

Xác thực và chú thích dữ liệu

- ◆ Định nghĩa và mô tả xác thực dữ liệu
- ◆ Giải thích cách sử dụng chú thích dữ liệu
- ◆ Giải thích và mô tả cách sử dụng ModelState

- ◆ Trong ứng dụng ASP.NET MVC, người dùng tương tác với ứng dụng bằng nhiều cách
 - Nhập 1 URL trên trình duyệt
 - Bấm vào một link trên ứng dụng
 - Điền vào form cung cấp bởi ứng dụng

Ở trong các trường hợp kể trên, một lập trình viên cần chắc chắn rằng bất kì dữ liệu nào được gửi bởi người dùng tới ứng dụng đều phải hợp lệ. Ta có thể xác thực dữ liệu người dùng bằng cách bao gồm logic xác thực trong ứng dụng. Logic xác thực đầu tiên sẽ phân tích xem dữ liệu người dùng có hợp lệ và có như mong đợi của ứng dụng hay không. Tiếp theo, logic xác thực sẽ cung cấp phản hồi cho người dùng để người dùng có thể xác nhận và sửa những input không hợp lệ trước khi họ gửi lại dữ liệu.

Luồng xác thực dữ liệu

- ◆ MVC Framework triển khai một luồng xác thực để xác thực dữ liệu người dùng
- ◆ Luồng xác thực này bắt đầu khi dữ liệu người dùng đã xác nhận đến được server
- ◆ Quy trình xác thực:
 - Quy trình xác thực này đầu tiên kiểm tra xem yêu cầu có ở dạng tấn công hay không, ví dụ như Cross Site Scripting (CSS).
 - Nếu quy trình xác nhận yêu cầu tấn công, nó ném ra một ngoại lệ
 - Còn không, quy trình kiểm tra dữ liệu với các yêu cầu xác thực của ứng dụng.
 - Nếu tất cả dữ liệu đều hợp lệ, quy trình sẽ chuyển tiếp yêu cầu tới ứng dụng để thực hiện các xử lý tiếp theo.
 - Nếu một hoặc nhiều dữ liệu không hợp lệ, quy trình gửi lại một thông báo xác thực lỗi dưới dạng một phản hồi.

- ◆ Trong ứng dụng ASP.NET MVC, ta có thể xác thực dữ liệu một cách thủ công trong hành động controller tiếp nhận dữ liệu người dùng cho một số loại xử lý.
- ◆ Để xác thực dữ liệu thủ công, ta có thể triển khai các quy trình xác thực đơn giản, ví dụ như quy trình kiểm tra độ dài của mật khẩu yêu cầu có hơn 7 kí tự.

- ◆ Code snippet sau biểu diễn xác thực thủ công triển khai trong một phương thức hành động:

Code Snippet:

```
public class HomeController : Controller    {
    Public ActionResult Index() {
        return View();
    }
    [HttpPost]
    Public ActionResult Index(User model) {
        String modelPassword = model.password;
        if (modelPassword.Length< 7)                {
            return View();
        }
        else {
            /*Implement registration process*/
            return Content("You have successfully registered");
        }
    }
}
```

- ◆ Trong đoạn code trên:
 - ◆ Tạo ra một class controller **HomeController** với 2 phương thức **Index()**.
 - ◆ Phương thức **Index()** đầu tiên trả về view tương ứng
 - ◆ Khi người dùng xác nhận một form hiển thị bởi view, phiên bản **HttpPost** của phương thức hành động nhận về một đối tượng model **User** đại diện cho dữ liệu được người dùng gửi.
 - ◆ Phương thức này xác thực xem mật khẩu có nhiều hơn 7 kí tự hay không
 - ◆ Nếu không, phương thức trả về một view. Trường hợp còn lại, phương thức trả về một tin nhắn hợp lệ.
 - ◆ Ta có thể sử dụng một biểu thức chính quy khi thực hiện xác thực thủ công
 - ◆ Ví dụ, ta có thể sử dụng một biểu thức chính quy để kiểm tra xem email người dùng gửi có ở đúng định dạng hay không, ví dụ *abc@abc.com*

- ◆ Xác thực thủ công ở trong một phương thức hành động có sử dụng biểu thức chính quy:

Code Snippet:

```
public class HomeController : Controller {
    public ActionResult Index() {
        return View();
    }
    [HttpPost]
    public ActionResult Index(User model) {
        string modelEmailId = model.email;
        string regexPattern=@"[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}";
        if (System.Text.RegularExpressions.Regex.IsMatch(modelEmailId,
            regexPattern)) {
            /*Implement registration process*/
            return Content("You have successfully registered"); }
        else {
            return View(); }
    }
}
```


- ◆ Trong đoạn code trên:
 - ◆ Sử dụng biến ***regexPattern*** lưu biểu thức chính quy để xác nhận một mẫu email hợp lệ..
 - ◆ Sau đó, phương thức này sử dụng phương thức ***IsMatch()*** của class ***System.Text.RegularExpressions.Regex*** để xác thực xem thuộc tính email có ở mẫu hợp lệ hay không. Nếu hợp lệ, phương thức hành động trả về view, còn không thì phương thức trả về một thông điệp hợp lệ.

- ◆ MVC Framework cung cấp các chú thích dữ liệu để bạn có thể áp dụng như thuộc tính của một model.
- ◆ Các chú thích dữ liệu này triển khai các tác vụ được yêu cầu trên ứng dụng.
- ◆ Một số chú thích quan trọng được sử dụng trong model của ứng dụng ASP.NET MVC:
 - Required
 - StringLength
 - RegularExpression
 - Range
 - Compare
 - DisplayName
 - ReadOnly
 - DataType
 - ScaffoldColumn

- ◆ Chú thích dữ liệu **Required** xác định thuộc tính mà chú thích này được liên kết cùng. Điều này có nghĩa giá trị của thuộc tính không được để trống. Thuộc tính này sẽ dẫn tới một lỗi xác thực nếu giá trị của thuộc tính là null hoặc bị để trống
- ◆ Cú pháp:

Syntax:

```
[Required]  
public string <property_name>;
```

Trong đó,

- ▮ `property_name`: tên của thuộc tính model.

