



Bài 03

Views trong ASP.NET MVC



- ◆ Giới thiệu về View và View Engines
- ◆ Xác định View cho Action
- ◆ Truyền dữ liệu từ Controller tới View
- ◆ Giới thiệu về Razor View Engine
- ◆ Ngôn ngữ Razor
- ◆ Các phương thức HTML Helper



- ◆ View cung cấp giao diện và nội dung cho người dùng.
- ◆ View hiển thị các form cho người dùng nhập dữ liệu.
- ◆ View chứa đựng cả code HTML và code chạy trên Web Server.



- ◆ Là một phần của MVC Framework mà nó chuyển đổi mã của view thành HTML để trình duyệt có thể hiển thị.
- ◆ View engine được chia làm 2 loại:
 - ◆ **Web Form view engine:** Là view engine sử dụng cú pháp giống ASP.NET Web Form.
 - ◆ **Razor view engine:** là view engine mặc định từ MVC 3 trở đi. View engine này không phải là một ngôn ngữ lập trình mới nhưng nó là một cú pháp markup mới để chuyển đổi giữa HTML và code lập trình đơn giản hơn.



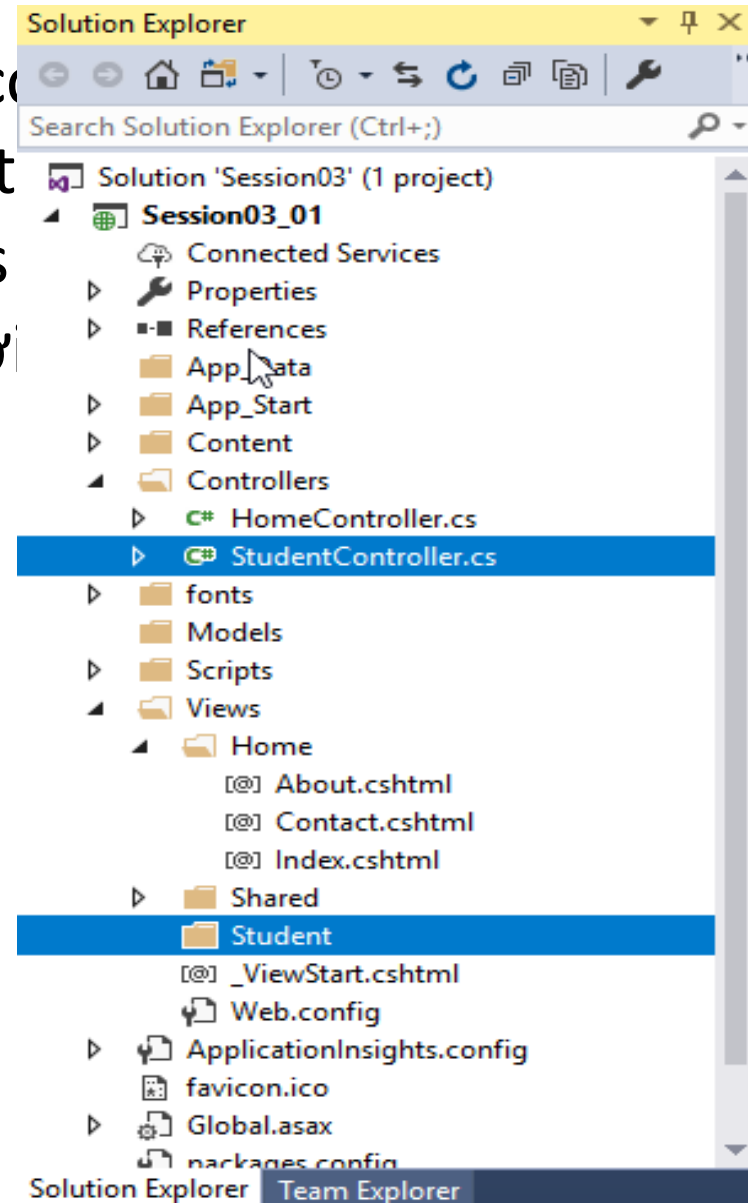
Xác định View cho 1 action

- ◆ Mỗi controller tạo ra với tên Xyzcontroller (từ khóa controller ở sau Xyz là bắt buộc) thì một thư mục Xyz sẽ được tạo ra trong thư mục Views của project, thư mục này sẽ chứa các view cùng tên với các action trả về view trong controller tương ứng.



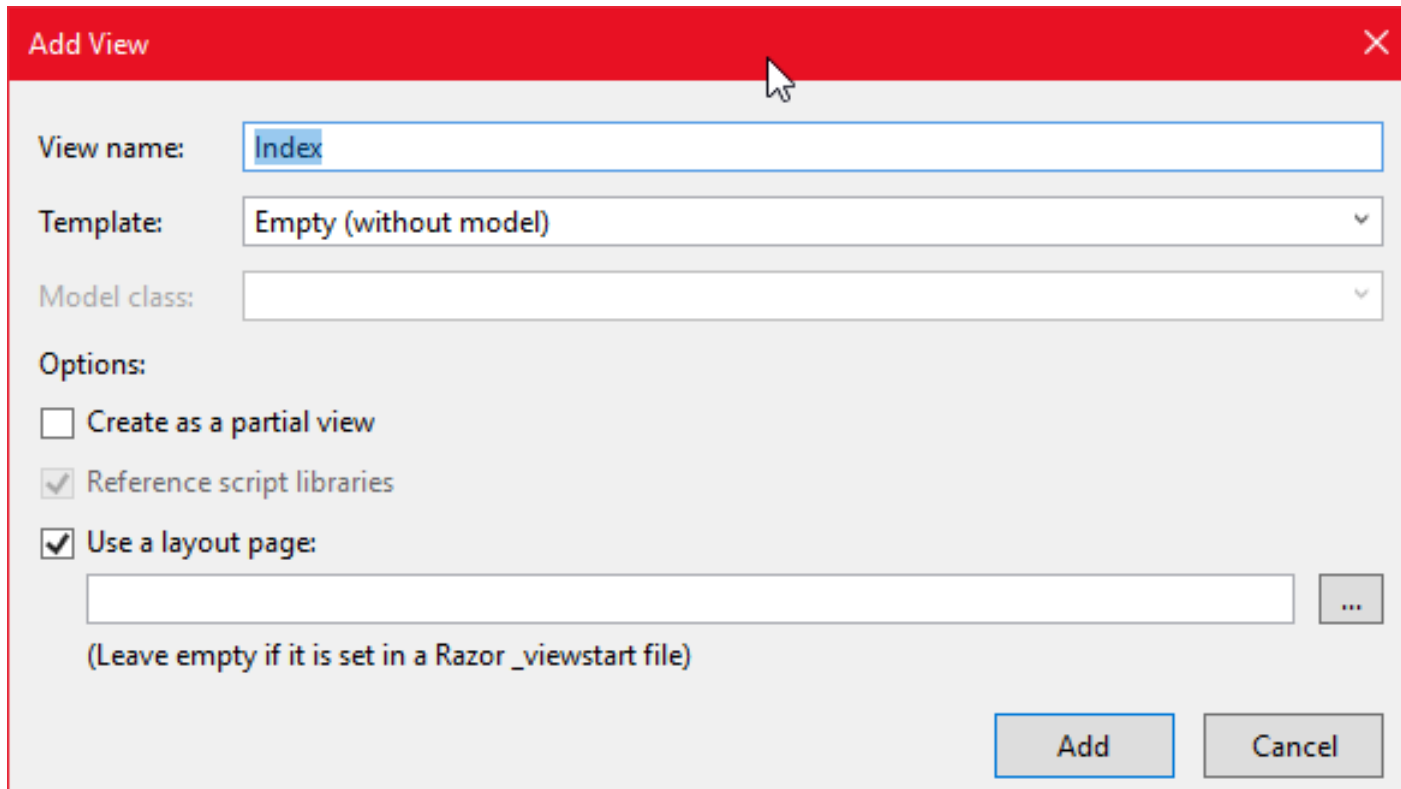
Xác định View cho 1 action

- ◆ Mỗi controller tạo ra với tên XyzController (controller ở sau Xyz là bắt buộc) thì được tạo ra trong thư mục Views này sẽ chứa các view cùng tên với trong controller tương ứng.



Xác định View cho 1 action

- ◆ Tạo view: Kích chuột phải vào tên action trong controller chọn Add View



Add View

View name:

Template:

Model class:

Options:

☐ Create as a partial view

☒ Reference script libraries

☒ Use a layout page:

...

(Leave empty if it is set in a Razor _viewstart file)

Xác định View cho 1 action

The screenshot shows the Microsoft Visual Studio IDE. The main editor window displays the code for `StudentController` in `Session03.Controllers`. The `Index()` action is highlighted, and the `StudentList()` action is also visible. The `StudentList()` action is highlighted with a red box, and a red arrow points from it to the `StudentList.cshtml` file in the Solution Explorer. The Solution Explorer shows the project structure, including the `Views` folder and the `Student` subfolder. The `StudentList.cshtml` file is selected. The Properties window at the bottom right shows the file name.

```
namespace Session03.Controllers
{
    // references
    public class StudentController : Controller
    {
        // GET: Student
        // references | 0 requests | 0 exceptions
        public ActionResult Index()
        {
            return View();
        }
        // references | 0 requests | 0 exceptions
        public ActionResult StudentList()
        {
        }
    }
}
```

