Partial views in ASP.NET Core

06/12/2019 • 9 minutes to read • +10

In this article

By Steve Smith, Maher JENDOUBI, Rick Anderson, and Scott Sauber

A partial view is a Razor markup file (*.cshtml*) without an @page directive that renders HTML output *within* another markup file's rendered output.

The term *partial view* is used when developing either an MVC app, where markup files are called *views*, or a Razor Pages app, where markup files are called *pages*. This topic generically refers to MVC views and Razor Pages pages as *markup files*.

View or download sample code (how to download)

When to use partial views

Partial views are an effective way to:

- Break up large markup files into smaller components.

In a large, complex markup file composed of several logical pieces, there's an advantage to working with each piece isolated into a partial view. The code in the markup file is manageable because the markup only contains the overall page structure and references to partial views.

- Reduce the duplication of common markup content across markup files.

  When the same markup elements are used across markup files, a partial view removes the duplication of markup content into one partial view file. When the markup is changed in the partial view, it updates the rendered output of the markup files that use the partial view.

Partial views shouldn't be used to maintain common layout elements. Common layout elements should be specified in _Layout.cshtml files.

Don't use a partial view where complex rendering logic or code execution is required to render the markup. Instead of a partial view, use a view component.

Declare partial views

A partial view is a *.cshtml* markup file without an @page directive maintained within the *Views* folder (MVC) or *Pages* folder (Razor Pages).

In ASP.NET Core MVC, a controller's ViewResult is capable of returning either a view or a partial view. In Razor Pages, a PageModel can return a partial view represented as a PartialViewResult object. Referencing and rendering partial views is described in the Reference a partial view section.

Unlike MVC view or page rendering, a partial view doesn't run *_ViewStart.cshtml*. For more information on *_ViewStart.cshtml*, see Layout in ASP.NET Core.

Partial view file names often begin with an underscore (`_`). This naming convention isn't required, but it helps to visually differentiate partial views from views and pages.

Reference a partial view

Use a partial view in a Razor Pages PageModel

In ASP.NET Core 2.0 or 2.1, the following handler method renders the *_AuthorPartialRP.cshtml* partial view to the response:

C# Copy

```
public IActionResult OnGetPartial() =>
    new PartialViewResult
    {
        ViewName = "_AuthorPartialRP",
        ViewData = ViewData,
    };
```

In ASP.NET Core 2.2 or later, a handler method can alternatively call the Partial method to produce a `PartialViewResult` object:

C# Copy

```
public IActionResult OnGetPartial() =>
    Partial("_AuthorPartialRP");
```

Use a partial view in a markup file

Within a markup file, there are several ways to reference a partial view. We recommend that apps use one of the following asynchronous rendering approaches:

- Partial Tag Helper
- Asynchronous HTML Helper

Partial Tag Helper

The Partial Tag Helper requires ASP.NET Core 2.1 or later.

The Partial Tag Helper renders content asynchronously and uses an HTML-like syntax:

CSHTML Copy

```cshtml
<partial name="_PartialName" />
```

When a file extension is present, the Tag Helper references a partial view that must be in the same folder as the markup file calling the partial view:

CSHTML Copy

```cshtml
<partial name="_PartialName.cshtml" />
```

The following example references a partial view from the app root. Paths that start with a tilde-slash (`~/`) or a slash (`/`) refer to the app root:

Razor Pages

CSHTML Copy

```cshtml
<partial name="~/Pages/Folder/_PartialName.cshtml" />
<partial name="/Pages/Folder/_PartialName.cshtml" />
```

MVC

CSHTML Copy

```
<partial name="~/Views/Folder/_PartialName.cshtml" />
<partial name="/Views/Folder/_PartialName.cshtml" />
```

The following example references a partial view with a relative path:

CSHTML

```
<partial name="../Account/_PartialName.cshtml" />
```

For more information, see Partial Tag Helper in ASP.NET Core.

