ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
VNUHCM - UIT

KIỂM SOÁT DỮ LIỆU

CHƯƠNG 5
KIỂM SOÁT DỮ LIỆU

Email: dungta@uit.edu.vn

NỘI DUNG

1. Kiểm tra dữ liệu?
2. ASP.NET MVC: ValidationMessage
3. ValidationMessageFor trong MVC
4. ValidationSummary trong MVC

1. KIỂM TRA DỮ LIỆU?

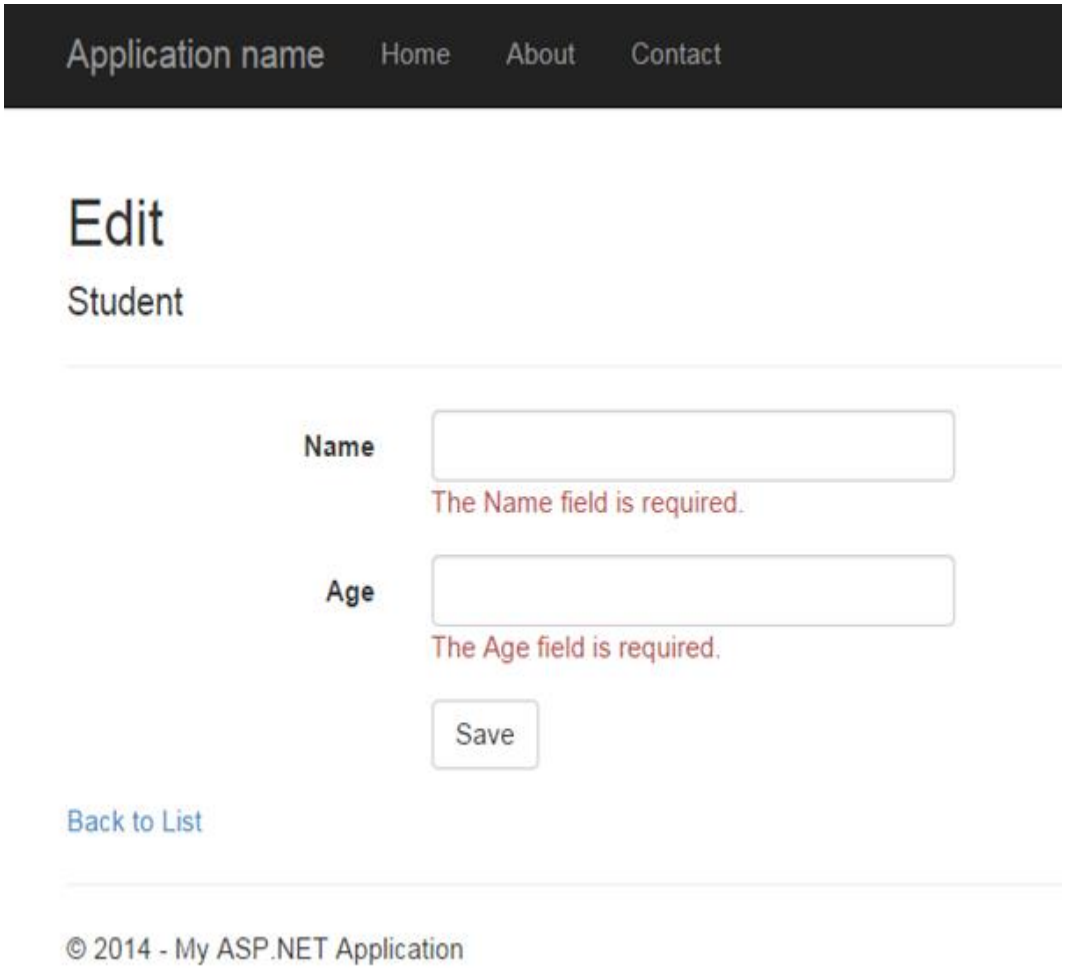
- Khi xây dựng ứng dụng, chúng ta nên kiểm tra dữ liệu nhập từ người dùng để hạn chế các sai sót dữ liệu nhập nhằm đảm bảo việc thực hiện xử lý dữ liệu được chính xác theo các yêu cầu nghiệp vụ. Nếu chúng ta viết mã để kiểm tra thì phải mất nhiều thời gian (sử dụng JavaScript hoặc VBScript).

