

# Views trong ASP.NET MVC

- Views là gì?
- Razor engine

1

1

## Làm việc với Views

- Để hiển thị nội dung HTML, ta có thể chỉ dẫn hành xử controller để đưa ra view
- View
  - Cung cấp giao diện người dùng
  - Hiển thị nội dung của ứng dụng và nhận đầu vào của người dùng
  - Sử dụng mô hình dữ liệu để tạo UI
  - Chứa cả HTML và code khi chạy trên Web server

2

2

# View Engines

- Chuyển code trong view sang HTML để các trình duyệt có thể đọc được
- Có hai loại
  - Web Form view engine:
  - Zaror view engine

3

3

## Xác định view cho action

- Mọi view được lưu mặc định trong thư mục view
- Nếu một action controller trả về một view, khi đó phần view có:
  - Một thư mục cho controller cùng tên với controller nhưng không có hậu tố Controller
  - Một file trong thư mục Home có cùng tên với action

4

4

## Xác định view cho action

```
public class HomeController : Controller{  
    public ActionResult Index(){  
        return View();  
    }  
}
```

- Action Index trả về một view
- Tên controller: HomeController
- Demo cách tạo một view

5

5

## Xác định view cho action

- Có thể trả về một view khác trong phương thức action bằng cách đặt tên của view là một thông số

```
public class HomeController: Controller{  
    public ActionResult Index(){  
        return View("TestIndex");  
    }  
}
```

- Đoạn code trên sẽ tìm view trong thư mục Views/Home và sinh ra Testdext thay vì Index

6

6

## Xác định view cho action

- Render một view trong thư mục khác: cần xác định đường dẫn cho view

```
public class HomeController : Controller{  
    public ActionResult Index()  
    {  
        return  
        View("~/Views/Demo/Welcome.cshtml");  
    }  
}
```

7

7

## Chuyển dữ liệu từ controller sang view

- Controller thực hiện chức năng, trả kết quả người dùng thông qua view
- Chuyển dữ liệu giữa controller và view qua:
  - ViewData
  - ViewBag
  - TempData

8

8

## Chuyển dữ liệu từ controller sang view

- ViewData
  - Chuyển dữ liệu từ controller đến view
  - Là thư viện gồm các đối tượng của lớp ViewDataDictionary
  - Các tính chất của ViewData:
    - Đối tượng ViewData chỉ tồn tại khi được gọi và là null khi chuyển hướng hay thay đổi URL (redirected)
    - ViewData yêu cầu mẫu khi sử dụng dữ liệu phức tạp để tránh lỗi
  - Cú pháp: `ViewData[<key>] = <Value>;`

9

9

## Chuyển dữ liệu từ controller sang view

- VD: ViewData kết nối HomeController và Index

```
public class HomeController : Controller {
    public ActionResult Index() {
        ViewData["Message"] = "Message from
                                ViewData";
        ViewData["CurrentTime"] =
            DateTime.Now; return View();
    }
}
```
- ViewData được tạo với 2 cặp giá trị
  - Message là chuỗi
  - CurrentTime chứa dữ liệu DateTime.Now

10

10

## Chuyển dữ liệu từ controller sang view

- Đoạn code sau lấy các giá trị của ViewData

```
<html>
<head>
  <title>Index View</title>
</head>
<body>
  <p> @ViewData["Message"] </p>
  <p> @ViewData["CurrentTime"] </p>
</body>
</html>
```

- ViewData trong đoạn code trên: Hiển thị giá trị của Message và CurrentTime

11

11

## Chuyển dữ liệu từ controller sang view

- ViewBag
  - Bao phủ ViewData
  - Chỉ tồn tại cho yêu cầu hiện tại và có giá trị null khi yêu cầu được chuyển hướng
  - Không yêu cầu mẫu khi sử dụng dữ liệu phức tạp
  - Cú pháp: ViewBag.<Property> = <Value>;
    - Property: chuỗi giá trị thể hiện thuộc tính của ViewBag
    - Value: giá trị của thuộc tính ViewBag

12

12

## Chuyển dữ liệu từ controller sang view

- VD: Đoạn code sau cho thấy 2 thuộc tính trong Index của lớp HomeController

```
public class HomeController:Controller{
    public ActionResult Index() {
        ViewBag.Message = "Message from
                               ViewBag";
        ViewBag.CurrentTime = DateTime.Now;
        return View();
    }
}
```

13

13

## Chuyển dữ liệu từ controller sang view

- Đoạn code hiển thị giá trị biểu diễn trong ViewBag

```
<html>
  <head>
    <title>Index View</title>
  </head>
  <body>
    <p> @ViewBag.Message</p>
    <p> @ViewBag.CurrentTime</p>
  </body>
</html>
```

