <textare>
DataType.Password	Sinh <input type="password">
DataType.PhoneNumber	Sinh phần tử nhập số điện thoại trên thiết bị di động
DataType.PostalCode	Chỉ cấp nhận dạng postal code
DataType.Text	Nhập văn bản thông thường
DataType.Time	Ô nhập thời gian
DataType.Upload	Ô nhập upload file
DataType.Url	Chỉ nhấp nhận địa chỉ tài nguyên



## 3. SINH GIAO DIỆN THEO MODEL

---

### 3.6. Hiển thị thông tin

❑ `@Html.DisplayNameFor (m=> m.Property)`

✎ Hiển thị tên của thuộc tính Property

❑ `@Html.DisplayFor (m=>m.Property)`

✎ Hiển thị giá trị cho thuộc tính Property

❑ `@Html.DisplayForModel ()`

✎ Hiển thị giá trị của tất cả các thuộc tính

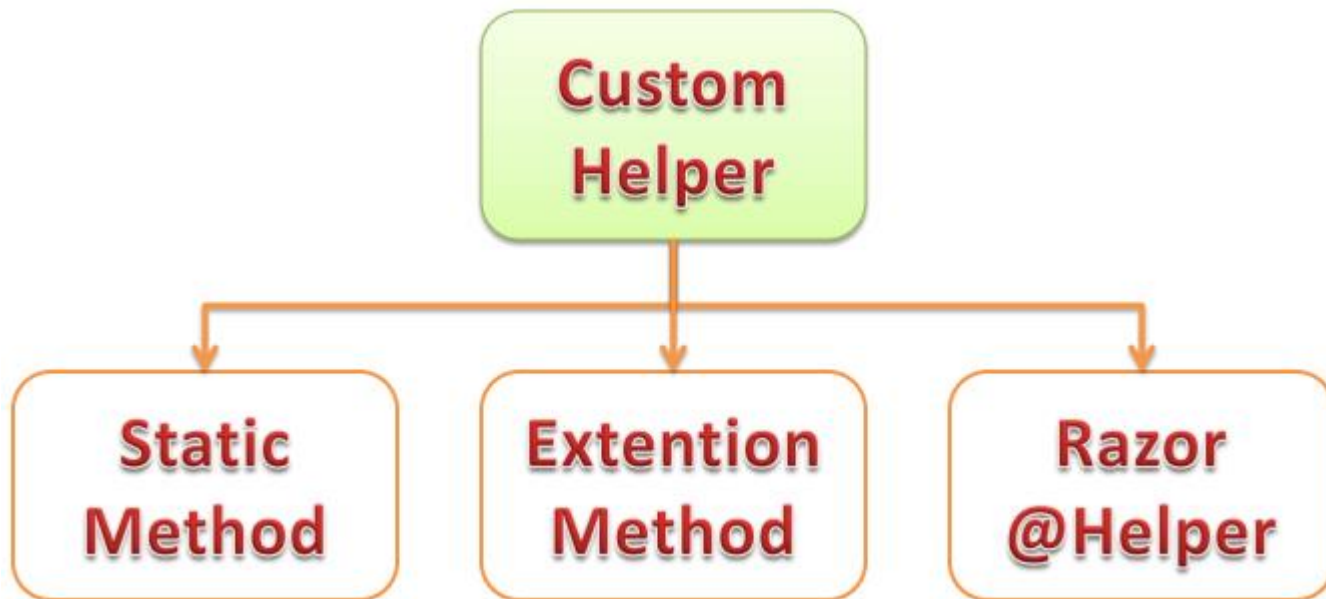
❑ `@Html.Display ("Mail")`

✎ Hiển thị giá trị của tất cả các thuộc tính của đối tượng trong ViewData hay ViewBag

## 4. CUSTOM HELPER

---

- ✓ Bên cạnh các helper dựng sẵn, bạn có thể tạo cho mình các helper có mục đích cho riêng mình.
- ✓ Trong MVC có 3 phương pháp tạo helper tùy biến





## 4. CUSTOM HELPER

### 4.1. Phương thức tĩnh

Static method

```
public class XString
{
    public static String Truncate(String input, int length)
    {
        if (input.Length <= length)
        {
            return input;
        }
        else
        {
            return input.Substring(0, length) + "...";
        }
    }
}
```

View

```
@{
    var input = @"Thật sự rất đơn giản với phương pháp này, bạn
                  chỉ cần định nghĩa phương thức tĩnh là có thể sử
                  dụng được như bạn từng thấy với String.Format().";
}
@XString.Truncate(input, 10);
```

Kết quả: Thật sự rất đơn giản với ...

