

LẬP TRÌNH WEB



Nguyễn Thị Mai Trang

1

Nội dung

- Chương 1: Tổng quan
- Chương 2: Ngôn ngữ PHP
- **Chương 3: Controller – View – Model**
- Chương 4: HTML Helpers
- Chương 5: Làm việc với Cơ sở dữ liệu

2

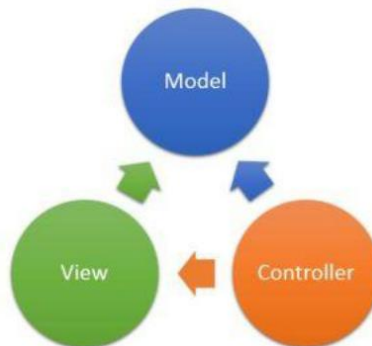
Chương 3

Controller – View - Model

3

Nội dung

- Controller
- Model
- View



4


3.1 Controller

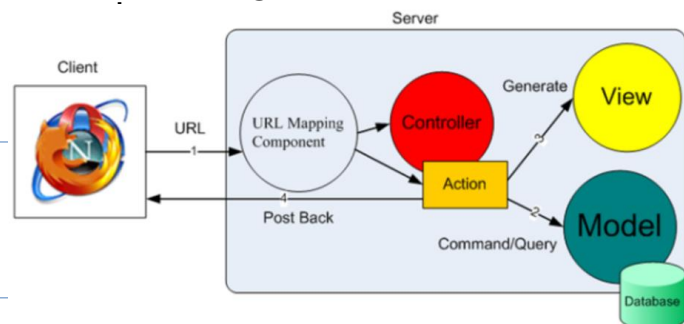
- Giới thiệu Controller
- Action trong Controller
- Thao tác với Controller
- Ví dụ sử dụng Controller

5

3.1.1 Giới thiệu Controller

- Controller là một trong các thành phần chính của ứng dụng ASP.NET MVC.
- Controller trong kiến trúc MVC xử lý các yêu cầu từ URL.
- Controller là một class kế thừa từ lớp cơ sở `System.Web.Mvc.Controller`
- Tất cả các class controller đều có hậu tố là "Controller"
 - HomeController
 - AccountController

 Controllers
 ▶ C# AccountController.cs
 ▶ C# HomeController.cs
 ▶ C# ManageController.cs



6

3.1.2 Action trong Controller

• Phương thức Action

- Class Controller chứa các phương thức có mức truy cập là public, gọi là các Action.
- Các phương thức Action không được overloaded và không được khai báo static.
- Tất cả các controller đều có phương thức action mặc định là **Index()**, được tạo tự động khi một Controller được tạo.
- Controller và các phương thức Action tiếp nhận các yêu cầu từ phía trình duyệt, truy vấn các thông tin cần thiết từ Model sau đó trả lại kết quả cho View

```
public class HomeController : Controller
{
    MusicStoreDB storeDB = new MusicStoreDB();
    public ActionResult Index()
    {
        return View();
    }

    public ActionResult About()
    {
        ViewBag.Message = "Your application description page.";
        return View();
    }
}
```

Lập trình web

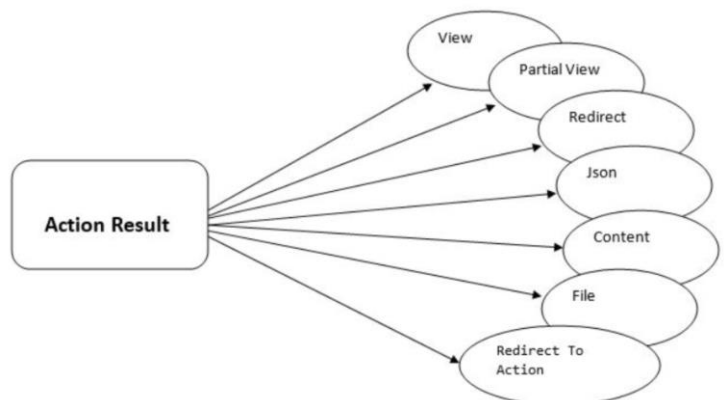
7

7

Action trong Controller (tt)

• ActionResult: là đối tượng trả về của các phương thức Action()

- Trả về HTML
- Chuyển hướng người dùng
- Trả về file
- Trả về nội dung văn bản
- Trả về lỗi và HTTP Code
- Kết quả liên quan đến bảo mật
- ...



Lập trình web

8

8

Action trong Controller(tt)

• Trả về HTML

– ViewResult

- Phương thức View() tìm kiếm View trong thư mục Views/<Controller> để tìm file .cshtml và chuyển nó cho Razor View Engine. View sẽ trả về một ViewResult và kết quả là một HTML Response.

```
public ActionResult Index() {
    var movie = new Movie() { Name = "Avatar" };
    return View(movie);
}
```

Action trong Controller(tt)

– ViewResult (tt)

- Ta có thể thay đổi bằng cách override và cung cấp một view name

```
public ActionResult Index() {
    return View("NotIndex");
}
```

- Khi đó, trong thư mục /Views/Home phải tồn tại tập tin **NotIndex.cshtml**
- Trường hợp tham chiếu đến tập tin View trong một thư mục khác, cần chỉ rõ đường dẫn dạng:

Ví dụ:

```
public ActionResult Index() {
    return ("~/Views/Example/Index.cshtml");
}
```

Action trong Controller(tt)

