Where to convert business model to view model?

asp.net-mvc c# entity-framework repository-pattern unit-of-work

English (en)

Question

In my ASP.NET MVC application, I am using unit of work and repository patterns for data access.

Using the unit of work class and the repository defined inside it I am fetching the related set of entities in my controller. With my beginner knowledge, I can think of two ways to fetch the business model and convert it to view model.

- Repository returns the business model to controller, this model than mapped to view model, or
- Repository itself converts business model to view model and then it is returned to controller.

Currently I am using first approach, but my controller code started to look ugly and long for view models with lots of properties.

On the other hand, I am thinking, since my repository is called UserRepository (for example), it should be returning the business model directly, instead of some model that is useful only for single view.

Which one of these do you think is better practice for large projects ? Is there an alternative way ?

emre nevayeshirazi

Accepted Answer

Repositories should return domain models, not view models. As far as the mapping between the models and the view models is concerned, personally I use AutoMapper so I have a separate mapping layer but this layer is called from the controller.

Here's how a typical GET controller action might look like:

```
public ActionResult Foo(int id)
{
    // the controller queries the repository to retrieve
a domain model
    Bar domainModel = Repository.Get(id);

    // The controller converts the domain model to a
view model
    // In this example I use AutoMapper, so the
controller actually delegates
    // this mapping to AutoMapper but if you don't have
a separate mapping layer
    // you could do the mapping here as well.
    BarViewModel viewModel = Mapper.Map<Bar,
BarViewModel>(domainModel);

    // The controller passes a view model to the view
    return View(viewModel);
}
```

which of course could be shortened with a custom action filter to avoid the repetitive mapping logic:

```
[AutoMap(typeof(Bar), typeof(BarViewModel))]
public ActionResult Foo(int id)
{
    Bar domainModel = Repository.Get(id);
    return View(domainModel);
}
```

The AutoMap custom action filter subscribes to the OnActionExecuted event, intercepts the model passed to the view result, invokes the mapping layer (AutoMapper in my case) to convert it to a view model and substitutes it for the view. The view is of course strongly typed to the view model.

Darin Dimitrov

Popular Answer

I think that your repository should return the business model.

You then can use a tool like Automapper to automatically map the properties to your viewmodel and can get rid of the manual mapping code. This approach is very usefull if you do not want to expose all of the business entitie's properties or it's comples structure to the view.

You also may find this post helpful, where you can get rid of the manual mapping calls (sort of) and it also provides a good example how to use viewmodels etc (in my opinion)- or get at least some kind of inspiration.

Excerpt from the post (the attribute does the conversion form busioness model to viewmodel):

```
[AutoMap(typeof(Product), typeof(ShowProduct))]
public ActionResult Details(int id)
{
    var product = _productRepository.GetById(id);

    return View(product);
}
```

Bernhard Kircher

View more on Stack Overflow