Session #3

LET'S MAKE A WEB APP

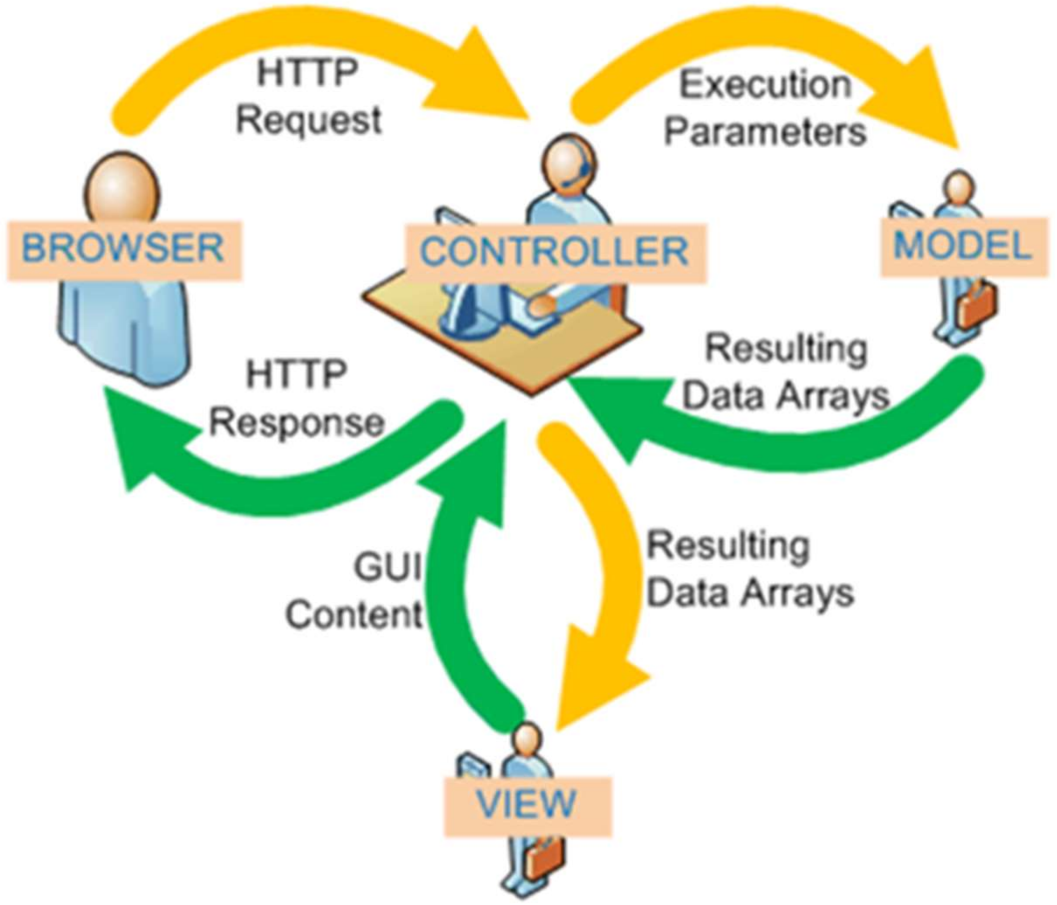
Agenda for today

- Recap on Week #2 and homework
- Revisiting some concepts and introducing a few new
- Demo
- Food
- Everybody codes

Master Class #3

After this session (and the accompanying homework) you should know:

- *How to create an ASP.NET Core MVC Web project*
- *Understand basic HTTP and Client/Server theory*
- *Understand the Model-View-Controller design pattern*
- *Learn how Razor can be used to generate views and layouts*
- *Understand how Routing works*