• Trả về HTML

– PartialViewResult

- PartialView là một phần của View (có thể xem như UserControl trong ASP.NET WebForm) PartialView Result sử dụng model để tạo ra một phần của View
- ViewResult tạo ra một view hoàn chỉnh còn PartialView trả về một phần của View.
- Thường dùng khi muốn cập nhật một phần của View thông qua AJAX.

```
public ActionResult Index() {
    var movie = new Movie() {Name = "Avatar" };
    return PartialView(movie);
}
```

Action trong Controller (tt)

• Chuyển hướng người dùng

- **Redirect result:** chuyển hướng người dùng đến một URL khác.
 - Ví dụ: Redirect("/Product/Index");
- **LocalRedirectResult:** tương tự như RedirectResult nhưng chỉ chuyển hướng đến các local URL (thường dùng để tránh việc bị tấn công [open redirect attack](#))
 - Ví dụ: LocalRedirect("/Product/Index");
- **RedirectToActionResult:** chuyển đến một action trong controller, tham số là tên Action method, tên controller và các giá trị tham số:

```
//Redirects đến action Index trong StoreController
RedirectToAction(actionName: "Index", controllerName: "Store");
//Redirects đến action Index trong cùng Controller
RedirectToAction(actionName: "Index");
```

Action trong Controller (tt)

• Chuyển hướng người dùng (tt)

- **RedirectToRouteResult**: chuyển đến một route, tham số là tên route, giá trị của route và chuyển người dùng đến vị trí được chỉ định trong route:

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new {controller = "Home", action = "Index", id = UrlParameter.Optional}
);
```

```
// Redirect đến route có tên là "default"
return RedirectToRoute("default");
//Redirect đến một route với tham số
return RedirectToRoute(new {action = "Index", controller = "Home", Id = "myId"});
//Redirect sử dụng Route Value
var routeValue = new RouteValueDictionary(new {action="Index", controller="Home"});
return RedirectToRoute(routeValue);
```

Lập trình web

13

13

Action trong Controller (tt)

- **FileResult**:

```
public FileResult DownloadFile() {
    return File("/Files/tutorial.pdf", "text/plain", "tutorial.pdf");
}
```

- **FileContentResult**: đọc một mảng byte và trả về như một file:

```
public FileContentResult DownloadContent(){
    var myfile = System.IO.File.ReadAllBytes("wwwroot/Files/FileContentResult.pdf")
    return new FileContentResult(myfile, "application/pdf");
}
```

- **FileStreamResult**: đọc một luồng stream và trả về một file:

```
public FileStreamResult CreateFile() {
    var stream = new MemoryStream(Encoding.ASCII.GetBytes("Hello World"));
    return new FileStreamResult(stream, "text/plain") {
        FileDownloadName = "test.txt"
    };
}
```

Lập trình web

14

14

Action trong Controller (tt)

- **ContentResult:** trả về chuỗi văn bản

```
public ContentResult About() { return Content("<b>Hello World</b>"); }
```

- **JsonResult:** trả về dữ liệu được định dạng JSON bằng cách chuyển một object sang JSON

```
private List<UserModel> GetUsers(){
    var userList = new List<UserModel>{
        new UserModel {UserId = 1, UserName = "Bill",
            Company = "ABC Solutions"},
        new UserModel {UserId = 2, UserName = "Jonh",
            Company = "ABC Solutions"}};
    return userList;
}
```

```
public class UserModel
{
    public int UserId { get; set; }
    public string UserName { get; set; }
    public string Company { get; set; }
}
```

```
public JsonResult GetUsersData() {
    var users = GetUsers();
    return Json(users, JsonRequestBehavior.AllowGet);
}
```

- **EmptyResult:** trả về dữ liệu rỗng, ví dụ: return new EmptyResult();

Action trong Controller (tt)

- **ContentResult:** trả về chuỗi văn bản

```
public ContentResult About() { return Content("<b>Hello World</b>"); }
```

- **JsonResult:** trả về dữ liệu được định dạng JSON bằng cách chuyển một object sang JSON

```
private List<UserModel> GetUsers(){
    var userList = new List<UserModel>{
        new UserModel {UserId = 1, UserName = "Bill",
            Company = "ABC Solutions"},
        new UserModel {UserId = 2, UserName = "Jonh",
            Company = "ABC Solutions"}};
    return userList;
}
```

```
public class UserModel
{
    public int UserId { get; set; }
    public string UserName { get; set; }
    public string Company { get; set; }
}
```

