

# Partial Views

As mentioned previously, tmpst is a single page application which means you are only interacting with one HTML page. However, putting all of the HTML required into one file would be messy and hard to manage. To make sure our page is easy to manage, tmpst uses partial views to separate sections of HTML

A partial view is a .cshtml file that contains HTML to be rendered within another .html or .cshtml page. Tmpst uses partial views to separate the different resources that we provide (weather, earthquakes etc.) as well as the content required within these services as in partial views within partial views.

## Advantages:

Partial views provide various advantages as detailed below:

- Separate logic within large files - By using partial views, logical systems (such as the previously mentioned tmpst systems) can be split up into smaller modules. This means that a “Landing Page”, such as the tmpst weather or earthquake system, would contain references to all the partial views that it requires, such as query builder and result containers, without containing (much) complex HTML within itself
- Reduce HTML duplication - By using partial views, any HTML that needs to be rendered for multiple pages can be encapsulated within a partial view and rendered as many times as required

## Useage:

The method to using when referencing a partial view is:

```
@await Html.PartialAsync("{PATH_TO_PARTIAL}");
```

In this tutorial:

Pre-requisites:

- Visual Studio 2017 (Community, Professional or Enterprise)
- .NET Core v2.2
- HTML you wish to render as a partial view
- ASP.NET Core Web Application

Aims:

- Learn how to create a partial view
- Learn how to reference a partial view

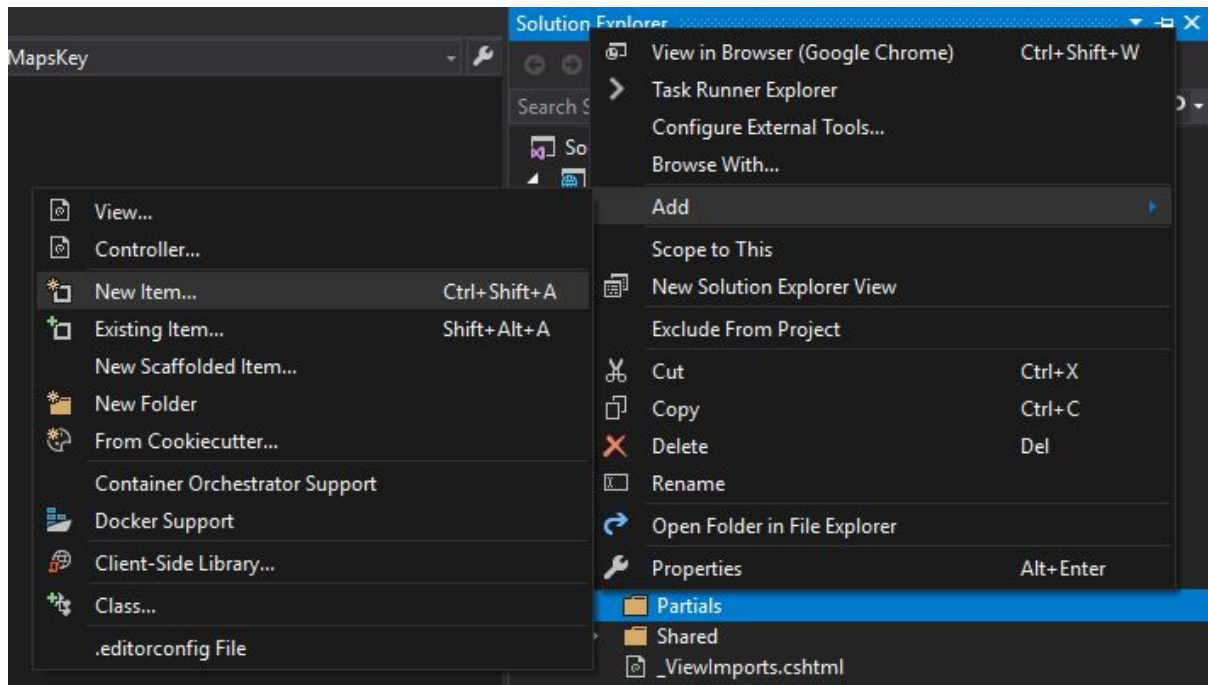
## Step 1: Create a View

In order to render a partial view, we will need to create one.