Solution Explorer

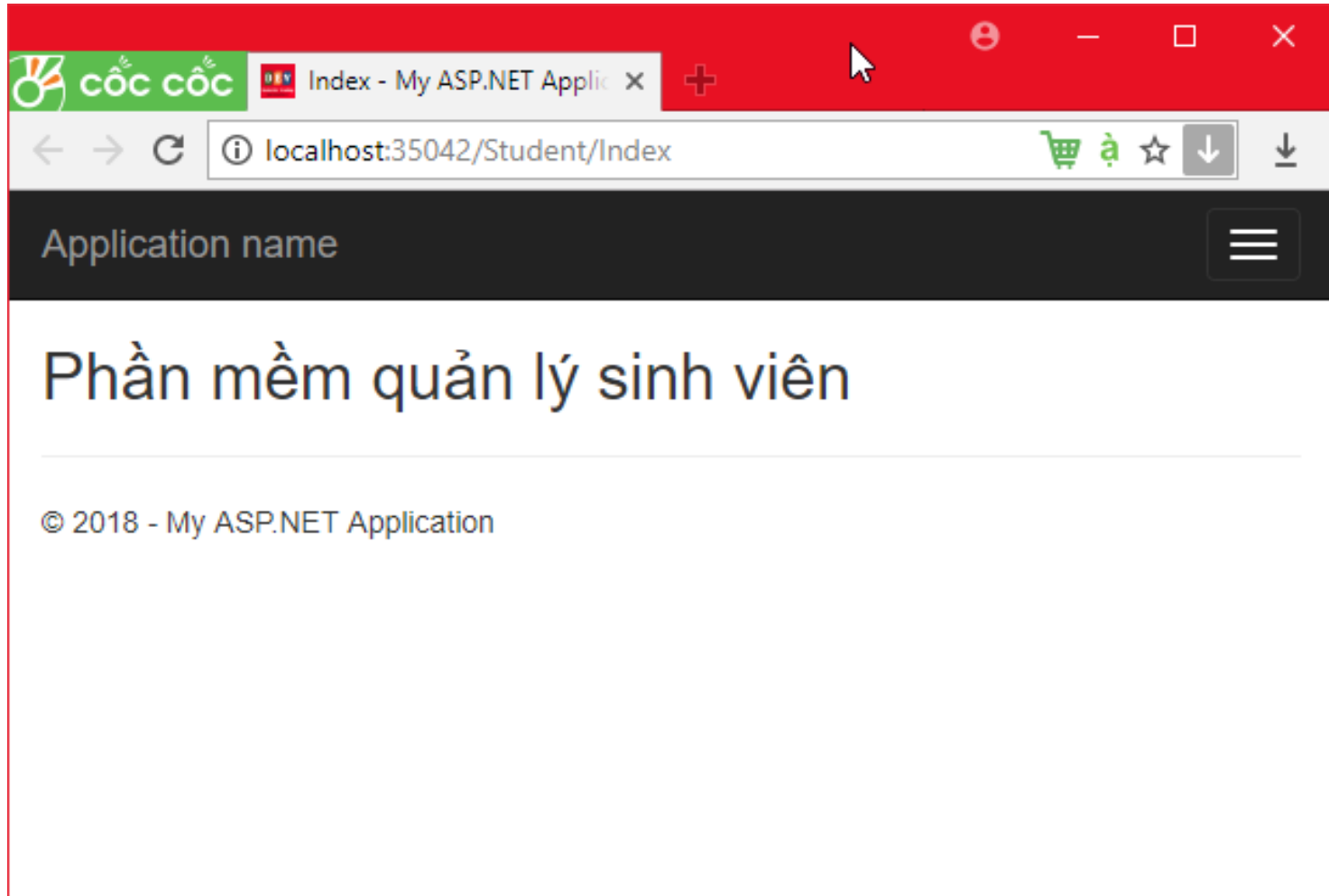
- Content
- Controllers
 - HomeController.cs
 - StudentController.cs
- fonts
- Models
- Scripts
- Views
 - Home
 - About.cshtml
 - Contact.cshtml
 - Index.cshtml
 - Shared
 - Student
 - Index.cshtml
 - StudentList.cshtml
 - _ViewStart.cshtml
 - Web.config

Properties

File Name

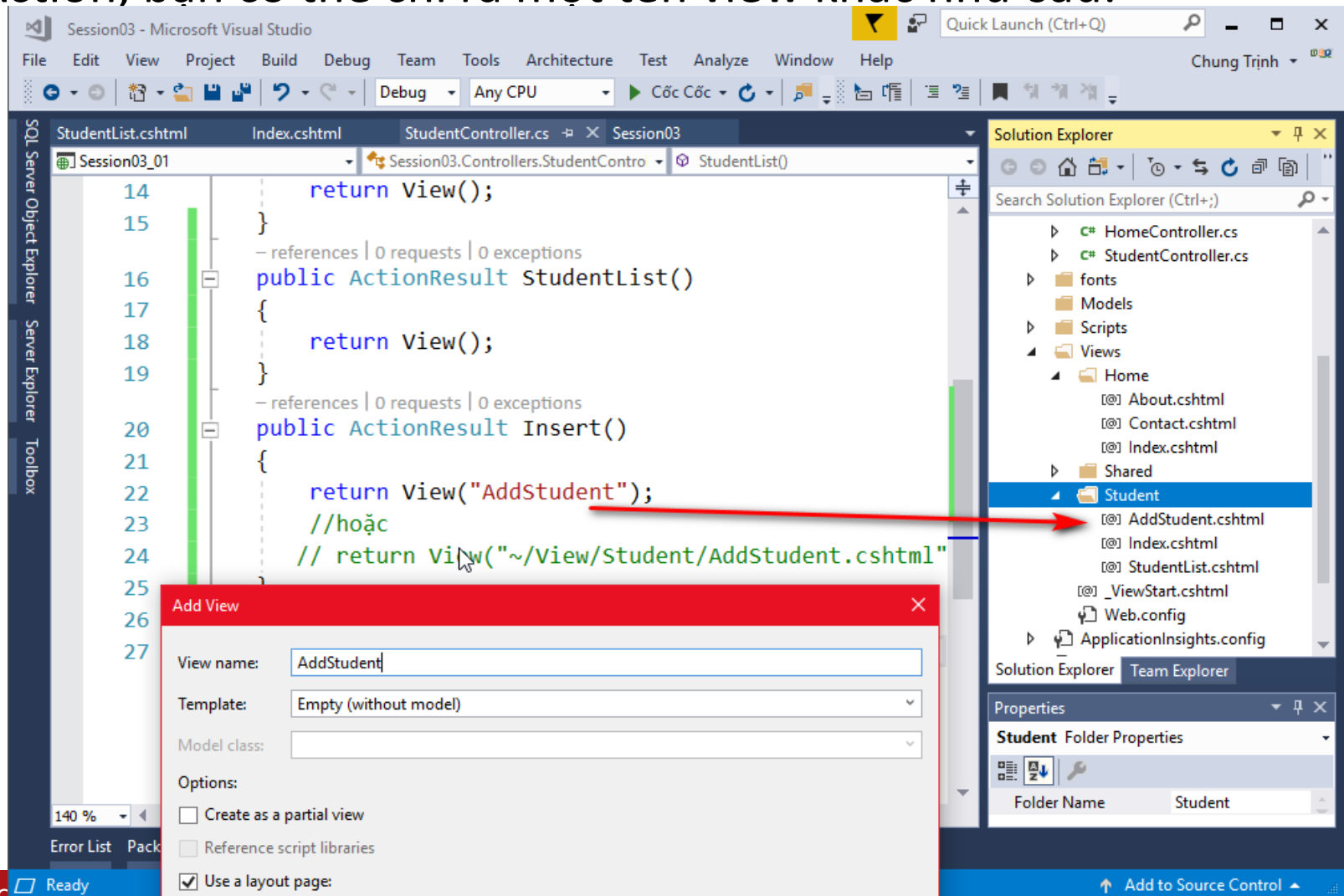
Xác định View cho 1 action

- ◆ Tạo view: Kích chuột phải vào tên action trong controller chọn Add View



Xác định View cho 1 action

- Trong trường hợp bạn không muốn tên View trả về trùng với tên Action, bạn có thể chỉ ra một tên view khác như sau:



- ◆ Trong ASP.NET MVC một controller thường thực hiện nghiệp vụ của ứng dụng và trả kết quả cho người dùng thông qua view. Các đối tượng được sử dụng để đưa kết quả từ Controller ra view là:
 - ◆ ViewData
 - ◆ ViewBag
 - ◆ TempData

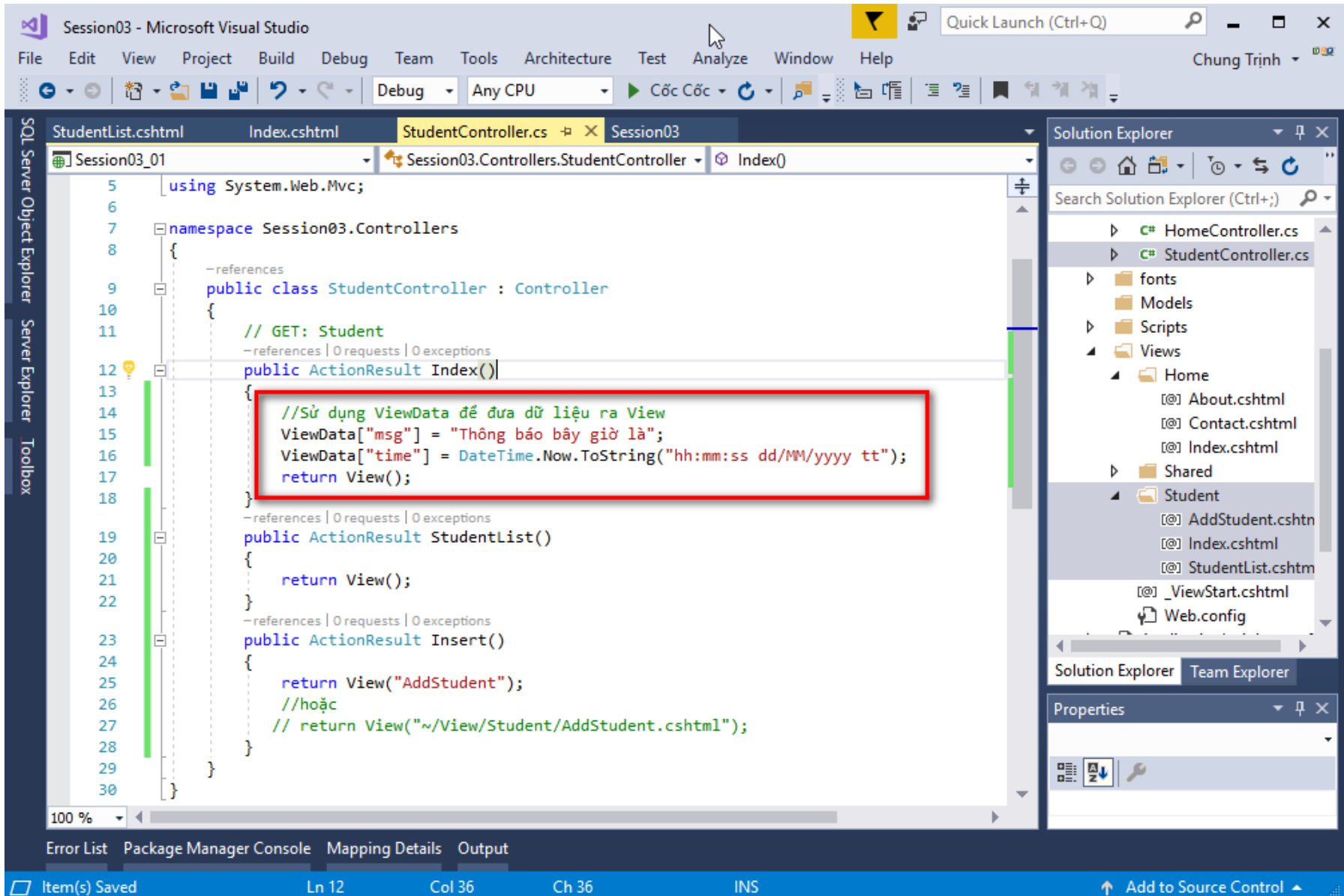


- ◆ ViewData làm nhiệm vụ chuyển dữ liệu từ Controller tới View
- ◆ Là đối tượng dạng Dictionary xuất phát từ lớp ViewDataDictionary.
- ◆ ViewData chỉ tồn tại trong request hiện tại.
- ◆ Giá trị của ViewData sẽ null nếu request thực hiện là redirected.
- ◆ ViewData đòi hỏi phải chuyển đổi kiểu khi sử dụng những kiểu dữ liệu phức tạp để tránh lỗi.
- ◆ Cú pháp