1. KIỂM TRA DỮ LIỆU?

- Với phiên bản trước của ASP.Net là ASP thì để khắc phục lỗi đó chúng ta phải thực hiện viết mã JavaScript để bắt lỗi việc đó, còn với ASP.NET đã cung cấp cho ta những điều khiển kiểm tra tính hợp lệ của các điều khiển nhập liệu trên Form.
- Webform hỗ trợ các validation controls để kiểm tra dữ liệu nhập từ người dùng trong các sever controls, mục đích là tránh để người dùng nhập sai hoặc không được bỏ trống các thông tin quan trọng bắt buộc,…

1. KIỂM TRA DỮ LIỆU?

- Trong phần trước, chúng ta đã tạo một View Edit Student. Bây giờ, chúng ta sẽ xác thực dữ liệu trong chế độ Edit, sẽ hiển thị thông báo lỗi khi click vào nút Save, như hình bên dưới nếu Name và Age để trống.



1. KIỂM TRA DỮ LIỆU?

❖ DataAnnotations

- Trong .NET Framework, Data Annotation dùng để thêm phần ý nghĩa mở rộng vào dữ liệu thông qua các thẻ thuộc tính.
- Tính năng Data Annotation được Microsoft giới thiệu lần đầu tiên ở phiển bản .NET 3.5 thuộc namespace System.ComponentModel.DataAnnotations. Namespace này chứa các lớp dùng để định nghĩa thuộc tính mở rộng cho dữ liệu.

1. KIỂM TRA DỮ LIỆU?

- DataAnnotations: Các thuộc tính DataAnnotations namespace System.ComponentModel.DataAnnotations

Thuộc tính	Giải thích
Required	Chỉ định thuộc tính phải có dữ liệu nhập vào trước khi submit về server.
StringLength	Định nghĩa chiều dài thuộc tính, cho phép đặc tả cả chiều dài tối thiểu và tối đa.
Range	Định nghĩa giá trị số tối thiểu và tối đa của một thuộc tính.
RegularExpression	Là biểu thức chính quy được dùng để xử lý chuỗi nâng cao thông qua biểu thức riêng của nó
CreditCard	Định nghĩa số thẻ tính dụng (credit card)

1. KIỂM TRA DỮ LIỆU?

- DataAnnotations: Các thuộc tính DataAnnotations namespace System.ComponentModel.DataAnnotations

Thuộc tính	Giải thích
CustomValidation	Cho phép người dùng tự viết hàm xử lý kiểm tra lỗi
EmailAddress	Kiểm tra email người dùng nhập vào có hợp lệ?
FileExtension	Kiểm tra phần mở rộng của một tập tin
MaxLength	Kích thước tối đa của trường dữ liệu nhập vào
MinLength	Kích thước tối thiểu của trường dữ liệu nhập vào
Phone	Kiểm tra số phone có hợp lệ?

1. KIỂM TRA DỮ LIỆU?

❖ Ví dụ kiểm tra việc chỉnh sửa Sinh viên:

➢ Bước 1: Chúng ta muốn xác thực rằng StudentName và Age không trống. Ngoài ra, Tuổi nên nằm trong khoảng từ 5 đến 50.

```
public class Student
{
    public int StudentId { get; set; }

    [Required]
    public string StudentName { get; set; }

    [Range(5,50)]
    public int Age { get; set; }
}
```

1. KIỂM TRA DỮ LIỆU?

❖ Ví dụ kiểm tra việc chỉnh sửa Sinh viên:

➢ Bước 2:

✓Tạo phương thức Edit GET và POST giống như phần trước.

✓Phương thức GET sẽ hiển thị Edit xem chỉnh sửa sinh viên đã chọn.

✓Phương thức Edit dạng POST sẽ lưu sinh viên đã chỉnh sửa như hiển thị bên dưới.

1. KIỂM TRA DỮ LIỆU?

❖ Ví dụ kiểm tra việc chỉnh sửa Sinh viên:

```
using MVC_BasicTutorials.Models;

namespace MVC_BasicTutorials.Controllers
{
    public class StudentController : Controller
    {
        public ActionResult Edit(int id)
        {
            var std = studentList.Where(s => s.StudentId == id)
                                 .FirstOrDefault();

            return View(std);
        }
```

