MVC Form Validation on Multiple Fields

Asked 8 years, 9 months ago Active 2 years, 6 months ago Viewed 19k times

20

8

How would I go about having multiple textboxes on an MVC 3 form treated as one for the purposes of validation?

It's a simple phone number field with one textbox for area code, one for prefix and one for the last four digits.

There are really two validation requirements:

1) They're all required. 2) They must all contain integers.

Now this is simple when doing it for individual fields but how can I create the equivalent of an ASP.NET CustomValidator with MVC so that I can validate all three fields as a whole?

c# asp.net-mvc validation forms asp.net-mvc-3

asked May 20 '11 at 17:06
Scott
12.7k 20 80 139

2 My comment doesn't address your validation question, but I am wondering whether this is the best UI to use for phone number entry. Isn't this design more work for the user; tab-tabing between the textboxes, more work if they want to paste a phone number in or delete an already-entered phone number. I'd have thought that a single textbox with regex-based validation would be more usual. – Richard Ev May 20 '11 at 17:26

1 If I were making the decisions, I'd say you're right, but for some reason someone's decided it needs to be three text boxes. Don't ask. :) – Scott May 20 '11 at 17:30

Ouch. Can you sway their opinion with the examples I've cited? Also, I'm presuming that you don't need any non-US people to enter their phone numbers into this system...as ours are often formatted differently. – Richard Ev May 20 '11 at 17:46

3 Answers

21

I actually ended up implementing a custom `ValidationAttribute` to solve this, using the same type of logic presented in `CompareAttribute` that allows you to use reflection to evaluate the values of other properties. This allowed me to implement this at the property level instead of the model level and also allows for client side validation via unobtrusive javascript:

```
public class MultiFieldRequiredAttribute : ValidationAttribute, IClientValidatable
    {
        private readonly string[] _fields;

        public MultiFieldRequiredAttribute(string[] fields)
        {
            _fields = fields;
        }

        protected override ValidationResult IsValid(object value, ValidationContext
validationContext)
        {
            foreach (string field in _fields)
            {
                PropertyInfo property = validationContext.ObjectType.GetProperty(field);
                if (property == null)
                    return new ValidationResult(string.Format("Property '{0}' is
undefined.", field));

                var fieldValue = property.GetValue(validationContext.ObjectInstance,
null);

                if (fieldValue == null || String.IsNullOrEmpty(fieldValue.ToString()))
                    return new
ValidationResult(this.FormatErrorMessage(validationContext.DisplayName));
            }

            return null;
        }

        public IEnumerable<ModelClientValidationRule>
GetClientValidationRules(ModelMetadata metadata, ControllerContext context)
        {
            yield return new ModelClientValidationRule
            {
                ErrorMessage = this.ErrorMessage,
                ValidationType = "multifield"
            };
        }
    }
```

Scott
12.7k 20 80 139

I had to change the ValidationType to "required" to have client-side validation work. I found this answer(stackoverflow.com/questions/13740489/…) to be helpful with the ValidationType accepted values. – gmasselli Sep 27 '16 at 14:54

18

You could handle this by putting `IValidatableObject` on the model class, and implementing the `Validate` method.

It could look something like this:

```
public IEnumerable<ValidationResult> Validate(ValidationContext validationContext)
{
      if (String.IsNullOrEmpty(_PhonePart1) || String.IsNullOrEmpty(_PhonePart2)
            || String.IsNullOrEmpty(_PhonePart3))
      {
           yield return new ValidationResult("You must enter all " +
                  "three parts of the number.");
      }

}
```

edited Aug 24 '17 at 18:24
user1477388
17.6k 22 111 214

answered May 20 '11 at 17:45
Thomas
1,154 1 11 23

1

Scott, Is using a custom model binder within your scope? If so, you can implement an IModelBinder to combine the results from the three text fields into a single phone number field, which can be decorated with validation attributes. Here's a stackoverflow question that has an example of how to do this: DataAnnotation Validations and Custom ModelBinder

edited May 23 '17 at 11:46

community wiki
2 revs
Shashi Penumarthy