Asynchronous HTML Helper

When using an HTML Helper, the best practice is to use PartialAsync. `PartialAsync` returns an IHtmlContent type wrapped in a Task<TResult>. The method is referenced by prefixing the awaited call with an `@` character:

CSHTML

```
@await Html.PartialAsync("_PartialName")
```

When the file extension is present, the HTML Helper references a partial view that must be in the same folder as the markup file calling the partial view:

CSHTML

```
@await Html.PartialAsync("_PartialName.cshtml")
```

The following example references a partial view from the app root. Paths that start with a tilde-slash (`~/`) or a slash (`/`) refer to the app root:

Razor Pages

CSHTML Copy

```
@await Html.PartialAsync("~/Pages/Folder/_PartialName.cshtml")
@await Html.PartialAsync("/Pages/Folder/_PartialName.cshtml")
```

MVC

CSHTML Copy

```
@await Html.PartialAsync("~/Views/Folder/_PartialName.cshtml")
@await Html.PartialAsync("/Views/Folder/_PartialName.cshtml")
```

The following example references a partial view with a relative path:

CSHTML Copy

```
@await Html.PartialAsync("../Account/_LoginPartial.cshtml")
```

Alternatively, you can render a partial view with RenderPartialAsync. This method doesn't return an IHtmlContent. It streams the rendered output directly to the response. Because the method doesn't return a result, it must be called within a Razor code block:

CSHTML Copy

```
@{
    await Html.RenderPartialAsync("_AuthorPartial");
}
```

Since `RenderPartialAsync` streams rendered content, it provides better performance in some scenarios. In performance-critical situations, benchmark the page using both approaches and use the approach that generates a faster response.

Synchronous HTML Helper

Partial and RenderPartial are the synchronous equivalents of `PartialAsync` and `RenderPartialAsync`, respectively. The synchronous equivalents aren't recommended because there are scenarios in which they deadlock. The synchronous methods are targeted for removal in a future release.

> **Important**
>
> If you need to execute code, use a **view component** instead of a partial view.

Calling `Partial` or `RenderPartial` results in a Visual Studio analyzer warning. For example, the presence of `Partial` yields the following warning message:

> Use of IHtmlHelper.Partial may result in application deadlocks. Consider using <partial> Tag Helper or IHtmlHelper.PartialAsync.

Replace calls to `@Html.Partial` with `@await Html.PartialAsync` or the Partial Tag Helper. For more information on Partial Tag Helper migration, see Migrate from an HTML Helper.

Partial view discovery

When a partial view is referenced by name without a file extension, the following locations are searched in the stated order:

Razor Pages

1. Currently executing page's folder
2. Directory graph above the page's folder
3. `/Shared`
4. `/Pages/Shared`
5. `/Views/Shared`

MVC

1. `/Areas/<Area-Name>/Views/<Controller-Name>`
2. `/Areas/<Area-Name>/Views/Shared`
3. `/Views/Shared`
4. `/Pages/Shared`

The following conventions apply to partial view discovery:

- Different partial views with the same file name are allowed when the partial views are in different folders.
- When referencing a partial view by name without a file extension and the partial view is present in both the caller's folder and the *Shared* folder, the partial view in the caller's folder supplies the partial view. If the partial view isn't present in the caller's folder, the partial view is provided from the *Shared* folder. Partial views in the *Shared* folder are called *shared partial views* or *default partial views*.
- Partial views can be *chained*—a partial view can call another partial view if a circular reference isn't formed by the calls. Relative paths are always relative to the current file, not to the root or parent of the file.

Note

A **Razor** `section` defined in a partial view is invisible to parent markup files. The `section` is only visible to the partial view in which it's defined.

Access data from partial views

When a partial view is instantiated, it receives a *copy* of the parent's `ViewData` dictionary. Updates made to the data within the partial view aren't persisted to the parent view. `ViewData` changes in a partial view are lost when the partial view returns.

The following example demonstrates how to pass an instance of ViewDataDictionary to a partial view:

CSHTML Copy

```
@await Html.PartialAsync("_PartialName", customViewData)
```

You can pass a model into a partial view. The model can be a custom object. You can pass a model with `PartialAsync` (renders a block of content to the caller) or `RenderPartialAsync` (streams the content to the output):

CSHTML Copy

```
@await Html.PartialAsync("_PartialName", model)
```

Razor Pages

The following markup in the sample app is from the *Pages/ArticlesRP/ReadRP.cshtml* page. The page contains two partial views. The second partial view passes in a model and `ViewData` to the partial view. The `ViewDataDictionary` constructor overload is used to pass a new `ViewData` dictionary while retaining the existing `ViewData` dictionary.