## 4. CUSTOM HELPER

---

### 4.2. HTML Helper tùy biến

- ❑ Phương thức tĩnh

  - ✂ Sử dụng: `@String.Format("{0:F}", Model.Price)`

- ❑ Phương thức mở rộng của `HtmlHelper`

  - ✂ Sử dụng: `@Html.Submit(String label)`

- ❑ Chỉ thị **@helper** để định nghĩa trực tiếp trên View

  - ✂ Sử dụng: `@Truncate(title, 20)`

## 4. CUSTOM HELPER

### 4.3. Mở rộng HTML Helper

Qui ước

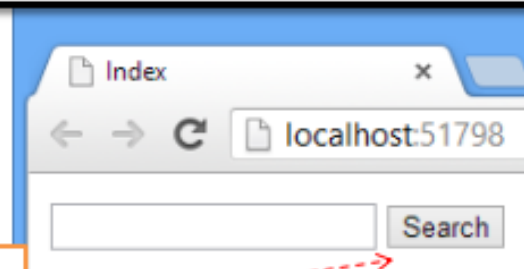
Lớp được mở rộng

```
public static class MyHelpers
{
    public static MvcHtmlString Submit(this HtmlHelper helper, string label)
    {
        TagBuilder tag = new TagBuilder("input");
        tag.MergeAttribute("type", "submit");
        tag.MergeAttribute("value", label);
        return MvcHtmlString.Create(tag.ToString(TagRenderMode.SelfClosing));
    }
}
```

```
@using (Html.BeginForm()){
    @Html.TextBox("txtSearch")
    @Html.Submit("Search")
}
```

Sử dụng helper  
mới định nghĩa

Control sinh ra



## 4. CUSTOM HELPER

### 4.3. Mở rộng HTML Helper

```
public static class XHtmlHelper
{
    public static MvcHtmlString Submit(this HtmlHelper helper, String label,
                                       String name = null, object htmlAttributes = null)
    {
        var tag = new TagBuilder("input");
        tag.Attributes["type"] = "submit";
        tag.Attributes["value"] = label;
        if (name != null)
        {
            tag.Attributes["name"] = name;
        }
        if (htmlAttributes != null)
        {
            var attributes = htmlAttributes.GetType().GetProperties();
            foreach (var a in attributes)
            {
                tag.Attributes[a.Name] = a.GetValue(htmlAttributes).ToString();
            }
        }
        return MvcHtmlString.Create(tag.ToString(TagRenderMode.SelfClosing));
    }
}
```

## 4. CUSTOM HELPER

### 4.4. Sử dụng Helper

Helper/HTML xuất ra

```
@Html.Submit("Save")
```

Sinh mã HTML: `<input type="submit" value="Save" />`

```
@Html.Submit("Save", "Command")
```

Sinh mã HTML: `<input name="Command" type="submit" value="Save" />`

```
@Html.Submit("Save", "Command", new { @class = "btn", id="save" })
```

Sinh mã HTML: `<input class="btn" id="save" name="Command" type="submit" value="Save" />`