- ◆ Đoạn code sau biểu thị chú thích ***Required*** trong các thuộc tính của model ***User***:

Code Snippet:

```
public class User {  
    public long Id { get; set; }  
    [Required]  
    public string Name { get; set; }  
    [Required]  
    public string Password { get; set; }  
    [Required]  
    public string ReenterPassword { get; set; }  
    [Required]  
    public int Age { get; set; }  
    [Required]  
    public string Email { get; set; } }  
}
```

- ◆ Trong đoạn code trên, ***Required*** được áp dụng cho các thuộc tính ***Name***, ***Password***, ***ReenterPassword***, ***Age***, và ***Email*** của model ***User***.

- ◆ Khi sử dụng **Required** trong các thuộc tính của model, đầu tiên ta cần sử dụng phương thức hỗ trợ **Html.ValidationSummary()** để truyền giá trị 'true' như một tham số. Phương thức này được sử dụng để hiển thị tin nhắn xác thực trên trang. Sau đó, với mỗi trường, ta cần sử dụng **Html.ValidationSummary()** truyền biểu thức lambda để kết nối phương thức với một thuộc tính model. Phương thức này trả về một tin nhắn lỗi xác thực cho thuộc tính được kết nối dưới dạng HTML

- ◆ Đoạn code sau là ví dụ về việc sử dụng các phương thức hỗ trợ để hiển thị các tin nhắn xác thực:

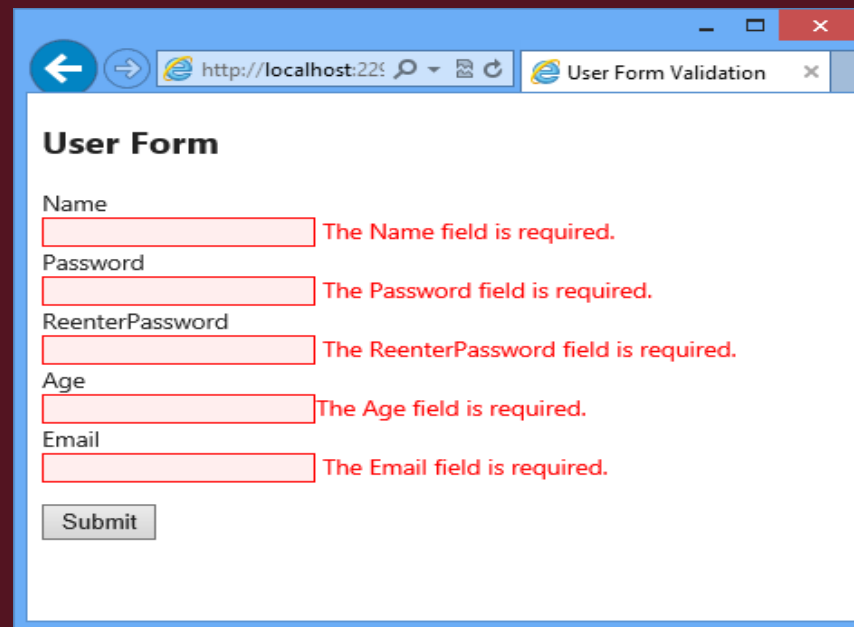
Code Snippet:

```
@model MVCModelDemo.Models.User @{
    ViewBag.Title = "User Form Validation";
}
<h2>User Form</h2>
@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)
    <div>
        @Html.LabelFor(model =>model.Name)
    </div>
    <div>
        @Html.EditorFor(model =>model.Name)
        @Html.ValidationMessageFor(model =>model.Name)
    </div>
    <div>
        @Html.LabelFor(model =>model.Password)
    </div>
    <div>
        @Html.EditorFor(model =>model.Password)
        @Html.ValidationMessageFor(model =>model.Password)
    </div>
}
```

Code Snippet:

```
<div>
@Html.LabelFor(model =>model.ReenterPassword)
</div>
<div>
@Html.EditorFor(model =>model.ReenterPassword)
@Html.ValidationMessageFor(model =>model.ReenterPassword)
</div>
<div>
@Html.LabelFor(model =>model.Age)
</div>
@Html.EditorFor(model =>model.Age)
@Html.ValidationMessageFor(model =>model.Age)
<div>
@Html.LabelFor(model =>model.Email)
</div>
<div>
@Html.EditorFor(model =>model.Email)
@Html.ValidationMessageFor(model =>model.Email)
</div>
<p>
<input type="submit" value="Submit" />
</p>
}
```

- ◆ Trong đoạn code trên:
 - ▮ ***Html.ValidationSummary(true)*** hiển thị tin nhắn xác thực dưới dạng danh sách..
 - ▮ Sau đó, với mỗi trường UI, ***Html.ValidationMessageFor()*** được sử dụng để xác thực các thuộc tính tương ứng của model.
- ◆ Khi sử dụng chú thích ***Required*** và người dùng xác nhận một form mà không có giá trị cho thuộc tính, MVC Framework hiển thị một tin nhắn lỗi xác thực mặc định:



The screenshot shows a web browser window with the address bar displaying 'http://localhost:22...' and the page title 'User Form Validation'. The main content area is titled 'User Form' and contains several input fields, each with a red border and a red error message to its right:

- Name: The Name field is required.
- Password: The Password field is required.
- ReenterPassword: The ReenterPassword field is required.
- Age: The Age field is required.
- Email: The Email field is required.

At the bottom of the form is a 'Submit' button.

- ◆ Ta cũng có thể tự chỉ định một tin nhắn xác thực cho chú thích ***Required***. Cú pháp như sau:

Syntax:

```
[Required(ErrorMessage = <error-message>)]
```

Trong đó,

- ▮ `error-message`: tin nhắn muốn hiển thị.