MVC – Model View Controller

HTTP Theory

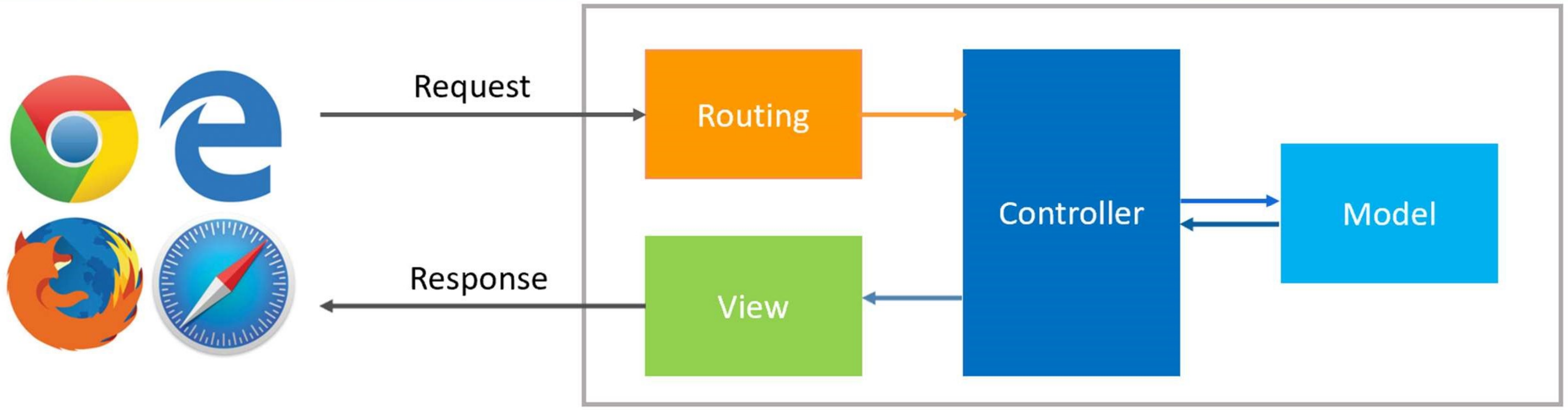
Browser sends HTTP Request

- GET / POST / HEAD / DELETE / …
- URL
- Query parameter or Body data
- Cookies and other headers

