What should you name your controller in MVC? When should you create a new one? [closed]

Asked 11 years, 3 months ago Active 2 years, 3 months ago Viewed 6k times

9

2

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 4 years ago.

I have a question that really applies to any MVC framework, I'm using the Zend Framework MVC.

When exactly should you create a new controller? What exactly should the Controller layer define?

I've created several apps with the MVC, progressively becoming more reusable, but I've always struggled with naming Controller classes. For the most part it matches whatever URL requests there are, so business/front end logic. But in some cases it seems totally arbitrary.

Does anybody have some heuristics/guidelines to follow? Seems like with all the hype about MVC, especially with PHP, there is little data on actual conventions and heuristics. As it's pretty easy to create a disorganized MVC application...

model-view-controller controller heuristics

edited Feb 24 '09 at 0:15

GEOCHET

20.1k ● 15 ● 68 ● 96

asked Feb 24 '09 at 0:13

AndreLiem

1,891 ● 5 ● 19 ● 26

2 Answers

Active Oldest Votes

I generally have one controller for each logical group of functions. Often this will correspond with one controller per model, sometimes not.

10

Imagine you're creating a simple online catalog that displays a list of categories then when the user selects a category, displays a list of products from that category, along with an admin panel for `CRUD` operations on categories and products. I'd have two models (`CategoryModel` and `ProductModel`). I'd have a controller which generated the category listings for the front end, and another controller which generated the product listings (e.g. `CategoryController` and `ProductController`). I'd then have a controller for categories and products on the back end (`AdminCategoryController` and `AdminProductController`). The two back end controllers would handle list/add/edit/delete/view operations for their respective models. If you think though your URL structure and put related functions on related urls, then your controller structure will often match your URL structure. In fact some frameworks (e.g. CodeIgniter) route requests based on the name of the controller as default behavior.

As for what goes in the controllers, I work in the idea that Models are for data access, and wrap and hide the database structure. Logic such as "assign the current time to completion_date when status is set to 'complete'" is a great fit models.

Views contain the entirety of your presentation. Controllers/Models shouldn't generate or handle HTML. Decisions such as 2 columns or 3 belong in views. Logic in views should be restricted to that which is required to generate the visible output. If you find yourself wanting to query the database from a view, you're probably putting too much logic into the view.

Controllers are for what's left. Generally that means, validating input, assigning form data to models, selecting the right views and instantiating the models required to handle the request.

edited Feb 18 '18 at 9:45

AmiNadimi

2,670 ● 1 ● 23 ● 39

answered Feb 24 '09 at 6:59

Jim OHalloran

5,589 ● 2 ● 33 ● 54

Thanks.... that's pretty much what I'm doing. One thing I'm trying to do is put more logic into the model layer. I use propel model objects, and was thinking that the validation should go into the model layer. The controller just sets the data in the model... – AndreLiem Feb 24 '09 at 19:09

1 Some developers prefer to put all validation into Models. I find that form validation is better done in the Controller (because it's tightly coupled to the UI), and basic data type validation (e.g. constraining an enum field to certain values) works well in a model. – Jim OHalloran Feb 25 '09 at 0:14

1

For the most part I follow the controller-per-model pattern. I have a few controllers that may serve multiple models (like the Admin controller which serves several administrative models), but the general rule is one controller per business model.

answered Feb 24 '09 at 1:15

tvanfosson

467k ● 91 ● 666 ● 765