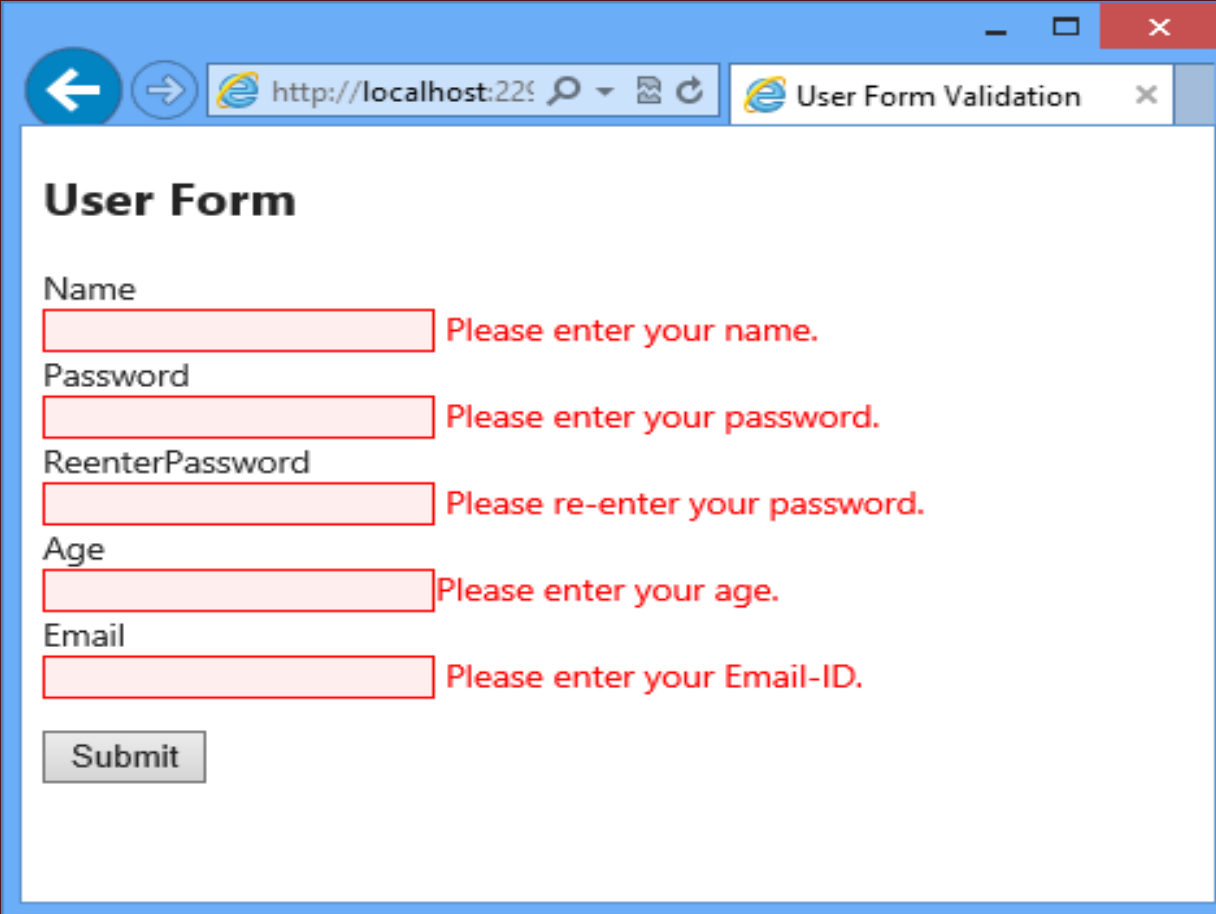
Required Annotation 8-9

- ◆ Ví dụ trên được biểu diễn ở code snippet sau:

Code Snippet:

```
public class User    {
    public long Id { get; set; }
    [Required(ErrorMessage = "Please enter your name.")]
    public string Name { get; set; }
    [Required(ErrorMessage = "Please enter your password.")]
    public string Password { get; set; }
    [Required(ErrorMessage = "Please re-enter your password.")]
    public string ReenterPassword { get; set; }
    [Required (ErrorMessage = "Please enter your age.")]
    public int Age { get; set; }
    [Required(ErrorMessage = "Please enter your Email-ID.")]
    public string Email { get; set; }
}
```

- ◆ Kết quả của đoạn code trên:



The screenshot shows a web browser window with the title 'User Form Validation'. The address bar displays 'http://localhost:22...'. The page content is titled 'User Form' and contains five input fields, each with a red border and a red error message to its right:

- Name: Please enter your name.
- Password: Please enter your password.
- ReenterPassword: Please re-enter your password.
- Age: Please enter your age.
- Email: Please enter your Email-ID.

At the bottom of the form is a 'Submit' button.

- ◆ Sử dụng chú thích ***StringLength*** để xác định độ dài ngắn nhất và dài nhất của một trường string.
- ◆ Cú pháp :

Syntax:

```
[StringLength(<max_length>, MinimumLength= <min_length>)]
```

Trong đó,

- ▮ `max_length`: Giá trị int chỉ định độ dài lớn nhất cho phép
- ▮ `min_length`: Giá trị int chỉ định độ dài nhỏ nhất cho phép

- ◆ Code snippet 6 biểu thị một model **User** với chú thích **StringLength** áp dụng cho các thuộc tính **Password** và **ReenterPassword**:

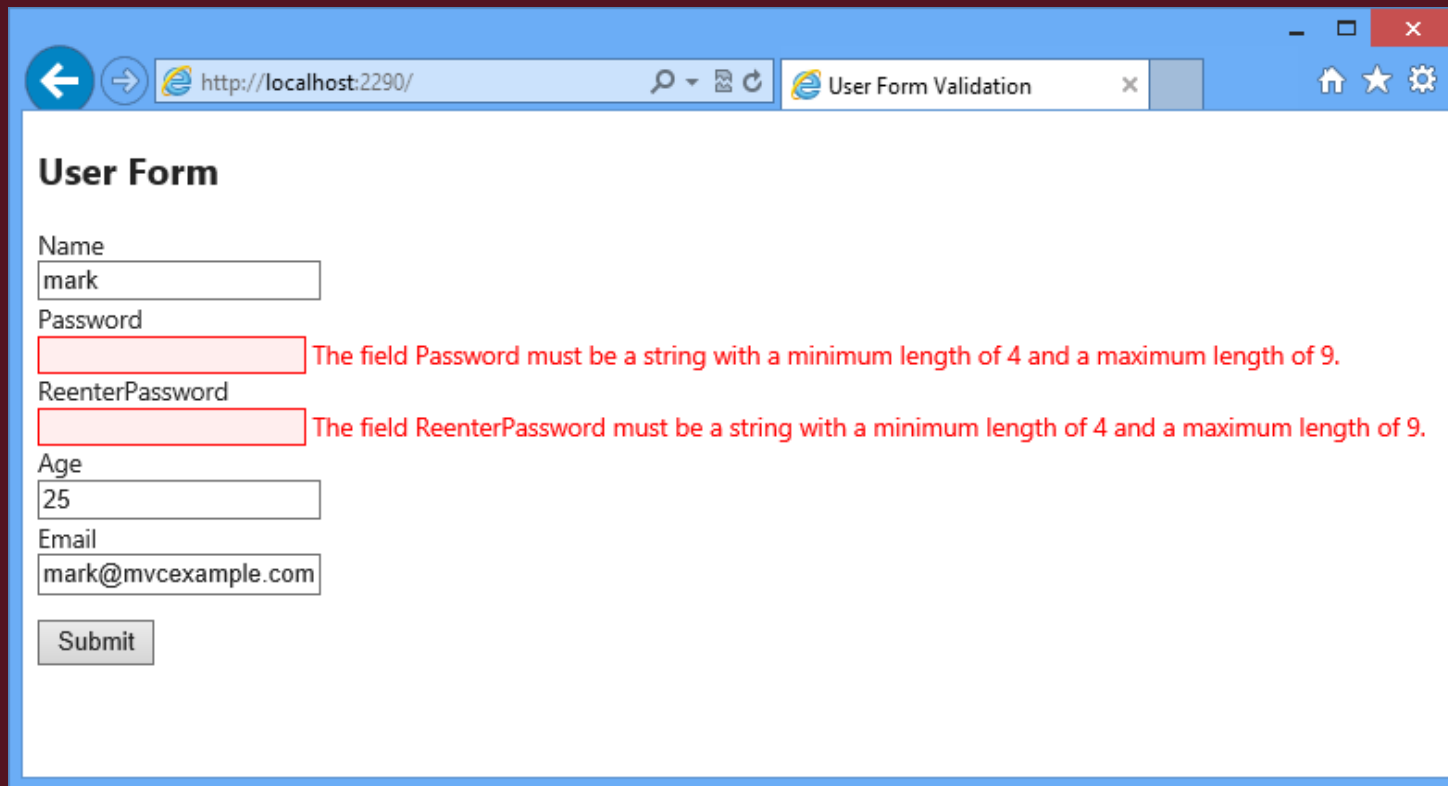
Code Snippet:

```
public class User { public long Id { get; set; }  
    [Required(ErrorMessage = "Please enter your name.")]  
    public string Name { get; set; }  
    [StringLength(9, MinimumLength = 4)]  
    [DataType(DataType.Password)]  
    public string Password { get; set; }  
    [StringLength(9, MinimumLength = 4)]  
    [DataType(DataType.Password)]  
    public string ReenterPassword { get; set; }  
    [Required (ErrorMessage = "Please enter your age.")]  
    public int Age { get; set; }  
    [Required(ErrorMessage = "Please enter your Email-ID.")]  
    public string Email { get; set; } }
```

- ◆ Trong đoạn code, chú thích **StringLength** xác định độ dài lớn nhất của **Password** và **ReenterPassword** là 9, độ dài nhỏ nhất là 4

StringLength

- ◆ Bất cứ khi nào giá trị mà người dùng nhập trong các trường này nằm ngoài khoảng từ 4 đến 9, một tin nhắn lỗi xác thực được hiển thị
- ◆ Kết quả :



The screenshot shows a web browser window with the address bar displaying `http://localhost:2290/` and the page title `User Form Validation`. The page content is titled **User Form** and contains several input fields and a submit button. The `Password` and `ReenterPassword` fields are highlighted with red borders, indicating validation errors. The error messages for these fields are: "The field Password must be a string with a minimum length of 4 and a maximum length of 9." and "The field ReenterPassword must be a string with a minimum length of 4 and a maximum length of 9." respectively. The other fields are filled with valid data: `Name` is "mark", `Age` is "25", and `Email` is "mark@mvcexample.com".

User Form

Name
mark

Password
The field Password must be a string with a minimum length of 4 and a maximum length of 9.

ReenterPassword
The field ReenterPassword must be a string with a minimum length of 4 and a maximum length of 9.

Age
25

Email
mark@mvcexample.com

Submit

RegularExpression

- ◆ Sử dụng chú thích **RegularExpression** để tiếp nhận input người dùng ở một định dạng xác định.
- ◆ Chú thích này cho phép bạn đối sánh một dòng văn bản string với một mẫu tìm kiếm có chứa 1 hoặc nhiều kí tự, toán tử, cấu trúc.
- ◆ Cú pháp:

Syntax:

```
[RegularExpression(<pattern>)]
```

Trong đó,

- ▮ <pattern>: định dạng xác định dựa theo input mà người dùng muốn

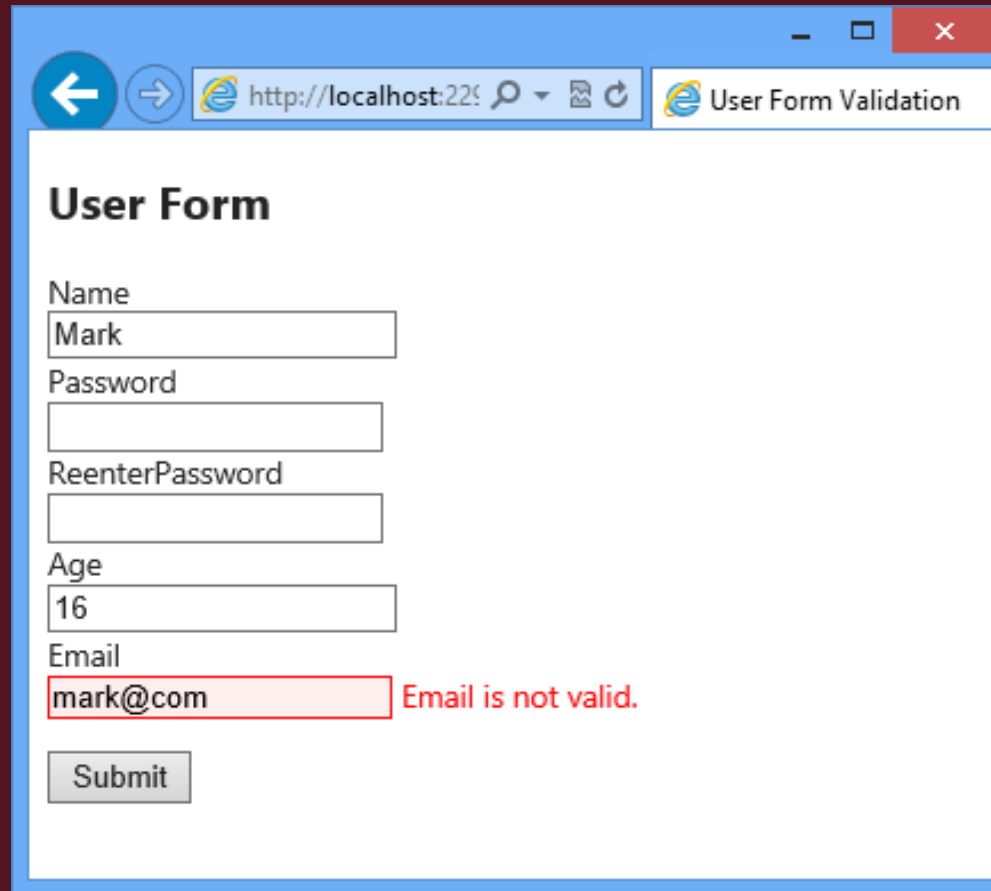
- ◆ Code snippet 7 sau đây biểu thị một model *User* với chú thích *RegularExpression* áp dụng cho thuộc tính Email:

Code Snippet:

```
public class User    {
    public long Id { get; set; }
    [Required(ErrorMessage = "Please enter your name.")]
    public string Name { get; set; }
    [StringLength(9, MinimumLength = 4)]
    public string Password { get; set; }
    [StringLength(9, MinimumLength = 4)]
    public string ReenterPassword { get; set; }
    [Required (ErrorMessage = "Please enter your age.")]
    public int Age { get; set; }
    [RegularExpression(@"[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}",
        ErrorMessage = "Email is not valid.")]
    public string Email { get; set; }
}
```


- ◆ Trong đoạn code trên:
 - ◆ Biểu thức chính quy `A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}` định nghĩa ra định dạng của địa chỉ email. Biểu thức này được chia làm 3 phần:
 - **Phần 1:** `[A-Za-z0-9._%+-]+`
 - **Phần 2:** `[A-Za-z0-9.-]+`
 - **Phần 3:** `[A-Za-z]{2,4}`
- ◆ Phần đầu tiên cho biết các kí tự và các dấu bên trong cặp ngoặc vuông sẽ được phép xuất hiện
- ◆ Tiếp sau đó, dấu '+' nằm giữa phần 1 và phần 2 thể hiện rằng phần 1 có thể chứa 1 hoặc nhiều kí tự thuộc các dạng ở bên trong dấu ngoặc vuông trước dấu '+'.
..
- ◆ Phần 3 có chứa '{2,4}' ở cuối thể hiện rằng phần này có thể chứa từ 2 đến 4 kí tự.

- ◆ Kết quả khi nhập một email sai định dạng:



The screenshot shows a web browser window titled "User Form Validation" at the URL "http://localhost:22...". The form is titled "User Form" and contains the following fields:

- Name:
- Password:
- ReenterPassword:
- Age:
- Email: (highlighted with a red border, with the message "Email is not valid." next to it)

A "Submit" button is located at the bottom of the form.

- ◆ Sử dụng chú thích ***Range*** để xác định giá trị lớn nhất và nhỏ nhất đối với một giá trị số.
- ◆ Cú pháp:

Syntax:

```
[Range (<minimum_range>, <maximum_range>)]
```

Trong đó,

- ▮ `minimum_range`: Giá trị số biểu thị giá trị nhỏ nhất
- ▮ `maximum_range`: Giá trị số biểu thị giá trị lớn nhất

- ◆ Đoạn code sau biểu thị ví dụ sử dụng chú thích **Range** cho class model **User**:

Code Snippet:

```
public class User    {
    public long Id { get; set; }
    [Required(ErrorMessage = "Please enter your name.")]
    public string Name { get; set; }
    [StringLength(9, MinimumLength = 4)]
    public string Password { get; set; }
    [StringLength(9, MinimumLength = 4)]
    public string ReenterPassword { get; set; }
    [Range(18, 60, ErrorMessage = "The age should be between 18 and 60.")]
    public int Age { get; set; }
    [RegularExpression(@"[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}",
        ErrorMessage = "Email is not valid.")]
    public string Email { get; set; }
}
```

- ◆ Kết quả của đoạn code trên:

The screenshot shows a web browser window titled 'User Form Validation' at the URL 'http://localhost:229'. The page contains a 'User Form' with the following fields and values:

- Name: Mark
- Password: (empty)
- ReenterPassword: (empty)
- Age: 16 (highlighted with a red border)
- Email: mark@mvcexample.com

A red text message next to the 'Age' field states: 'The age should be between 18 and 60.' Below the fields is a 'Submit' button.