CSHTML Copy

```
@model ReadRPModel

<h2>@Model.Article.Title</h2>
@* Pass the author's name to Pages\Shared\_AuthorPartialRP.cshtml *@
@await Html.PartialAsync("../Shared/_AuthorPartialRP", Model.Article.AuthorName)
@Model.Article.PublicationDate
```

```
@* Loop over the Sections and pass in a section and additional ViewData to
   the strongly typed Pages\ArticlesRP\_ArticleSectionRP.cshtml partial view. *@
@{
    var index = 0;

    foreach (var section in Model.Article.Sections)
    {
        await Html.PartialAsync("_ArticleSectionRP",
                                section,
                                new ViewDataDictionary(ViewData)
                                {
                                    { "index", index }
                                });

        index++;
    }
}
```

Pages/Shared/_AuthorPartialRP.cshtml is the first partial view referenced by the *ReadRP.cshtml* markup file:

CSHTML Copy

```
@model string
<div>
    <h3>@Model</h3>
    This partial view from /Pages/Shared/_AuthorPartialRP.cshtml.
</div>
```

Pages/ArticlesRP/_ArticleSectionRP.cshtml is the second partial view referenced by the *ReadRP.cshtml* markup file:

CSHTML Copy

```
@using PartialViewsSample.ViewModels
@model ArticleSection
```

```
<h3>@Model.Title Index: @ViewData["index"]</h3>
<div>
    @Model.Content
</div>
```

MVC

The following markup in the sample app shows the *Views/Articles/Read.cshtml* view. The view contains two partial views. The second partial view passes in a model and `ViewData` to the partial view. The `ViewDataDictionary` constructor overload is used to pass a new `ViewData` dictionary while retaining the existing `ViewData` dictionary.

CSHTML Copy

```
@model PartialViewsSample.ViewModels.Article

<h2>@Model.Title</h2>
@* Pass the author's name to Views\Shared\_AuthorPartial.cshtml *@
@await Html.PartialAsync("_AuthorPartial", Model.AuthorName)
@Model.PublicationDate

@* Loop over the Sections and pass in a section and additional ViewData to
   the strongly typed Views\Articles\_ArticleSection.cshtml partial view. *@
@{
    var index = 0;

    foreach (var section in Model.Sections)
    {
        await Html.PartialAsync("_ArticleSection",
                                section,
                                new ViewDataDictionary(ViewData)
                                {
                                    { "index", index }
                                });

        index++;
    }
}
```

Views/Shared/_AuthorPartial.cshtml is the first partial view referenced by the *Read.cshtml* markup file:

CSHTML Copy

```cshtml
@model string
<div>
    <h3>@Model</h3>
    This partial view from /Views/Shared/_AuthorPartial.cshtml.
</div>
```

Views/Articles/_ArticleSection.cshtml is the second partial view referenced by the *Read.cshtml* markup file:

CSHTML Copy

```cshtml
@using PartialViewsSample.ViewModels
@model ArticleSection

<h3>@Model.Title Index: @ViewData["index"]</h3>
<div>
    @Model.Content
</div>
```

At runtime, the partials are rendered into the parent markup file's rendered output, which itself is rendered within the shared *_Layout.cshtml*. The first partial view renders the article author's name and publication date:

> Abraham Lincoln
>
> This partial view from <shared partial view file path>. 11/19/1863 12:00:00 AM

The second partial view renders the article's sections:

> Section One Index: 0
>
> Four score and seven years ago ...
>
> Section Two Index: 1
>
> Now we are engaged in a great civil war, testing ...
>
> Section Three Index: 2
>
> But, in a larger sense, we can not dedicate ...

Additional resources

- razor syntax reference for ASP.NET Core
- Tag Helpers in ASP.NET Core
- Partial Tag Helper in ASP.NET Core
- View components in ASP.NET Core
- Areas in ASP.NET Core

Is this page helpful?

Yes No