1. KIỂM TRA DỮ LIỆU?

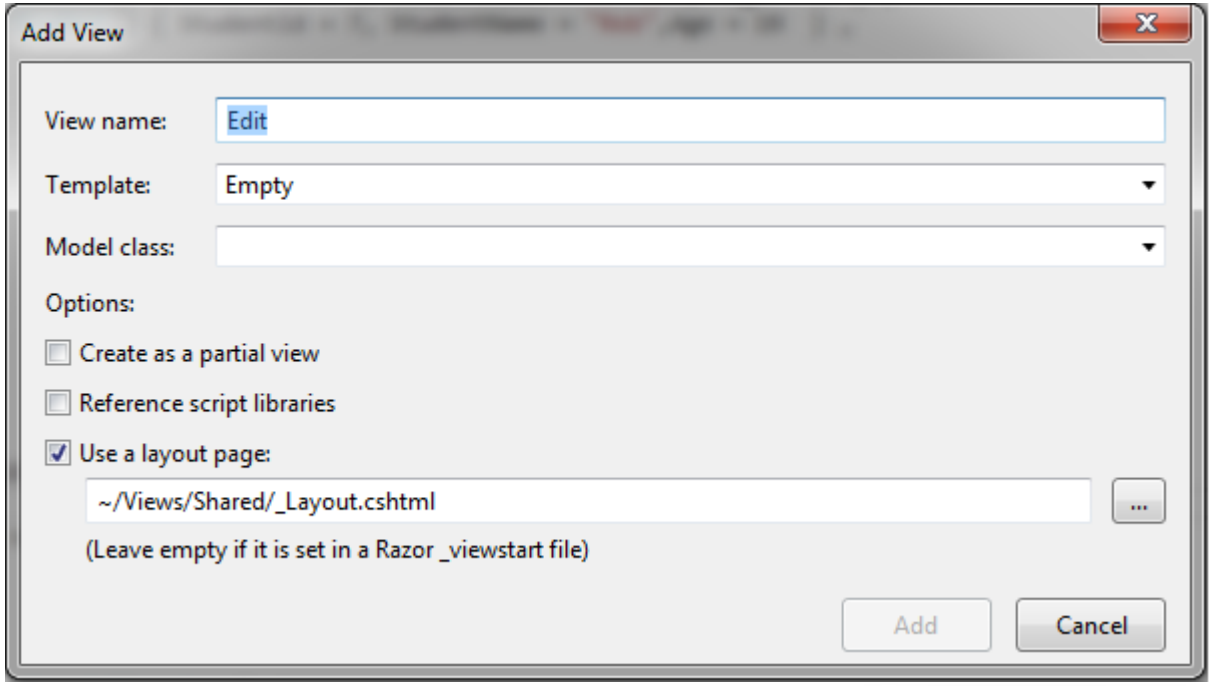
❖ Ví dụ kiểm tra việc chỉnh sửa Sinh viên:

```
        [HttpPost]
        public ActionResult Edit(Student std)
        {
            if (ModelState.IsValid) {

                //write code to update student

                return RedirectToAction("Index");
            }

            return View(std);
        }
    }
}
```

1. KIỂM TRA DỮ LIỆU?

❖ Ví dụ kiểm tra việc chỉnh sửa Sinh viên:

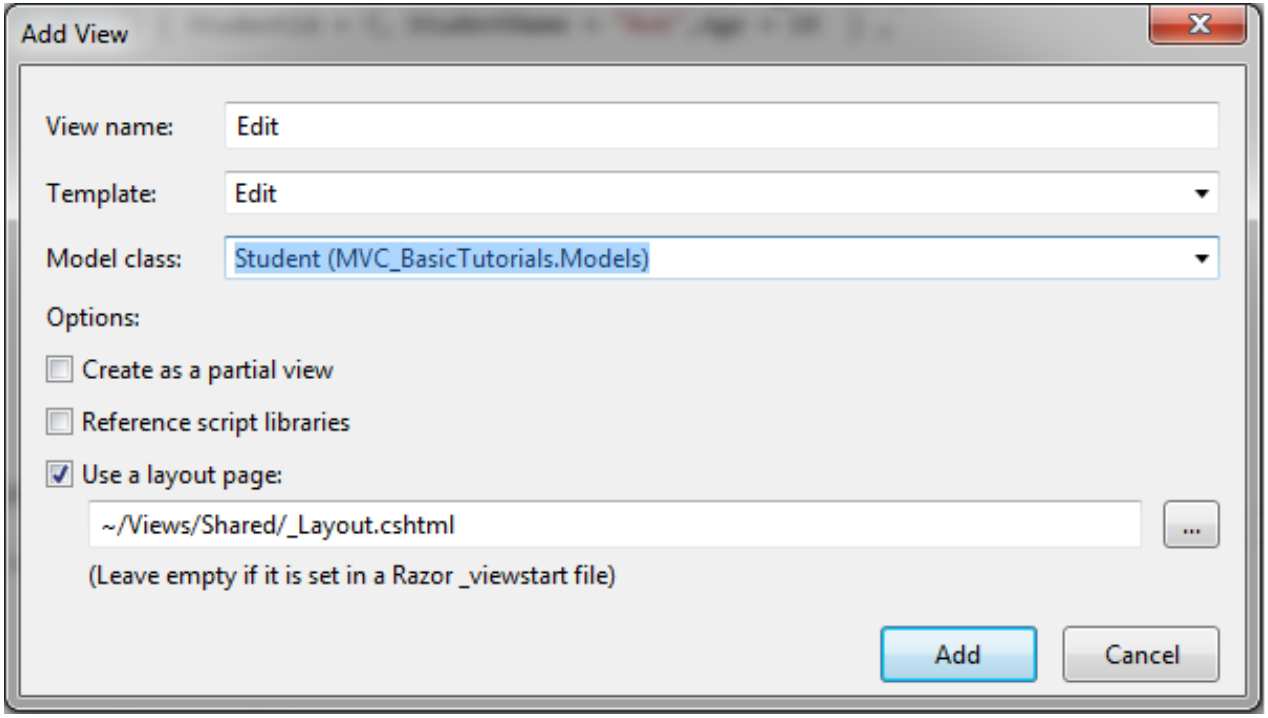
➢ Bước 3: tạo phương thức Edit, nhấp chuột phải vào bên trong Chỉnh sửa phương thức hành động -> click vào Add View...



