# How to validate multiple fields in ASP.NET Core

## Introduction

This sample demonstrates how to make a custom validation in ASP.NET Core.

## Sample prerequisites

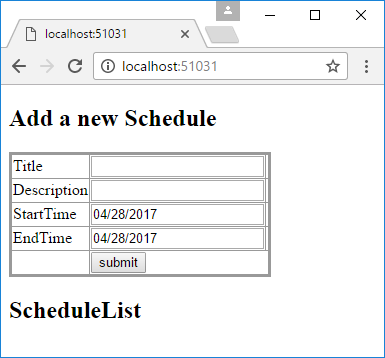
* Visual Studio 2017 or above. [[Visual Studio Home Page](https://www.visualstudio.com/)]
* Visual Studio enabled ASP.NET Core develop component.

## Building the sample

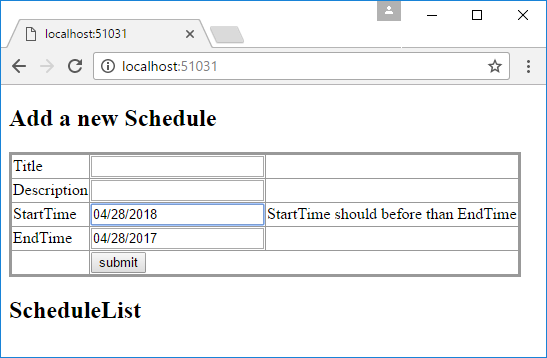
Use Visual Studio to open the sample solution, then press **F6 Key** to build the sample project.

## Running the sample

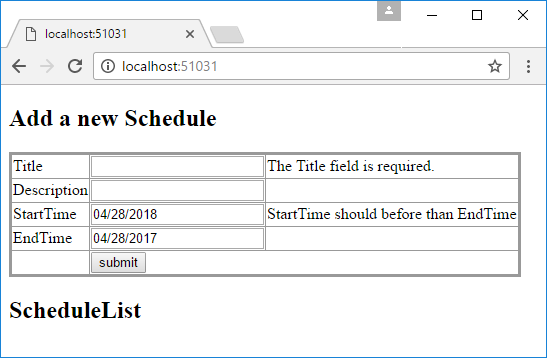
* Use Visual Studio to open the sample solution, then press F5 Key or select **Debug -> Start Debugging** from the menu.
* When the project is running, you will see below page in browser.



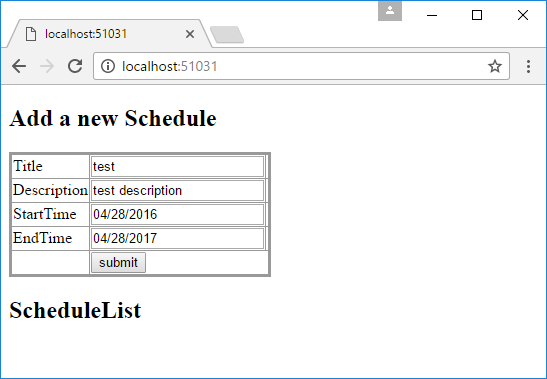
* Set StartTime as a value that earlier than EndTime field.

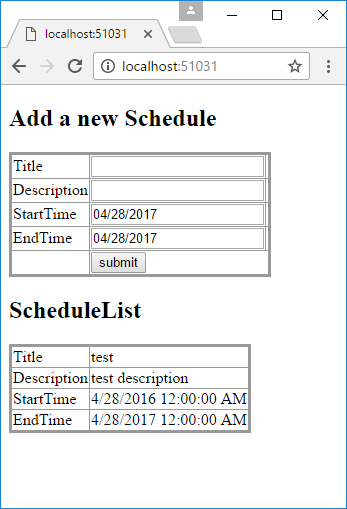


* Set Title as empty, and click submit button.



* Set all filed as valid value, and click button.





## Using the code

**Backend custom validation attribute**

[AttributeUsage(AttributeTargets.Property, AllowMultiple = false)]

public class BeforeThanAttribute : ValidationAttribute, IClientModelValidator

{

public string ComparePropertyName { get; private set; }

public BeforeThanAttribute(string comparePropertyName)

{

ComparePropertyName = comparePropertyName;

}

protected override ValidationResult IsValid(object value, ValidationContext validationContext)

{

var comparePropertyValue = validationContext.ObjectType.GetProperty(ComparePropertyName).GetValue(validationContext.ObjectInstance);

if (value != null && value is DateTime

& comparePropertyValue != null && comparePropertyValue is DateTime)

{

if ((DateTime)value > (DateTime)comparePropertyValue)

{

return new ValidationResult(ErrorMessage);

}

}

return ValidationResult.Success;

}

public void AddValidation(ClientModelValidationContext context)

{

var errorMessage = FormatErrorMessage(context.ModelMetadata.GetDisplayName());

MergeAttribute(context.Attributes, "data-val-beforthan", errorMessage);

MergeAttribute(context.Attributes, "data-val-beforthan-other", ComparePropertyName);

}

private bool MergeAttribute(IDictionary<string, string> attributes, string key, string value)

{

if (attributes.ContainsKey(key))

{

return false;

}

attributes.Add(key, value);

return true;

}

}

**Use backend custom validation attribute**

public class Schedule

{

[Required]

public string Title { get; set; }

public string Description { get; set; }

[Required]

[BeforeThan(nameof(EndTime),ErrorMessage = "StartTime should before than EndTime")]

public DateTime StartTime { get; set; }

[Required]

public DateTime EndTime { get; set; }

}

Enable validation in frontend

<script src="~/lib/jquery.js"></script>

<script src="~/lib/jquery.validate.js"></script>

<script src="~/lib/jquery.validate.unobtrusive.js"></script>

<script>

(function () {

function getModelPrefix(fieldName) {

return fieldName.substr(0, fieldName.lastIndexOf(".") + 1);

}

function appendModelPrefix(value, prefix) {

if (value.indexOf("\*.") === 0) {

value = value.replace("\*.", prefix);

}

return value;

}

function escapeAttributeValue(value) {

return value.replace(/([!"#$%&'()\*+,./:;<=>?@\[\\\]^`{|}~])/g, "\\$1");

}

$.validator.addMethod('beforthan', function (value, element, params) {

var currentValue = new Date(value);

var compareValue = new Date(params.other.value);

return currentValue <= compareValue;

});

$.validator.unobtrusive.adapters.add("beforthan", ["other"], function (options) {

var prefix = getModelPrefix(options.element.name),

other = options.params.other,

fullOtherName = appendModelPrefix(other, prefix),

element = $(options.form).find(":input").filter("[name='" + escapeAttributeValue(fullOtherName) + "']")[0];

options.rules["beforthan"] = {

other: element

};

options.messages["beforthan"] = options.message;

});

}());

## </script>

## More information

Validating Dependent Fields in ASP.NET MVC

<http://www.dotnetspeak.com/asp-net-mvc/validating-dependent-fields-in-asp-net-mvc/>