```
public JsonResult GetUsersData() {
    var users = GetUsers();
    return Json(users, JsonRequestBehavior.AllowGet);
}
```

- **EmptyResult:** trả về dữ liệu rỗng, ví dụ: return new EmptyResult();

Action trong Controller (tt)

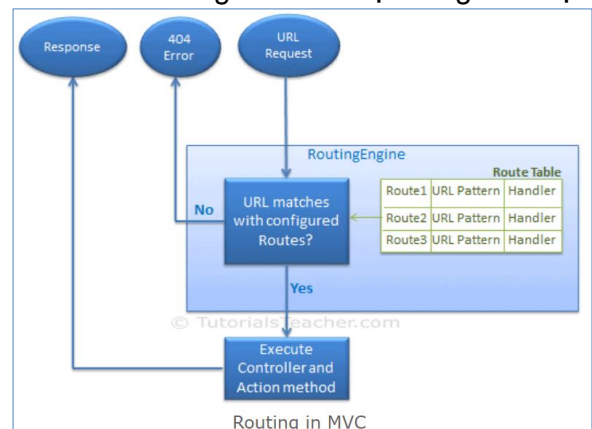
• Gọi phương thức Action

- Có thể gọi phương thức Action thông qua địa chỉ URL trên trình duyệt thông qua tên controller và tên phương thức.
 - Cú pháp :`http:// <domain_name> /<controller_name>/<action_name>/[parameter]`
 - <domain_name>: tên máy chủ Web theo DNS
 - <controller_name>: tên controller
 - <action_name>: tên phương thức Action trong <controller_name>
 - Ví dụ: `http:// cunghoclaptrinh.com/Home/TopicList`
- Gọi trong Route

Action trong Controller (tt)

• Route (Định tuyến)

- Nhận và định hướng từ các yêu cầu đến từ client mà không cần tên một trang web cụ thể và ánh xạ các yêu cầu tới một action trong Controller.
- Xác định các URL gửi đi tương ứng với các Action trong Controller



Action trong Controller (tt)

• Route trong RouteConfig.cs

```
public static void RegisterRoutes(RouteCollection routes)
{
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

    routes.MapRoute(
        name: "Default",
        url: "{controller}/{action}/{id}",
        defaults: new {controller = "Home", action="Index", id = UrlParameter.Optional}
    );
    routes.MapRoute(
        name: "ProductsbyCategory",
        url: "Products/{category}",
        defaults: new {controller = "Products", action = "Index"}
    );
}
```

Annotations in the code:

- Route name**: points to the `name` parameter in the first `MapRoute` call.
- URL Pattern**: points to the `url` parameter in the first `MapRoute` call.
- Default for Route**: points to the `defaults` parameter in the first `MapRoute` call.

Lập trình web

19

19

Action trong Controller (tt)

• Route (tt)

- URL Pattern: phần nằm sau tên miền, nếu url là rỗng, mặc định Controller là Home và action là index

Diagram illustrating URL patterns and their components:

Top URL: `http://localhost:1234/home/index/100`

- `home`: Controller
- `index`: Action method
- `100`: Id parameter value

Bottom URL: `http://localhost:1234/home/index`

- `home`: Controller
- `index`: Action method

Lập trình web

20

20

Action trong Controller (tt)

• Route (tt):

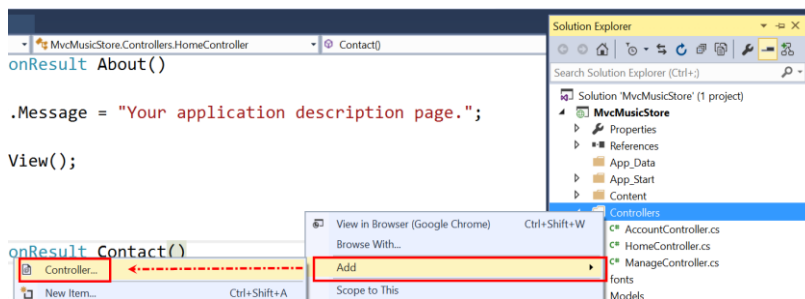
– Mối liên quan giữa địa chỉ URL và các tham số trong URL Pattern

URL	Controller	Action	Id
http://localhost/home	HomeController	Index	null
http://localhost/home/index/123	HomeController	Index	123
http://localhost/home/about	HomeController	About	null
http://localhost/home/contact	HomeController	Contact	null
http://localhost/student	StudentController	Index	null
http://localhost/student/edit/123	StudentController	Edit	123

3.1.3 Thao tác với Controller

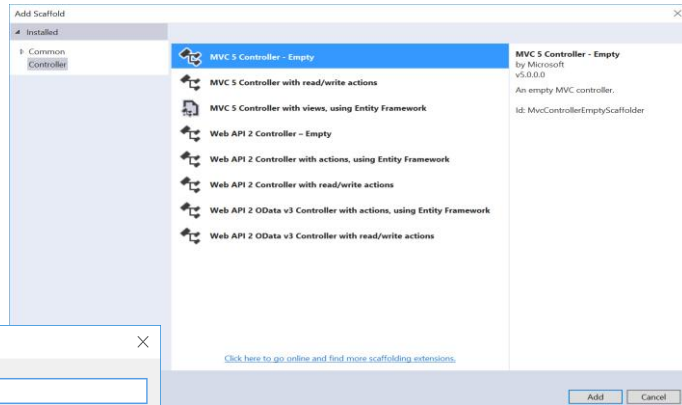
• Thêm mới Controller:

– Click chuột phải trên thư mục Controller → Add → Controller

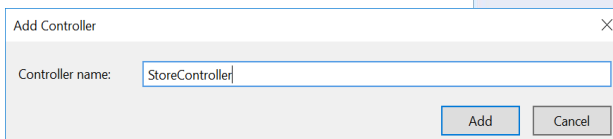


Thao tác với Controller (tt)

- Để thêm một Controller rỗng, chọn MVC 5 Controller-Empty



- Đặt tên cho Controller



Lập trình web

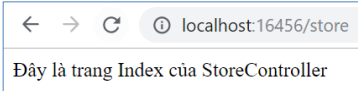
23

23

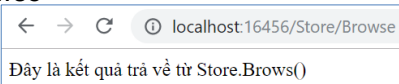
Thao tác với Controller (tt)

- Trong StoreController mới tạo chỉ có duy nhất một phương thức Index()
- Bổ sung hai phương thức như hình:
- Thực thi ứng dụng với các URL:

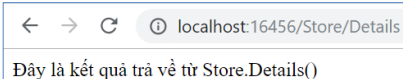
/Store:



/Store/Browse



/Store/Details