1. KIỂM TRA DỮ LIỆU?

❖ Ví dụ kiểm tra việc chỉnh sửa Sinh viên:

➢ Chọn mẫu Edit template trong danh sách thả xuống template và cũng chọn lớp Model Student như dưới đây.



1. KIỂM TRA DỮ LIỆU?

- Ví dụ kiểm tra việc chỉnh sửa Sinh viên:
  - Bây giờ, click vào Add để tạo Edit trong thư mục View/Student. Tập tin Edit.cshtml sẽ được tạo ra:

Ví dụ kiểm tra việc chỉnh sửa Sinh viên

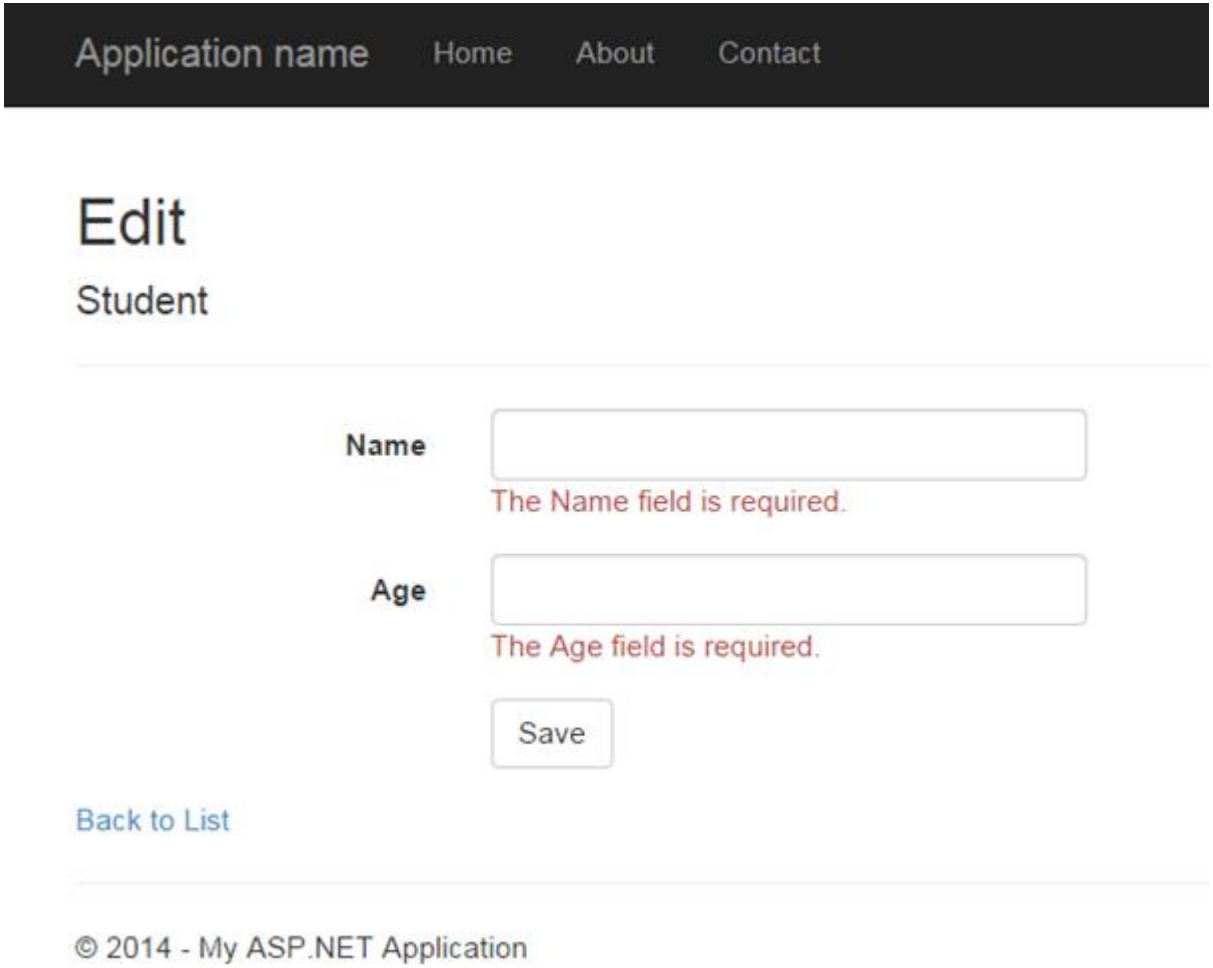
```
@model MVC_BasicTutorials.Models.Student
@{
    ViewBag.Title = "Edit";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Edit</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Student</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.StudentId)

        <div class="form-group">
            @Html.LabelFor(model => model.StudentName, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.StudentName, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.StudentName, "", new { @class = "text-danger" })
            </div>
        </div>
```

