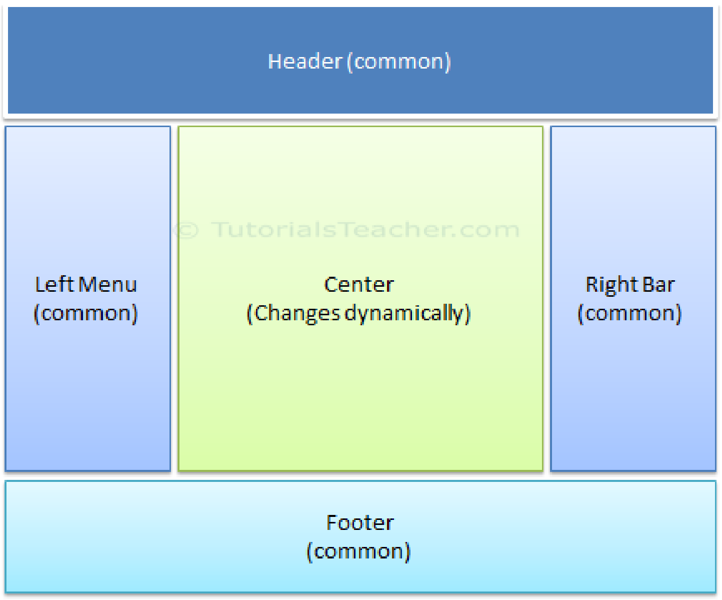
Layout View:

In this section, you will learn about the layout view in ASP.NET MVC.

What is Layout view?

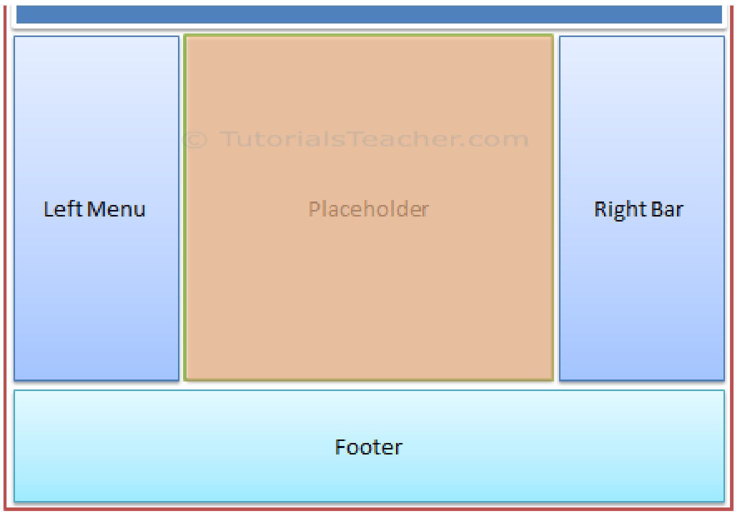
An application may contain common parts in the UI which remains the same throughout the application such as the logo, header, left navigation bar, right bar or footer section. ASP.NET MVC introduced a Layout view which contains these common UI parts, so that we don't have to write the same code in every page. The layout view is same as the master page of the ASP.NET webform application.

For example, an application UI may contain Header, Left menu bar, right bar and footer section that remains same in every page and only the centre section changes dynamically as shown below.



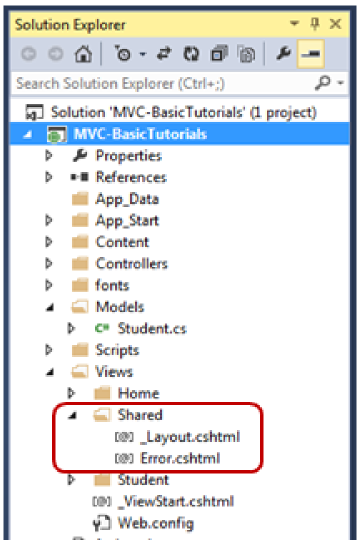
Sample Application UI Parts

The layout view allows you to define a common site template, which can be inherited in multiple views to provide a consistent look and feel in multiple pages of an application. The layout view eliminates duplicate coding and enhances development speed and easy maintenance. The layout view for the above sample UI would contain a Header, Left Menu, Right bar and Footer sections. It contains a placeholder for the center section that changes dynamically as shown below.



Layout View

The razor layout view has same extension as other views, .cshtml or .vbhtml. Layout views are shared with multiple views, so it must be stored in the Shared folder. For example, when we created our first MVC application in the previous section, it also created _Layout.cshtml in the Shared folder as shown below.