`ViewData[<key>] = <Value>;`



ViewData



Session03 - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help

Debug Any CPU Cốt Cốt

StudentList.cshtml Index.cshtml StudentController.cs Session03

Session03_01 Session03.Controllers.StudentController Index()

```
5 using System.Web.Mvc;
6
7 namespace Session03.Controllers
8 {
9     -references
10     public class StudentController : Controller
11     {
12         // GET: Student
13         -references | 0 requests | 0 exceptions
14         public ActionResult Index()
15         {
16             //Sử dụng ViewData để đưa dữ liệu ra View
17             ViewData["msg"] = "Thông báo bây giờ là";
18             ViewData["time"] = DateTime.Now.ToString("hh:mm:ss dd/MM/yyyy tt");
19             return View();
20         }
21         -references | 0 requests | 0 exceptions
22         public ActionResult StudentList()
23         {
24             return View();
25         }
26         -references | 0 requests | 0 exceptions
27         public ActionResult Insert()
28         {
29             return View("AddStudent");
30             //hoặc
31             // return View("~/View/Student/AddStudent.cshtml");
32         }
33     }
34 }
```

Solution Explorer

Search Solution Explorer (Ctrl+;)

- C# HomeController.cs
- C# StudentController.cs
 - fonts
 - Models
 - Scripts
 - Views
 - Home
 - About.cshtml
 - Contact.cshtml
 - Index.cshtml
 - Shared
 - Student
 - AddStudent.cshtml
 - Index.cshtml
 - StudentList.cshtml
 - _ViewStart.cshtml
 - Web.config

Solution Explorer Team Explorer

Properties

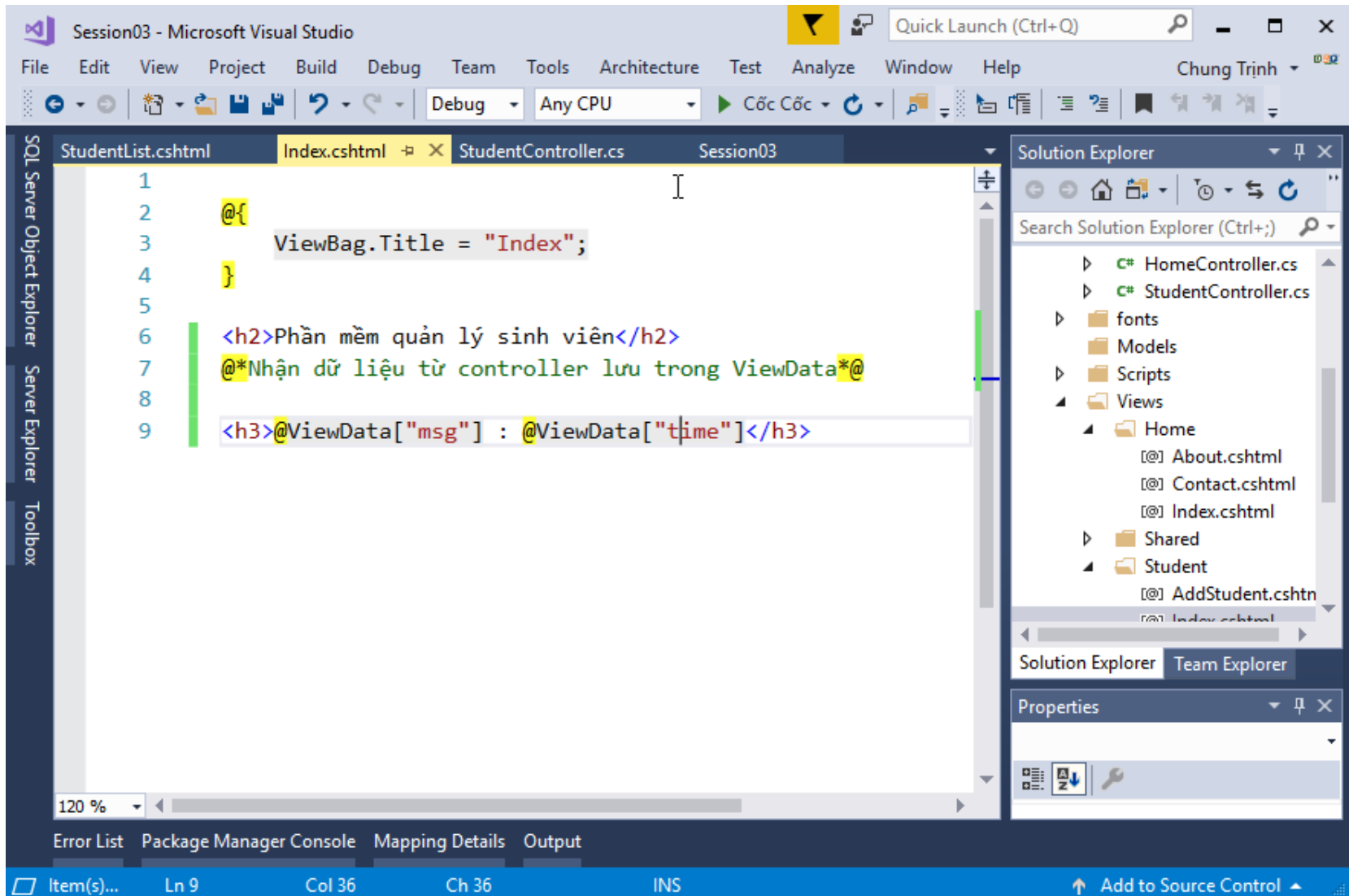
100 %

Error List Package Manager Console Mapping Details Output

Item(s) Saved Ln 12 Col 36 Ch 36 INS

Add to Source Control

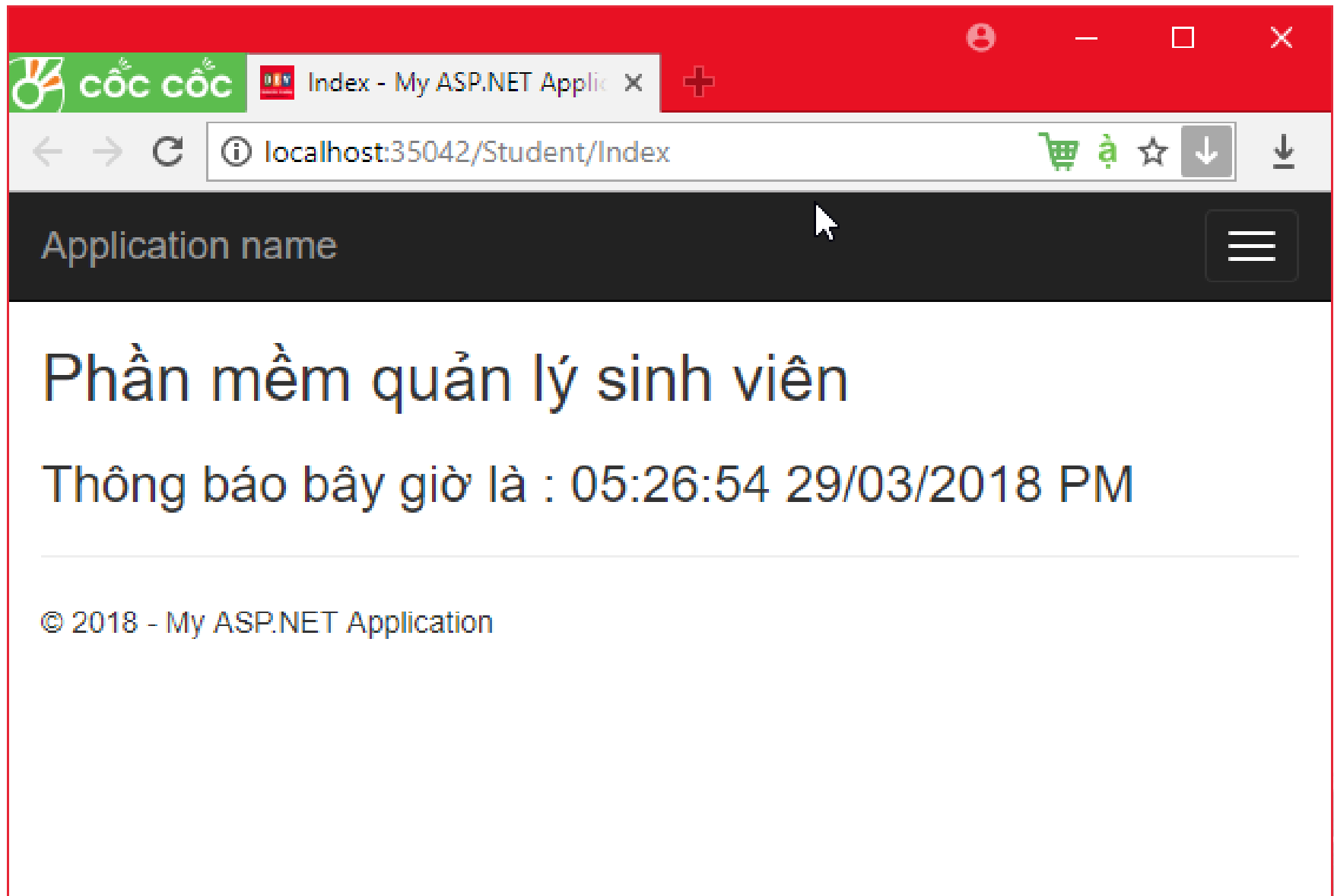
ViewData



```
1 @{
2     ViewBag.Title = "Index";
3 }
4
5 <h2>Phần mềm quản lý sinh viên</h2>
6 *@Nhận dữ liệu từ controller lưu trong ViewData*@
7
8 <h3>@ViewData["msg"] : @ViewData["time"]</h3>
```

The screenshot shows the Visual Studio IDE with the following details:

- File Explorer:** Shows the project structure for Session03, including folders for fonts, Models, Scripts, Views, and sub-folders like Home, Shared, and Student.
- Properties Window:** Currently empty.
- Bottom Bar:** Displays the current file as Item(s)..., Ln 9, Col 36, Ch 36, INS, and an option to Add to Source Control.