Ví dụ kiểm tra việc chỉnh sửa Sinh viên

```
        <div class="form-group">
            @Html.LabelFor(model => model.Age, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Age, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Age, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}
<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

Ví dụ kiểm tra việc chỉnh sửa Sinh viên

- Bây giờ, nó sẽ hiển thị thông báo xác thực mặc định khi bạn lưu mà không nhập Tên hoặc Tuổi.



1. KIỂM TRA DỮ LIỆU?

- Các thuộc tính xác thực mặc định:
  - Required
  - StringLength
  - Range
  - RegularExpression
  - CreditCard
  - CustomValidation
  - EmailAddress
  - FileExtension
  - MaxLength
  - MinLength
  - Phone

2. ASP.NET MVC: ValidationMessage

- Html.ValidationMessage() là một phương thức mở rộng. Nó hiển thị thông báo xác nhận nếu có lỗi tồn tại cho trường được chỉ định trong đối tượng ModelStateDipedia.
- Cú pháp ValidationMessage():

```
MvcHtmlString ValidateMessage(string modelName,
string validationMessage, object htmlAttributes)
```

2. ASP.NET MVC: ValidationMessage

- ValidationMessage có nhiều phương thức. Vui lòng truy cập MSDN để biết tất cả ValidationMessage.
- Ví dụ:

```
@model Student

@Html.Editor("StudentName") <br />
@Html.ValidationMessage("StudentName", "", new { @class = "text-danger" })
```

2. ASP.NET MVC: ValidationMessage

- Phương thức ValidationMessage() sẽ chỉ hiển thị một lỗi, nếu bạn đã cài đặt thuộc tính DataAnnotations cho thuộc tính được chỉ định trong lớp model.

Ví dụ Model Student

```
public class Student
{
    public int StudentId { get; set; }
    [Required]
    public string StudentName { get; set; }
    public int Age { get; set; }
}
```

2. ASP.NET MVC: ValidationMessage

```
@model Student

@Html.Editor("StudentName") <br />
@Html.ValidationMessage("StudentName", "", new { @class = "text-danger" })
```

❖ Đoạn mã trên sẽ tạo ra html sau:

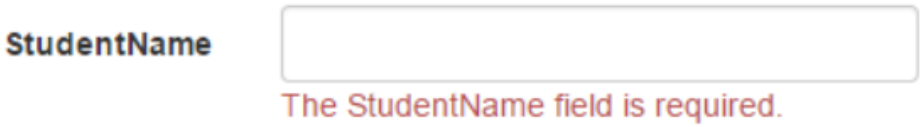
```
<input id="StudentName"
       name="StudentName"
       type="text"
       value="" />

<span class="field-validation-valid text-danger"
       data-valmsg-for="StudentName"
       data-valmsg-replace="true">
</span>
```

2. ASP.NET MVC: ValidationMessage

- Bây giờ, khi người sử dụng gửi form mà không nhập StudentName, thì ASP.NET MVC sử dụng thuộc tính dữ liệu của Html5 để xác thực và thông báo xác thực mặc định sẽ được đưa vào, khi xảy ra lỗi xác thực, như hiển thị bên dưới.

