Should data annotations be on the Model or the View Model?

Asked 9 years, 2 months ago Active 9 years, 2 months ago Viewed 3k times

11

7

I've been used to decorating data model classes with data annotation attributes, but the purist in me baulks slightly at including purely presentational attributes such as display format here. I am, however, quite happy to keep validation centric attributes here. One good reason I have to continue keeping all annotations etc. in the data model is that my view model aggregates data model classes, e.g. my `ViewModelBase.DetailItem<TEntity>` property in the view model is just a reference to an entity class in my data model. If I wanted to move presentational annotations to the view model, I would have to quite radically revise my design to one where I duplicate data model properties in my view model and use an object mapping tool to populate view model objects based on data model objects.

Where should I be doing my data annotations?

Just BTW, this is what my rough draft `ViewModelBase` looks like:

```
public class ViewModelBase<T>
{
    public virtual string PageTitle { get; set; }
    public virtual string ViewHeading { get; set; }

    public virtual ViewMode ViewMode { get; set; }
    public virtual IEnumerable<T> ItemList { get; set; }
    public virtual T DetailItem { get; set; }
}
```

asp.net asp.net-mvc asp.net-mvc-3

edited Dec 15 '10 at 17:47

asked Dec 15 '10 at 16:13
ProfK
41.2k 97 336 671

Also check out: stackoverflow.com/questions/3338919/… – DarrellNorton Dec 15 '10 at 18:45

2 Answers

7

I share the same concern around the DRY principal and validation, which is why I prefer to keep most validation requirements in the model. But why does it have to be one or the other? Model Validation belongs in the model, but there are certain view-specific validations which belong in the viewmodel.

That being said, data annotations are just that: annotations around data. Not proper validation logic. Validation logic is a completely different concept to data annotations (A Required attribute is merely one aspect of validation). I personally find it difficult to place where real validation fits within a MVVM implementation as some validation requires context rather than just required or not.

Short answer: If it's in your Model then it gets aggregated to your viewmodels. If there is a view specific requirement, the viewmodel can cater for additional requirements if needed.

answered Dec 15 '10 at 18:40

WilliamB
154 1

Hey @William, long time no read :-) I like what you're saying about validation being aggregated to the viewmodels, but I have some hurdles in adding extra validation to aggregate type viewmodels. I'll see if I come up with something funky later tonight. – ProfK Dec 15 '10 at 18:56

Perhaps the viewmodels aren't being abstracted far enough away from the models (or conversely, aren't being abstracted close enough to the views) to make the decision easy? I still believe validation isn't specific to views. There might be context involved (e.g. A name is required in a hospital system. Except for newborns where no name is applicable yet) but validation logic and reasoning is rarely applicable to a specific view only. – WilliamB Dec 15 '10 at 19:24

I wonder who voted you down here. Really not called for if you ask me. – ProfK Dec 15 '10 at 19:52

12

Validation should at least be performed at the view model because this is what you receive from the view. Also the validation is always performed in the context of a given view. So you could have two different view models corresponding to two different views but mapped to a single model class and because the validation could be different depending on the view, this validation should be performed on the view model. If you performed the validation on the model then you would have hard time to distinguish between the two cases, because you could have a situation where a property is required in the first view, but not required on the second view. So if you are using data annotations to perform validation then you should decorate your view model with them.

edited Dec 15 '10 at 17:55

answered Dec 15 '10 at 17:50

Darin Dimitrov
900k 236 3104 2809

good point about the two different validations, but it seems to consider the model more of a DTO than a full domain model with business logic to cater for different use cases, where each use case will be catered for by a different view & view model. – ProfK Dec 15 '10 at 18:24