```
public class StoreController : Controller
{
    // GET: Store
    0 references
    public string Index()
    {
        return "Đây là trang Index của StoreController";
    }
    0 references
    public string Browse()
    {
        return "Đây là kết quả trả về từ Store.Brows()";
    }
    0 references
    public string Details()
    {
        return "Đây là kết quả trả về từ Store.Details()";
    }
}
```

Lập trình web

24

24

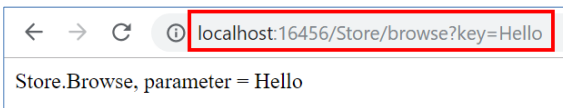
Thao tác với Controller (tt)

- Sử dụng tham số trong Controller Actions:

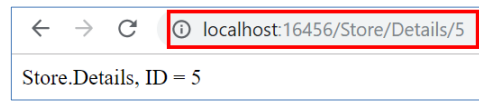
```
public string Browse(string key)
{
    string message = HttpUtility.HtmlEncode("Store.Browse, parameter = " + key);
    return message;
}
0 references
public string Details(int id)
{
    string message = "Store.Details, ID = " + id;
    return message;
}
```

- Thực thi ứng dụng với các URL:

<http://localhost:16456/Store/browse?key=Hello>



<http://localhost:16456/Store/Details/5>



3.1.4 Ví dụ sử dụng Controller

- Tạo ứng dụng MVC: BookManager

- Tạo và hiển thị danh mục sách
- Mỗi quyển sách có ba trường:

- BookId
- Title
- Author

- Tạo class Books trong Model

```
namespace BookManager.Models
{
    11 references
    public class Books
    {
        6 references
        public int BookId { get; set; }
        6 references
        public string Title { get; set; }
        6 references
        public string Author { get; set; }
    }
}
```

Ví dụ sử dụng Controller (tt)

- Tạo Controller BookController

```
public IActionResult BookList()
{
    List<Books> listBooks = GetAllBooks();
    return View(listBooks);
}
```

