Multiple models in a view

Asked 9 years, 4 months ago Active 11 months ago Viewed 409k times

300

173

I want to have 2 models in one view. The page contains both `LoginViewModel` and `RegisterViewModel`.

e.g.

```
public class LoginViewModel
{
    public string Email { get; set; }
    public string Password { get; set; }
}

public class RegisterViewModel
{
    public string Name { get; set; }
    public string Email { get; set; }
    public string Password { get; set; }
}
```

Do I need to make another ViewModel which holds these 2 ViewModels?

```
public BigViewModel
{
    public LoginViewModel LoginViewModel{get; set;}
    public RegisterViewModel RegisterViewModel {get; set;}
}
```

I need the validation attributes to be brought forward to the view. This is why I need the ViewModels.

Isn't there another way such as (without the `BigViewModel`):

```
@model ViewModel.RegisterViewModel
@using (Html.BeginForm("Login", "Auth", FormMethod.Post))
{
       @Html.TextBoxFor(model => model.Name)
       @Html.TextBoxFor(model => model.Email)
       @Html.PasswordFor(model => model.Password)
}
```

```
@model ViewModel.LoginViewModel
@using (Html.BeginForm("Login", "Auth", FormMethod.Post))
{
       @Html.TextBoxFor(model => model.Email)
       @Html.PasswordFor(model => model.Password)
}
```

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

edited Jun 21 '19 at 15:00
live-love
25.9k ● 13 ● 111 ● 108

asked Jan 21 '11 at 21:33

Shawn Mclean
51.9k ● 89 ● 260 ● 392

3 see this: codeproject.com/Articles/687061/… – S.Serpooshan May 18 '14 at 9:28

1 @saeed serpooshan, thank you so much for the link with different options, after 4 years you posted a comment and it helped me, i just used `ViewBag` with for each in the view, works great – shaijut Apr 13 '15 at 17:41

@stom Just an FYI: the post author always gets a notification, but if you want to notify someone else, you need to put `@` in front of their name, as I did here. – jpaugh Apr 25 '17 at 21:01

12 Answers

Active | Oldest | Votes

259

There are lots of ways...

1. with your BigViewModel you do:

```
@model BigViewModel
@using(Html.BeginForm()) {
    @Html.EditorFor(o => o.LoginViewModel.Email)
    ...
}
```

2. you can create 2 additional views

   Login.cshtml

```
@model ViewModel.LoginViewModel
@using (Html.BeginForm("Login", "Auth", FormMethod.Post))
{
    @Html.TextBoxFor(model => model.Email)
    @Html.PasswordFor(model => model.Password)
}
```

and register.cshtml *same thing*

after creation you have to render them in the main view and pass them the viewmodel/viewdata

so it could be like this:

```
@{Html.RenderPartial("login", ViewBag.Login);}
@{Html.RenderPartial("register", ViewBag.Register);}
```

or

```
@{Html.RenderPartial("login", Model.LoginViewModel)}
@{Html.RenderPartial("register", Model.RegisterViewModel)}
```

3. using ajax parts of your web-site become more independent
4. `iframes`, but probably this is not the case

edited Oct 4 '13 at 17:40

Cosmo Harrigan
850 ● 7 ● 20

answered Jan 21 '11 at 23:58

Omu
61.4k ● 83 ● 246 ● 391

2 Is it a problem if 2 textboxes have the same name on the form due to using partialviews? – Shawn Mclean Jan 22 '11 at 2:44

2 No, should be fine- click on the element itself using something like firebug (on firefox) and you will see something like id="LoginViewModel_Email" name = "LoginViewModel.Email", so in actual fact they are unique! A view model should be what you need, just post each page to a different URL and you should be fine – Haroon Jan 22 '11 at 9:50

@Lol coder actually it would be 2 forms, one for each viewmodel, but anyway if you would have 2 or 3 or more with same name you would just get an array with that name on the server side (if you put it in the params of the post action method) – Omu Jan 22 '11 at 10:58

@Chuck Norris I am using asp.net mvc 4 and implemented your partialviewresult technique but `@Html.RenderAction` is reporting a error that **Expression must return a value** – Deeptechtons Oct 16 '12 at 4:10