```
<span class="field-validation-error text-danger"
              data-valmsg-for="StudentName"
              data-valmsg-replace="true">The StudentName field is required.</span>
```

Lỗi thông báo :



2. ASP.NET MVC: ValidationMessage

❖ Tùy chỉnh thông báo lỗi:

➢ Bạn có thể hiển thị thông báo lỗi của riêng bạn thay vì thông báo lỗi mặc định như được hiển thị ở trên. Bạn có thể cung cấp một thông báo lỗi tùy chỉnh trong thuộc tính DataAnnotations hoặc phương thức ValidationMessage().

```
public class Student
{
    public int StudentId { get; set; }
    [Required(ErrorMessage="Please enter student name.")]
    public string StudentName { get; set; }
    public int Age { get; set; }
}
```

2. ASP.NET MVC: ValidationMessage

❖ Tùy chỉnh thông báo lỗi:

➢ Ngoài ra, bạn có thể chỉ định một thông báo là tham số thứ hai trong phương thức ValidationMessage () như hiển thị bên dưới.

```
@model Student

@Html.Editor("StudentName") <br />
@Html.ValidationMessage("StudentName",
                        "Please enter student name.",
                        new { @class = "text-danger" })
```

3. ValidationMessageFor TRONG MVC

- Html.ValidationMessageFor() là một phương thức mở rộng với thuộc tính xác định sử dụng một biểu thức lambda. Nó hiển thị thông báo xác nhận nếu có lỗi tồn tại cho trường được chỉ định trong đối tượng ModelStateDipedia.
- Cú pháp ValidationMessageFor():

```
MvcHtmlString ValidateMessage(Expression<Func<dynamic,TProperty>>
expression, string validationMessage, object htmlAttributes)
```

3. ValidationMessageFor TRONG MVC

❖ Ví dụ về ValidationMessageFor:

```
@model Student

@Html.EditorFor(m => m.StudentName) <br />
@Html.ValidationMessageFor(m => m.StudentName, "", new { @class = "text-danger" })
```

❖ Phương thức ValidationMessageFor() sẽ chỉ hiển thị lỗi nếu bạn đã cấu hình thuộc tính DataAnnotations cho thuộc tính được chỉ định trong lớp mô hình.

3. ValidationMessageFor TRONG MVC

- Ví dụ sau đây là lớp mô hình Sinh viên trong đó thuộc tính "Required" được áp dụng cho thuộc tính StudentName.

```
public class Student
{
    public int StudentId { get; set; }
    [Required]
    public string StudentName { get; set; }
    public int Age { get; set; }
}
```

3. ValidationMessageFor TRONG MVC

```
public class Student
{
    public int StudentId { get; set; }
    [Required]
    public string StudentName { get; set; }
    public int Age { get; set; }
}
```

- Đoạn mã trên sẽ tạo ra html sau:

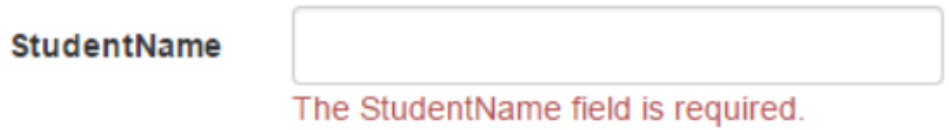
```
<input id="StudentName"
       name="StudentName"
       type="text"
       value="" />

<span class="field-validation-valid text-danger"
       data-valmsg-for="StudentName"
       data-valmsg-replace="true">
</span>
```

3. ValidationMessageFor TRONG MVC

- Khi user submit mà không nhập StudentName thì ASP.NET MVC sử dụng thuộc tính dữ liệu của Html5 để xác thực và thông báo xác thực mặc định sẽ được đưa vào khi xảy ra lỗi xác thực, như hiển thị sau:

```
<span class="field-validation-error text-danger"
              data-valmsg-for="StudentName"
              data-valmsg-replace="true">The StudentName field is required.</span>
```

Thông báo lỗi sẽ xuất hiện như hình ảnh hiển thị bên dưới.



3. ValidationMessageFor TRONG MVC

- Tùy chỉnh thông báo lỗi:
  - Bạn có thể hiển thị thông báo lỗi của riêng bạn thay vì thông báo lỗi mặc định như được hiển thị ở trên.
  - Bạn có thể cung cấp một thông báo lỗi tùy chỉnh trong thuộc tính DataAnnotations hoặc phương thức ValidationMessageFor().

3. ValidationMessageFor TRONG MVC

- Tùy chỉnh thông báo lỗi:
  - Ví dụ thông báo lỗi tùy chỉnh trong Model

```
public class Student
{
    public int StudentId { get; set; }
    [Required(ErrorMessage="Please enter student name.")]
    public string StudentName { get; set; }
    public int Age { get; set; }
}
```

3. ValidationMessageFor TRONG MVC

❖ Tùy chỉnh thông báo lỗi:

➢ Ngoài ra, bạn có thể chỉ định một thông báo là tham số thứ hai trong phương thức ValidationMessage() như hiển thị bên dưới.

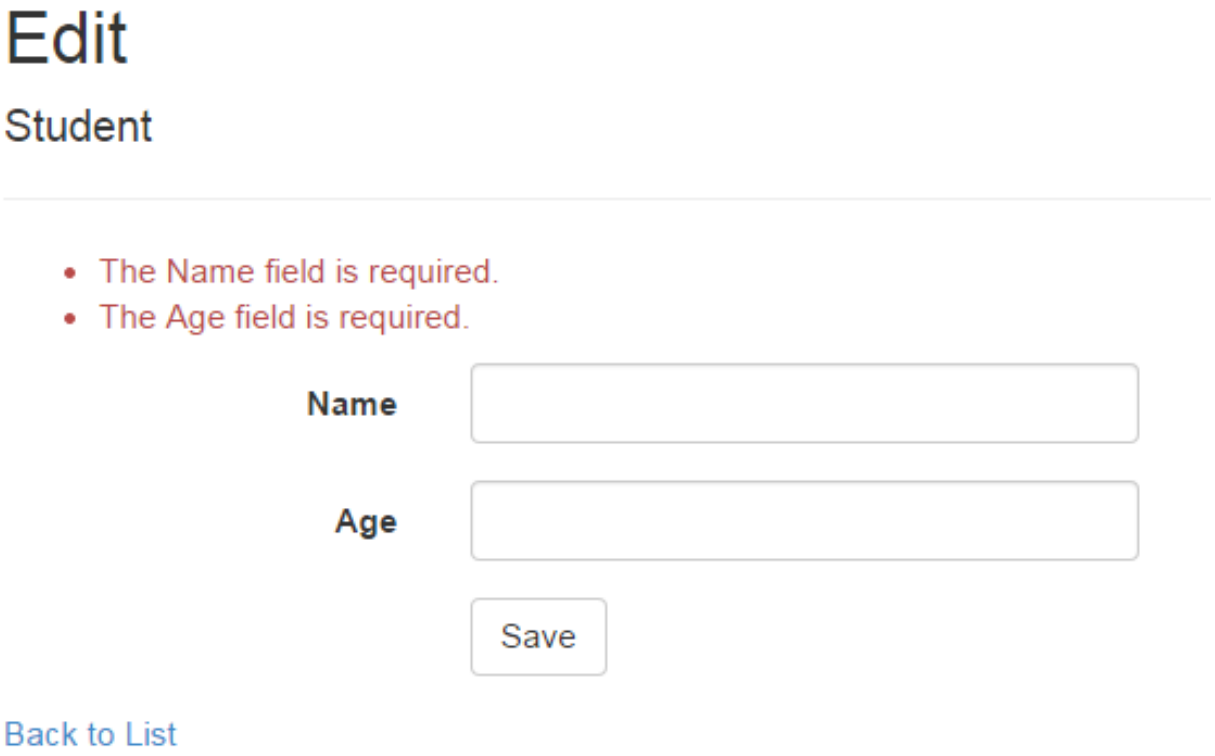
```
@model Student