```
public List<Books> GetAllBooks()
{
    List<Books> listBooks = new List<Books>()
    {
        new Books () { BookId = 1, Title = "Ngọc Trọng đá", Author = "Nguyễn Đông Thức" },
        new Books () { BookId = 2, Title = "Tôi thấy hoa vàng trên cỏ xanh", Author = "Nguyễn Nhật Ánh"},
        new Books () { BookId = 3, Title = "Yêu như là sống", Author = "Nguyễn Mạnh Tuấn"},
        new Books () { BookId = 4, Title = "Lão Hạc", Author = "Nam Cao" }
    };
    return listBooks;
}
```

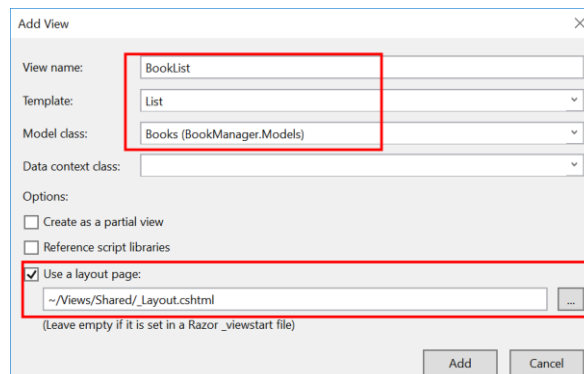
Lập trình web

27

27

Ví dụ sử dụng Controller (tt)

- Tạo View: click chuột phải trên tên Action BookList trong Controller
- Chọn như hình để tạo View tự động hiển thị dữ liệu nhận được



Add View

View name: BookList

Template: List

Model class: Books (BookManager.Models)

Data context class:

Options:

☐ Create as a partial view

☐ Reference script libraries

☒ Use a layout page: ~/Views/Shared/_Layout.cshtml

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

Add Cancel

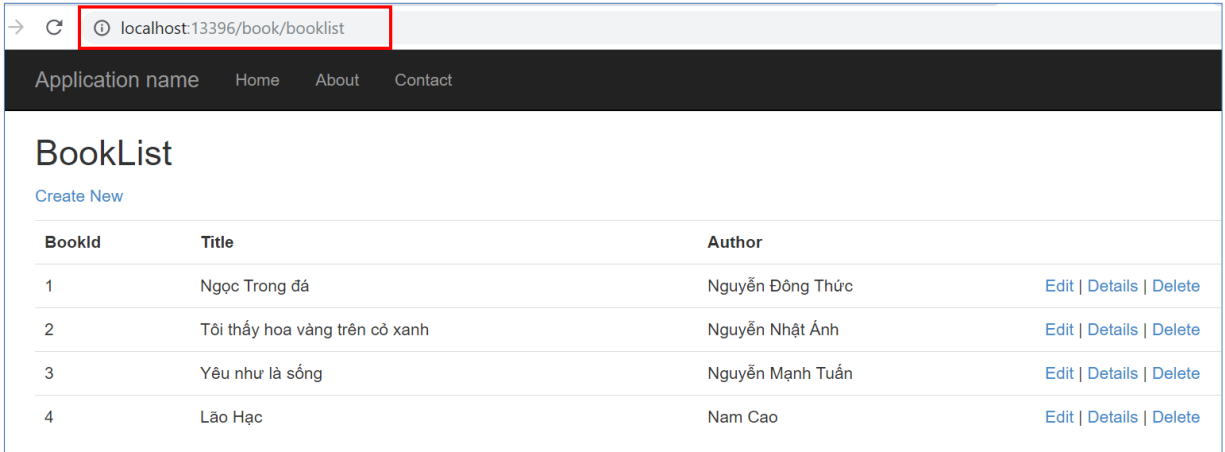
Lập trình web

28

28

Ví dụ sử dụng Controller (tt)

- Thực thi ứng dụng với URL .../Book/BookList



BookId	Title	Author	
1	Ngọc Trọng đá	Nguyễn Đồng Thức	Edit Details Delete
2	Tôi thấy hoa vàng trên cỏ xanh	Nguyễn Nhật Ánh	Edit Details Delete
3	Yêu như là sống	Nguyễn Mạnh Tuấn	Edit Details Delete
4	Lão Hạc	Nam Cao	Edit Details Delete

Lập trình web

29

29

3.2 View

- Giới thiệu View
- Razor
- Xử lý Form trong View
- View To Controller

Lập trình web

30

30

3.2.1 Giới thiệu View

- View là một thành phần trong mô hình MVC, có chức năng hiển thị giao diện người dùng
- View hiển thị dữ liệu đến từ Model
- View chứa mã lệnh HTML với các phần tử trình bày dữ liệu và nội dung trang web
- Trong ứng dụng APS.NET MVC, tập tin “.cshtml” sử dụng Razor view engine, cho phép kết hợp mã HTML với mã lệnh của ngôn ngữ lập trình (C#, VB.NET)
- Thông thường kiểu trả về cho View là “ActionResult” hoặc “ActionResult”

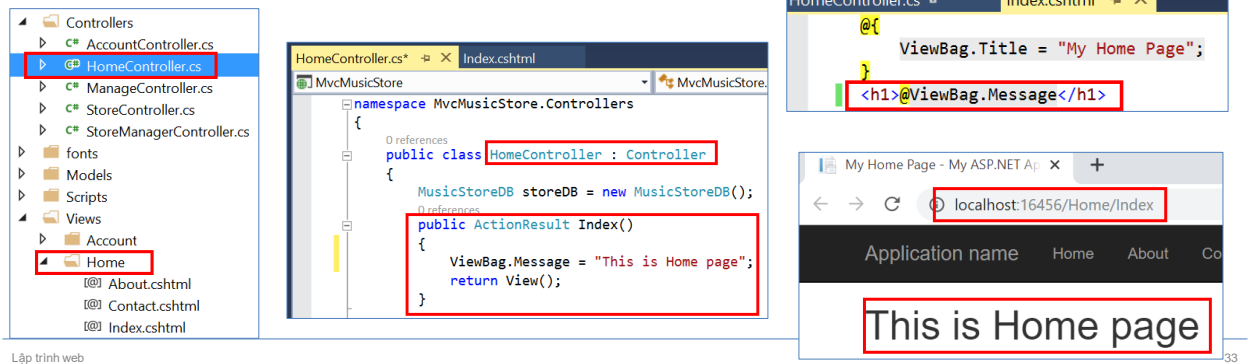
Giới thiệu View (tt)

- Thông thường, mỗi Controller tương ứng với một thư mục trong View.
- Mỗi action trong Controller tương ứng với tên một tập tin trong View
- Tương tác giữa View và Controller:
 - Truyền dữ liệu từ Controller sang View sử dụng đối tượng **ViewBag**
 - Truyền dữ liệu từ Controller sang View sử dụng đối tượng **ViewData**
 - Truyền dữ liệu từ Controller sang View thông qua **Model**

Giới thiệu View (tt)

• ViewBag:

- Trong Controller: gán biến/đối tượng cho đối tượng ViewBag: **ViewBag.Tenbien;**
- Trong View: truy xuất đối tượng ViewBag: **@ViewBag.Tenbien**
- Ví dụ truyền một chuỗi sang View:



The screenshot illustrates the use of ViewBag in an ASP.NET MVC application. On the left, the file explorer shows the project structure with 'Home' selected under 'Views'. The center pane shows the `HomeController.cs` file with the `Index()` action method. The right pane shows the `Index.cshtml` view file. Below these, a browser preview shows the rendered output.

```

// HomeController.cs
namespace MvcMusicStore.Controllers
{
    public class HomeController : Controller
    {
        MusicStoreDB storeDB = new MusicStoreDB();

        public ActionResult Index()
        {
            ViewBag.Message = "This is Home page";
            return View();
        }
    }
}

// Index.cshtml
@{
    ViewBag.Title = "My Home Page";
}
<h1>@ViewBag.Message</h1>

```

Browser Preview: localhost:16456/Home/Index
 Application name Home About Co
 This is Home page

33

Giới thiệu View (tt)

• ViewBag (tt):

- Ví dụ truyền một danh sách sang View:

```

public ActionResult List()
{
    var albums = new ArrayList();
    for (int i = 0; i < 10; i++)
    {
        albums.Add("Product " + i.ToString());
    }
    ViewBag.Albums = albums;
    return View();
}

```

