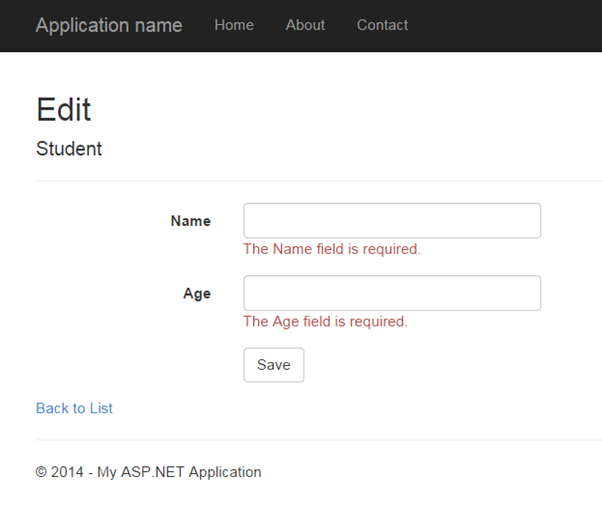
Implement Data Validation in MVC

In this section, you will learn how to implement data validations in the ASP.NET MVC application.

We have created an Edit view for Student in the previous section. Now, we will implement data validation in the Edit view, which will display validation messages on the click of Save button, as shown below if Student Name or Age is blank.

[](https://www.tutorialsteacher.com/Content/images/mvc/validation.png)Validation

DataAnnotations

ASP.NET MVC uses DataAnnotations attributes to implement validations. DataAnnotations includes built-in validation attributes for different validation rules, which can be applied to the properties of model class. ASP.NET MVC framework will automatically enforce these validation rules and display validation messages in the view.

The DataAnnotations attributes included in *System.ComponentModel.DataAnnotations* namespace. The following table lists DataAnnotations validation attributes.

| **Attribute** | **Description** |
| --- | --- |
| Required | Indicates that the property is a required field |
| StringLength | Defines a maximum length for string field |
| Range | Defines a maximum and minimum value for a numeric field |
| RegularExpression | Specifies that the field value must match with specified Regular Expression |
| CreditCard | Specifies that the specified field is a credit card number |
| CustomValidation | Specified custom validation method to validate the field |
| EmailAddress | Validates with email address format |
| FileExtension | Validates with file extension |
| MaxLength | Specifies maximum length for a string field |
| MinLength | Specifies minimum length for a string field |
| Phone | Specifies that the field is a phone number using regular expression for phone numbers |

Let's start to implement validation in Edit view for student.

**Step 1:** First of all, apply DataAnnotation attribute on the properties of Student model class. We want to validate that StudentName and Age is not blank. Also, Age should be between 5 and 50. Visit [Model](https://www.tutorialsteacher.com/mvc/mvc-model) section if you don't know how to create a model class.

Example: Apply DataAnnotation Attributes

public class Student

{

public int StudentId { get; set; }

[Required]

public string StudentName { get; set; }

[Range(5,50)]

public int Age { get; set; }

}

You can also apply multiple DataAnnotations validation attributes to a single property if required.

In the above example, we have applied a ***Required*** attribute to the StudentName property. So now, the MVC framework will automatically display the default error message, if the user tries to save the Edit form without entering the Student Name. In the same way, the ***Range*** attribute is applied with a min and max value to the Age property. This will validate and display an error message if the user has either not entered Age or entered an age less than 5 or more than 50.

**Step 2:** Create the GET and POST Edit Action method in the same as previous section. The GET action method will render Edit view to edit the selected student and the POST Edit method will save edited student as shown below.

Example: Edit Action methods:

using MVC\_BasicTutorials.Models;

namespace MVC\_BasicTutorials.Controllers

{

public class StudentController : Controller

{

public ActionResult Edit(int id)

{

var std = studentList.Where(s => s.StudentId == StudentId)

.FirstOrDefault();

return View(std);

}

[HttpPost]

public ActionResult Edit(Student std)

{

if (ModelState.IsValid) {

//write code to update student

return RedirectToAction("Index");

}

return View(std);

}

}

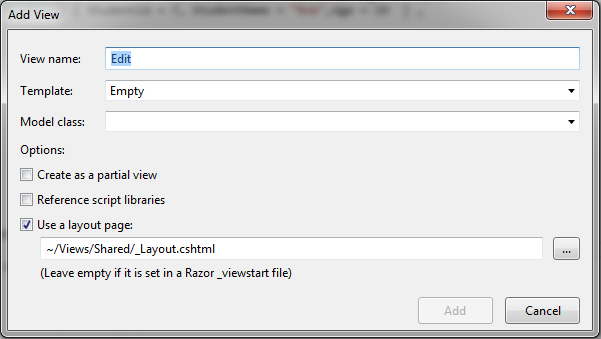
}

As you can see in the POST Edit method, we first check if the ModelState is valid or not. If ModelState is valid then update the student into database, if not then return Edit view again with the same student data.

