Prefered way of building multi view layouts in ASP.NET MVC?

Ask Question

2

I'm currently wondering about the prefered way to build multi view layouts using ASP.NET MVC.

For example in angular-based SPAs, loading different views into a layout is pretty straightforward:

```
<div ui-view="header"></div>
<div ui-view="content"></div>

.state("home", {
    views: {
        "header": { templateUrl: "app/_header.html" }
...
```

However in MVC, there seem to be multiple ways to achieve something, but I'm extremely confused about what 'method' is originally intended for what purpose.

- @Html.RenderPartial() - renders a 'partial', defined by the parent
- @Html.Partial() - seems to be the same as above
- @RenderPage() - renders a 'page', defined by the parent
- @RenderSection() - renders a 'section' by id, section defined by 'body view'

For a common "header, body, footer, nav" layout, what is the prefered (or the right) method? Or is it just a question of personal preference?

c# asp.net asp.net-mvc angularjs razor

asked Jun 10 '15 at 9:03

user4614918

Just a suggestion to add to answer below - if you have an MVC project, you can look at the `/Views/Shared` folder and inspect the scaffolded `_Layout.cshtml` file. I think this will give you a starting point. Depending on what you're going for, you can also look at `_ViewStart` - link is relatively "old" but still great/quick read. Hth. – EdSF Jun 10 '15 at 13:49

1 Answer

3

In your scenario, I would go for `@Html.Partial` just in case you need to manipulate the HTML inside your SPA (as you can assign it to a variable).

As for common elements, use `RenderPartial` , this can be slightly faster (*albeit nothing to write home about*).

Here's the explanations:

> Html.RenderPartial

Renders the view directly to the response stream, similarly to how a normal web page is served by the server.

> Html.Partial

Renders the view as a string, if you want to assign the result into a variable and manipulate it, then use this.

> RenderPage

Same as `RenderPartial` , but allows you to specify a path to the view should you need to.

> RenderSection

Usually used in layouts to define sections that will be included within child pages, this is especially useful if you want scripts/styles included in the `head` tag that you wouldn't be able to dictate within a child view.

There's also:

> RenderAction

This takes a controller and action and will render the result of that action, useful if said view requires a different model and the data to populate that model is stored in a common function/library.

edited Jun 10 '15 at 9:17

answered Jun 10 '15 at 9:10

mattytommo
48k 15 103 133