- ◆ Sử dụng chú thích **Compare** để so sánh các giá trị ở 2 trường.
- ◆ Sử dụng chú thích **Compare** để chắc chắn rằng 2 thuộc tính này ở đối tượng model có giá trị giống nhau.
- ◆ Ví dụ ở Code snippet sau:

Code Snippet:

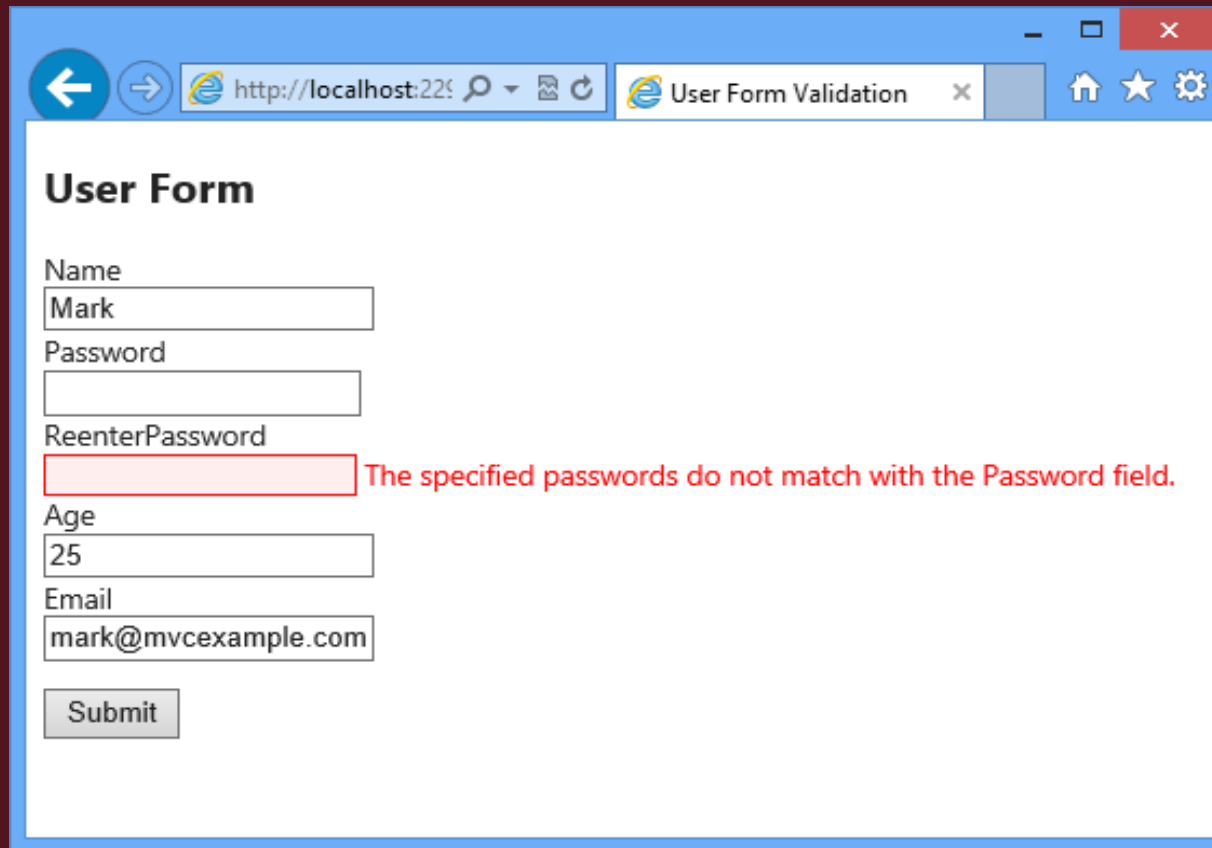
```
public class User    {
    public long Id { get; set; }
    [Required(ErrorMessage = "Please enter your name.")]
    public string Name { get; set; }
    [StringLength(9, MinimumLength = 4)]
    public string Password { get; set; }
    [StringLength(9, MinimumLength = 4)]
    [Compare("Password", ErrorMessage = "The specified passwords do not
        match with the Password field.")]
}
```

Code Snippet:

```
public string ReenterPassword { get; set; }
    [Range(18, 60, ErrorMessage = "The age should be between 18 and 60.")]
    public int Age { get; set; }
    [RegularExpression(@"[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}",
        ErrorMessage = "Email is not valid.")]
    public string Email { get; set; }
}
```

Đoạn code trên sử dụng chú thích ***Compare*** để kiểm tra trường ***ReenterPassword*** có giá trị giống với trường ***Password***.

◆ Kết quả:



The screenshot shows a web browser window with the address bar displaying 'http://localhost:228' and the page title 'User Form Validation'. The form is titled 'User Form' and contains the following fields:

- Name: Mark
- Password: (empty)
- ReenterPassword: (empty, highlighted with a red border)
- Age: 25
- Email: mark@mvcexample.com

A red error message is displayed next to the 'ReenterPassword' field: "The specified passwords do not match with the Password field." Below the form is a 'Submit' button.