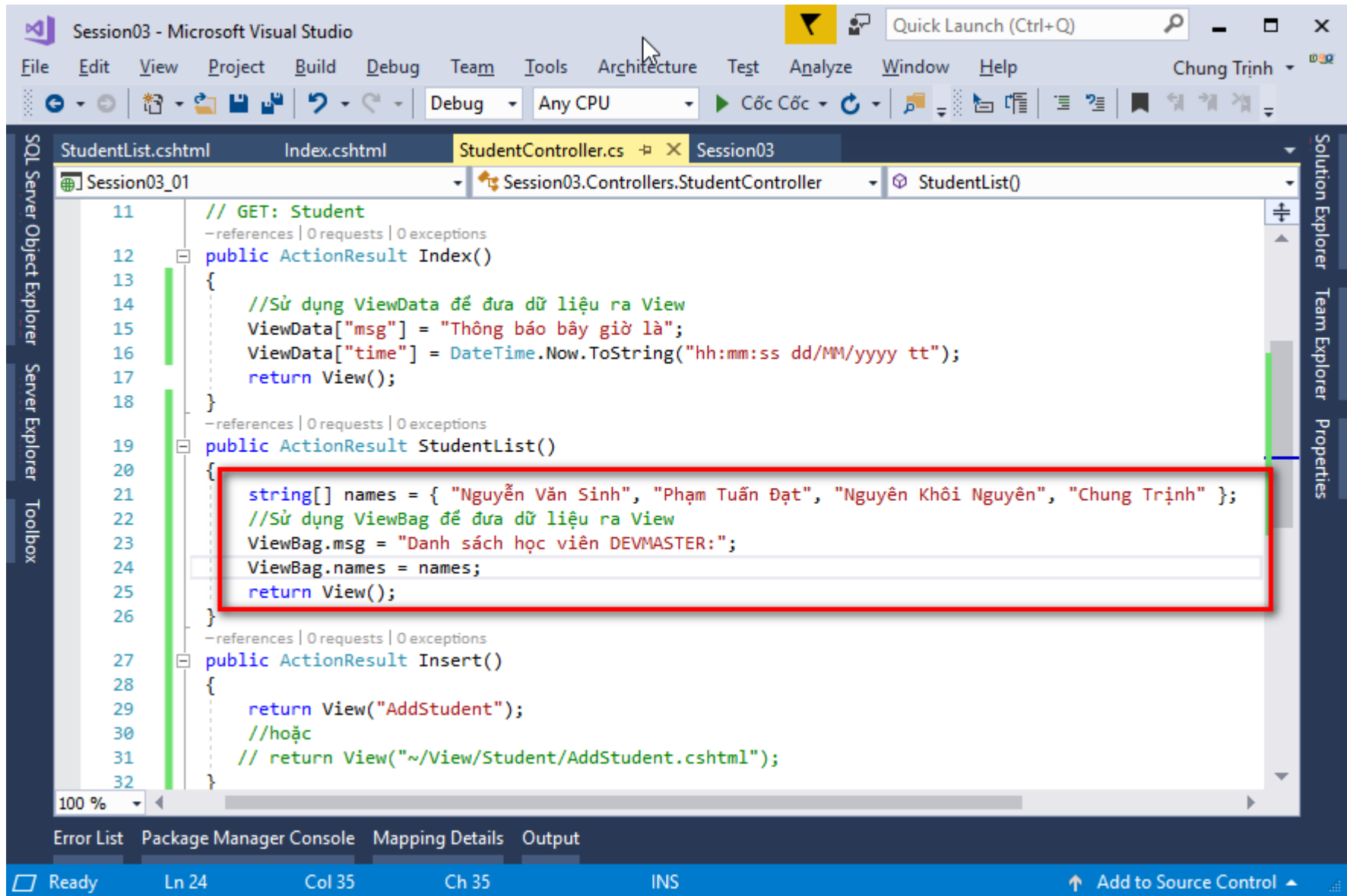


- ◆ Là đối tượng bao bọc ViewData. Chỉ tồn tại trong request hiện tại và null nếu request thực thi redirected.
- ◆ Cho phép tạo thuộc tính động dựa trên tính năng động của C# 4.0.
- ◆ Không đòi hỏi chuyển đổi kiểu khi sử dụng kiểu phức tạp.
- ◆ Cú pháp

ViewBag.<Property> = <Value>;



ViewBag



Session03 - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help Chung Trình

Debug Any CPU Cốc Cốc

StudentList.cshhtml Index.cshhtml StudentController.cs Session03

Session03_01 Session03.Controllers.StudentController StudentList()

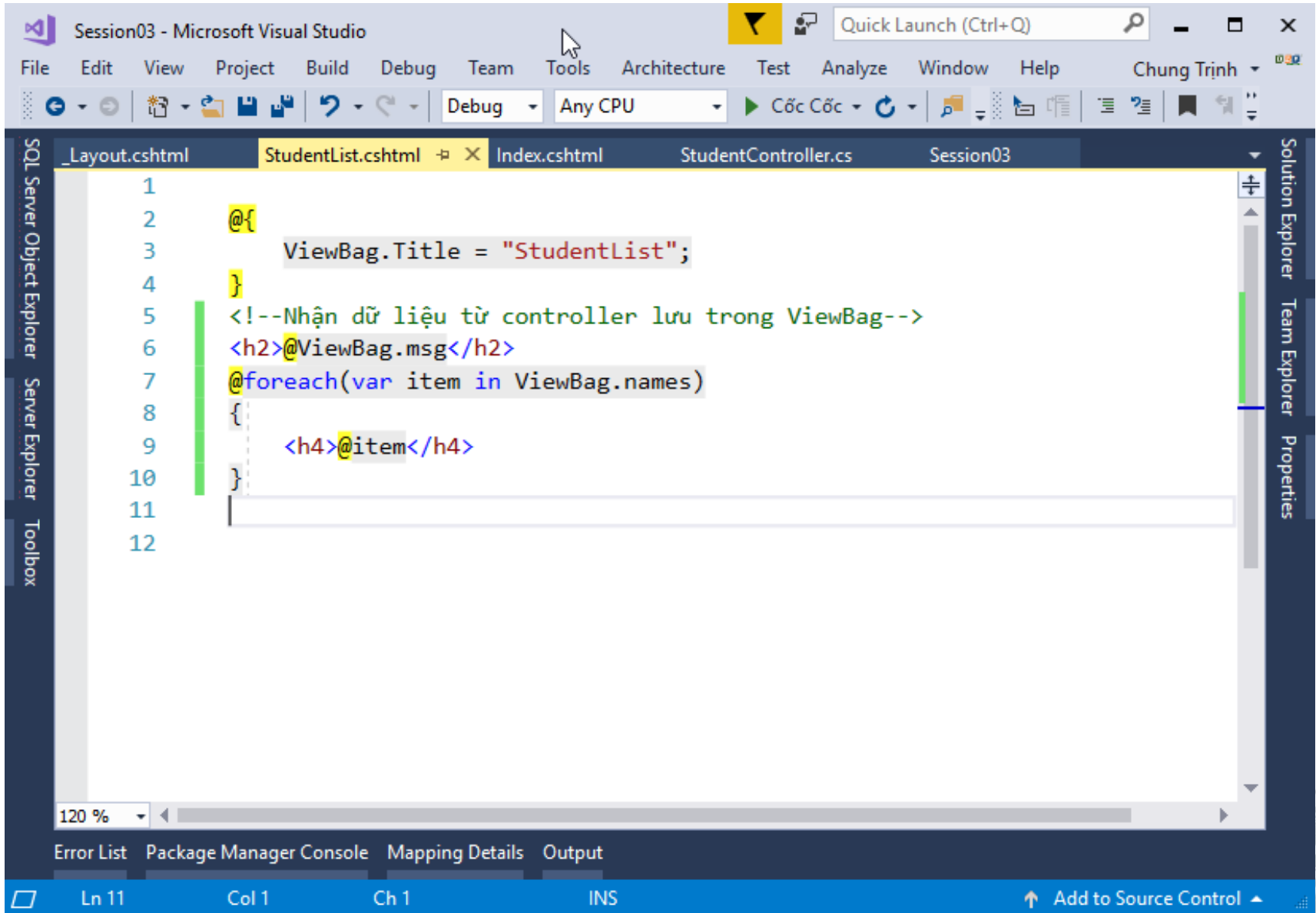
```
11 // GET: Student
12 -references | 0 requests | 0 exceptions
13 public ActionResult Index()
14 {
15     //Sử dụng ViewData để đưa dữ liệu ra View
16     ViewData["msg"] = "Thông báo bây giờ là";
17     ViewData["time"] = DateTime.Now.ToString("hh:mm:ss dd/MM/yyyy tt");
18     return View();
19 }
20 -references | 0 requests | 0 exceptions
21 public ActionResult StudentList()
22 {
23     string[] names = { "Nguyễn Văn Sinh", "Phạm Tuấn Đạt", "Nguyễn Khôi Nguyên", "Chung Trịnh" };
24     //Sử dụng ViewBag để đưa dữ liệu ra View
25     ViewBag.msg = "Danh sách học viên DEVMASTER:";
26     ViewBag.names = names;
27     return View();
28 }
29 -references | 0 requests | 0 exceptions
30 public ActionResult Insert()
31 {
32     return View("AddStudent");
33     //hoặc
34     // return View("~/View/Student/AddStudent.cshhtml");
35 }
```

100 %

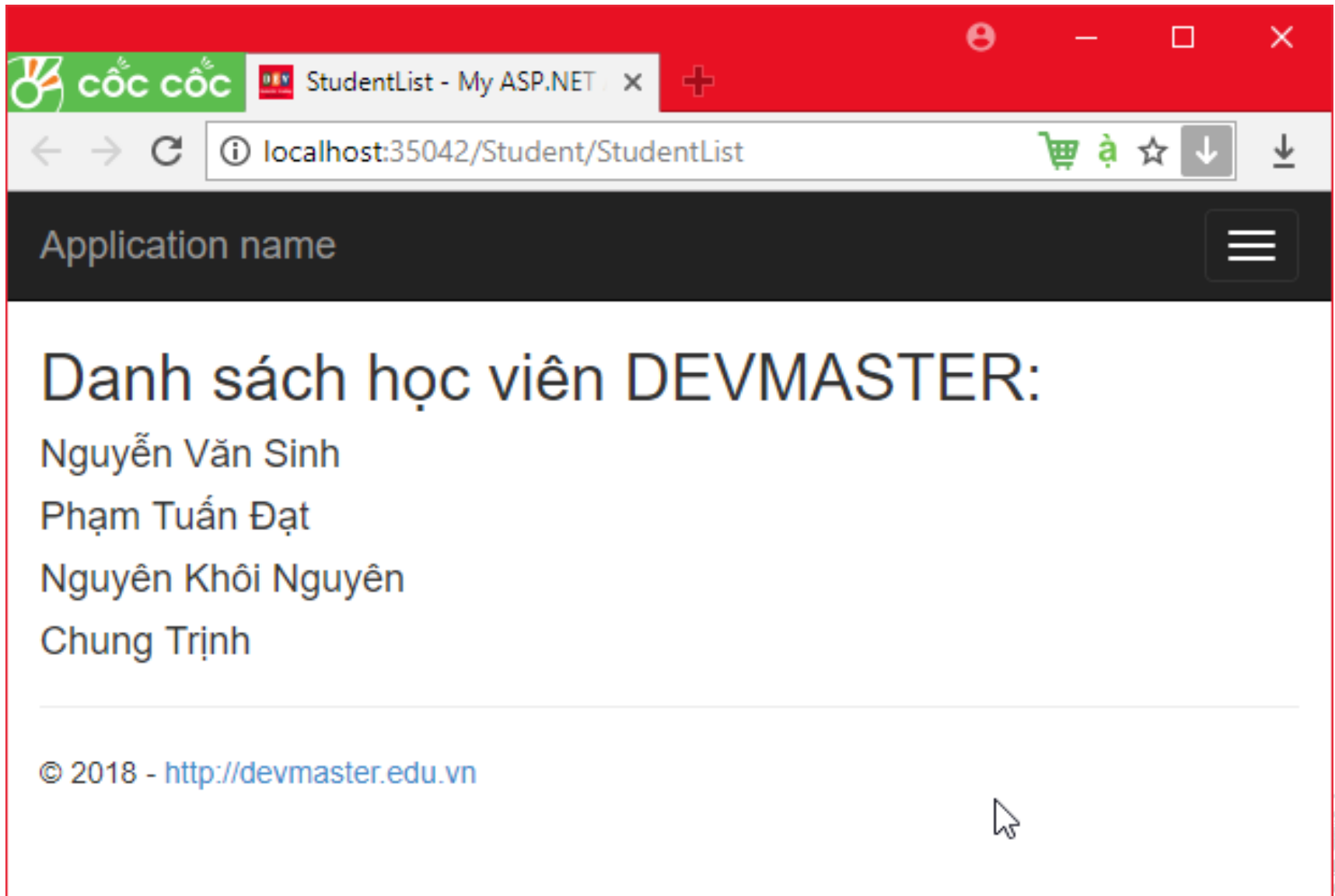
Error List Package Manager Console Mapping Details Output

