What's the difference between Model, ViewModel and DTO? [closed]

Asked 6 years, 4 months ago Active 6 years, 4 months ago Viewed 3k times

4

3

Closed. This question is off-topic. It is not currently accepting answers.

Want to improve this question? Update the question so it's on-topic for Stack Overflow.

Closed 6 years ago.

I am an ASP.NET MVC developer and I am confused. What's the difference between Model, ViewModel and DTO (Data Transfer Object)? Is it ok for model to have methods that will save itself to database?

asp.net-mvc model-view-controller

asked Nov 13 '13 at 8:29

Art Drozdov
127 1 9

it's not a question of coding it's a question of concept of coding – Art Drozdov Nov 14 '13 at 14:02

In my opinion there is nothing wrong with your question as I had the same doubt. – Mohit Shah Mar 11 '16 at 7:58

1 Answer

11

DTO is an object for passing data in case of communication between layers. It's a general pattern that is not tied to ASP.NET MVC.

ViewModel contains a data specific to particular view, is passed to that view in controller and is used in the view for rendering. It's a pattern specific to ASP.NET MVC (don't mix up with ViewModel from MVVM - they are different)

Model is a set of objects that represent your business domain. It can contain methods that will save it to DB depending of what pattern you will choose to build it (something like Active Record in your case).

answered Nov 13 '13 at 8:39

Oleksandr Kobylianskyi
2,826 15 31

Thank you! Can ViewModel have something like Init() method to retrieve initial data from database? – Art Drozdov Nov 13 '13 at 8:43

I don't entirely agree - a ViewModel in MVC and MVVM is essentially the same thing. You may have different methods, and technically MVC is stateless so you don't *really* have a persistent viewmodel, but you are still modelling the same thing - the logic behind a particular area of functionality. I would also say that it's entirely possible to have multiple views on the same viewmodel (and sometimes desirable) - I know you didn't directly say that this **isn't** the case, but just wanted to point it out. – Charleh Nov 13 '13 at 8:47

1 It also might be worth mentioning that the `Model` part of MVC has different meanings to different people. I always use a ViewModel which wraps the actual `Model` (which is usually an entity framework object or a repository), so for me it really should be called VMMVC or something! Depending on complexity, the model could be the DTO or an entity class (bad, but it works fine for some simple views) or something of that nature. Obviously, it makes sense to structure your project as best as possible (the more layered the better usually) but it really does depend on project size/scope/developer. – Charleh Nov 13 '13 at 8:51