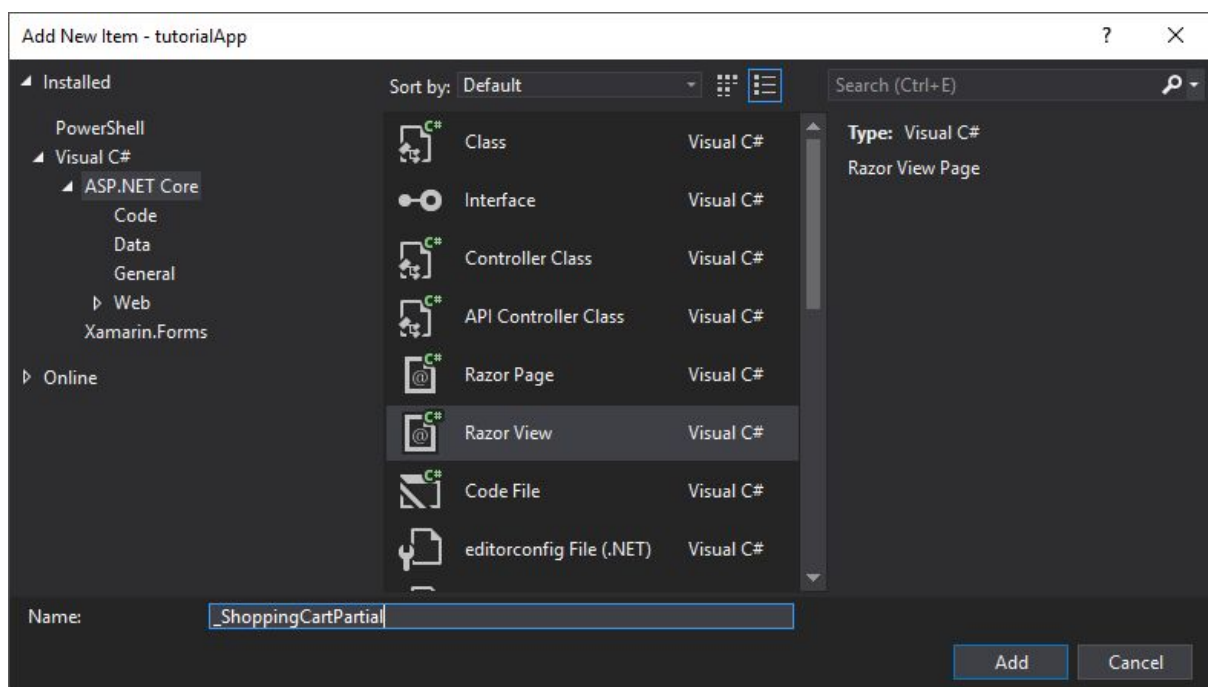
Steps:

1. Right click on the view folder that you want to add the partial to (it is often good practice to have a separate folder for partial views)
2. Select “Add New Item” from the context menu
3. Name the partial view (it is good practice to prefix the name with an underscore (\_) such as “\_ShoppingCartPartial”

Below is a breakdown of all the above mentioned steps:



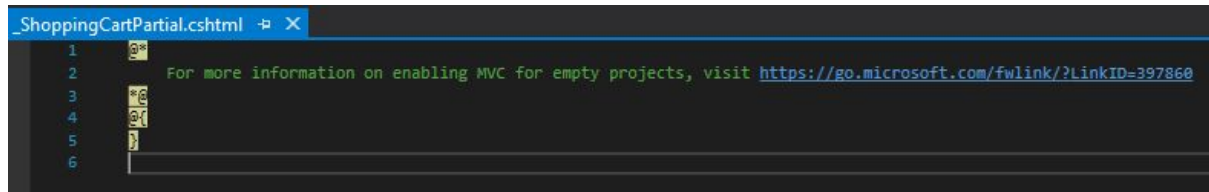
Above is an example of accessing the context menu