Ready Ln 24 Col 35 Ch 35 INS Add to Source Control

ViewBag



```
1
2  @{
3      ViewBag.Title = "StudentList";
4  }
5  <!--Nhận dữ liệu từ controller lưu trong ViewBag-->
6  <h2>@ViewBag.msg</h2>
7  @foreach (var item in ViewBag.names)
8  {
9      <h4>@item</h4>
10 }
11
12
```

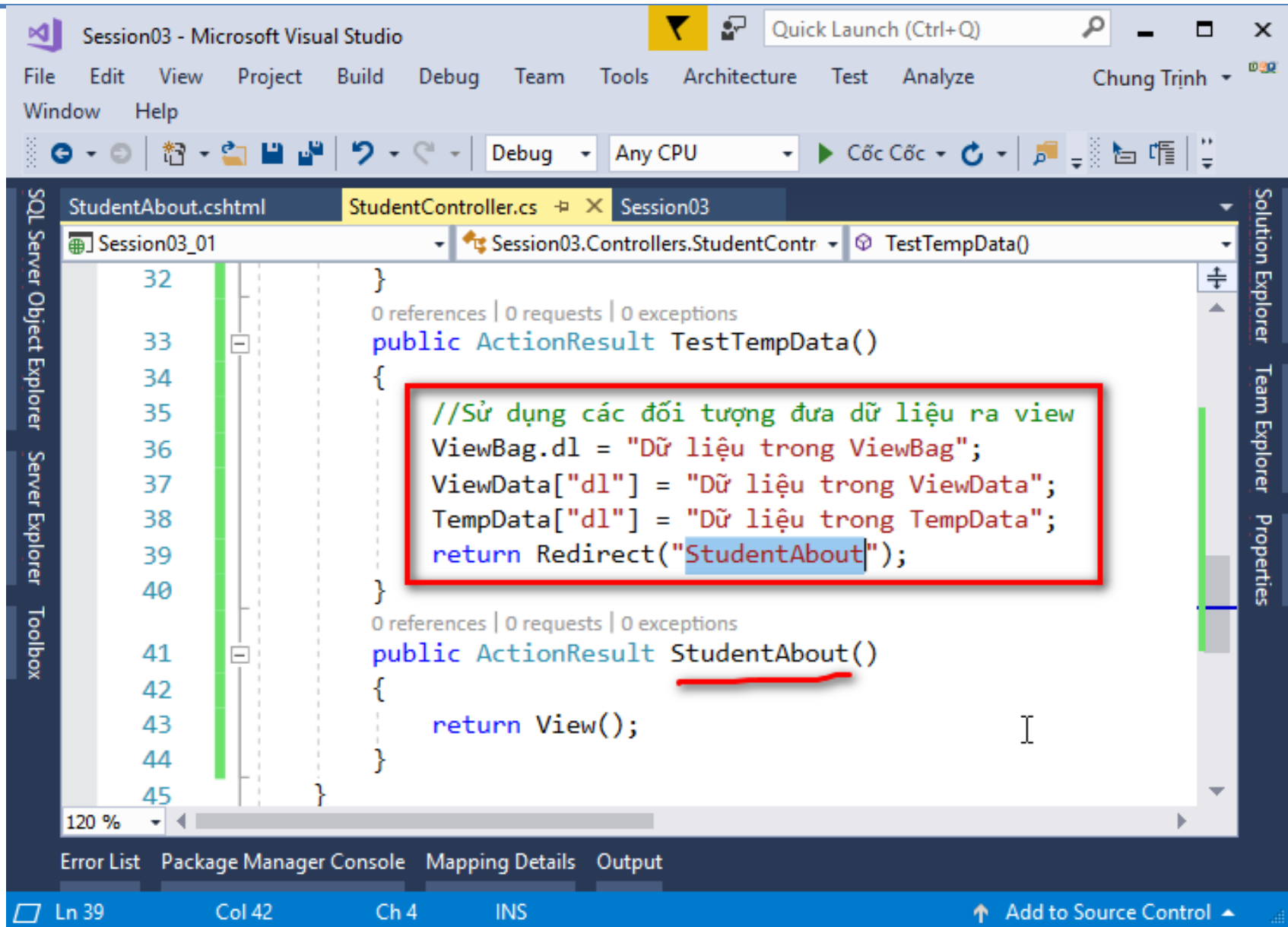


- ◆ Là đối tượng dạng Dictionary xuất phát từ lớp TempDataDictionary, tương tự ViewData tuy nhiên nó cho phép chuyển dữ liệu từ request hiện tại tới các request tiếp theo trong khi request redirection.

TempData[<Key>] = <Value>;



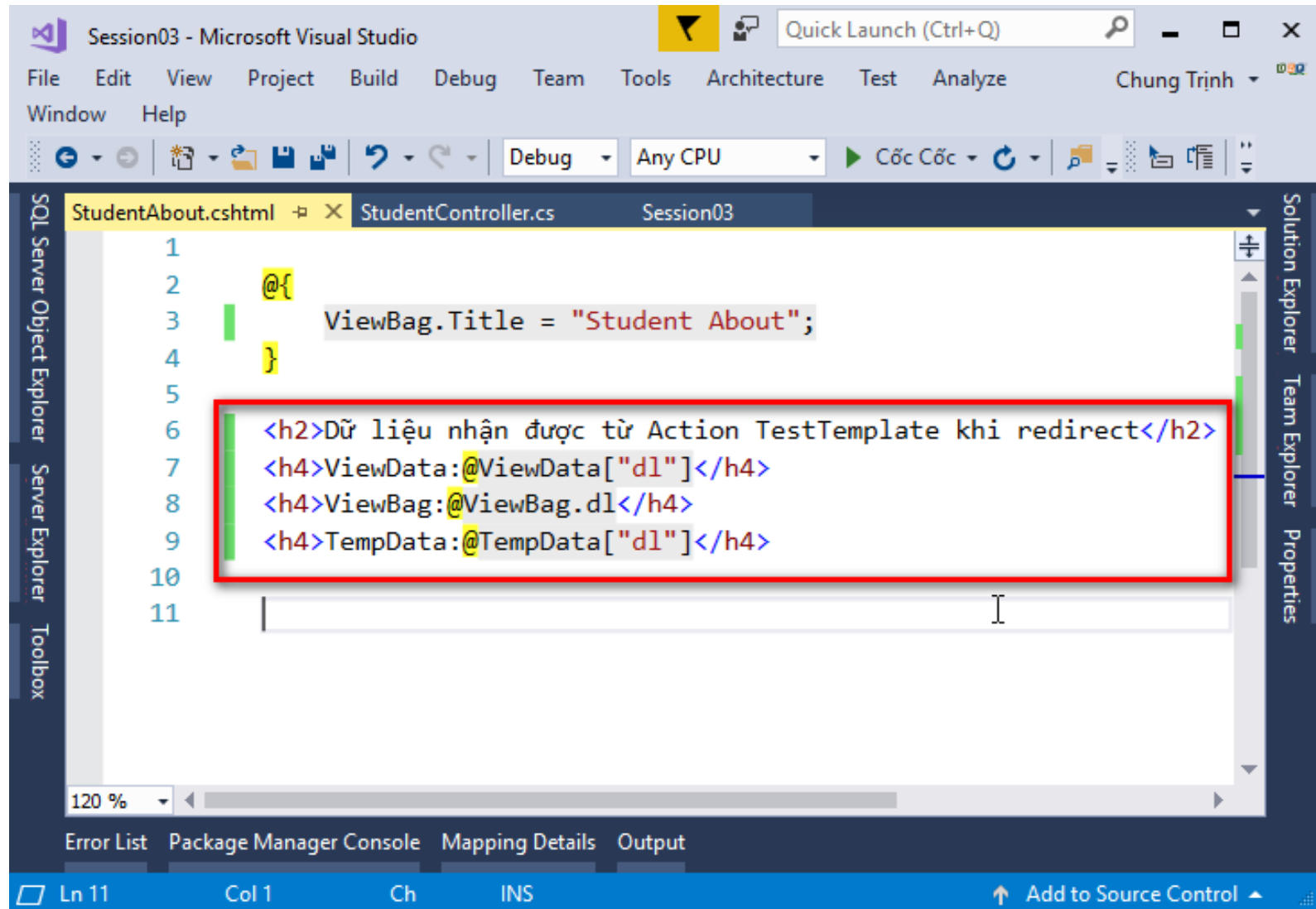
TempData



The screenshot shows the Visual Studio IDE with the file `StudentController.cs` open. The code is for a controller named `StudentController` in the namespace `Session3.Controllers`. The `TestTempData()` method is highlighted with a red box, and the `StudentAbout()` method is underlined with a red line.