## 4. CUSTOM HELPER

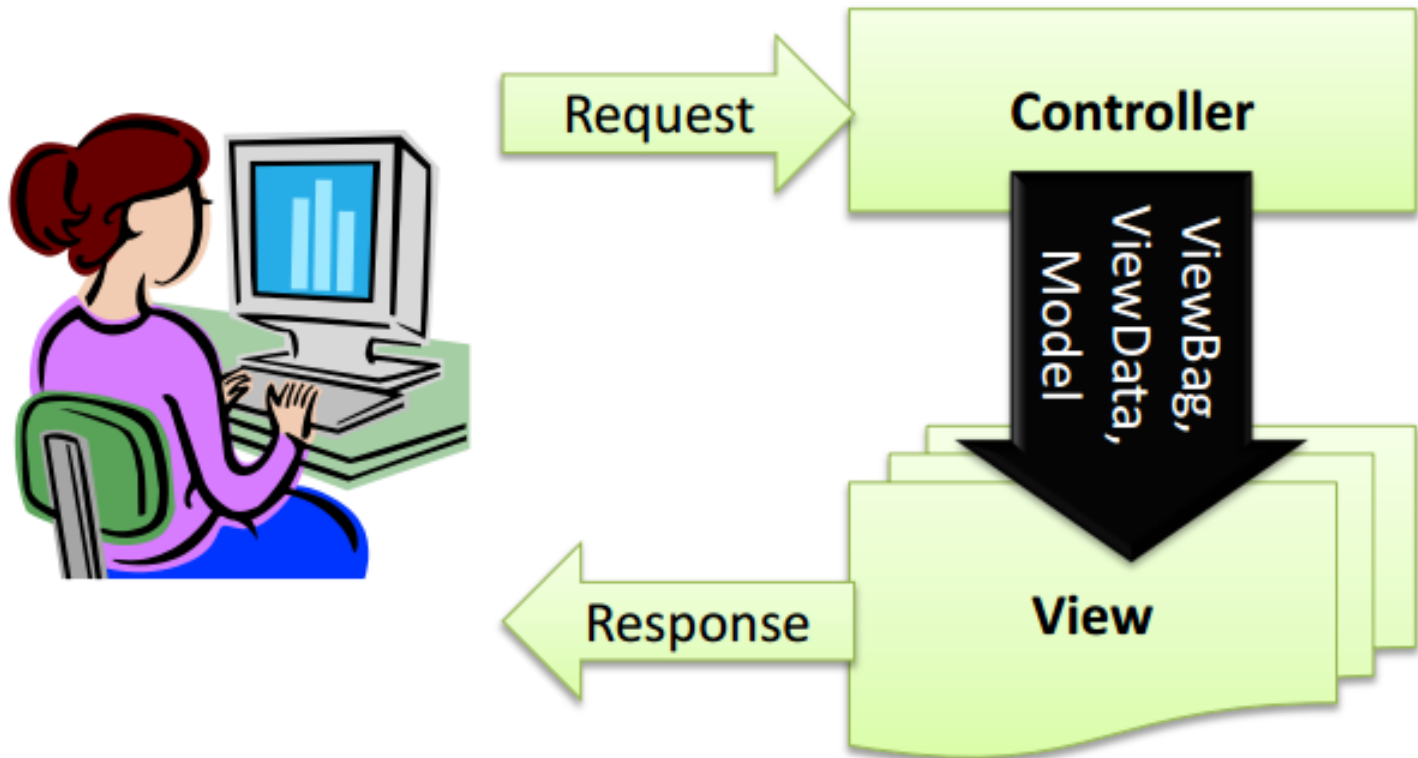
### 4.5. Sử dụng @Helper

```
@helper Truncate(String input, int length)
{
    if(input.Length < length)
    {
        @input
    }
    else
    {
        @input.Substring(0, length) <text>...</text>
    }
}
```

```
@foreach (var item in Model) {
    <tr>
        <td>
            @Truncate(item.Artist.Name, 25)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Genre.Name)
        </td>
        <td>
            @Truncate(item.Title, 25)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Price)
        </td>
    </tr>
}
```

## 5. VIEWBAG, MODEL

**ViewBag/ViewData** và **Model** được sử dụng để chia sẻ dữ liệu giữa Controller và View



## 5. VIEWBAG, MODEL

### ViewBag/ViewData

Đã được sử dụng để truyền dữ liệu từ Controller sang View. View sẽ sử dụng để xây dựng giao diện phù hợp trả lại cho người dùng.

```
public ActionResult Detail()
{
    ViewBag.Id = "SV001";
    ViewBag.Name = "Nguyễn Anh Tuấn";
    ViewData["Marks"] = 9.5;
    return View();
}
```

**ViewBag.Id ~  
ViewData["Id"]**

```
<h2>Student Detail</h2>
<ul>
    <li>Id: @ViewBag.Id</li>
    <li>Name: @ViewData["Name"]</li>
    <li>Marks: @ViewBag.Marks</li>
</ul>
```