- Khi sử dụng ViewBag để lưu thuộc tính và giá trị trong action thì thuộc tính đó có thể được truy cập từ cả hai: ViewBag và ViewData

14

14

## Chuyển dữ liệu từ controller sang view

- Đoạn code sau cho thấy một controller action lưu thuộc tính ViewBag

```
public class HomeController: Controller{  
    public ActionResult Index(){  
        ViewBag.CommonMessage = "Common  
                                message accessible to both  
                                ViewBag and ViewData";  
        return View();  
    }  
}
```

15

15

## Chuyển dữ liệu từ controller sang view

- Đoạn code sau cho thấy cả ViewData và ViewBag đều truy cập đến thuộc tính CommonMessage trong ViewBag

```
<html>  
    <head><title>Index View</title></head>  
    <body>  
        <p><em>Accessed from ViewData:</em>  
        @ViewData["CommonMessage"]</p>  
        <p><em>Accessed from ViewBag:</em>  
        @ViewBag.CommonMessage</p>  
    </body>  
</html>
```

16

16



## Chuyển dữ liệu từ controller sang view

- TempData
  - Là đối tượng Dictionary của lớp TempDataDictionary
  - Lưu trữ dữ liệu dạng key-value
  - Cho phép chuyển dữ liệu từ yêu cầu hiện tại đến yêu cầu sau khi chuyển hướng
  - Cú pháp: TempData[<Key>] = <Value>;
    - Key; chuỗi nhận dạng đối tượng hiện tại trong TempData
    - Value: là đối tượng biểu diễn trong TempData

17

17

## Xác định view cho action

- Đoạn code sau cho thấy sử dụng TempData để chuyển giá trị từ view này đến view khác khi chuyển hướng

```
public class HomeController:Controller{
    public ActionResult Index(){
        ViewData["Message"] = "ViewData Message";
        ViewBag.Message = "ViewBag Message";
        TempData["Message"] = "TempData Message";
        return Redirect("Home/About");
    }
    public ActionResult About() {
        return View();
    }
}
```

18

18

## View bộ phận (Partial Views)

- Partial View:
  - View con của view chính
  - Cho phép dùng lại các thẻ của view khác
- Tạo Partial View
  - Kích chuột phải thư mục Views/Shared, Add/View
  - Đánh dấu hộp kiểm: Create as a partial view

19

19

## Partial Views

- Cú pháp: @Html.Partial(<partial\_view\_name>
  - partial\_view\_name: tên view cục bộ không kèm phần mở rộng .cshtml

```
<html>
  <head><title>Index View</title></head>
  <body>
    <h1> Welcome to the Website</h1>
    <div>@Html.Partial("_TestPartialView")</div>
  </body>
</html>
```

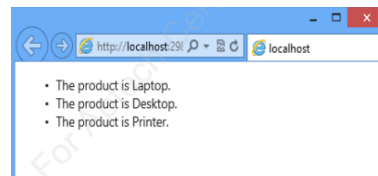
20

20

# Razor

- Dựa trên ASP.NET tạo ra view
- Dễ hiểu cho người lập trình: ngôn ngữ lập trình tương tự C#.NET và VB.NET

```
@{var products = new string[] { "Laptop",  
"Desktop", "Printer"};}  
<html>  
<head><title>Test View</title></head>  
<body>  
    <ul>  
        @foreach (var product in  
            products){  
            <li>The product is  
                @product.</li>  
        }  
    </ul>  
</body>  
</html>
```



21

21

# Razor Engine

- Là engine sinh view mặc định của MVC
- Biên dịch view cho lần gọi đầu
- Cung cấp view được biên dịch cho lần gọi sau
- Không cung cấp ngôn ngữ lập trình mới mà đưa ra cú pháp tích hợp HTML
- Hỗ trợ Test Driven Development (TDD) cho phép kiểm tra view độc lập với ứng dụng

22

22

# Cú pháp Razor

- @{ <code> }
  - Code: cú pháp c# hoặc vb# được thực hiện trên server
- Biến được khai báo với từ khóa var
  - @{ var myMessage = "Hello World"; }
  - @{  
    var myMessage = "Hello World";  
    var num = 10;  
}
- Chuỗi để trong ngoặc kép
- Dòng lệnh kết thúc bởi (;)
- Dùng .cshtml để lưu kiểu file c#

23

23

# Biến - Cấu trúc lệnh

- Khai báo biến, sử dụng cấu trúc lệnh tương tự như trong C#

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

24

24

```

<!DOCTYPE html>
<html><body>
    @{ var b = 0;
        while (b < 7) {
            b += 1;
            <p>Text @b</p>}
        }
</body></html>