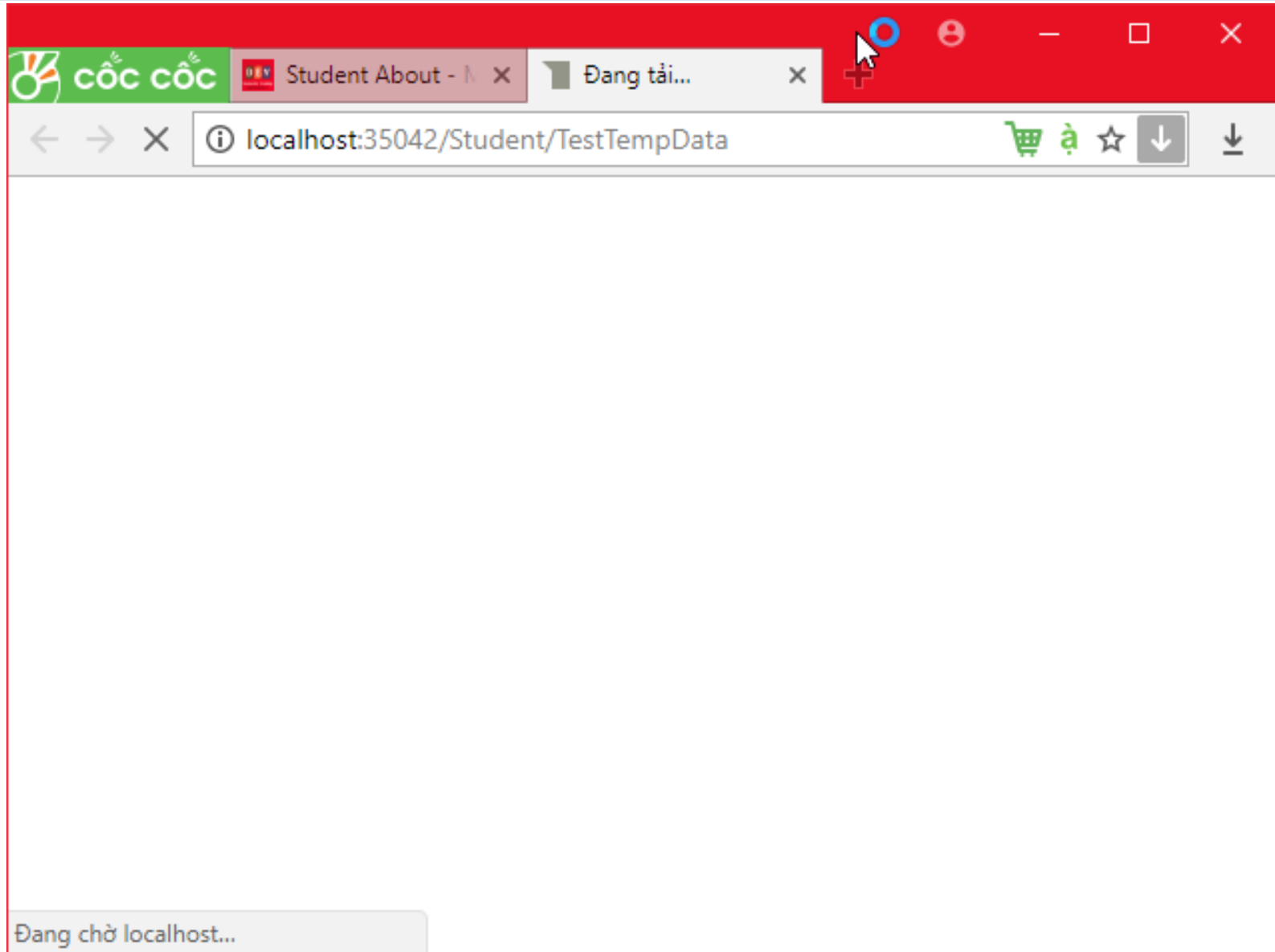
```
32 }  
33 0 references | 0 requests | 0 exceptions  
34 public ActionResult TestTempData()  
35 {  
36     //Sử dụng các đối tượng đưa dữ liệu ra view  
37     ViewBag.dl = "Dữ liệu trong ViewBag";  
38     ViewData["dl"] = "Dữ liệu trong ViewData";  
39     TempData["dl"] = "Dữ liệu trong TempData";  
40     return Redirect("StudentAbout");  
41 }  
42 0 references | 0 requests | 0 exceptions  
43 public ActionResult StudentAbout()  
44 {  
45     return View();  
}
```

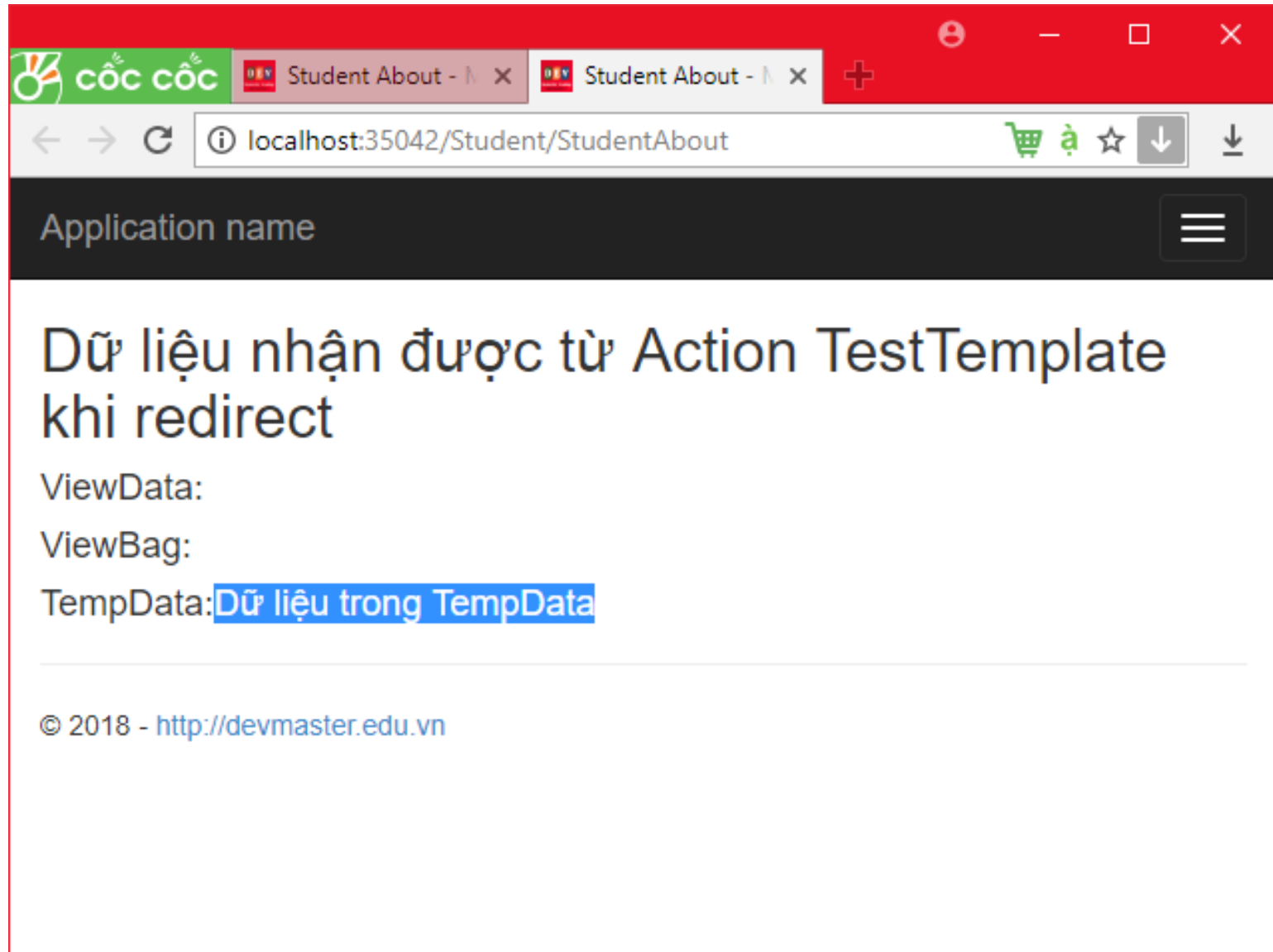
TempData



```
1
2  @{
3      ViewBag.Title = "Student About";
4  }
5
6  <h2>Dữ liệu nhận được từ Action TestTemplate khi redirect</h2>
7  <h4>ViewData:@ViewData["dl"]</h4>
8  <h4>ViewBag:@ViewBag.dl</h4>
9  <h4>TempData:@TempData["dl"]</h4>
10
11
```

TempData

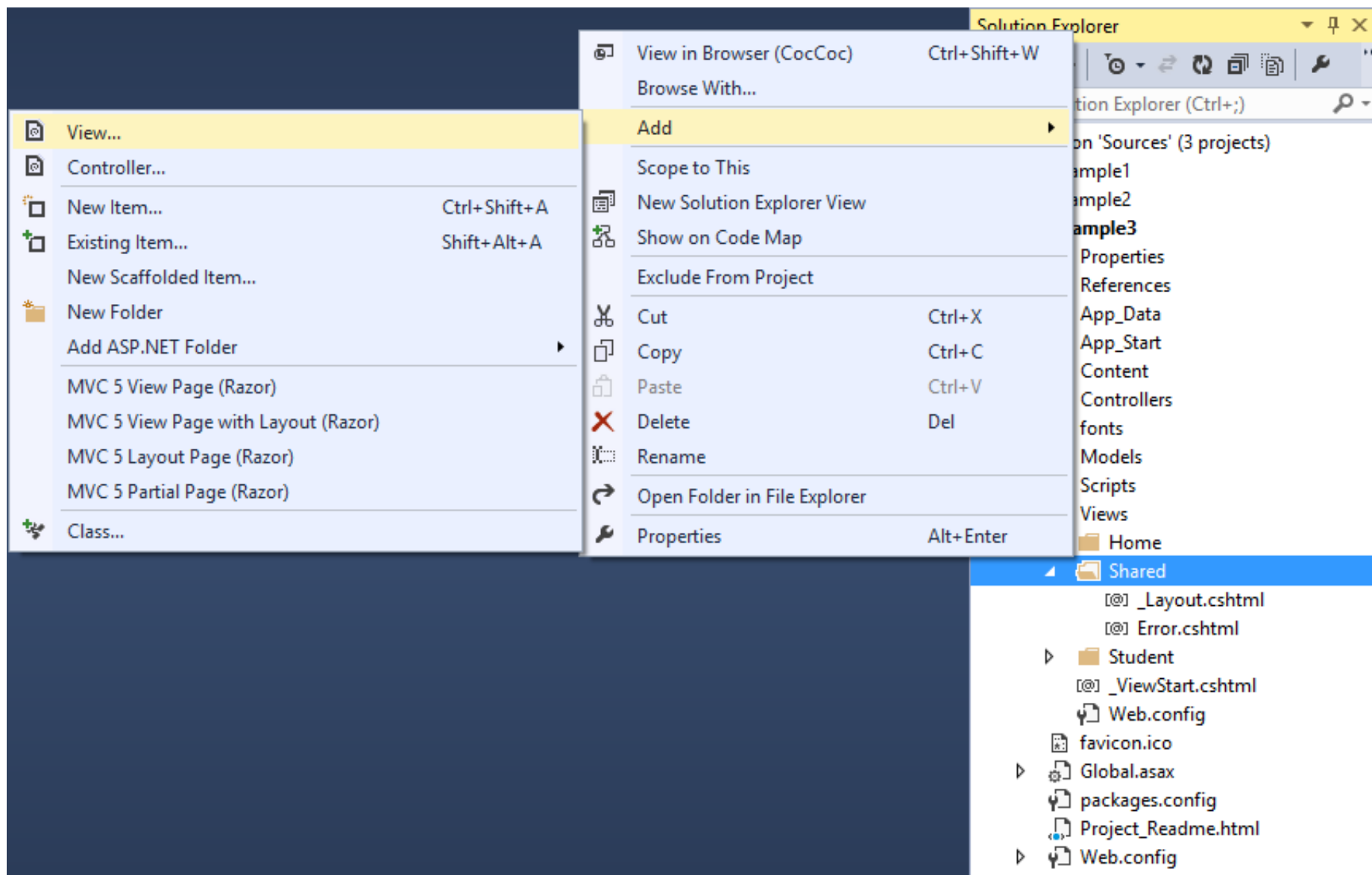




- ◆ Là một sub-view của main view.
- ◆ Cho phép tái sử dụng những mã ở các View khác nhau trong ứng dụng.