Above is an example of naming the partial view

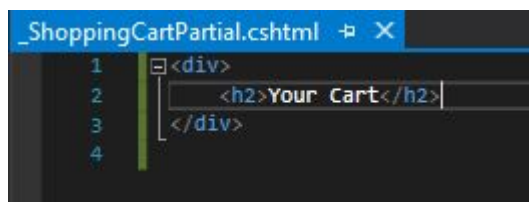
## Step 2: Populate the Partial View

Now that the partial view has been created, we need to add content that we want to render. Currently the partial view we created should look like this:



```
_ShoppingCartPartial.cshtml
1
2 For more information on enabling MVC for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=397860
3
4
5
6
```

There is no actual HTML within the partial, so referencing it would make no difference. As a simple example add a header to the partial as is shown below:



```
_ShoppingCartPartial.cshtml
1 <div>
2   <h2>Your Cart</h2>
3 </div>
4
```

The image to the left shows a div tag that contains a h2 tag that states “Your Cart”

## Step 3 (Final Step): Reference Partial

Before continuing with this step, ensure that you have the following setup:

- A partial view with content - Make sure that you have a partial content with valid HTML inside. Your HTML content does not have to be the same as the examples given

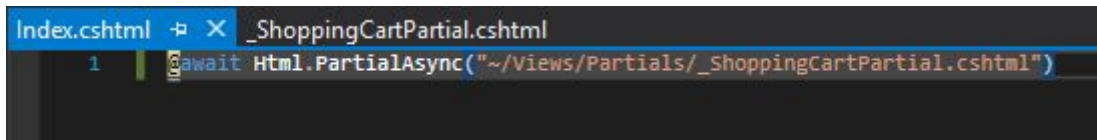
If you have completed all of the above items, then you can proceed to actually reference the partial view in another HTML page.

Note: For this section of the tutorial, we will be referencing our partial view in the Index.cshtml view as this is often the page that is configured to display first

Steps:

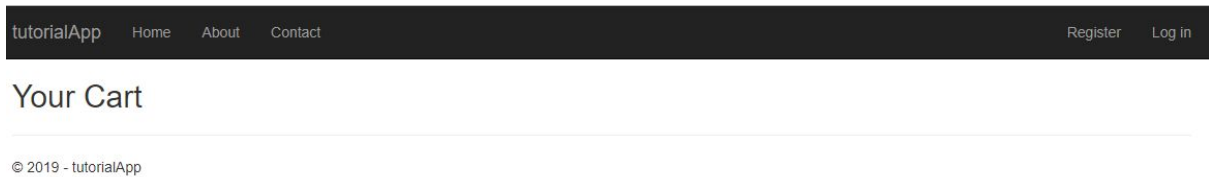
1. Remove all predefined HTML from the index view
2. Reference the partial view

After removing all of the predefined HTML from the index view, add a reference to the partial view using the syntax mentioned in the usage section of this tutorial



```
Index.cshtml  X  _ShoppingCartPartial.cshtml
1  @await Html.PartialAsync("~/Views/Partials/_ShoppingCartPartial.cshtml")
```

After you have added this line to your chosen view, run the application and you should see a result similar to the one below:



## Conclusion

That is the code required to create and reference a partial view. The content, complexity and number of partial views in an application depend on the developer's choice. We use the `@await Html.PartialAsync` as this ensures that the partial is fully loaded before the line after is processed, which prevents lock ups and unresponsive pages

### Common Issues:

Normally, partial views can be created and referenced without too much difficulty, however, some of the most common issues are detailed below:

- Referencing the path - In our example we reference the `_ShoppingCartPartial` which is in the `Partials` folder of the `Views` folder. If we instead try and reference the `_ShoppingCartPartial` view in just the `Views` folder we will get an error. If the path to the chosen partial view is incorrect, then the application will crash and throw an error
- Forgetting to reference - Sometimes an error can occur purely because the partial was not referenced at all. It is generally good practice to reference a partial as soon as it is created to mitigate errors like this

## Outcomes

You should now be able to:

- Create a partial view
- Reference a partial view

## Useful Resources:

- Microsoft Partial View Docs - <https://docs.microsoft.com/en-us/aspnet/core/mvc/views/partial?view=aspnetcore-2.2>