Layout View in Shared folder

The following is an auto-generated _Layout.cshtml.

_Layout.cshtml:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
```

```
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            @Html.ActionLink("Application name", "Index", "Home", new { area = "" }, new { @clas
        </div>
        <div class="navbar-collapse collapse">
            <ul class="nav navbar-nav">
                <li>@Html.ActionLink("Home", "Index", "Home")</li>
                <li>@Html.ActionLink("About", "About", "Home")</li>
                <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
            </ul>
        </div>
    </div>
</div>
<div class="container body-content">
    @RenderBody()
    <hr />
    <footer>
        <p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>
    </footer>
</div>

@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/bootstrap")
@RenderSection("scripts", required: false)
dy>
ml>
```

As you can see, the layout view contains html Doctype, head and body as normal html, the only difference is call to RenderBody() and RenderSection() methods. RenderBody acts like a placeholder for other views. For example, Index.cshtml in the home folder will be injected and rendered in the layout view, where the RenderBody() method is being called. You will learn about these rendering methods later in this section.

Use Layout View:

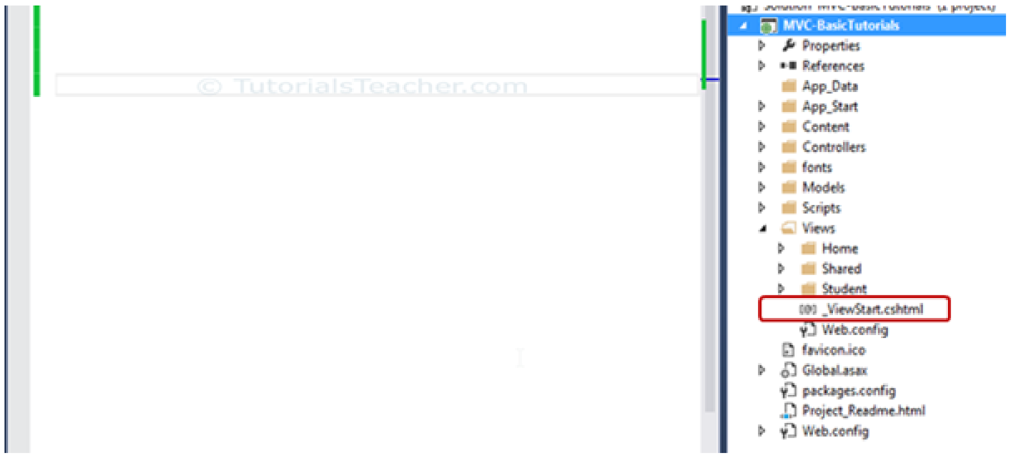
You must be wondering that how would the View know which layout view to use?

You can set the layout view in multiple ways, by using _ViewStart.cshtml or setting up path of the layout page using Layout property in the individual view or specifying layout view name in the action method.

_ViewStart.cshtml:

_ViewStart.cshtml is included in the Views folder by default. It sets up the default layout page for all the views in the folder and its subfolders using the Layout property. You can assign a valid path of any Layout page to the Layout property.

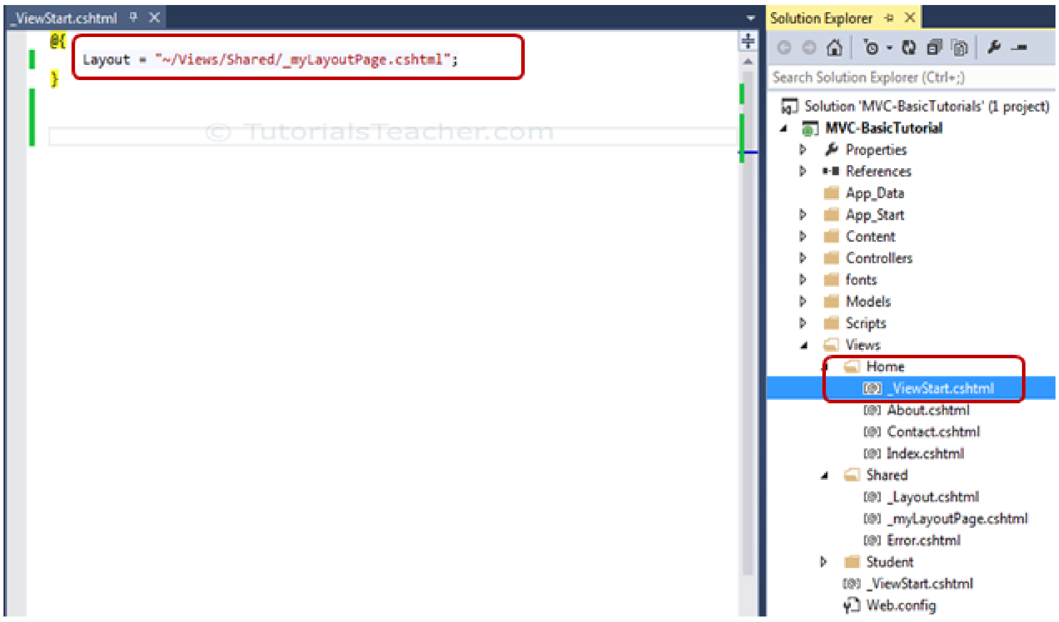
For example, the following _ViewStart.cshtml in the **Views** folder, sets the Layout property to "~/Views/Shared/_Layout.cshtml". So now, _layout.cshtml would be layout view of all the views included in Views and its subfolders. So by default, all the views derived default layout page from _ViewStart.cshtml of Views folder.



_ViewStart.cshtml

_ViewStart.cshtml can also be included in sub folder of View folder to set the default layout page for all the views included in that particular subfolder only.

For example, the following _ViewStart.cshtml in Home folder, sets Layout property to _myLayoutPage.cshtml. So this _ViewStart.cshtml will influence all the views included in the Home folder only. So now, Index, About and Contact views will be rendered in _myLayoutPage.cshtml instead of default _Layout.cshml.