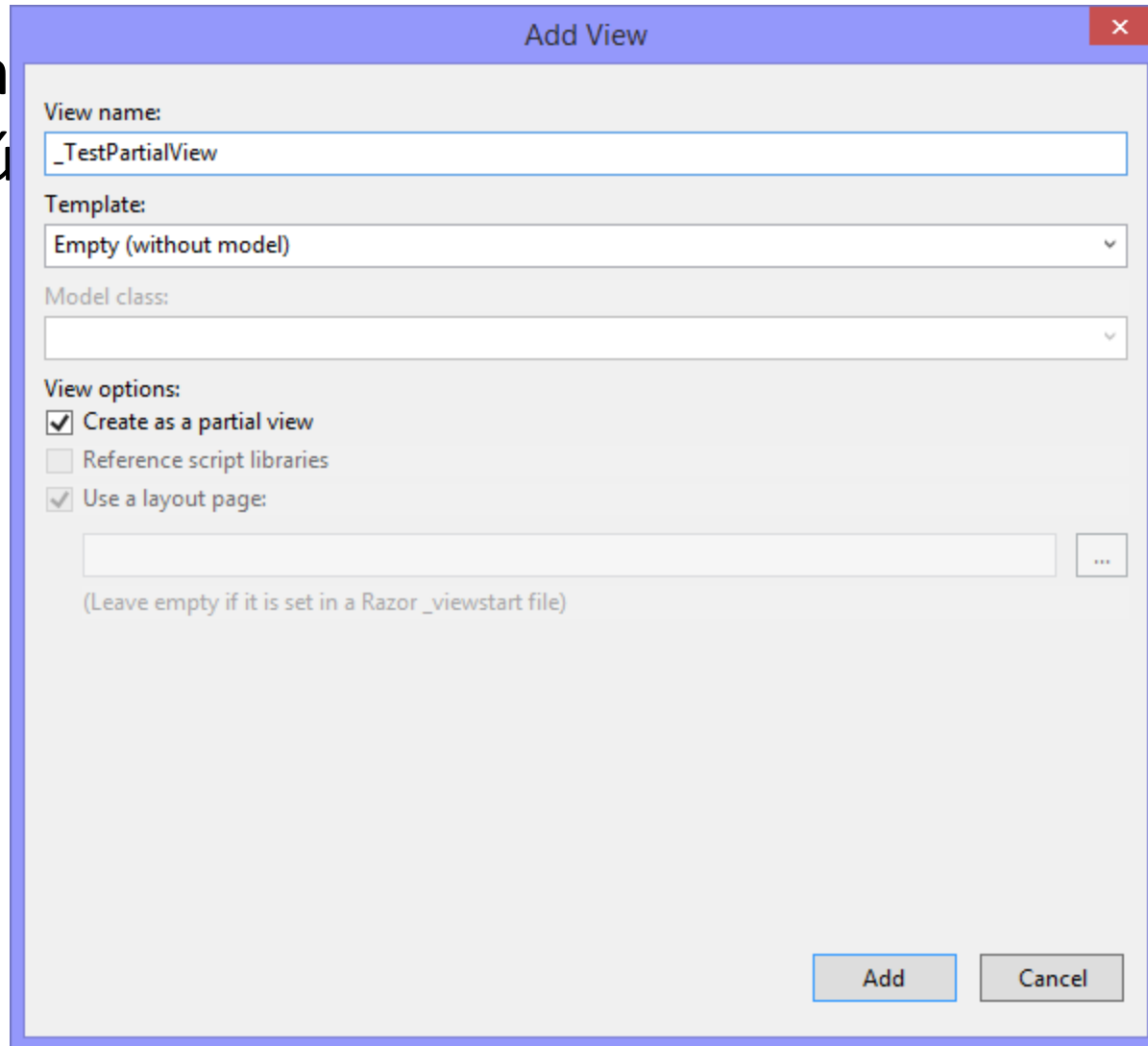


Sử dụng Partial Views



Sử dụng Partial Views

- ◆ Là một sub-view của main view.
- ◆ Cho phép chia sẻ các thành phần UI giữa các view khác nhau trong ứng dụng.



View name:
_TestPartialView

Template:
Empty (without model)

Model class:

View options:

☒ Create as a partial view

☐ Reference script libraries

☒ Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

Add Cancel

ác nhau

Sử dụng Partial Views

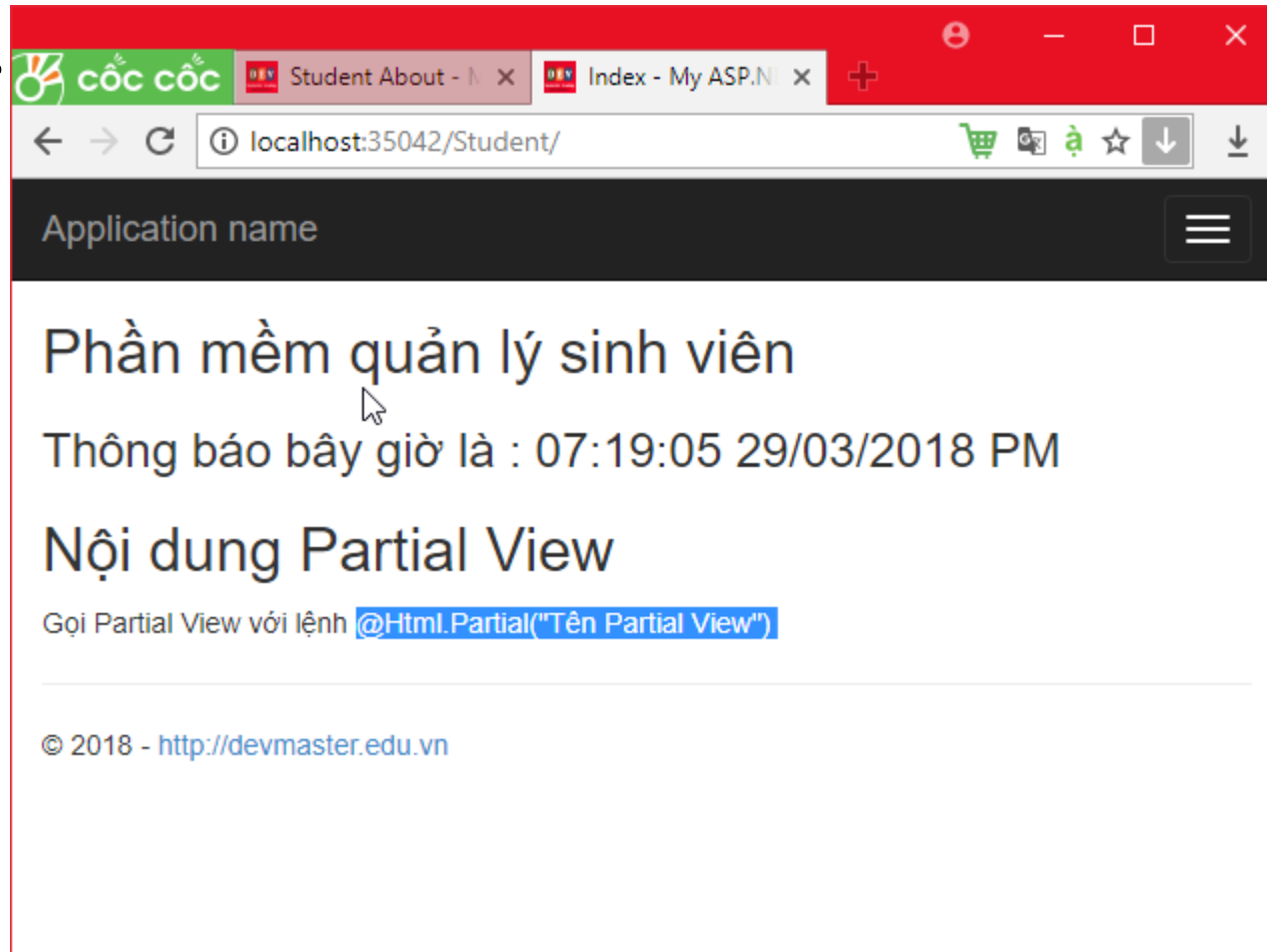
- ◆ Là một sub-view của main view.
- ◆ Cho phép tái sử dụng những mã ở các View khác nhau trong ứng dụng.

```
_TestPartialPage.cshtml  TestTempData.cshtml  StudentAbout.cshtml  StudentController.cs
1  <h2>Nội dung Partial View</h2>
2  Gọi Partial View với lệnh @Html.Partial("_TestPartialView")
3  |
```



Sử dụng Partial Views

- ◆ Là một sub-view của main view.
- ◆ Cho phép tái sử dụng những mã ở các View khác nhau trong ứng dụng.



Giới thiệu về Razor view engine

- ◆ Razor không phải là một ngôn ngữ lập trình. Đó là một cú pháp đánh dấu cho phép người dùng nhúng server-code (Visual Basic và C #) vào các trang web.
- ◆ Dễ sử dụng và dễ học, mã sử dụng trong Razor là sự phối hợp giữa ngôn ngữ lập trình C# hoặc VB.NET với mã HTML. Sự phối hợp này rất tự nhiên và dễ viết. Razor không lệ thuộc vào ASP.NET runtime
- ◆ Razor rất thông minh, có thể phân biệt được dễ dàng đâu là code C#, đâu là mã HTML, nó chỉ cần ký tự @ để cho biết điểm khởi đầu của một đoạn code C# và thậm chí chúng ta còn có thể trộn lẫn mã HTML và C# với nhau một cách phức tạp hơn mà Razor vẫn nhận ra được.
- ◆ Sự ra đời của Razor View Engine đi kèm với ASP.NET MVC 3 đã giải tỏa cơn khát cần phải có một View Engine phù hợp hơn cho ASP.NET MVC và chính thức được Microsoft hậu thuẫn.



