**When to use Html.RenderPartial and Html.RenderAction in ASP.NET MVC Razor Views**

[Tweet](http://twitter.com/share)

Posted on December 13th, 2010

In this blog post I want to show examples when to use Html.RenderPartial and Html.RenderAction when creating web sites in ASP.NET MVC using the Razor view engine.

**Some background information on Html.RenderAction and Html.RenderPartial**

Html.RenderAction is a somewhat contentious feature because it invokes a controller action.

This means each time it’s called a new request (which has to go through the ASP.NET MVC pipeline) is made.

As shown below ASP.NET MVC requests are routed to a controller. It’s the controller that decides what view to call and what data (model) to pass to it.

A view has one responsibility and that’s to render HTML. It’s supposed to be simple, it doesn’t know (or care) where the data in the model comes from.

The thing MVC traditionalists frown upon is the fact that Html.RenderAction can execute child actions in ASP.NET MVC views.

However given that Html.RenderAction just like an AJAX request is a form of sub request, the case against seems somewhat outdated and doesn’t take into the account using Html.RenderAction will allow you to separate concerns of models and controllers as we will see later in this post.

For Html.RenderPartial the content of the partial view is added to content of the host view, either directly to the response output (or by using a StringWriter if you use or Html.Partial).

**Examples of when to use Html.RenderPartial and Html.RenderAction**

The mockup below shows a blog post with comments and a list of categories in a side bar.

**When to use Html.RenderPartial (or Html.Partial)**

For the page we’re designing I would create a view to show the blog post and call Html.RenderPartial to output the comments

[view source](#viewSource)

[print](#printSource)[?](#about)

|  |
| --- |
| @model WhenToUseRenderActionAndRenderPartial.Models.ShowPostViewModel |
| @{ |

|  |
| --- |
| ViewBag.Title = Model.Post.Title; |
| } |

|  |
| --- |
|  |
| <section> |

|  |
| --- |
| <article> |
| <header> |

|  |
| --- |
| <h1>@Model.Post.Title</h1> |
| Posted on | <time datetime="@Model.Post.DatePublished.ToString("s")"> |

|  |
| --- |
| @Model.Post.DatePublished.ToLongDateString() |
| </time> | @Model.Comments.Count() Comments |

|  |
| --- |
| </header> |
| @Html.Raw(Model.Post.Content) |

|  |
| --- |
| </article> |
| </section> |

|  |
| --- |
| @{Html.RenderPartial("\_Comments", Model.Comments);} |

The partial view for comments would look like this

[view source](#viewSource)

[print](#printSource)[?](#about)

|  |
| --- |
| @model IEnumerable<WhenToUseRenderActionAndRenderPartial.Models.Comment> |
| <section> |

|  |
| --- |
| <header> |
| <h3> |

|  |
| --- |
| Comments</h3> |
| </header> |

|  |
| --- |
| @foreach (var comment in Model) |
| { |

|  |
| --- |
| <article> |
| <header> |

|  |
| --- |
| @comment.Author on |
| <time datetime="@comment.DateCreated.ToString("s") "> |

|  |
| --- |
| @comment.DateCreated.ToLongDateString() |
| </time> |

|  |
| --- |
| </header> |
| <img alt="@comment.Author" src=@comment.ImageUrl /> |

|  |
| --- |
| <p> |
| @comment.Content |

|  |
| --- |
| </p> |
| </article> |

|  |
| --- |
| } |
| </section> |

This means the ShowPost action in the Blog controller

[view source](#viewSource)