- ◆ Khi sử dụng phương thức hỗ trợ **@Html.LabelFor()** trong một strongly typed view, phương thức hiển thị một label với giá trị là tên thuộc tính tương ứng.
- ◆ Bạn có thể đặt phương thức **@Html.LabelFor()** hiển thị một giá trị khác bằng cách sử dụng chú thích **DisplayName**.
- ◆ Cú pháp:

Syntax:

```
[DisplayName (<text> )]
```

Trong đó,

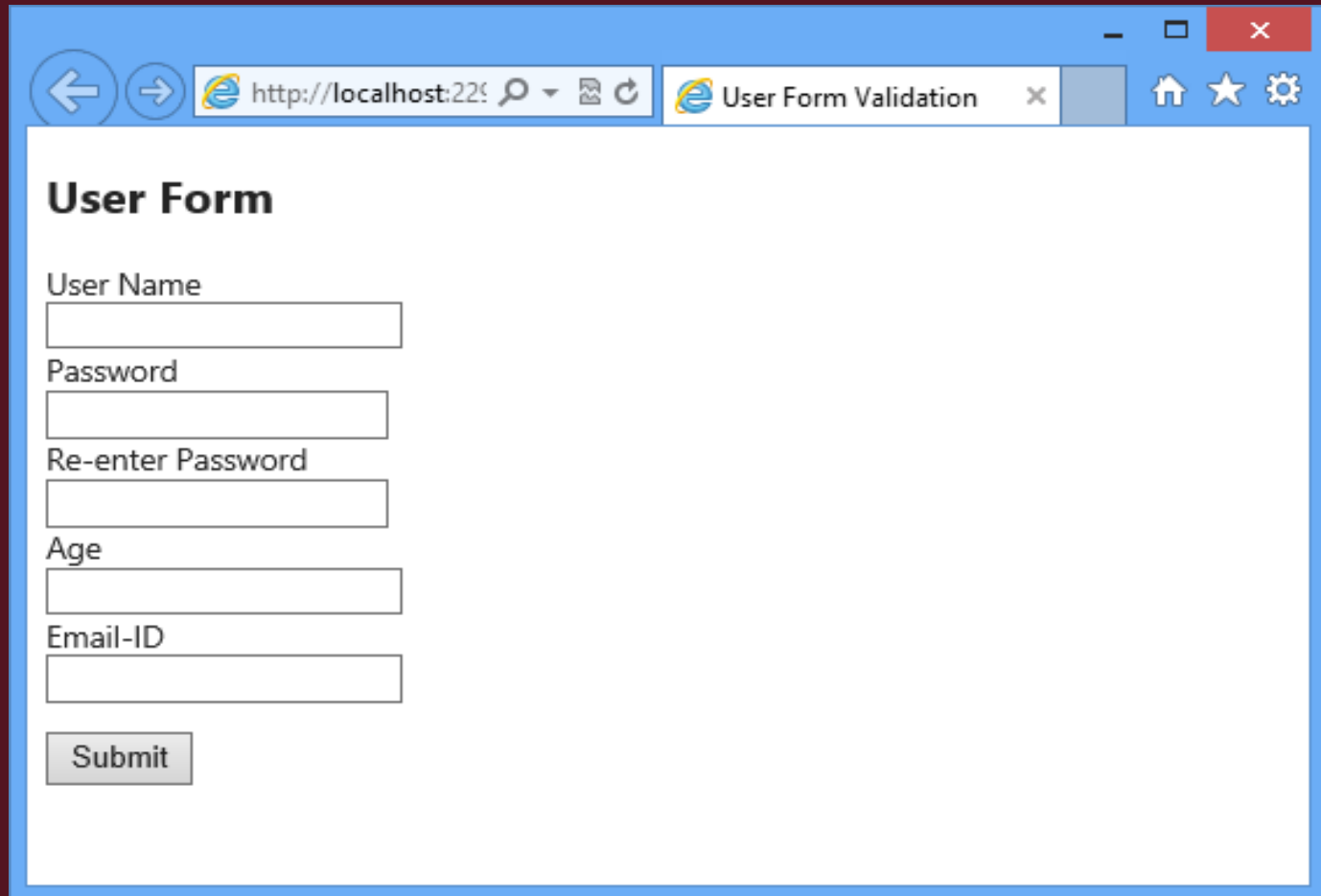
- ▮ `text`: đoạn văn bản mà bạn muốn hiển thị cho thuộc tính

- ◆ Code snippet sau biểu thị model **User** với chú thích **DisplayName** áp dụng cho các thuộc tính *Name*, *ReenterPassword*, *Email*:

Code Snippet:

```
public class User {  
    public long Id { get; set; }  
    [DisplayName("User  Name")]  
    public string Name { get; set; }  
    public string Password { get; set; }  
    [DisplayName("Re-enter  Password")]  
    public string ReenterPassword { get; set; }  
    public int Age { get; set; }  
    [DisplayName("Email- ID")]  
    public string Email { get; set; }  
}
```

- ◆ Kết quả:



The screenshot shows a web browser window with the address bar displaying 'http://localhost:225'. The page title is 'User Form Validation'. The main content area is titled 'User Form' and contains the following fields and buttons:

- User Name:
- Password:
- Re-enter Password:
- Age:
- Email-ID:
- Submit:

- ◆ Sử dụng chú thích **ReadOnly** để hiển thị các trường chỉ đọc.
- ◆ Sử dụng chú thích **ReadOnly** sẽ định hướng model binder mặc định không đặt thuộc tính **DiscountedAmount** với một giá trị mới từ yêu cầu.
- ◆ Cú pháp:

Syntax:

```
[ReadOnly(<boolean_value>)]
```

where,

- ▮ `boolean_value`: là một giá trị boolean, có thể là true hoặc false. Nếu là true, thuộc tính của trường được chỉ định sẽ có giá trị chỉ đọc. Nếu là false, model binder mặc định sẽ đặt một giá trị mới cho thuộc tính dựa trên yêu cầu.

- ◆ Ví dụ ở code snippet sau:

Code Snippet:

```
[ReadOnly(true)]  
public int DiscountedAmount { get; set; }  
}
```

- ◆ Đoạn code sử dụng chú thích ***ReadOnly*** được đặt là true đối với thuộc tính ***DiscountedAmount***.

- ◆ Sử dụng chú thích ***DataType*** để cung cấp thông tin về mục đích cụ thể của một thuộc tính tại thời gian chạy.
- ◆ Cú pháp:

Syntax:

```
[DataType(DataType.<value>)]
```

Trong đó,

- ▮ `value`: giá trị của `DataType`.