1 Can you explain how this works? I understand this is a solution to the problem, but I can't make any sense of it. I have a similar(slightly different)

example of this problem and I cannot figure out how to get around it. – Andre Feb 7 '18 at 12:23

126

I'd recommend using `Html.RenderAction` and PartialViewResults to accomplish this; it will allow you to display the same data, but each partial view would still have a single view model and removes the need for a `BigViewModel`

So your view contain something like the following:

```
@Html.RenderAction("Login")
@Html.RenderAction("Register")
```

Where `Login` & `Register` are both actions in your controller defined like the following:

```
public PartialViewResult Login( )
{
    return PartialView( "Login", new LoginViewModel() );
}

public PartialViewResult Register( )
{
    return PartialView( "Register", new RegisterViewModel() );
}
```

The `Login` & `Register` would then be user controls residing in either the current View folder, or in the Shared folder and would like something like this:

/Views/Shared/Login.cshtml: (or /Views/MyView/Login.cshtml)

```
@model LoginViewModel
@using (Html.BeginForm("Login", "Auth", FormMethod.Post))
{
    @Html.TextBoxFor(model => model.Email)
    @Html.PasswordFor(model => model.Password)
}
```

/Views/Shared/Register.cshtml: (or /Views/MyView/Register.cshtml)

```
@model ViewModel.RegisterViewModel
@using (Html.BeginForm("Login", "Auth", FormMethod.Post))
{
```

```
    @Html.TextBoxFor(model => model.Name)
    @Html.TextBoxFor(model => model.Email)
    @Html.PasswordFor(model => model.Password)
}
```

And there you have a single controller action, view and view file for each action with each totally distinct and not reliant upon one another for anything.

answered Jan 22 '11 at 21:19

TheRightChoyce
2,906 ● 1 ● 16 ● 17

4 This makes alot of sense in terms of designing, but in terms of efficiency, doesn't it have to go through 3 full cycles of the mvc cycle? stackoverflow.com/questions/719027/renderaction-renderpartial/… – Shawn Mclean Jan 24 '11 at 17:30

6 Yes you're correct: it causes an additional full MVC cycle for each `RenderAction`. I always forget its part of the futures pack since my project always includes that dll by default. Its really up to preference and application requirements as to if the additional mvc cycles are worth the separation it gives on the design side. A lot of times you can cache the `RenderAction` results so the only hit you take is the slight extra processing via the controller factory. – TheRightChoyce Jan 24 '11 at 18:14

I've implemented the above.. what am i missing? Please help: stackoverflow.com/questions/9677818/… – diegohb Mar 13 '12 at 3:02

Holy crap! This worked perfectly for me right out of the gate. I'm building an internal site with only a couple users... so efficiency isn't a real concern of mine. THANK YOU! – Derek Evermore Aug 14 '13 at 15:23

1 Had to use curly brackets to get PartialView working. – null Jan 16 '18 at 15:07

113

Another way is to use:

```
@model Tuple<LoginViewModel,RegisterViewModel>
```

I have explained how to use this method both in the view and controller for another example: Two models in one view in ASP MVC 3

In your case you could implement it using the following code:

In the view:

```
@using YourProjectNamespace.Models;
@model Tuple<LoginViewModel,RegisterViewModel>
```

```
@using (Html.BeginForm("Login1", "Auth", FormMethod.Post))
{
        @Html.TextBoxFor(tuple => tuple.Item2.Name, new {@Name="Name"})
        @Html.TextBoxFor(tuple => tuple.Item2.Email, new {@Name="Email"})
        @Html.PasswordFor(tuple => tuple.Item2.Password, new {@Name="Password"})
}

@using (Html.BeginForm("Login2", "Auth", FormMethod.Post))
{
        @Html.TextBoxFor(tuple => tuple.Item1.Email, new {@Name="Email"})
        @Html.PasswordFor(tuple => tuple.Item1.Password, new {@Name="Password"})
}
```

Note that I have manually changed the Name attributes for each property when building the form. This needs to be done, otherwise it wouldn't get properly mapped to the method's parameter of type model when values are sent to the associated method for processing. I would suggest using separate methods to process these forms separately, for this example I used Login1 and Login2 methods. Login1 method requires to have a parameter of type RegisterViewModel and Login2 requires a parameter of type LoginViewModel.

if an actionlink is required you can use:

```
@Html.ActionLink("Edit", "Edit", new { id=Model.Item1.Id })
```

in the controller's method for the view, a variable of type Tuple needs to be created and then passed to the view.