- ◆ Razor sử dụng cú pháp rất đơn giản giống với PHP

```
<ul>
@for (int i = 0; i < 10; i++) {
<li>@i</li>
}
</ul>
```

PHP:

```
<ul>
<?php
for ($i = 0; $i < 10; $i++) {
echo("<li>$i</li>");
}
?>
</ul>
```

Web Forms (and Classic ASP):

```
<ul>
<% for (int i = 0; i < 10; i++) { %>
<li><% =i %></li>
<% } %>
</ul>
```



Quy tắc viết Razor trong C#

C# Example

```
<!-- Single statement block -->
@{ var myMessage = "Hello World"; }

<!-- Inline expression or variable -->
<p>The value of myMessage is: @myMessage</p>

<!-- Multi-statement block -->
@{
var greeting = "Welcome to our site!";
var weekDay = DateTime.Now.DayOfWeek;
var greetingMessage = greeting + " Today is: " + weekDay;
}
<p>The greeting is: @greetingMessage</p>
```



- ◆ Các biến sử dụng để lưu trữ dữ liệu.
- ◆ Các biến phải bắt đầu bằng 1 ký tự chữ cái không có khoảng trắng hoặc ký tự đặc biệt.



```
<!DOCTYPE html>
<html><body>
@{
var heading = "Using variables";
string greeting = "Welcome ASP.NET MVC Application";
int num = 103;
DateTime today = DateTime.Today;
<h3>@heading</h3>
<p>@greeting</p>
<p>@num</p>
<p>@today</p>
}
</body></html>
```

Type	Description	Examples
int	Integer (whole numbers)	103, 12, 5168
float	Floating-point number	3.14, 3.4e38
decimal	Decimal number (higher precision)	1037.196543
bool	Boolean	true, false
string	String	"Hello W3Schools", "John"



Các toán tử

Operator	Description	Example
=	Assigns a value to a variable.	i=6
+ - * /	Adds a value or variable. Subtracts a value or variable. Multiplies a value or variable. Divides a value or variable.	i=5+5 i=5-5 i=5*5 i=5/5
+= -=	Increments a variable. Decrements a variable.	i += 1 i -= 1
==	Equality. Returns true if values are equal.	if (i==10)
!=	Inequality. Returns true if values are not equal.	if (i!=10)
< > <= >=	Less than. Greater than. Less than or equal. Greater than or equal.	if (i<10) if (i>10) if (i<=10) if (i>=10)
+	Adding strings (concatenation).	"w3" + "schools"
.	Dot. Separate objects and methods.	DateTime.Hour
()	Parenthesis. Groups values.	(i+5)
()	Parenthesis. Passes parameters.	x=Add(i,5)
[]	Brackets. Accesses values in arrays or collections.	name[3]
!	Not. Reverses true or false.	if (!ready)
&& 	Logical AND. Logical OR.	if (ready && clear) if (ready clear)

Chuyển đổi các kiểu dữ liệu

Method	Description	Example
AsInt() IsInt()	Converts a string to an integer.	<pre>if (myString.IsInt()) {myInt=myString.AsInt();}</pre>
AsFloat() IsFloat()	Converts a string to a floating-point number.	<pre>if (myString.IsFloat()) {myFloat=myString.AsFloat();}</pre>
AsDecimal() IsDecimal()	Converts a string to a decimal number.	<pre>if (myString.IsDecimal()) {myDec=myString.AsDecimal();}</pre>
AsDateTime() IsDateTime()	Converts a string to an ASP.NET DateTime type.	<pre>myString="10/10/2012"; myDate=myString.AsDateTime();</pre>
AsBool() IsBool()	Converts a string to a Boolean.	<pre>myString="True"; myBool=myString.AsBool();</pre>
ToString()	Converts any data type to a string.	<pre>myInt=1234; myString=myInt.ToString();</pre>



- ◆ Câu lệnh if

Example

```
@{var price=50;}  
<html>  
<body>  
@if (price>30)  
{  
    <p>The price is too high.</p>  
}  
</body>  
</html>
```



- ◆ Câu lệnh else

Example

```
@{var price=20;}  
<html>  
<body>  
  @if (price>30)  
  {  
    <p>The price is too high.</p>  
  }  
  else  
  {  
    <p>The price is OK.</p>  
  }  
</body>  
</html>
```



◆ Câu lệnh Else If

Example

```
@{var price=25;}  
<html>  
<body>  
@if (price>=30)  
{  
    <p>The price is high.</p>  
}  
else if (price>20 && price<30)  
{  
    <p>The price is OK.</p>  
}  
else  
{  
    <p>The price is low.</p>  
}  
</body>  
</html>
```



◆ Câu lệnh switch

Example

```
@{
var weekday=DateTime.Now.DayOfWeek;
var day=weekday.ToString();
var message="";
}
<html>
<body>
@switch(day)
{
case "Monday":
    message="This is the first weekday.";
    break;
case "Thursday":
    message="Only one day before weekend.";
    break;
case "Friday":
    message="Tomorrow is weekend!";
    break;
default:
    message="Today is " + day;
    break;
}
<p>@message</p>
</body>
</html>
```



- ◆ Vòng lặp For

Example

```
<html>
<body>
  @for(var i = 10; i < 21; i++)
    {<p>Line @i</p>}
</body>
</html>
```



◆ Vòng lặp For Each

Example

```
<html>
<body>
<ul>
@foreach (var x in Request.ServerVariables)
    {<li>@x</li>}
</ul>
</body>
</html>
```



◆ Vòng lặp While

Example

```
<html>
<body>
@{
var i = 0;
while (i < 5)
{
    i += 1;
    <p>Line @i</p>
}
}
</body>
</html>
```



- ◆ Một mảng rất hữu ích khi bạn muốn lưu trữ các biến tương tự nhưng không muốn để tạo ra một biến riêng biệt cho mỗi trong số họ:

Example

```
@{
    string[] members = {"Jani", "Hege", "Kai", "Jim"};
    int i = Array.IndexOf(members, "Kai")+1;
    int len = members.Length;
    string x = members[2-1];
}
<html>
<body>
<h3>Members</h3>
@foreach (var person in members)
{
    <p>@person</p>
}
<p>The number of names in Members are @len</p>
<p>The person at position 2 is @x</p>
<p>Kai is now in position @i</p>
</body>
</html>
```



- ◆ HTML Helper là các phương thức mở rộng từ của lớp `HtmlHelper`, chúng chỉ được gọi ở View, hầu hết các phương thức này trả về 1 chuỗi mã HTML để hiển thị trên trình duyệt.
- ◆ Một số phương thức phổ biến
 - ◆ `Html.ActionLink()`
 - ◆ `Html.BeginForm()` and `Html.EndForm()`
 - ◆ `Html.Label()`
 - ◆ `Html.TextBox()`
 - ◆ `Html.TextArea()`
 - ◆ `Html.Password()`
 - ◆ `Html.CheckBox()`



- ◆ `Html.ActionLink()` sẽ sinh ra một HyperLink html mà nó thi trở tới một Action trong Controller.
- ◆ Cú pháp
`@Html.ActionLink(<link_text>,<action_method>,<optional_controller>)`

```
<!DOCTYPE html>
<html>
<body>
@Html.ActionLink("Click to Browse", "Browse", "Home")
</body>
</html>
```

- ◆ `Html.BeginForm()` sẽ sinh ra một thẻ Form mở và phối hợp với routing engine để sinh ra URL.
- ◆ Cú pháp

```
@{Html.BeginForm(<action_method>,<controller_name>);}
```
- ◆ `Html.EndForm()` sẽ sinh ra thẻ đóng của form
- ◆ Cú pháp

```
@{Html.EndForm();}
```



HTML Helper Methods

```
<!DOCTYPE html>
<html>
<body>
@{Html.BeginForm("Browse","Home");}
<p>Inside Form</p>
@{Html.EndForm();}
</body>
</html>
```

- ◆ Để tránh sử dụng phương thức `Html.EndForm()` bạn có thể sử dụng câu lệnh `@using` trước phương thức `Html.BeginForm()`.

```
@using(Html.BeginForm("Browse", "Home"))  
{  
    <p>Inside Form</p>  
}
```



- ◆ `Html.Label()`: Cho phép hiển thị 1 nhãn trên form, bổ sung thông tin mô tả cho các control khác, tăng khả năng truy cập ứng dụng.
- ◆ Cú pháp:
`@Html.Label(<label_text>)`

```
@{Html.BeginForm("Browse","Home");}  
@Html.Label("User Name:")</br>  
@{Html.EndForm();}
```



- ◆ `Html.TextBox()`: Cho phép hiển thị một thẻ input textbox để nhận dữ liệu từ người dùng.

- ◆ Cú pháp

`@Html.TextBox("textbox_text")`

```
@{Html.BeginForm("Browse","Home");}  
@Html.Label("User Name:")</br>  
@Html.TextBox("textBox1")</br>  
<input type="submit" value="Submit">  
@{Html.EndForm();}
```



- ◆ `Html.TextArea()`: Cho phép hiển thị một ô văn bản có thể nhập được nhiều dòng, bạn có thể chỉ ra số dòng, số cột của `TextArea`.
- ◆ Cú pháp:

```
@{Html.BeginForm("Browse","Home");}  
@Html.Label("User Name:")</br>  
@Html.TextBox("textBox1")</br></br>  
@Html.Label("Address:")</br>  
@Html.TextArea("textarea1")</br></br>  
<input type="submit" value="Submit">  
@{Html.EndForm();}
```

- ◆ `Html.Password()` cho phép hiển thị một thẻ input để nhập mật khẩu
- ◆ Cú pháp:

```
@{Html.BeginForm("Browse","Home");}  
@Html.Label("User Name:")</br>  
@Html.TextBox("textBox1")</br></br>  
@Html.Label("Address:")</br> @Html.TextArea("textarea1")</br></br>  
@Html.Label("Password:")</br> @Html.Password("password")</br></br>  
<input type="submit" value="Submit">
```



- ◆ `Html.CheckBox()` hiển thị một input `CheckBox` cho phép người dùng chọn 1 giá trị `true` hoặc `false`.

```
@{Html.BeginForm("Browse","Home");}  
@Html.Label("User Name:")</br>  
@Html.TextBox("textBox1")</br></br>  
@Html.Label("Address:")</br> @Html.TextArea("textarea1")</br></br>  
@Html.Label("Password:")</br> @Html.Password("password")</br></br>  
@Html.Label("I need updates on my mail:")  
@Html.CheckBox ("checkbox1")</br> </br>  
<input type="submit" value="Submit"> @{Html.EndForm();}
```



- ◆ `Html.RadioButton()` hiển thị một option radio, bạn có thể tạo ra nhiều option radio và chọn 1 trong các giá trị đó.



HTML Helper Methods

```
@{Html.BeginForm("Browse","Home");}  
@Html.Label("User Name:")</br>  
@Html.TextBox("textBox1")</br></br>  
@Html.Label("Address:")</br> @Html.TextArea("textarea1")</br></br>  
@Html.Label("Password:")</br>@Html.Password("password")</br></br>  
@Html.Label("I need updates on my  
mail:")@Html.CheckBox("checkbox1")</br> </br>  
@Html.Label("Select your city:") @Html.DropDownList("myList", new  
SelectList(new [] { "New York", "Philadelphia", "California" } ),  
"Choose")</> </br></br>  
Male @Html.RadioButton("Gender", "Male", true)</br>  
Female @Html.RadioButton("Gender", "Female")</br> </br>  
<input type="submit" value="Submit">  
@{Html.EndForm();}
```

HTML Helper Methods

```
@{Html.BeginForm("Browse","Home");}  
@Html.Label("User Name:")</br>  
@Html.TextBox("textBox1")</br></br>  
@Html.Label("Address:")</br> @Html.TextArea("textarea1")</br></br>  
@Html.Label("Password:")</br> @Html.Password("password")</br></br>  
@Html.Label("I need updates on my mail:")  
@Html.CheckBox("checkbox1")</br> </br>  
@Html.Label("Select your city:")  
@Html.DropDownList("myList", new SelectList(new [] {"New York",  
"Philadelphia", "California"}), "Choose")</> </br></br>  
<input type="submit" value="Submit">  
@{Html.EndForm();}
```

- ◆ The `Url.Action()` sinh ra một URL mà nó trở tới một action trong controller.
- ◆ Cú pháp:
`@Url.Action(<action_name>, <controller_name>)`

```
<!DOCTYPE html>  
<html> <body>  
<a href='@Url.Action("Browse", "Home")'>Browse</a>  
</body> </html>
```