```

@{
    ViewBag.Title = "List";
}
<h2>List</h2>
<ul>
    @foreach (string x in ViewBag.Albums)
    {
        <li>@x</li>
    }
</ul>

```

List

- Product 0
- Product 1
- Product 2
- Product 3
- Product 4
- Product 5
- Product 6
- Product 7
- Product 8
- Product 9

Lập trình web

34

34

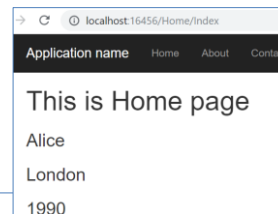
Giới thiệu View (tt)

• ViewData

- ViewData là một tập hợp mà mỗi phần tử là một đối tượng ViewDataDictionary, được truy xuất thông qua một trường dạng string gọi là key.
- Ví dụ:

```
public ActionResult Index()
{
    ViewBag.Message = "This is Home page";
    ViewData["Name"] = "Alice";
    ViewData["Address"] = "London";
    ViewData["Birthday"] = 1990;
    return View();
}
```

```
@{
    ViewBag.Title = "My Home Page";
}
<h1>@ViewBag.Message</h1>
<h3>@ViewData["Name"]</h3>
<h3>@ViewData["Address"]</h3>
<h3>@Convert.ToInt16(ViewData["Birthday"])</h3>
```



Lập trình web

35

35

Giới thiệu View (tt)

• Model

- Ví dụ:

```
public class Person
{
    2 references
    public string Name { get; set; }
    2 references
    public int Age { get; set; }
}
```

Model

```
public class HomeController : Controller
{
    // GET: Home
    0 references
    public ActionResult Index()
    {
        Person person = new Models.Person();
        person.Name = "Tom";
        person.Age = 20;
        return View(person);
    }
}
```

Controller

```
@model DataPassing.Models.Person
```

```
<h3>@Model.Name</h3>
<h3>@Model.Age</h3>
```

View

Lập trình web

36

36

3.2.2 Razor

- Razor là một View Engine được hỗ trợ trong APS.NET MVC, cho phép người phát triển kết hợp mã lệnh ngôn ngữ lập trình với mã HTML.
 - Mã C# hoặc VB.NET được đặt trong khối lệnh **@{...}**
- Đặc điểm của Razor:
 - Compact: cú pháp ngắn gọn
 - Easy to Learn: dễ học, có thể dùng các ngôn ngữ lập trình: C#, Visual Basic
 - Hỗ trợ Intellisense
 - Unit Testable: hỗ trợ khả năng unit test, không cần các controller hoặc web-server
- Lợi ích của Razor:
 - Web Grid
 - Web Graphics
 - Google Analytics
 - Facebook Integration
 - Twitter Integration
 - Sending Email
 - Validation

Razor (tt)

- Cú pháp Razor:
 - Kết thúc lệnh với dấu chấm phẩy (;)
 - Chuỗi đặt giữa dấu nháy kép ("")
 - Mã lệnh C# có phân biệt chữ hoa, chữ thường
 - Khai báo biến với từ khóa var hoặc khai báo kiểu dữ liệu (ASP.NET thường có thể tự động xác định kiểu dữ liệu)
 - Ví dụ:


```
var str = "My String"; // hoặc string str = "My String";
var counter = 100; // hoặc int counter = 100;
var today = DateTime.Today; // hoặc DateTime today = DateTime.Today;
```
 - Các kiểu dữ liệu: int, float, decimal, bool, string
 - Toán tử: như C#

Razor (tt)

– Chuyển đổi kiểu dữ liệu

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

Lập trình web

39

39

Razor (tt)

– Biểu thức nội tuyến bắt đầu với @:

<h3>Giá trị của number là @number </h3>

<h2>@DateTime.Now.ToShortDateString()</h2>

– Câu lệnh, khối lệnh Razor luôn đặt trong @{ ... }

```
@{ var number = 100; }
```

```
...
```

```
@{  
    var a = 10;  
    var b = 5;  
    var c = a + b;  
}
```

Lập trình web

40

40

Razor (tt)

– Chú thích trong Razor:

- Các dòng chú thích đặt trong `@*` và `*@`

```
@* A one-line code comment. *@
@*
    This is a multiline code comment.
    It can continue for any number of lines.
*@
```

- Nếu là code C# có thể chú thích dùng `//` hoặc `/*` và `*/`

```
@{
    // This is a comment.
    var myVar = 17;
    /* This is a multi-line comment that uses C#
    commenting syntax. */
}
```

- Nếu là mã HTML: dùng `<!-- This is a comment. -->`

Razor (tt)

– Kết hợp mã lệnh lập trình và mã HTML

```
@foreach (var item in items) {
    <span>Item @item.Name.</span>
}
```

– Kết hợp mã lệnh và plain text dùng thẻ `<text></text>` và `@:`

```
@if (showMessage) {
    <text>This is plain text</text>
}
```

hoặc

```
@if (showMessage) { @:This is plain text.}
```

– Hiển thị ký tự `@`: `@@`

The ASP.NET Twitter Handle is `@aspnet` → The ASP.NET Twitter Handle is @aspnet

Razor (tt)

- Cấu trúc điều kiện:

– if.. else: **@if** ...

```
<p>
  @if (DateTime.IsLeapYear(DateTime.Now.Year))
  {
    <text>@DateTime.Now.Year : is a leap year</text>
  }
  else
  {
    <text>@DateTime.Now.Year: is not a leap year</text>
  }
</p>
```

2021: is not a leap year

Razor (tt)

- Cấu trúc điều kiện:

– switch: **@switch** ...