[print](#printSource)[?](#about)

|  |
| --- |
| using System; |
| using System.Collections.Generic; |

|  |
| --- |
| using System.Web.Mvc; |
| using WhenToUseRenderActionAndRenderPartial.Models; |

|  |
| --- |
|  |
| namespace WhenToUseRenderActionAndRenderPartial.Controllers |

|  |
| --- |
| { |
| public class BlogController : Controller |

|  |
| --- |
| { |
| public ActionResult ShowPost() |

|  |
| --- |
| { |
| ShowPostViewModel viewModel = new ShowPostViewModel(); |

|  |
| --- |
| viewModel.Post = new Post |
| { |

|  |
| --- |
| Title = "Blog Title", |
| Content = "<p>Lorem ipsum dolor sit amet</p>", |

|  |
| --- |
| DatePublished = new DateTime(2010, 10, 5) |
| }; |

|  |
| --- |
| viewModel.Comments = new List<Comment>() |
| { |

|  |
| --- |
| new Comment() |
| { |

|  |
| --- |
| Author = "Andrew", |
| Content = "I think", |

|  |
| --- |
| DateCreated = new DateTime(2010, 10, 5) |
| }, |

|  |
| --- |
| new Comment() |
| { |

|  |
| --- |
| Author = "Bill", |
| Content = "I reckon", |

|  |
| --- |
| DateCreated = new DateTime(2010, 10, 5) |
| }, |

|  |
| --- |
| new Comment() |
| { |

|  |
| --- |
| Author = "Chris", |
| Content = "My opinion", |

|  |
| --- |
| DateCreated = new DateTime(2010, 10, 6) |
| } |

|  |
| --- |
| }; |
| return View(viewModel); |

|  |
| --- |
| } |
| } |

|  |
| --- |
| } |

needs to pass a model with properties for the blog post and its comments to the view.

[view source](#viewSource)

[print](#printSource)[?](#about)

|  |
| --- |
| public class Post |
| { |

|  |
| --- |
| public string Title { get; set; } |
| public string Content { get; set; } |

|  |
| --- |
| public DateTime DatePublished { get; set; } |
| } |

|  |
| --- |
|  |
| public class Comment |

|  |
| --- |
| { |
| public string Content { get; set; } |

|  |
| --- |
| public string Author { get; set; } |
| public string ImageUrl { get; set; } |

|  |
| --- |
| public DateTime DateCreated { get; set; } |
| } |

|  |
| --- |
|  |
| public class ShowPostViewModel |

|  |
| --- |
| { |
| public Post Post { get; set; } |

|  |
| --- |
| public IEnumerable<Comment> Comments { get; set; } |
| } |

**When to use Html.RenderAction (or Html.Action)**

So while we could add a property for categories to the ShowPostViewModel used for the ShowPost page and use Html.RenderPartial to display them, we have to add a category property to every model in our solution where categories are shown.

[view source](#viewSource)

[print](#printSource)[?](#about)

|  |
| --- |
| public class ShowPostViewModel |
| { |

|  |
| --- |
| public Post Post { get; set; } |
| public IEnumerable<Comment> Comments { get; set; } |

|  |
| --- |
| public IEnumerable<Category> Categories { get; set; } |
| } |

|  |
| --- |
|  |
| public class AnotherViewModel |

|  |
| --- |
| { |
| public string StringToOutput { get; set; } |

|  |
| --- |
| public IEnumerable<Category> Categories { get; set; } |
| } |

|  |
| --- |
|  |
| public class Category |

|  |
| --- |
| { |
| public int Id { get; set; } |

|  |
| --- |
| public string Name { get; set; } |
| } |

And using a base class with a categories property and have ViewModel classes inherit from it would just be bloating models with unrelated data and concerns.

[view source](#viewSource)

[print](#printSource)[?](#about)

|  |
| --- |
| public class ViewModelBase |
| { |

|  |
| --- |
| public IEnumerable<Category> Categories { get; set; } |
| } |

|  |
| --- |
|  |
| public class ShowPostViewModel : ViewModelBase |

|  |
| --- |
| { |
| public Post Post { get; set; } |

|  |
| --- |
| public IEnumerable<Comment> Comments { get; set; } |
| } |

So if you want to display the same information on multiple pages and the data doesn’t necessarily depend on or isn’t related to the rest of the page, something like a widget to display categories then I would use Html.RenderAction.

As shown in the markup below Html.RenderAction is called from the layout page for the blog post.