# 5. VIEWBAG, MODEL

## Model

**Action**

```
public ActionResult Detail()
{
    // Tạo đối tượng
    var model = new StudentInfo
    {
        Id = "SV001",
        Name = "Nguyễn Anh Tuấn",
        Marks = 9.5
    };
    // Truyền đối tượng model cho view
    return View(model);
}
```

**Model**

```
public class StudentInfo
{
    public string Id { get; set; }
    public string Name { get; set; }
    public double Marks { get; set; }
}
```

Khai báo kiểu dữ liệu của đối tượng Model để tận dụng intelligence (chấm xỏ)

**View**

```
@model Mvc5.Models.StudentInfo
<h2>Student Detail</h2>
<ul>
    <li>Id: @Model.Id</li>
    <li>Name: @Model.Name</li>
    <li>Marks: @Model.Marks</li>
</ul>
```

### Student Detail

- Id: SV001
- Name: Nguyễn Anh Tuấn
- Marks: 9.5

## 5. VIEWBAG, MODEL

### Khai báo kiểu dữ liệu của Model

```
@model Mvc5.Models.StudentInfo
@{
    ViewBag.Title = "Student Detail";
}
```

Chỉ thị **@model** được sử dụng để khai báo kiểu cho đối tượng **Model**

```
<h2>Student Detail</h2>
```

```
<ul>
```

```
<li>Id: @Model.Id</li>
```

```
<li>Name: @Model.</li>
```

```
<li>Marks: @Model
```

```
</ul>
```

Equals  
GetHashCode  
GetType  
Id  
Marks  
Name  
ToString

Tên dùng được tính năng intelligence của công cụ VS2013 tránh được sai sót

Chú ý: phân biệt **@model** và **@Model**

- **@model**: Dùng để khai báo kiểu của **@Model**
- **@Model**: Đối tượng chứa dữ liệu truyền từ Controller

# 5. VIEWBAG, MODEL

## Chia sẻ với Model

```
public ActionResult Browse()
{
    ViewBag.Title = "List of your mails";

    Mail mail1 = new Mail
    {
        From = "receiver1@gmail.com",
        To = "sender1@gmail.com",
        Subject = "Mail subject 1",
        Body = "Mail content 1"
    };
    Mail mail2 = new Mail
    {
        From = "receiver2@gmail.com",
        To = "sender2@gmail.com",
        Subject = "Mail subject 2",
        Body = "Mail content 2"
    };

    List<Mail> mails = new List<Mail>();
    mails.Add(mail1);
    mails.Add(mail2);

    return View(mails);
}
```