ModelState.IsValid determines that whether submitted values satisfy all the DataAnnotation validation attributes applied to model properties.

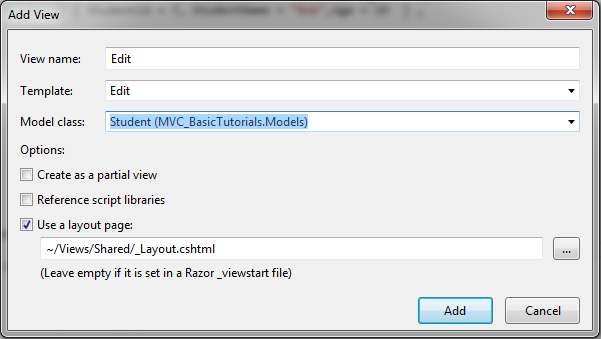
**Step 3:** Now, create an Edit view for Student.

To create an Edit view, right click inside Edit action method -> click **Add View..**

[](https://www.tutorialsteacher.com/Content/images/mvc/addview.png)Create Edit View

In the Add View dialogue, keep the view name as Edit. (You can change as per your requirement.)

Select the Edit template in the Template dropdown and also select Student Model class as shown below.

[](https://www.tutorialsteacher.com/Content/images/mvc/addview-2.png)Create Edit View

Now, click **Add** to generate Edit view under View/Student folder. Edit.cshtml will be generated as shown below.

Edit.cshtml:

@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>

<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>

}

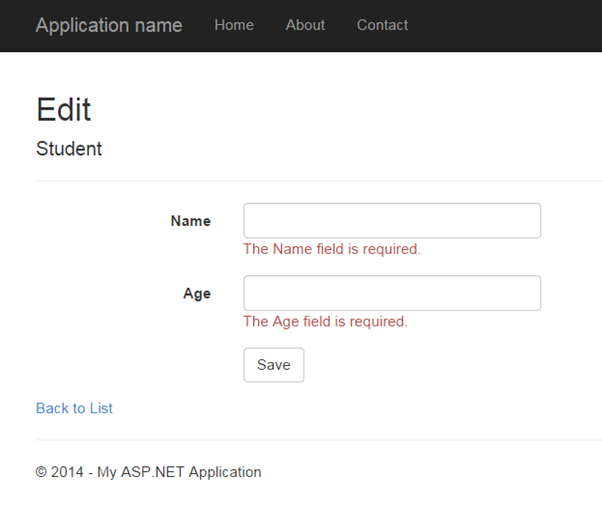
<div>

@Html.ActionLink("Back to List", "Index")

</div>

As you can see in the above Edit.cshtml, it calls Html Helper method **ValidationMessageFor** for every field and **ValidationSummary** method at the top. ValidationMessageFor is responsible to display error message for the specified field. ValidationSummary displays a list of all the error messages at once.

So now, it will display default validation message when you submit an Edit form without entering a Name or Age.

[](https://www.tutorialsteacher.com/Content/images/mvc/validation-editview.png)Validation

Thus, you can implement validations by applying various DataAnnotation attributes to the model class and using ValidationMessage() or ValidationMessageFor() method in the view.

Learn how to [implement client side validation in ASP.NET MVC](https://www.tutorialsteacher.com/articles/enable-client-side-valiation-in-mvc).

 Points to Remember :

1. ASP.NET MVC uses DataAnnotations attributes for validation.
2. DataAnnotations attributes can be applied to the properties of the model class to indicate the kind of value the property will hold.
3. The following validation attributes available by default
   1. Required
   2. StringLength
   3. Range
   4. RegularExpression
   5. CreditCard
   6. CustomValidation
   7. EmailAddress
   8. FileExtension
   9. MaxLength
   10. MinLength
   11. Phone
4. Use **ValidationSummary** to display all the error messages in the view.
5. Use **ValidationMessageFor** or **ValidationMessage** helper method to display field level error messages in the view.
6. Check whether the model is valid before updating in the action method using ModelState.IsValid.
7. Enable client side validation to display error messages without postback effect in the browser.

ASP.NET MVC: ValidationMessage

You have learned how to implement validation in a view in the presious section. Here, we will see the HtmlHelper extension method ValidtionMessage in detail.

The Html.ValidationMessage() is an extension method, that is a loosely typed method. It displays a validation message if an error exists for the specified field in the ModelStateDictionary object.