@Html.Editor("StudentName") <br />
@Html.ValidationMessageFor(m => m.StudentName,
    "Please enter student name.", new { @class = "text-danger" })
```

4. ValidationSummary TRONG MVC

- Phương thức ValidationSummary() tạo ra một danh sách không có thứ tự (phần tử ul) các thông báo xác thực nằm trong đối tượng ModelStateDipedia.
- Có thể sử dụng ValidationSummary để hiển thị tất cả các thông báo lỗi cho tất cả các trường. Nó cũng có thể được sử dụng để hiển thị các thông báo lỗi tùy chỉnh.

4. ValidationSummary TRONG MVC

- Hình dưới đây cho thấy cách xác thực hiển thị các thông báo lỗi.



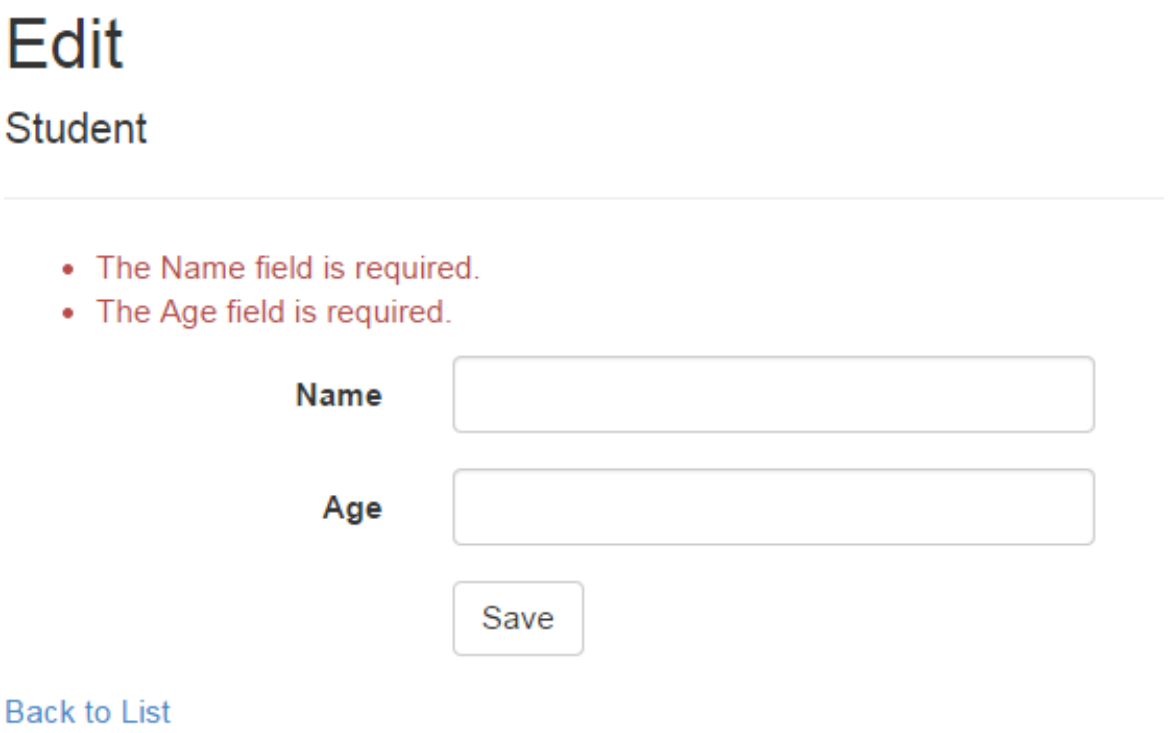
4. ValidationSummary TRONG MVC

- Hiển thị thông báo bằng cách sử dụng phương thức ValidationSummary
- Theo mặc định, ValidationSummary lọc ra các thông báo lỗi các trường. Nếu bạn muốn hiển thị các thông báo lỗi các trường dưới dạng tóm tắt, hãy chỉ định ElimPropertyErrors = false.

4. ValidationSummary TRONG MVC

❖ Ví dụ:

```
@Html.ValidationSummary(false, "", new { @class = "text-danger" })
```



4. ValidationSummary TRONG MVC

- Hiển thị thông báo lỗi tùy chỉnh:
  - Bạn cũng có thể hiển thị thông báo lỗi tùy chỉnh bằng cách sử dụng ValidationSummary. Ví dụ: chúng tôi muốn hiển thị một thông báo nếu Tên sinh viên đã tồn tại trong cơ sở dữ liệu.
  - Để hiển thị thông báo lỗi tùy chỉnh, trước hết, bạn cần thêm lỗi tùy chỉnh vào ModelState trong phương thức hành động thích hợp.

4. ValidationSummary TRONG MVC

❖ Hiển thị thông báo lỗi tùy chỉnh:

```
if (ModelState.IsValid) {

    //check whether name is already exists in the database or not
    bool nameAlreadyExists = * check database *

    if(nameAlreadyExists)
    {
        ModelState.AddModelError(string.Empty, "Student Name already exists.");

        return View(std);
    }
}
```

Email: dungta@uit.edu.vn