@model IEnumerable<BasicDemo.Models.Mail>

```
<h2>@ViewBag.Title</h2>

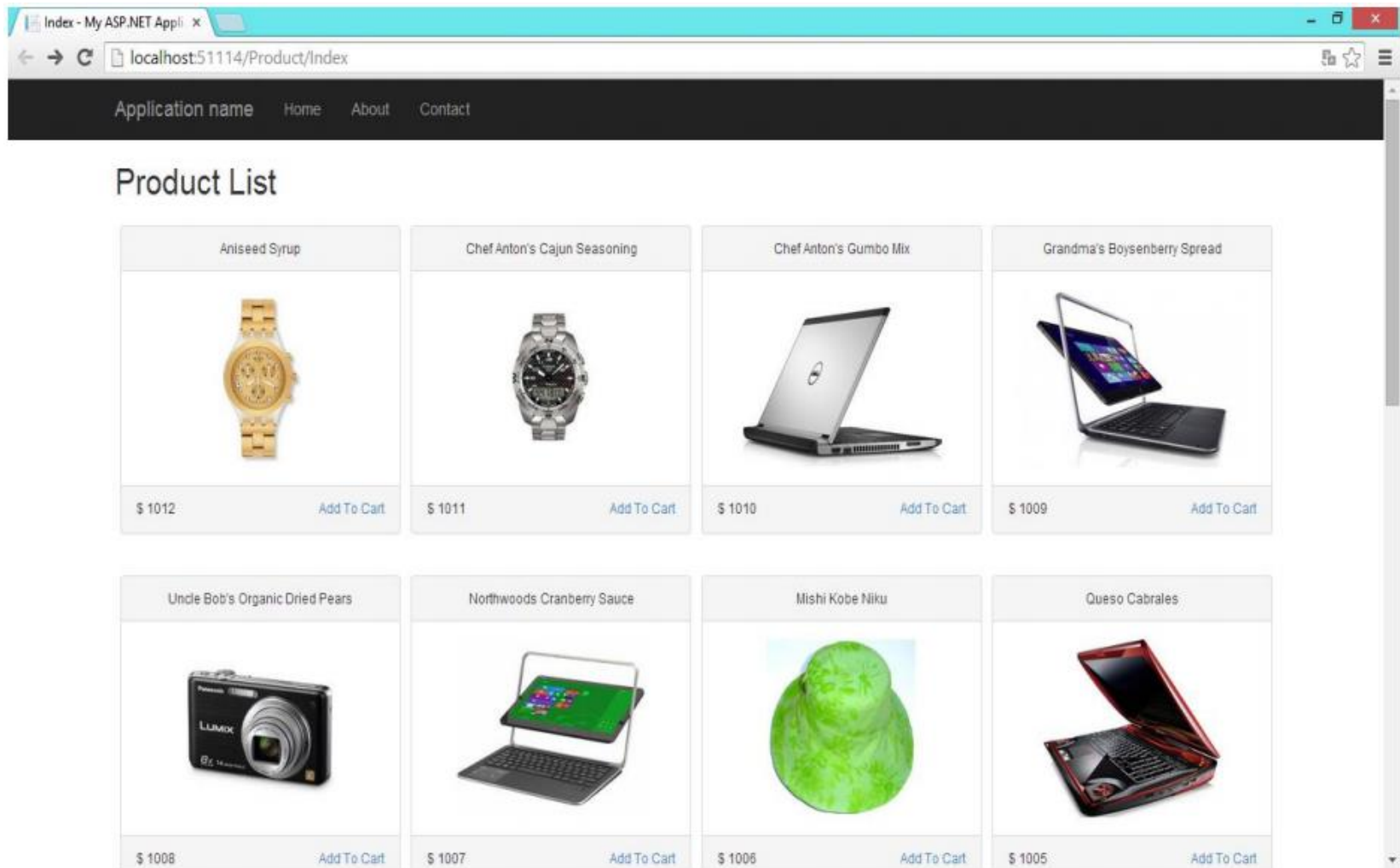
<table border="1" style="width:100%">
    <tr style="background:yellow;">
        <th>FROM</th>
        <th>TO</th>
        <th>SUBJECT</th>
        <th>BODY</th>
        <th>&nbsp;</th>
    </tr>
    @foreach (var item in Model) {
        <tr>
            <td>@item.From</td>
            <td>@item.To</td>
            <td>@item.Subject</td>
            <td>@item.Body</td>
            <td><a href="Home/Single">Single</a></td>
        </tr>
    }
</table>
```

List of your mails

FROM	TO	SUBJECT	BODY	
receiver1@gmail.com	sender1@gmail.com	Mail subject 1	Mail content 1	<a href="#">Single</a>
receiver2@gmail.com	sender2@gmail.com	Mail subject 2	Mail content 2	<a href="#">Single</a>