Example:

```
public ActionResult Details()
{
    var tuple = new Tuple<LoginViewModel, RegisterViewModel>(new LoginViewModel(),new
RegisterViewModel());
    return View(tuple);
}
```

or you can fill the two instances of LoginViewModel and RegisterViewModel with values and then pass it to the view.

edited May 23 '17 at 11:47

Community ♦

1 ● 1

answered Dec 11 '12 at 15:27

Hamid Tavakoli

4,027 ● 1 ● 28 ● 33

This was a great way to handle it, thanks! Did what I needed. – Todd Davis Mar 21 '13 at 22:04

I've tried this, but if I use `EditorFor` or `HiddenFor` (which is ideally what I want to use) the model properties are not set when the `Login1` / `Login2` controller methods are called. Presumably the `@Name=` mapping is being ignored. Does `HiddenFor` require some other trick for this situation? – Gary Chapman Nov 20 '14 at 4:27

1 This will not work at all - you cannot bind to the model when the form is submitted – user3559349 Nov 27 '16 at 21:52

@Hamid Thanks Hamid, for a newbie in MVC, this was the most simplistic answer for me. Thank you. – Harold_Finch May 12 '17 at 10:21

How did you bind the Model when submitting the form? – Mohammed Noureldin Apr 16 '18 at 11:37

28

Use a view model that contains multiple view models:

```
namespace MyProject.Web.ViewModels
{
    public class UserViewModel
    {
        public UserDto User { get; set; }
        public ProductDto Product { get; set; }
        public AddressDto Address { get; set; }
    }
}
```

In your view:

```
@model MyProject.Web.ViewModels.UserViewModel

@Html.LabelFor(model => model.User.UserName)
@Html.LabelFor(model => model.Product.ProductName)
@Html.LabelFor(model => model.Address.StreetName)
```

edited Sep 4 '16 at 22:59

smulholland2

909 ● 2 ● 10 ● 26

answered Feb 21 '14 at 5:47

Yini

671 ● 8 ● 8

1 This is a great solution, and model validation still works with no qualms. Thanks! – AFM-Horizon Oct 23 '19 at 13:36

Do I need to make another view which holds these 2 views?

11

Answer:No

> Isn't there another way such as (without the BigViewModel):

Yes, you can use Tuple (brings magic in view having multiple model).

Code:

```
@model Tuple<LoginViewModel, RegisterViewModel>


   @using (Html.BeginForm("Login", "Auth", FormMethod.Post))
   {
    @Html.TextBoxFor(tuple=> tuple.Item.Name)
    @Html.TextBoxFor(tuple=> tuple.Item.Email)
    @Html.PasswordFor(tuple=> tuple.Item.Password)
   }


   @using (Html.BeginForm("Login", "Auth", FormMethod.Post))
    {
     @Html.TextBoxFor(tuple=> tuple.Item1.Email)
     @Html.PasswordFor(tuple=> tuple.Item1.Password)
    }
```

edited Jan 28 '16 at 16:30
sparkhee93
1,327 ● 3 ● 18 ● 26

answered Jan 28 '16 at 16:07

VCody
480 ● 3 ● 12

Wouldn't this not map properly to the controller that was accepting the form? I thought it would look for "Item" in your case before looking for "name". – SolidSnake4444 May 15 '17 at 4:16

Add this ModelCollection.cs to your Models

7

```
using System;
using System.Collections.Generic;

namespace ModelContainer
```

```
{
  public class ModelCollection
  {
   private Dictionary<Type, object> models = new Dictionary<Type, object>();

   public void AddModel<T>(T t)
   {
      models.Add(t.GetType(), t);
   }

   public T GetModel<T>()
   {
     return (T)models[typeof(T)];
   }
 }
}
```