```
@{
    Layout = "~/Views/Shared/_myLayoutPage.cshtml";
}
```

Layout View

Setting Layout property in individual view:

You can also override default layout page set by _ViewStart.cshtml by setting Layout property in each individual .cshtml view. For example, the following Index view use _myLayoutPage.cshtml even if _ViewStart.cshtml set _Layout.cshtml.

Index view:

```
@{
```

```
<div class="jumbotron">
    <h1>ASP.NET</h1>
    <p class="lead">ASP.NET is a free web framework for building great Web sites and Web application
    <p><a href="http://asp.net" class="btn btn-primary btn-lg">Learn more &raquo;</a></p>
</div>

<div class="row">
    <div class="col-md-4">
        <h2>Getting started</h2>
        <p>
            ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that
            enables a clean separation of concerns and gives you full control over markup
            for enjoyable, agile development.
        </p>
        <p><a class="btn btn-default" href="http://go.microsoft.com/fwlink/?LinkId=301865">Learn mor
    </div>
    <div class="col-md-4">
        <h2>Get more libraries</h2>
        <p>NuGet is a free Visual Studio extension that makes it easy to add, remove, and update lib
        <p><a class="btn btn-default" href="http://go.microsoft.com/fwlink/?LinkId=301866">Learn mor
    </div>
    <div class="col-md-4">
        <h2>Web Hosting</h2>
        <p>You can easily find a web hosting company that offers the right mix of features and price
        <p><a class="btn btn-default" href="http://go.microsoft.com/fwlink/?LinkId=301867">Learn mor
    </div>
</div>
```

Specify Layout page in ActionResult method:

You can also specify which layout page to use in while rendering view from action method using View() method.

The following example, View() method renders Index view using _myLayoutPage.cshtml.

C#:

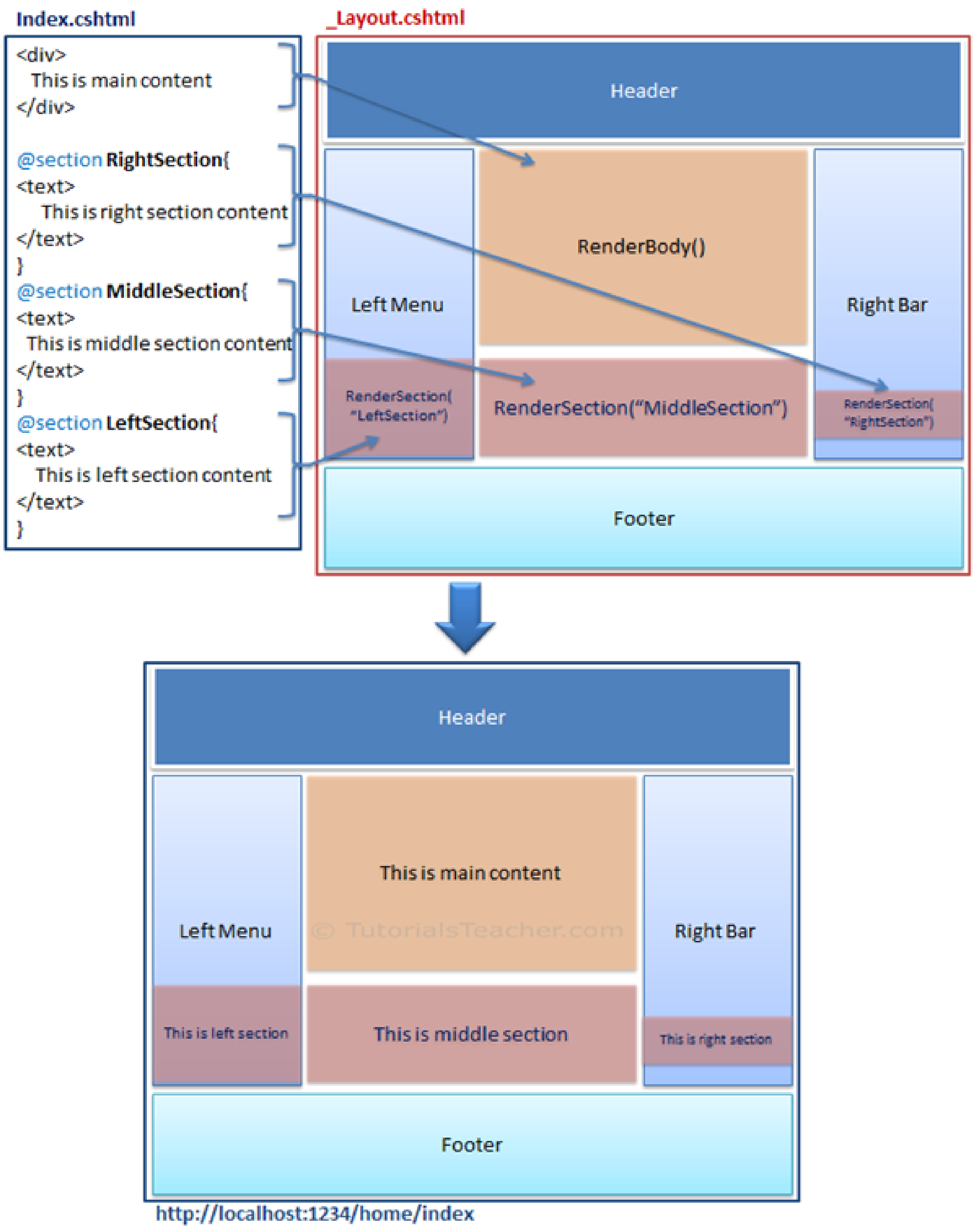
```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View("Index", "_myLayoutPage");
    }

    public ActionResult About()
    {
        return View();
    }

    public ActionResult Contact()
    {
        return View();
```

Rendering Methods:

ASP.NET MVC layout view renders child views using the following methods.

Method	Description
RenderBody()	Renders the portion of the child view that is not within a named section. Layout view must include RenderBody() method.
RenderSection(string name)	Renders a content of named section and specifies whether the section is required. RenderSection() is optional in Layout view.

The following figure illustrates the use of RenderBody and RenderSection methods.



Rendering Methods

RightSection in the above figure. Index.cshtml defines named section using @section where name of each section matches with the name specified in RenderSection method of _Layout.cshtml, such as @Section RightSection etc. At run time, the named sections of Index.cshtml such as LeftSection, RightSection and MiddleSection will be rendered at appropriate place where RenderSection method is called. Rest of the part of Index view, which is not in any of the named section will render where RenderBody() method is being called.

Let's create a new layout view to understand the above render methods in the next section.

Further Reading:

- Difference between RenderBody and RenderSection methods

Points to Remember :

1. The Layout view contains common parts of a UI. It is same like masterpage of ASP.NET webforms.
2. _ViewStart.cshtml file can be used to specify path of layout page, which in turn will be applicable to all the views of the folder and its subfolder.
3. You can set the Layout property in the individual view also, to override default layout page setting of _ViewStart.cshtml
4. Layout view uses two rendering methods: RenderBody() and RenderSection().
5. RenderBody can be used only once in the layout view, whereas the RenderSection method can be called multiple time with different name.
6. RenderBody method renders all the content of view which is not wrapped in named section.
7. RenderSection method renders the content of a view which is wrapped in named section.
8. RenderSection can be configured as required or optional. If required, then all the child views must included that named section.

« Previous Next »

We respect your privacy.