```
@{
  var weekday = DateTime.Now.DayOfWeek;
  var day = weekday.ToString();
  var message = "";
}
<p>
@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>
```

Today is Sunday

Razor (tt)

• Cấu trúc lặp: for, while, do, foreach:

```
@for (var i = 1; i <= 5; i++)
{
    <p>Line @i</p>
}
```

Line 1
Line 2
Line 3
Line 4
Line 5

```
@{ var i = 1; sum = 0;}
@while (i <= 10)
{
    sum += i;
    i++;
}
```

```
int[] arrInt = { 7, 8, 4, 5, 3, 2 };
var sum = 0;
@foreach (var n in arrInt)
{
    sum += n;
    <span>Number is @n</span><br />
}
<h3>Tổng = @sum</h3>
```

```
@{ i = 1; sum = 0;}
@do
{
    sum += i;
    i++;
}while (i <= 10);
```

Razor (tt)

• Đối tượng Request, Response

- Đối tượng Response: chứa thông tin phản hồi từ server trả về cho trình duyệt.
 - Ví dụ: `@Response.ContentType`
- Đối tượng Request: chứa các thông tin về yêu cầu hiện tại, bao gồm loại trình duyệt, URL của trang, thông tin người dùng, phương thức GET/POST

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

- ALL_HTTP
- ALL_RAW
- APPL_MD_PATH
- APPL_PHYSICAL_PATH
- AUTH_TYPE
- AUTH_USER
- AUTH_PASSWORD
- LOGON_USER
- REMOTE_USER

Razor (tt)

- Ví dụ sử dụng đối tượng Request để truy xuất thông tin về trình duyệt, URL, đường dẫn vật lý,...

Requested URL	https://localhost:44345/Home/RequestDemo
Relative Path	/Home/RequestDemo
Full Path	D:\TaiLieu\Lap trinh Web\CodeDemo\RazorDemo\RazorDemo\Home\RequestDemo
HTTP Request Type	GET

- Lưu ý: có thể sử dụng **Server.MapPath** để chuyển đường dẫn thư mục ảo về đường dẫn vật lý

```
@{ var dataFilePath = "~/dataFile.txt"; }
<!-- Displays a physical path C:\Websites\MyWebSite\datafile.txt -->
<p>@Server.MapPath(dataFilePath)</p>
```

Lập trình web

```
<table class="table-condensed table-bordered">
```

```
<tr>
  <th>Requested URL</th>
  <td>@Request.Url</td>
```

```
</tr>
<tr>
  <th>Relative Path</th>
  <td>@Request.FilePath</td>
```

```
</tr>
<tr>
  <th>Full Path</th>
  <td>@Request.MapPath(Request.FilePath)</td>
```

```
</tr>
<tr>
  <th>HTTP Request Type</th>
  <td>@Request.RequestType</td>
```

```
</tr>
</table>
```

47

47

Razor (tt)

- Tập hợp:
 - Array: sử dụng cú pháp C#
 - Dictionary

```
@{
  var myScores = new Dictionary<string, int>();
  myScores.Add("test1", 71);
  myScores.Add("test2", 82);
  myScores.Add("test3", 100);
  myScores.Add("test4", 59);
}
<p>My score on test 3 is: @myScores["test3"]%</p>
@{ myScores["test4"] = 79; }
<p>My corrected score on test 4 is: @myScores["test4"]%</p>
```

My score on test 3 is:
100%

My corrected score on test
4 is: 79%

Lập trình web

48

48

3.2.3 Xử lý Form trong View

- **Thuộc tính IsPost:** được dùng để kiểm tra trạng thái postback của trang web
 - IsPost = true: trang web được post back (người dùng submit trang)
 - IsPost = false: trang web được nạp lần đầu tiên

```
@{
    var result = "";
    if(IsPost)
    {
        result = "This page was posted using the Submit button.";
    }
    else
    {
        result = "This was the first request for this page.";
    }
}
```

```
<form method="POST" action="" >
    <input type="Submit" name="Submit" value="Submit"/>
    <p>@result</p>
</form>
```

Xử lý Form trong View (tt)

- **Đọc dữ liệu từ form: Request ["fieldname"]**

```
<form class="form-horizontal" action="" method="post">
    <div class="form-group">
        <label for="txtNumber1">Số thứ 1:</label><br>
        <input type="text" name="txtNumber1">
    </div>
    <div class="form-group">
        <label for="txtNumber2">Số thứ 2:</label><br>
        <input type="text" name="txtNumber2">
    </div>
    <div class="form-group">
        <input type="submit" value="Cộng">
    </div>
    <div class="form-group">
        <label>Total: @result</label>
    </div>
</form>
```

```
@{
    var result = 0;
    if (IsPost)
    {
        var a = Request["txtNumber1"];
        var b = Request["txtNumber2"];
        result = a.AsInt() + b.AsInt();
    }
}
```

3.2.4 View To Controller

- Sử dụng đối tượng của lớp `HttpRequestBase`

```
string strName = Request["txtName"].ToString();
(formfield có thuộc tính name ="txtName")
```

- Sử dụng `FormCollection`:

```
[HttpPost]
public ActionResult Calculate(FormCollection form) {
    string strName = form["txtName"].ToString();
}
```

View To Controller

- Sử dụng tham số: **View:**