ValidationMessage() Signature

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

Visit MSDN to know all the [overloads of ValidationMessage() method](https://msdn.microsoft.com/en-us/library/system.web.mvc.html.validationextensions.validationmessage(v=vs.118).aspx).

Example: ValidationMessage

@model Student

@Html.Editor("StudentName") <br />

@Html.ValidationMessage("StudentName", "", new { @class = "text-danger" })

In the above example, the first parameter in the ValidationMessage method is a property name for which we want to show the error message e.g. StudentName. The second parameter is for custom error message and the third parameter is for html attributes like css, style etc.

The ValidationMessage() method will only display an error, if you have configured the DataAnnotations attribute to the specifed property in the model class. The following is a Student model class where the DataAnnotations attribute "Required" is applied to the StudentName property.

Example: Student Model

public class Student

{

public int StudentId { get; set; }

[Required]

public string StudentName { get; set; }

public int Age { get; set; }

}

The above code will generate following html.

<input id="StudentName"

name="StudentName"

type="text"

value="" />

<span class="field-validation-valid text-danger"

data-valmsg-for="StudentName"

data-valmsg-replace="true">

</span>

Now, when the user submits a form without entering a StudentName, then ASP.NET MVC uses a data- attribute of Html5 for the validation and a default validation message will be injected, when the validation error occurs, as shown below.

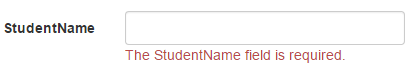
Html with Validation message:

<span class="field-validation-error text-danger"

data-valmsg-for="StudentName"

data-valmsg-replace="true">The StudentName field is required.</span>

The error message will look like below.

[](https://www.tutorialsteacher.com/Content/images/mvc/validationmessage.png)Output of ValidationMessage() method

Custom Error Message

You can display your own error message instead of the default error message as shown above. You can provide a custom error message either in the DataAnnotations attribute or ValidationMessage() method.

Use the parameter of the DataAnnotation attributes to provide your own custom error message as shown below.

Example: Custom error message in the 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; }

}

Also, you can specify a message as a second parameter in the ValidationMessage() method as shown below.

Example: Custom error message

@model Student

@Html.Editor("StudentName") <br />

@Html.ValidationMessage("StudentName", "Please enter student name.", new { @class = "text-danger" })

# ASP.NET MVC: ValidationMessageFor

The Html.ValidationMessageFor() is a strongly typed extension method. It displays a validation message if an error exists for the specified field in the ModelStateDictionary object.

## ValidationMessageFor() Signature

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

Visit MSDN to know all the [overloads of ValidationMessageFor() method](https://msdn.microsoft.com/en-us/library/system.web.mvc.html.validationextensions.validationmessagefor(v=vs.118).aspx).

Consider the following ValidationMessageFor() example.

Example: ValidationMessageFor

@model Student

@Html.EditorFor(m => m.StudentName) <br />

@Html.ValidationMessageFor(m => m.StudentName, "", new { @class = "text-danger" })

In the above example, the first parameter in ValidationMessageFor method is a lambda expression to specify a property for which we want to show the error message. The second parameter is for custom error message and the third parameter is for html attributes like css, style etc.

The ValidationMessageFor() method will only display an error if you have configured DataAnnotations attribute to the specifed property in the model class. The following example is a Student model class where the DataAnnotations attribute "Required" is applied to the StudentName property.

Example: Student Model

public class Student

{

public int StudentId { get; set; }

[Required]

public string StudentName { get; set; }

public int Age { get; set; }

}

The above code will generate the following html.

Html Result:

<input id="StudentName"

name="StudentName"

type="text"

value="" />

<span class="field-validation-valid text-danger"

data-valmsg-for="StudentName"

data-valmsg-replace="true">

</span>

Now, when the user submits a form without entering the StudentName then ASP.NET MVC uses the data- attribute of Html5 for the validation and the default validation message will be injected when validation error occurs, as shown below.

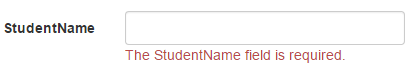
Html with Validation message:

<span class="field-validation-error text-danger"

data-valmsg-for="StudentName"

data-valmsg-replace="true">The StudentName field is required.</span>

The error message will appear as the image shown below.

[](https://www.tutorialsteacher.com/Content/images/mvc/validationmessage.png)Output of ValidationMessageFor() method

## Custom Error Message

You can display your own error message instead of the default error message as above. You can provide a custom error message either in the DataAnnotations attribute or the ValidationMessageFor() method.

Use the ErrorMessage parameter of the DataAnnotation attributes to provide your own custom error message as shown below.

