How are we doing? Please help us improve Stack Overflow. **Take our short survey**

Data validation attribute for a condition between two properties asp.net mvc

Asked 5 years, 5 months ago Active 5 years, 5 months ago Viewed 12k times

9

4

I want to put a rule between two properties that is one property have to be greater than the other. So what is the data validation attribute that can let me do this ?

Here are my properties

```
public int Min{get;set;}
public int Max{get;set;}
```

As you can easily understand Max have to be greater than Min.

Thank you for your help!

c# asp.net asp.net-mvc

edited Sep 9 '14 at 15:17

AlexC
9,705 4 31 54

asked Sep 5 '14 at 9:05

intern
195 1 1 13

1 use jquery instead of model validation – Kartikeya Khosla Sep 5 '14 at 9:06

I prefer model validation I don't know how to handle Jquery – intern Sep 5 '14 at 9:19

Okk..but as far as i understand mvc its difficult to get your functionality only way is to make as custom validation attribute but with jquery its very easy.. –

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

these Min and Max are textboxes???? – Kartikeya Khosla Sep 5 '14 at 10:00

4 Answers

21

Data validations on your object strike me as a good thing (as well as using client side validation).

This is an attribute that you can use to do what you are asking (which will be able to compare pairs of types that implement IComparable)

```
public class GreaterThanAttribute : ValidationAttribute
{

    public GreaterThanAttribute(string otherProperty)
        : base("{0} must be greater than {1}")
    {
        OtherProperty = otherProperty;
    }

    public string OtherProperty { get; set; }

    public string FormatErrorMessage(string name, string otherName)
    {
        return string.Format(ErrorMessageString, name, otherName);
    }

    protected override ValidationResult
        IsValid(object firstValue, ValidationContext validationContext)
    {
        var firstComparable = firstValue as IComparable;
        var secondComparable = GetSecondComparable(validationContext);

        if (firstComparable != null && secondComparable != null)
        {
            if (firstComparable.CompareTo(secondComparable) < 1)
            {
                object obj = validationContext.ObjectInstance;
                var thing = obj.GetType().GetProperty(OtherProperty);
                var displayName = (DisplayAttribute)Attribute.GetCustomAttribute(thing,
typeof(DisplayAttribute));
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
displayName.GetName()));
            }
        }

        return ValidationResult.Success;
    }

    protected IComparable GetSecondComparable(
        ValidationContext validationContext)
    {
        var propertyInfo = validationContext
                           .ObjectType
                           .GetProperty(OtherProperty);
        if (propertyInfo != null)
        {
            var secondValue = propertyInfo.GetValue(
                validationContext.ObjectInstance, null);
            return secondValue as IComparable;
        }
        return null;
    }
}
```

You can then decorate your model:

```
public int Min{get;set;}

[GreaterThan("Min")]
public int Max{get;set;}
```

This is a useful question regarding less than validations MVC custom validation: compare two dates but applies to dates rather than integers but the same approach applies

edited May 23 '17 at 12:32
Community ♦
1 1

answered Sep 5 '14 at 11:55
AlexC
9,705 4 31 54

I personally like this method. Even though these days, everyone probably has javascript enabled, it's still best practice to perform validation on both client and server. If you're only going to do it in one place, I would still do it server-side. You should probably do this AND the JQuery solution below this answer. – Mike C. Sep 5 '14 at 12:11

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

5

You could use a Attribute or your view model could implement IValidatableObject. What's nice is that the asp.net mvc modelbinder will automatically run this on post.

```
public class TestCompareModel : IValidatableObject
{
    [Required]
    public Int32 Low { get; set; }

    [Required]
    public Int32 High { get; set; }

    public IEnumerable<ValidationResult> Validate(ValidationContext validationContext)
    {
        var results = new List<ValidationResult>();

        if (High < Low)
            results.Add(new ValidationResult("High cannot be less than low"));

        return results;
    }
}
```

Controller action:

```
    [HttpPost]
    public ActionResult Test(TestCompareModel viewModel)
    {
        if (!ModelState.IsValid)
            return View(viewModel);

        return RedirectToAction("Index");
    }
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
@model Scratch.Web.Models.TestCompareModel

@{
    ViewBag.Title = "Test";
}

<h2>Test</h2>

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

    <div class="form-horizontal">
        <h4>TestCompareModel</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.Low, htmlAttributes: new { @class = "control-
label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Low, new { htmlAttributes = new { @class
= "form-control" } })
                @Html.ValidationMessageFor(model => model.Low, "", new { @class = "text-
danger" })
            </div>
        </div>

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

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
    @Html.ActionLink("Back to List", "Index")
</div>
```

answered Sep 5 '14 at 13:16

Fran
5,673 1 19 33

This should be marked as the answer! It conforms to DRY and it generates validation code on the client side. I'm OK with requiring a POST as this kind of validation is more business/domain validation. Also much less code to write (no new attribute class or adding logic to the view). – ShooShoSha Aug 10 '17 at 4:25

1

I agree with Exception, JQuery is much easier to work with than the model itself for accomplishing this kind of functionality. However that being said with no experience in Javascript/Jquery its worth having a look at the documentation for JQuery here.

You will also find great tutorials here

And the most important part, the actual JQuery library file here. You can download the file and include it in your solution OR simply include a link to a server hosted CDN version of the file in the header of your view. (both options have instructions provided on the link I gave you)

An update to Exceptions answer however, you do not include the functionality required to only allow integer values in the input controls. To fix this simply change the inputs type attribute to "number" like this.

```
<input type="number" id="Max" name="Max"  />
```

And modify the Script to remove parsing of String to Integer like this:

```
$("#Max").focusout(function(){
      if( $(this).val() < $("#Min").val() )
      {
         $("#errormess").html('Max value cannot be lower then min Value');
      }
      else{ $("#errormess").html(''); }
   });

   $("#Min").focusout(function(){
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
    }
    else{ $("#errormess").html(''); }
});
```

edited Sep 5 '14 at 11:42

answered Sep 5 '14 at 11:23

Master Yoda

3,612 6 30 61

Thank you very much for your help!! – intern Sep 5 '14 at 14:38

1

The functionality which you require can be easily achieved using Jquery as shown :-

HTML :-

```
<input type="text" id="Max" name="Max"  />  //with model validations just make sure user
can input numbers in Max and Min textboxes.
<input type="text" id="Min" name="Min" />
<div id="errormess"></div>
```

Jquery :

```
$(document).ready(function(){
   $("#Max").focusout(function(){
      if(parseInt($(this).val()) < parseInt($("#Min").val()))
      {
         $("#errormess").html('Max value cannot be lower then Min Value');
      }
      else{ $("#errormess").html(''); }
   });

   $("#Min").focusout(function(){
      if(parseInt($(this).val()) > parseInt($("#Max").val()))
      {
         $("#errormess").html('Max value cannot be lower then Min Value');
      }
      else{ $("#errormess").html(''); }
   });
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

edited Sep 5 '14 at 11:50

answered Sep 5 '14 at 10:43

Kartikeya Khosla

17.4k 8 35 60

Thank you for your help Exception it is more clear now! – intern Sep 5 '14 at 14:39

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.