DataAnnotations on Model or ViewModel? [Answered]

7 replies

Last post Jan 09, 2011 12:33 PM by francesco abbruzzese

DataAnnotations on Model or ViewModel?

Dec 09, 2009 06:23 PM | darmak

Hi everybody,

Where should I place DataAnnotation attributes like Required, StringLength, etc - in my **Model** classes or in my **ViewModel** classes?

Which approach is better?

MODEL DATAANNOTATIONS VIEWMODEL

Re: DataAnnotations on Model or ViewModel?

Dec 09, 2009 08:33 PM | CodeHobo

Generally those would go in your Model classes since that's where the actual data lives. The ViewModel is just the wrapper around different kinds of model objects. While you can add data annotation validation to the ViewModel (such as in the case of there being something specific you want to check for) but I think that should be in addition to the Model and not instead of.

Re: DataAnnotations on Model or ViewModel?

Dec 10, 2009 02:30 AM | bradwils

I am a believer that these things belong on your view model.

They are describing what is essential user interaction: form generation and validation. They don't necessarily replace any business layer validation you have in place; they supplement it, in UI-specific form.

Re: DataAnnotations on Model or ViewModel?

Dec 10, 2009 07:03 AM | Augi

I agree with Brad.

If you are using DataAnnnotation in your Model (business layer) then you can use these classes in your ViewModels and validation will work well for you. Sure, you must use some "DataAnnotation checker" in your business layer too. So this is first scenario - you are using DataAnnotation in your Model and validation rules are bubbling to ViewModel. But...if mapping between Model validation and ViewModel validation is different (it's not the same) then it's hacky to change default behaviour. So I recommend to write ViewModel classes and use DataAnnotation attributes here.

If you are not using DataAnnotation in your Model then it depends on how validation rules are described in your business layer. You can implement own ModelBinder (that will execute your custom validation) and ModelMetadata provider that will be able to perform your validation (i.e. will read validation rules from XML, database, custom attributes, ...). Again, you must perform validation in your business layer too. Validation in presentation layer (ASP.NET MVC) is improvement of user interface only and business layer validation can be different.

So my suggestion is to create ViewModels and use DataAnnotation here. The reason is that "business layer validation" and "UI validation" can be different (i.e. UI validation is more precise).

Re: DataAnnotations on Model or ViewModel?

Dec 10, 2009 01:43 PM | darmak

Thank you very much for all the replies!

DataAnnotations on ViewModels seemed also better for me, but then the validation is inside a ModelBinder and not in my "service layer". I know that in the ModelBinder would be some sort of UI-only-validation and the rest ("business rules") would remain in Services (and be called from a Controller), but still - checking

that int is in a specific range **can be** a business rule in some domains, so it would be nice to check that only once.

My understanding is that a Model should be independent, stand-alone, etc. Marking it with DataAnnotation attributes makes it dependent. Should I be that stubborn to avoid any dependencies or is it acceptable in this case?

It is a good point to have UI- and business- validations, but I think I would have to duplicate logic between them very often.

Re: DataAnnotations on Model or ViewModel?

Apr 01, 2010 12:54 PM | jkar

This is interesing. If my ViewModel looks like this:

```
public class DinnerViewModel()
{
  public Dinner Dinner { get; set }
  public SelectList Countries { get; set; }
}
```

How can I put the validation of the Dinner class' properties in my ViewModel rather than the Model class?

Re: DataAnnotations on Model or ViewModel?

Jan 08, 2011 05:54 PM | c9806103

You can add the following to the top of your ViewModel:

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
```

And then lay the DataAnnotation attributes directly over each property you need to validate. Such as:

```
[DisplayName("Account Status")]
[Required(ErrorMessage = "Acount Status is required")]
public int StatusID { get; set; }
```

However where I run into limitations is the scenario where you need two different validation rule sets for different situations. For example a public-facing www site and an internal admin site.

One the www site all fields are mandatory, but on the admin site, some fields are not mandatory.

There is no nice/flexible way to be able to create two validation rule sets and specify which one to apply over the same object depending on the situation.

You can create two different ViewModels no problems, but you run into difficulties when needing to retrieve the "object to be edited" from the database and then having the manually populate each and every value inside the ViewModel manually by hand. I.e. you can no longer use model binding to do it because your ViewModel is now a different class than the object that you are ultimately trying to manipulate. Some suggest AutoMapper for this task etc, but it doesn't work with my auto-generated LINQ to SQL classes.

DataAnnotations needs to be more flexible to allow you to specify which validation rule set to apply in which situation in order to be useful and practical.

Re: DataAnnotations on Model or ViewModel?

Jan 09, 2011 12:33 PM | francesco abbruzzese

In my opinion there is NOT A GENERAL RULE ! It dependeds on the attribute and on the situation. I am referring mainly to the validation attributes that are the ones that may give more "doubts". Moreover, one need to distinguish the PLACE they are defined and WHEN they are checked. About the WHEN they may be checked more than once in the different layers for security related issues.

About WHERE defining them I do some examples to clarify:

1) The Required attribute may be applied and ALSO DUPLICATED on both Model and View Model. The Model class may need a property as "Required" but this doesn't mean it is required also in ALL View Models connected with the original Model because the property might be taken from a different place than user interaction.

2) Whole model level properties CANNOT be checked in general in the View Model for the same reason of the Required attribute, that is because The View Model doesn't contain necessarily all information that will be transferred to the Model. On the other side a whole model level constraints might be applied ONLY to the View Model because it translates a constraint that is specific of a user-machine interaction.

3) Property level constraints like for instance the constraint on an email field for sure ARE THE SAME both on the MODEl and the VIEW MODEL, because they are a kind of specialization of the general DataType "string". The same applies to SOME range constraints such as the one saying that a % nedd to be between 0 and 100. I prefer defining SUCH CONSTRAINTS in a metaclass that I use for both the model and the View Model in order to avoid duplicating the constraint.

4) On the others side there are property level constraints that can be applied ONLY ON the View model. The typical instance is that a date be in the future. This is a concept connected to the time the user interaction takes place and is difficult to be defined on the DB layer, since the concept of NOW...has nothing to do with data storage.