```

```

<!DOCTYPE html>
@{var mark=60;}
<html><body>
    @if (mark<80) {
        <p>You have failed in
            the exam.</p>}
    else {
        <p>You have passed
            the exam.</p> }
</body></html>

```

```

<!DOCTYPE html>
<html><body>
    <h1>Even Numbers</h1>
    @{ var num=1;
        for (num = 1; num <= 11; num++){
            if ((num % 2) == 0)
                { <p> @num</p> }
            } }
</body></html>

```

25

25

```

<!DOCTYPE html>
@{
    var day=DateTime.Now.DayOfWeek.ToString();
    var msg=""; }
<html> <body>
    @switch(day) {
        case "Monday":
            msg="Today is Monday, the first working day.";
            break;
        case "Friday":
            msg="Today is Friday, the last working day.";
            break;
        default:
            msg="Today is " + day;
            break; }
    <p>@msg</p>
</body> </html>

```

26

26

```

<!DOCTYPE html>
@{
    string[] members = {"Joe", "Mark", "Stella"};
    int i = Array.IndexOf(members, "Stella")+1;
    int len = members.Length;
    string x = members[2-1]; }
<html> <body>
    <h3>Student Details</h3>
    @foreach (var person in members) {
        <span>@person &nbsp;</span> }
        <p>The number of students in class are
            @len</p>
        <p>The student at position 2 is @x</p>
        <p>Stella is now in position @i</p>
    </body></html>

```

27

27

## HTML Helper Methods

- Là phương thức của lớp HtmlHelper, chỉ được gọi từ view
- Đơn giản hóa thi hành view
- Cho phép sinh HTML và tái sử dụng
- Một số phương thức thường dùng:
  - Html.ActionLink()
  - Html.BeginForm() và Html.EndForm()
  - Html.Label()
  - Html.TextBox()
  - Html.TextArea()
  - Html.Password()
  - Html.CheckBox()

28

28

## HTML Helper Methods

- `Html.ActionLink()`: cho phép sinh ra một hyperlink dựa trên action method của controller class

```
@Html.ActionLink(<link_text>,<action_method>
,<optional_controller>)
```

- VD:

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

29

29

## HTML Helper Methods

- `Html.BeginForm()`
  - Cho phép đánh dấu điểm bắt đầu của form
  - Phối hợp với routing engine để sinh URL
  - Quản lý thẻ `<form>`

- Cú pháp:

```
@{Html.BeginForm(<action_method>,<controller_name>);}
```

- Khi sử dụng `HTML.BeginForm`, cần sử dụng `HTML.EndForm`

30

30

# HTML Helper Methods

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

31

31

# HTML Helper Methods

- `Html.Label()`
  - Cho phép hiển thị nhãn trong form
  - Cho phép kết nối đến các thẻ khác
  - Cú pháp: `@Html.Label(<label_text_name>)`

- VD:

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

32

32



# HTML Helper Methods

- **Html.TextBox():**

- Cho phép hiển thị thẻ đầu vào
- Sử dụng chấp nhận đầu vào của người dùng
- Cú pháp:

```
@Html.TextBox("textbox_text_name")
```

- VD:

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

33

33

# HTML Helper Methods

- **Html.TextArea()**

- Cho phép hiển thị thẻ <textarea>
- Cho phép xác định số cột, hàng hiển thị

- VD:

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

34

34

## HTML Helper Methods

- **Html.Password():** hiển thị trường mật khẩu

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

35

35

## HTML Helper Methods

- **Html.CheckBox():** hiển thị checkbox

```
<!DOCTYPE html>
<html><body>
    @{Html.BeginForm("Browse", "Home");}
    @Html.Label("User Name:")</br>
    @Html.TextBox("textBox1")</br></br>
    @Html.Label("Address:")</br>
    @Html.TextArea("textareal")</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();}
</body> </html>
```

36

36

# HTML Helper Methods

- `Html.DropDownList()`: sinh thẻ `<select>`
- **Cú pháp:**  
`@Html.DropDownList("myList",  
new SelectList(new []  
{<value1>,<value2>, <  
value3>}), "Choose")`
  - Value1, value2, value3: các giá trị danh sách
  - Choose: giá trị đầu danh sách

37

37

# HTML Helper Methods

- **VD:**

```
<!DOCTYPE html>  
<html><body>  @{Html.BeginForm("Browse", "Home");}  
    @Html.Label("User Name:")</br>  
    @Html.TextBox("textBox1")</br></br>  
    @Html.Label("Address:")</br>  
    @Html.TextArea("textareal")</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();}  
</body></html>
```

38

38

## HTML Helper Methods

- **Html.RadioButton():** tạo nút lựa chọn  
`@Html.RadioButton("name",  
"value", isChecked)`
- **Url.Action():** sinh url  
`@Url.Action(<action_name>,  
<controller_name>)`