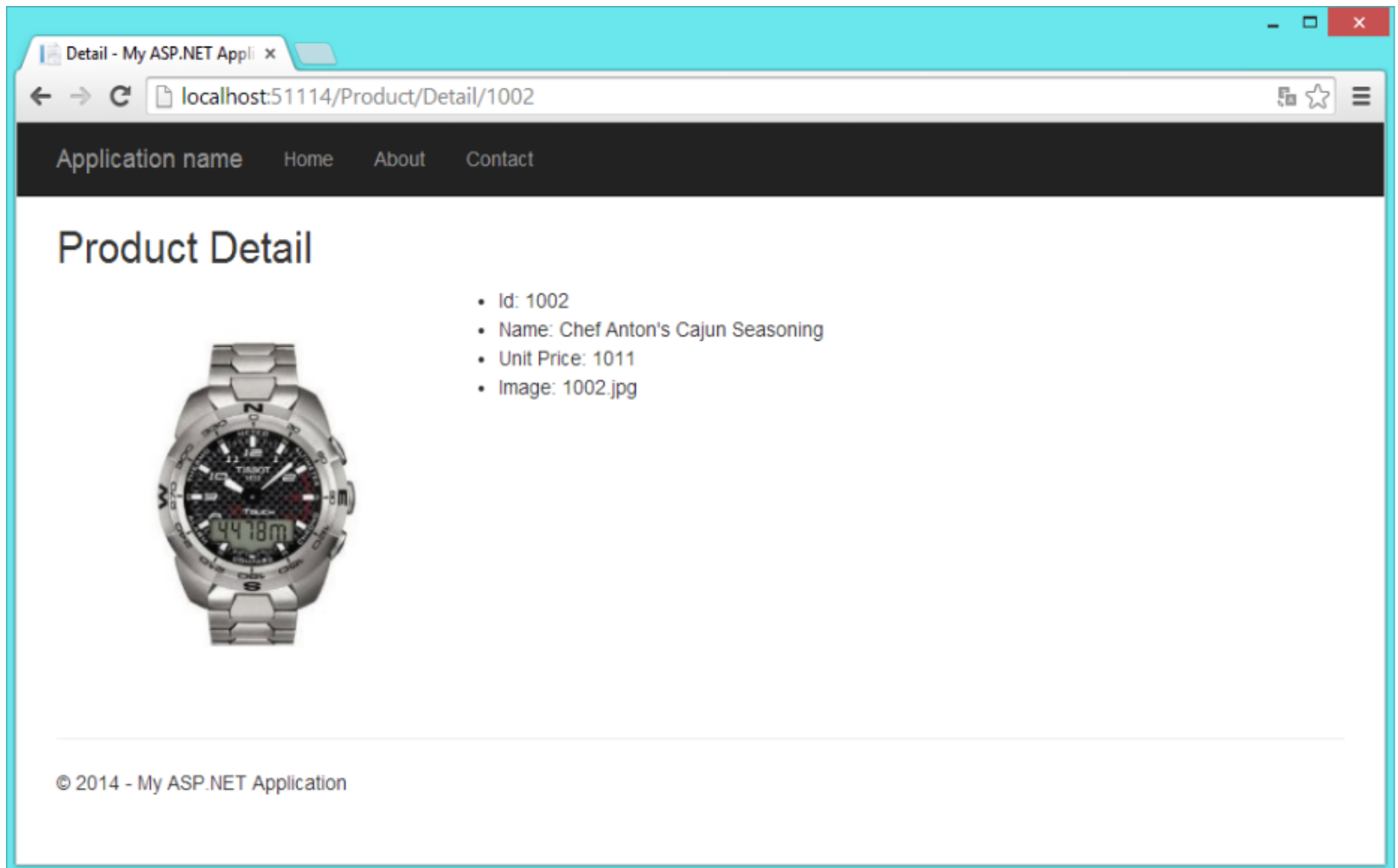
# 5. VIEWBAG, MODEL

## Minh họa: ProductList



## 5. VIEWBAG, MODEL

### Minh họa: ProductDetail



## 6. CHIA SẺ DỮ LIỆU THEO PHIÊN - SESSION

- ✓ Phiên làm việc (session) được tính từ lúc bắt đầu truy cập website cho đến khi đóng cửa sổ trình duyệt hoặc hết hạn sử dụng.
- ✓ Dữ liệu được lưu vào session sẽ được duy trì trong suốt phiên làm việc và được truy cập bởi bất kỳ thành phần nào hoạt động trong phiên làm việc đó.





## 6. CHIA SẺ DỮ LIỆU THEO PHIÊN - SESSION

### Ứng dụng:

- ✓ Duy trì giỏ hàng: giỏ hàng cần duy trì trong suốt quá trình tìm kiếm và chọn mua hàng hóa.
- ✓ Duy trì tài khoản đăng nhập: tài khoản đăng nhập cần được duy trì trong suốt phiên làm việc để khi cần có thể sử dụng.