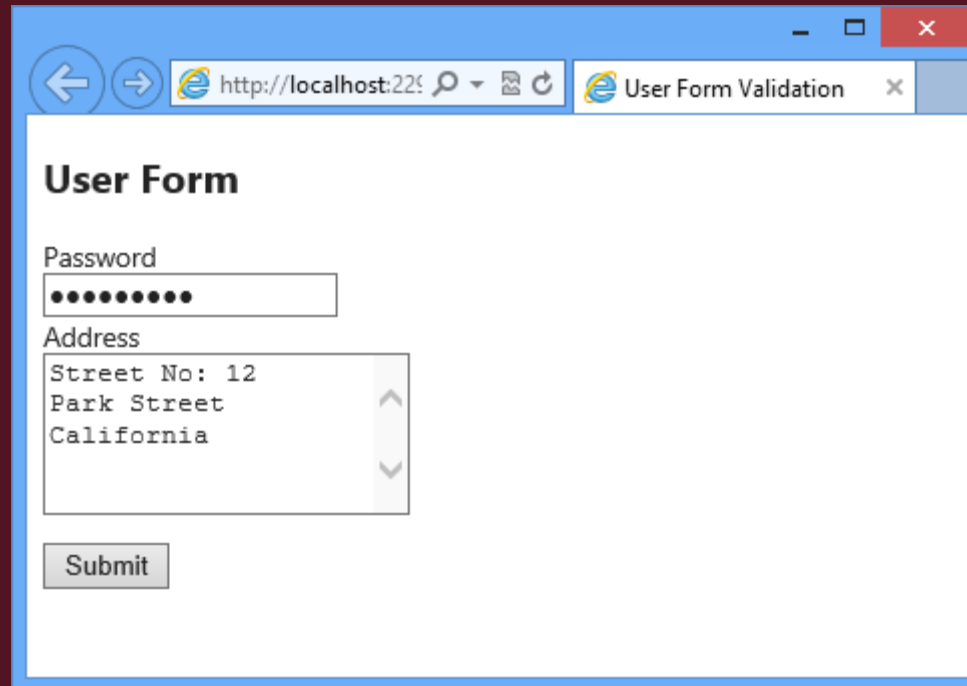
- ◆ Ví dụ ở Code snippet sau:

Code Snippet:

```
[DataType(DataType.Password)]  
public string Password { get; set; }  
[DataType(DataType.MultilineText)]  
public string Address { get; set; }
```

Trong đoạn code trên, **DataType** áp dụng cho các thuộc tính ***Password*** và ***Address***. Trường hiển thị thuộc tính ***Password*** là các ký tự được ẩn, thuộc tính ***Address*** là một vùng cho một đoạn text nhiều dòng.

- ◆ Kết quả:



The screenshot shows a web browser window with the title 'User Form Validation'. The address bar displays 'http://localhost:22...'. The main content area is titled 'User Form' and contains the following elements:

- A 'Password' label above a text input field filled with ten black dots.
- An 'Address' label above a list box containing three items: 'Street No: 12', 'Park Street', and 'California'. The list box has up and down arrow buttons on its right side.
- A 'Submit' button located below the address list.

- ◆ Khi sử dụng scaffolding với **Create** template, view theo mặc định sẽ tạo ra các trường UI cho tất cả các thuộc tính của model.
- ◆ Tuy vậy, có thể ta lại muốn rằng view này không tạo các trường UI cho một số các thuộc tính nhất định.
- ◆ Ở các tình huống như vậy, ta sử dụng chú thích **ScaffoldColumn** với giá trị false.
- ◆ Cú pháp:

Code Snippet:

```
[ScaffoldColumn (false)]
```

```
public long Id { get; set; }
```

- ◆ Trong đoạn code, chú thích **ScaffoldColumn** với giá trị false sẽ giúp cho **Create** scaffolding template không tạo ra một trường UI cho thuộc tính Id

Xác thực ModelState

- ◆ Trong đoạn code, chú thích ***ScaffoldColumn*** với giá trị false sẽ giúp cho ***Create*** scaffolding template không tạo ra một trường UI cho thuộc tính Id.
- ◆ Ta sử dụng ***ModelState.IsValid*** để kiểm tra trạng thái của model, theo ở ví dụ trên là false. Vì vậy, ta có thể trả về view hiển thị một tin nhắn xác thực lỗi bằng phương thức hỗ trợ ***Html.ValidationMessage()***.
- ◆ Nếu model binding thành công, ***ModelState.IsValid()*** trả về true và lúc này ta có thể thực hiện các chức năng yêu cầu với dữ liệu model..

- ◆ Ví dụ ở Code snippet sau:

Code Snippet:

```
public ActionResult Index(User model)
{
    if (ModelState.IsValid)
    {
        /*Perform required function with the model data*/
        return Content("You have successfully registered");
    }
    else
    return View();
}
```

Trong đoạn code, ***ModelState.IsValid()*** được sử dụng để kiểm tra trạng thái của model ***User***. Nếu thuộc tính này trả về true, một tin nhắn xác thực thành công được hiển thị. Nếu thuộc tính trả về false, view cho phép người dùng nhập lại giá trị hợp lệ.

- MVC Framework triển khai một luồng xác thực để xác thực dữ liệu người dùng
- Trong ứng dụng ASP.NET MVC, ta có thể xác thực dữ liệu trong hành động controller một cách thủ công
- MVC Framework cung cấp các chú thích dữ liệu có thể áp dụng như thuộc tính đối với các thuộc tính của model
- Chú thích Required khi áp dụng với một thuộc tính xác thực một giá trị được chỉ định cho một thuộc tính
- RegularExpression khi áp dụng với một thuộc tính xác thực một giá trị được chỉ định cho một thuộc tính, đối sánh với biểu thức chính quy xác định
- Phương thức hỗ trợ @Html.LabelFor() khi sử dụng trong một strongly typed view hiển thị một label với một dòng text có giá trị là tên thuộc tính tương ứng
- ModelState là một class trong namespace System.Web.Mvc đóng gói trạng thái của model binding trong ứng dụng.