Controller:

```
public class SampleController : Controller
{
  public ActionResult Index()
  {
    var model1 = new Model1();
    var model2 = new Model2();
    var model3 = new Model3();

    // Do something

    var modelCollection = new ModelCollection();
    modelCollection.AddModel(model1);
    modelCollection.AddModel(model2);
    modelCollection.AddModel(model3);
    return View(modelCollection);
  }
}
```

The View:

```
enter code here
@using Models
@model ModelCollection

@{
  ViewBag.Title = "Model1: " + ((Model.GetModel<Model1>()).Name);
}
```

```
<h2>Model2: @((Model.GetModel<Model2>()).Number</h2>

@((Model.GetModel<Model3>()).SomeProperty
```

edited Aug 27 '16 at 7:01

Gytis Tenovimas

6,334 ● 11 ● 30 ● 48

answered Oct 22 '15 at 14:53

Morten Frederiksen

4,848 ● 1 ● 37 ● 59

I like this approach since it allows me to use different models in the same view without needing to have them intersect. – Matt Small Mar 18 '19 at 12:41

6

a simple way to do that

we can call all model first

```
@using project.Models
```

then send your model with viewbag

```
// for list
ViewBag.Name = db.YourModel.ToList();

// for one
ViewBag.Name = db.YourModel.Find(id);
```

and in view

```
// for list
List<YourModel> Name = (List<YourModel>)ViewBag.Name ;

//for one
YourModel Name = (YourModel)ViewBag.Name ;
```

then easily use this like Model

answered Nov 16 '13 at 16:44

Pnsadeghy
1,171 ● 1 ● 12 ● 19

3

My advice is to make a big view model:

```
public BigViewModel
{
    public LoginViewModel LoginViewModel{get; set;}
    public RegisterViewModel RegisterViewModel {get; set;}
}
```

In your Index.cshtml, if for example you have 2 partials:

```
@addTagHelper *,Microsoft.AspNetCore.Mvc.TagHelpers
@model .BigViewModel

@await Html.PartialAsync("_LoginViewPartial", Model.LoginViewModel)

@await Html.PartialAsync("_RegisterViewPartial ", Model.RegisterViewModel )
```

and in controller:

```
model=new BigViewModel();
model.LoginViewModel=new LoginViewModel();
model.RegisterViewModel=new RegisterViewModel();
```

edited Apr 14 '19 at 20:57

Vega
19.8k ● 13 ● 58 ● 75

answered Apr 14 '19 at 20:25

alin
35 ● 5

2

I want to say that my solution was like the answer provided on this stackoverflow page: [ASP.NET MVC 4, multiple models in one view?]

However, in my case, the linq query they used in their Controller did not work for me.

This is said query:

```
var viewModels =
        (from e in db.Engineers
         select new MyViewModel
         {
             Engineer = e,
             Elements = e.Elements,
         })
        .ToList();
```

Consequently, "in your view just specify that you're using a collection of view models" did not work for me either.

However, a slight variation on that solution did work for me. Here is my solution in case this helps anyone.

Here is my view model in which I know I will have just one team but that team may have multiple boards (and I have a ViewModels folder within my Models folder btw, hence the namespace):

```
namespace TaskBoard.Models.ViewModels
{
    public class TeamBoards
    {
        public Team Team { get; set; }
        public List<Board> Boards { get; set; }
    }
}
```

Now this is my controller. This is the most significant difference from the solution in the link referenced above. I build out the ViewModel to send to the view differently.

```
public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }

            TeamBoards teamBoards = new TeamBoards();
            teamBoards.Boards = (from b in db.Boards
                                 where b.TeamId == id
                                 select b).ToList();
            teamBoards.Team = (from t in db.Teams
                               where t.TeamId == id
                               select t).FirstOrDefault();

            if (teamBoards == null)
```

```
            {
                return HttpNotFound();
            }
            return View(teamBoards);
        }
```