Example: Custom error message in the 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; }

}

Also, you can specify a message as a second parameter in the ValidationMessage() method as shown below.

Example: Custom error message

@model Student

@Html.Editor("StudentName") <br />

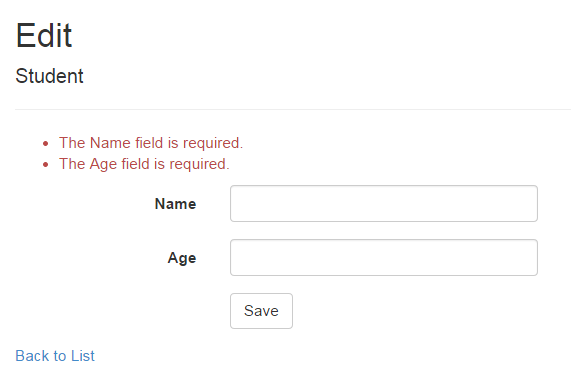
@Html.ValidationMessageFor(m => m.StudentName, "Please enter student name.", new { @class = "text-danger" })

Learn about ValidationSummary method in the next section.

# ASP.NET MVC: ValidationSummary

The ValidationSummary helper method generates an unordered list (ul element) of validation messages that are in the ModelStateDictionary object.

The ValidationSummary can be used to display all the error messages for all the fields. It can also be used to display custom error messages. The following figure shows how ValidationSummary displays the error messages.

[](https://www.tutorialsteacher.com/Content/images/mvc/validationsummary.png)ValidationSummary

## ValidationSummary() Signature

*MvcHtmlString ValidateMessage(bool excludePropertyErrors, string message, object htmlAttributes)*

Visit MSDN to know all the [overloads of ValidationMessage() method](https://msdn.microsoft.com/en-us/library/system.web.mvc.html.validationextensions.validationsummary(v=vs.118).aspx).

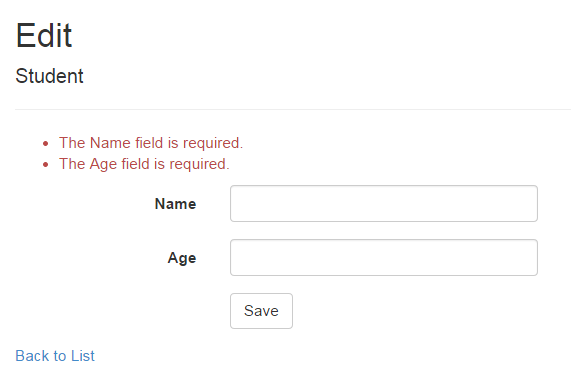
## Display Field Level Error Messages using ValidationSummary

By default, ValidationSummary filters out field level error messages. If you want to display field level error messages as a summary then specify excludePropertyErrors = false.

Example: ValidationSummary to display field errors

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

So now, the following Edit view will display error messages as a summary at the top. Please make sure that you don't have a ValidationMessageFor method for each of the fields.

[](https://www.tutorialsteacher.com/Content/images/articles/validationsummary-demo.png)Error Message using ValidationSummary

## Display Custom Error Messages

You can also display a custom error message using ValidationSummary. For example, we want to display a message if Student Name already exists in the database.

To display a custom error message, first of all, you need to add custom errors into the ModelState in the appropriate action method.

Example: Add error in ModelState

if (ModelState.IsValid) {

//check whether name is already exists in the database or not

bool nameAlreadyExists = \* check database \*

if(nameAlreadyExists)

{

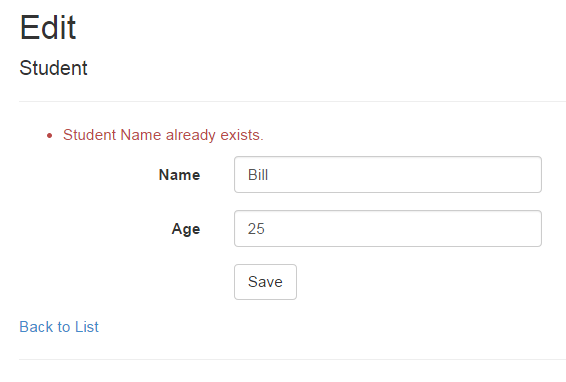
ModelState.AddModelError(string.Empty, "Student Name already exists.");

return View(std);

}

}

As you can see in the above code, we have added custom error messages using the ModelState.AddModelError method. The ValidationSummary method will automatically display all the error messages added into ModelState.

[](https://www.tutorialsteacher.com/Content/images/articles/validationsummary-demo2.png)Display error message using ValidationSymmary

Thus, you can use the ValidationSummary helper method to display error messages.