[view source](#viewSource)

[print](#printSource)[?](#about)

|  |
| --- |
| <!DOCTYPE html> |
| <html> |

|  |
| --- |
| <head> |
| <title>@ViewBag.Title</title> |

|  |
| --- |
| <link href="@Url.Content("~/Content/Site.css")" rel="stylesheet" type="text/css" /> |
| <script src="@Url.Content("~/Scripts/jquery-1.4.4.min.js")" type="text/javascript"></script> |

|  |
| --- |
| </head> |
| <body> |

|  |
| --- |
| @RenderBody() |
| @{Html.RenderAction("Show", "Category");} |

|  |
| --- |
| </body> |
| </html> |

**Jag Reehal’s Final Thought on ‘When to use Html.RenderPartial and Html.RenderAction in ASP.NET MVC Razor Views’**

In cases where a model for the page is already populated with all the information required I would use Html.RenderPartial.

If the same information is shown on multiple pages it’s likely that the data needs to be in its own model so I would use Html.RenderAction. As shown in this post I would say layout pages are going to the most common place Html.RenderAction will be used.

**Related Posts**

1. [ASP.NET MVC Razor View Best Practices – Keep logic out of your Razor views](http://www.arrangeactassert.com/asp-net-mvc-razor-view-best-practices-keep-logic-out-of-your-razor-views/)
2. [ASP.NET MVC View Best Practices – Save time creating views with MvcContrib.FluentHtml](http://www.arrangeactassert.com/asp-net-mvc-view-best-practices-save-time-creating-views-with-mvccontrib-fluenthtml/)
3. [ASP.NET MVC View Best Practices – Keep logic out of your views](http://www.arrangeactassert.com/asp-net-mvc-view-best-practices-keep-logic-out-of-your-views/)

**Comments**

* [Disqus](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)
  + [**Login**](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)
  + [**About Disqus**](http://disqus.com/)
* [**Like**](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)
* [Dislike](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)
  + 18 people liked this.

**Glad you liked it. Would you like to share?**

Facebook

Twitter

* [Share](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)
* [No thanks](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)

Sharing this page …

Thanks! [Close](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)

[Login](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)

**Add New Comment**

* Post as …
* Image

**Showing 17 comments**

[**niaher**](http://www.lessonist.com/) 1 comment collapsed [CollapseExpand](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)

Good post. Thanks.

* + A[Like](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)
  + [Reply](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)
  + [1 year ago](#comment-141261906)
  + [1 Like](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)
  + [F](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)

**Cesar** 1 comment collapsed [CollapseExpand](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)

I want to be the first one to thank you for such a good article. 10!

* + A[Like](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)
  + [Reply](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)
  + [1 year ago](#comment-141261905)
  + [1 Like](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)
  + [F](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)

**nick4eva** 1 comment collapsed [CollapseExpand](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)

Great post. It was very helpful for me.

* + A[Like](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)
  + [Reply](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)
  + [1 year ago](#comment-141261903)
  + [1 Like](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)
  + [F](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)

**Wiebe Tijsma** 1 comment collapsed [CollapseExpand](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)

Hi, thanks for the article... Maybe a bit late, but what's good to mention here is that one of the nice side effects of going through the entire ASP.NET MVC stack with RenderAction is that action filters like OutputCache are also working, giving you an easy way to cache parts of your page...

* + A[Like](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)
  + [Reply](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)
  + [2 months ago](#comment-423934955)
  + [0 Like](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)
  + [F](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)

**Leon** 2 comments collapsed [CollapseExpand](http://www.arrangeactassert.com/when-to-use-html-renderpartial-and-html-renderaction-in-asp-net-mvc-razor-views/)

Thanks for your article. It is over a year old, but I hope you are still responding to comments on it.

I am new to MVC and I am using version 3. In your example, in the layout page line that contains Html.RenderAction('Show", "Category") , where is the "Show" action located? I am trying to implement a menu of dropdowns that will render via the layout regardless of the View in the "content" area (RenderBody()). I have tried the default Index action in a specific MenuController as well as actions in the HomeController. Below is the Index action in the MenuController:

<header id="header"> <h1>Menu Example</h1> </header>  
<div id="menu"> @{Html.RenderAction("Index", "MenuController");} </div>  
<div id="content"> @RenderBody() </div>  
<footer>

This is the footer

</footer>