```
<form action="/Form/Register" method="post">
    <input type="text" name=" txtUserName" value="" />
    <input type="number" name=" txtAge" value="" />
    <input type="submit" value="Send" />
</form>
```

Controller:

```
[HttpPost]
public ActionResult Register (string txtUserName, int txtAge)
{
}
}
```

View To Controller

• Strongly Typed model binding to view

Controller: `CalculateController`

```
public ActionResult Index(Expressions exp)
{
    return View(exp);
}
[HttpPost]
public ActionResult Calculate(Expressions exp)
{
    switch (exp.Operator)
    {
        case "+": exp.Result = exp.Number1 + exp.Number2; break;
        case "-": exp.Result = exp.Number1 - exp.Number2; break;
        case "x": exp.Result = exp.Number1 * exp.Number2; break;
        case "/":
            if (exp.Number2 != 0)
                exp.Result = exp.Number1 / exp.Number2;
            break;
    }
    return RedirectToAction("Index", exp);
}
```

Model:

```
public class Expressions
{
    public double Number1 { get; set; }
    public double Number2 { get; set; }
    public string Operator { get; set; }
    public double Result { get; set; }
}
```

Lập trình web

53

53

View To Controller

• Strongly Typed model binding to view (tt)

View:

```
@model ViewToControllerByModel.Models.Expressions
@using (Html.BeginForm("Calculate", "Calculate", FormMethod.Post))
{
    <div class="form-group"> ActionName ControllerName
        @Html.TextBoxFor(model => model.Number1, new { @class = "form-control" })
    </div>
    <div class="form-group">
        @Html.DropDownListFor(model => model.Operator, new[] {
            new SelectListItem { Text = "+", Value = "+" },
            new SelectListItem { Text = "-", Value = "-" },
            new SelectListItem { Text = "x", Value = "x" },
            new SelectListItem { Text = "/", Value = "/" },
        },
        new { @class = "form-control" })
    </div>
```

Lập trình web

54

54

View To Controller

- Strongly Typed model binding to view (tt)

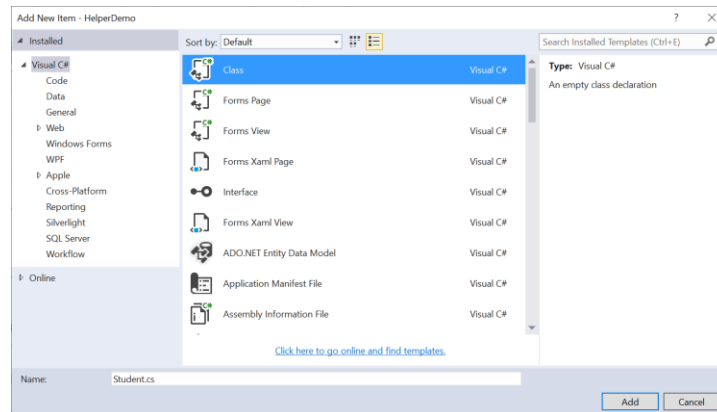
```
<div class="form-group">
    @Html.TextBoxFor(model => model.Number2, new { @class = "form-control" })
</div>
<div class="form-group">
    Result:
</div>
<div class="form-group">
    @Html.TextBoxFor(model => model.Result, null, new { @class = "form-control",
                                                         @readonly="readonly" })
</div>
<div class="form-group">
    <button type="submit" class="btn btn-primary">Calculate</button>
</div>
}
```

3.3. Model

- Model là một thành phần trong mô hình MVC, chịu trách nhiệm lưu trữ dữ liệu của ứng dụng
- Model là tập hợp các lớp mô tả dữ liệu và các quy tắc, thao tác dữ liệu
- Dữ liệu từ Model sẽ được truyền sang View để hiển thị trên giao diện người dùng
- Các lớp model được đặt trong thư mục Models của ứng dụng

Model (tt)

- Tạo một Model:
 - Click chuột phải trên thư mục Model → Add → Class..



Lập trình web

57

57

Model (tt)

- Tạo một Model (tt):
 - Khai báo các trường dữ liệu cần thiết:

```
public class Student
{
    0 references
    public int StudentId { get; set; }
    [Display(Name = "Name")]
    0 references
    public string StudentName { get; set; }
    0 references
    public int Age { get; set; }
    0 references
    public bool isNewlyEnrolled { get; set; }
    0 references
    public string Password { get; set; }
    0 references
    public DateTime DoB { get; set; }
}
```

Lập trình web

58

58

Model (tt)

- Các cách làm việc với cơ sở dữ liệu trong Model:
 - **EF CodeFirst**: viết code trước để sau đó phát sinh cơ sở dữ liệu (database), sử dụng Entity Framework (nếu chưa có phải cài)
 - Cấu hình chuỗi kết nối (nếu cần)
 - Tạo các lớp biểu diễn dữ liệu trong Model
 - **EF Database First**: tạo mô hình mã nguồn dựa vào database đã có sẵn
 - Sử dụng đối tượng ADO.NET Entity Data Model
 - **EF Model First**: xây dựng mô hình trên tập tin sơ đồ trước, sau đó sinh mã tự động
 - Thường dùng trong các ứng dụng có quy mô lớn
 - Sử dụng Entity Framework Designer, tạo lược đồ cơ sở dữ liệu từ mô hình, lưu trữ ở tập tin EDMX (.edmx)
 - Có thể xem, chỉnh sửa ở Entity Framework Designer