Then in my view I do not specify it as a list. I just do "@model TaskBoard.Models.ViewModels.TeamBoards" Then I only need a for each when I iterate over the Team's boards. Here is my view:

```
@model TaskBoard.Models.ViewModels.TeamBoards

@{
    ViewBag.Title = "Details";
}

<h2>Details</h2>

<div>
    <h4>Team</h4>
    <hr />


    @Html.ActionLink("Create New Board", "Create", "Board", new { TeamId =
@Model.Team.TeamId}, null)
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => Model.Team.Name)
        </dt>

        <dd>
            @Html.DisplayFor(model => Model.Team.Name)
            <ul>
                @foreach(var board in Model.Boards)
                {
                    <li>@Html.DisplayFor(model => board.BoardName)</li>
                }
            </ul>
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.Team.TeamId }) |
    @Html.ActionLink("Back to List", "Index")
</p>
```

I am fairly new to ASP.NET MVC so it took me a little while to figure this out. So, I hope this post helps someone figure it out for their project in a shorter timeframe. :-)

edited May 23 '17 at 11:55
Community ♦
1 ● 1

answered Nov 29 '13 at 19:31
Notso
176 ● 2 ● 5

2

you can always pass the second object in a ViewBag or View Data.

answered Feb 27 '18 at 12:28
Nada N. Hantouli
935 ● 1 ● 9 ● 20

1

1. Create one new class in your model and properties of `LoginViewModel` and `RegisterViewModel`:

```
public class UserDefinedModel()
{
    property a1 as LoginViewModel
    property a2 as RegisterViewModel
}
```

2. Then use `UserDefinedModel` in your view.

edited May 16 '15 at 18:10
Andrew
13k ● 4 ● 42 ● 61

answered Dec 29 '14 at 14:02
user4282639

1 yes, that works for me. then i referenced it like this: (model was declared at the top of the view. There were 2 models inside it: profile, and emailstuff . . . . . @Html.DisplayNameFor(model => model.profile.BlackoutBegin) In the controller I filled one of the models using @notso post below. I didn't need to fill the other becuase I was just using it for an input. – JustJohn Apr 19 '16 at 22:22

This is a simplified example with IEnumerable.

0

I was using two models on the view: a form with search criteria (SearchParams model), and a grid for results, and I struggled with how to add the IEnumerable model and the other model on the same view. Here is what I came up with, hope this helps someone:

```
@using DelegatePortal.ViewModels;

@model SearchViewModel

@using (Html.BeginForm("Search", "Delegate", FormMethod.Post))
{

                Employee First Name
                @Html.EditorFor(model => model.SearchParams.FirstName,
new { htmlAttributes = new { @class = "form-control form-control-sm " } })

                <input type="submit" id="getResults" value="SEARCH" class="btn btn-
primary btn-lg btn-block" />

}
<br />
    @(Html
        .Grid(Model.Delegates)
        .Build(columns =>
        {
            columns.Add(model => model.Id).Titled("Id").Css("collapse");
            columns.Add(model => model.LastName).Titled("Last Name");
            columns.Add(model => model.FirstName).Titled("First Name");
        })
```

...)

SearchViewModel.cs:

```
namespace DelegatePortal.ViewModels
{
    public class SearchViewModel
    {
        public IEnumerable<DelegatePortal.Models.DelegateView> Delegates { get; set; }

        public SearchParamsViewModel SearchParams { get; set; }
....
```

DelegateController.cs:

```
// GET: /Delegate/Search
    public ActionResult Search(String firstName)
    {
        SearchViewModel model = new SearchViewModel();
        model.Delegates = db.Set<DelegateView>();
        return View(model);
    }

    // POST: /Delegate/Search
    [HttpPost]
    public ActionResult Search(SearchParamsViewModel searchParams)
    {
        String firstName = searchParams.FirstName;
        SearchViewModel model = new SearchViewModel();

        if (firstName != null)
            model.Delegates = db.Set<DelegateView>().Where(x => x.FirstName ==
firstName);

        return View(model);
    }
```

SearchParamsViewModel.cs:

```
namespace DelegatePortal.ViewModels
{
    public class SearchParamsViewModel
    {
        public string FirstName { get; set; }
    }
}
```

edited Jun 21 '19 at 15:30

answered Jun 21 '19 at 14:58

live-love

25.9k ● 13 ● 111 ● 108