### Mã lấy đối tượng session:

Nơi viết mã	Mã lệnh	Ví dụ
Controller	Session	<code>Session["A"]="Hello"</code>
View	@Session	<code>@Session["A"]</code>
Class bất kỳ	HttpContext.Current.Session	<code>HttpContext.Current.Session["A"]="Hello"</code>

## 6. CHIA SẺ DỮ LIỆU THEO PHIÊN - SESSION

### Thao tác Session:

Method/Property	Mô tả	Ví dụ
Add(Key, Value)	Thêm mới một đối tượng vào Session	Session.Add("Now", DateTime.Now)
[Key]=Value	Thêm mới hoặc thay thế một đối tượng	Session["Cart"] = new ShoppingCart()
Remove(Key)	Xóa đối tượng	Session.Remove("Cart")
Clear()	Xóa sạch các đối tượng trong Session	Session.Clear()
Abandon()	Hủy session	Session.Abandon()
SessionID	Mã của phiên làm việc	Var id = Session.SessionID



## 6. CHIA SẺ DỮ LIỆU THEO PHIÊN - SESSION

---

### Minh họa:

- ✓ Shopping Cart: duy trì danh sách hàng đã chọn trong session
- ✓ CRUD: duy trì dữ liệu trong List

## 7. APPLICATION

- ❑ Application là phạm vi chia sẻ dữ liệu trên toàn ứng dụng (tất cả mọi user có thể tạo và sử dụng)
- ❑ Ứng dụng: Cache dữ liệu cho toàn ứng dụng
  - ✗ Bộ đếm số người truy cập
  - ✗ Queue mails
  - ✗ Queue chat messages



# 7. APPLICATION

---

## Application Scope

### □ Truy xuất đối tượng Application

✎ Trong Controller: `HttpContext.Application`

✎ Trong View: `@HttpContext.Current.Application`

✎ Trong lớp bất kỳ: `HttpContext.Current.Application`

### □ Thao tác

✎ `Application.Add (name, value)`

✎ `Application[name] = <value>`

✎ `Application.Remove (name)`

✎ `Application.Clear ()`

✎ `Application.Lock ()`

✎ `Application.Unlock ()`

## 7. APPLICATION

---

### Minh họa:

- ✓ Bộ đến khách thăm web
- ✓ Chat
- ✓ Queue mails

## 8. COOKIE

---

- ✓ Cookie là mẫu tin nhỏ được lưu trên máy client và truyền thông với server trong các request và response.
- ✓ Ứng dụng: Chia sẻ dữ liệu giữa các trang trong website được truy cập từ máy
  - Tài khoản đăng nhập
  - Hàng hóa đã xem



## 8. COOKIE

---

### ❑ Lấy cookie từ client

✎ Trong controler: `Request.Cookies[name]`

✎ Trong View: `@Request.Cookies[name]`

### ❑ Gửi cookie về client

✎ `Response.Cookies.Add(cookie)`

### ❑ Tạo cookie

✎ `HttpCookie cookie = new HttpCookie(name, value)`

✓ Tạo cookie với tên và giá trị

✎ `HttpCookie cookie = new HttpCookie(name)`

✓ Tạo cookie với tên

## 8. COOKIE

---

### Cookie API

- ❑ `cookie.Expires`
  - ✎ Thời hạn của cookie.
- ❑ `cookie.Name`
  - ✎ Tên của cookie
- ❑ `cookie.Value`
  - ✎ Giá trị đơn của cookie
- ❑ `cookie.Values`
  - ✎ Các giá trị của một cookie
- ❑ `cookie.Values.Add(Key, Value)`
  - ✎ Thêm một giá trị vào cookie
- ❑ `cookie.Values[Key] = <Value>`
  - ✎ Thêm hoặc thay thế 1 giá trị của cookie



## 8. COOKIE

---

### Minh họa:

Duy trì hàng hóa đã xem

## 9. GLOBAL.ASAX

- ❑ Tập tin Global.asax chứa các điều khiển sự kiện quản lý vòng đời của application, session và request

```
public class MvcApplication : System.Web.HttpApplication
{
    // Chạy sau khi ứng dụng start thành công
    protected void Application_Start()...

    // Chạy trước khi ứng dụng shutdown
    protected void Application_End()...

    // Chạy sau khi có một phiên làm việc được tạo
    protected void Session_Start()...

    // Chạy trước khi 1 phiên làm việc hết hạn
    protected void Session_End()...

    // Chạy trước khi request chưa được phục vụ
    protected void Application_BeginRequest()...

    // Chạy trước khi request đã được phục vụ
    protected void Application_EndRequest()...
}
```

## 9. GLOBAL.ASAX

---

**Minh họa:**

HitCounter

# HẾT CHƯƠNG 6