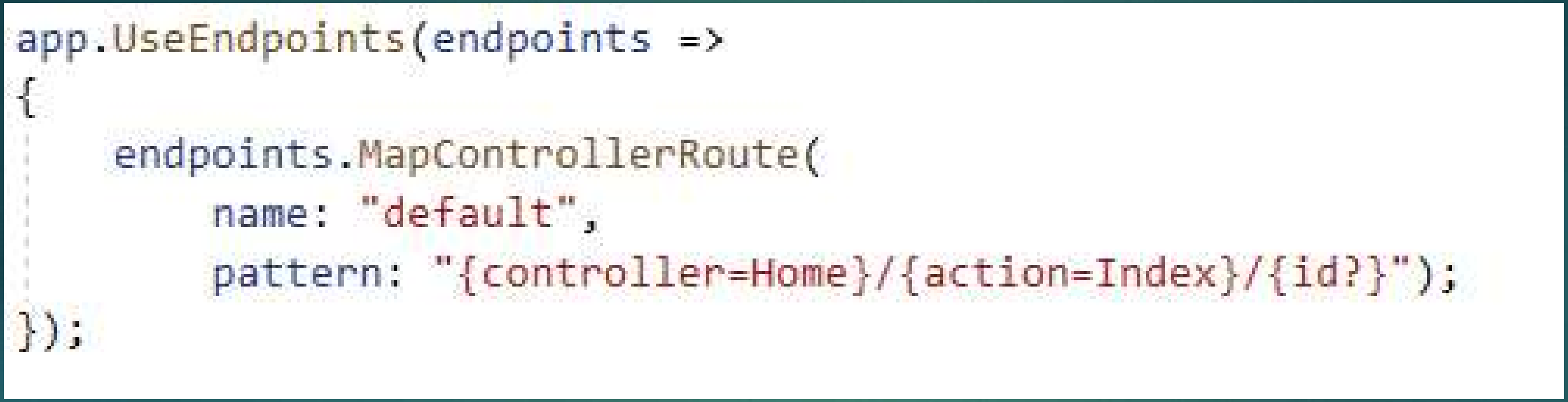
Server returns HTTP Response

- Headers – including content type
- Set-cookies
- Body content

Routing



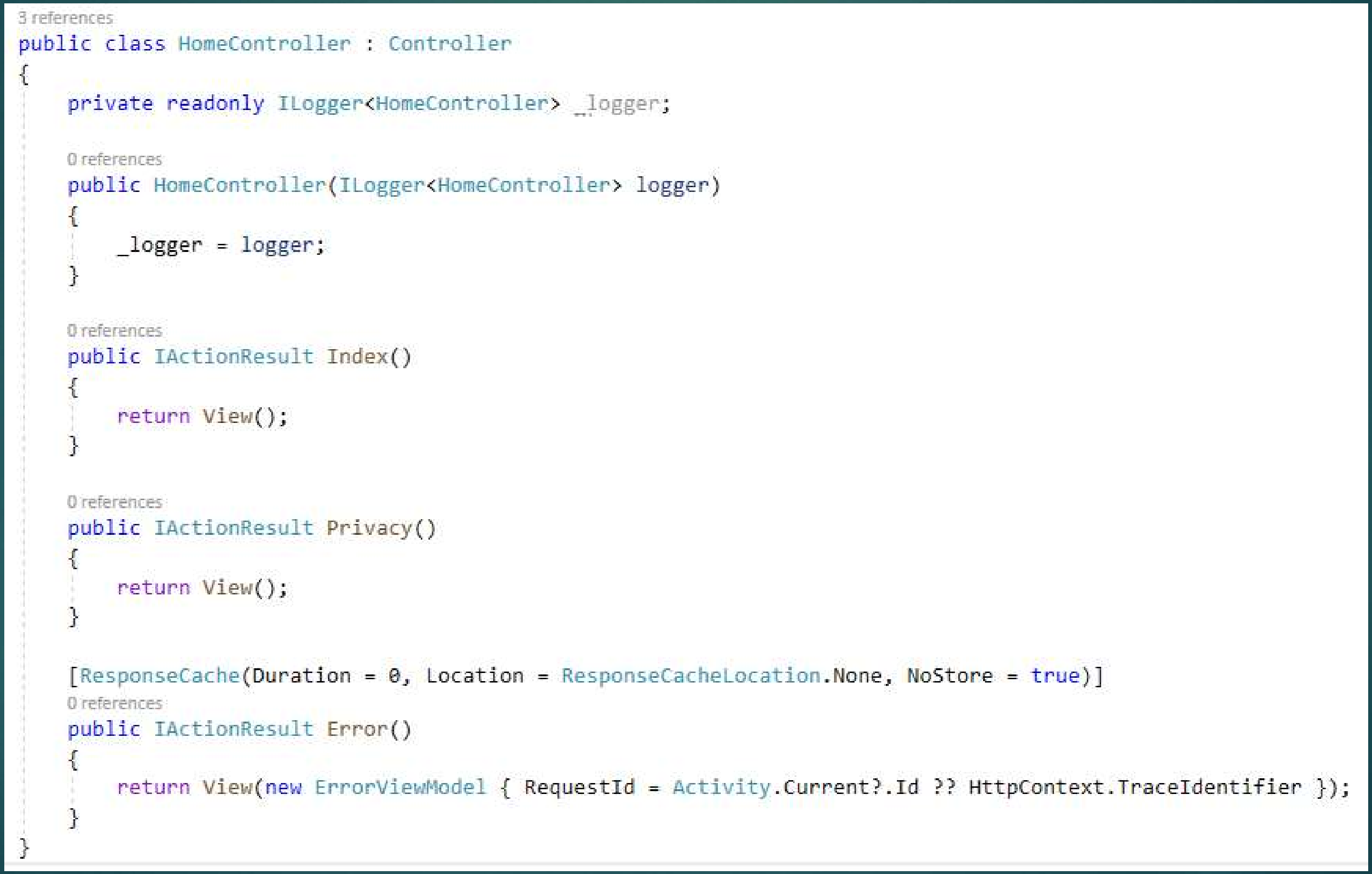
https://www.aspnethostingnews.com/asp-net-core-routing/



```
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");
});
```

Routing configured in Startup class

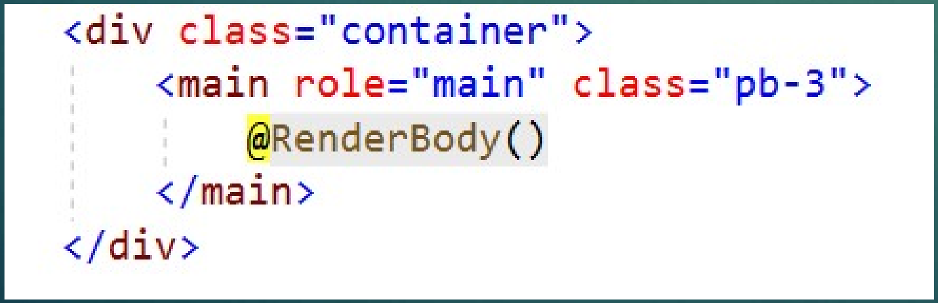
Controllers



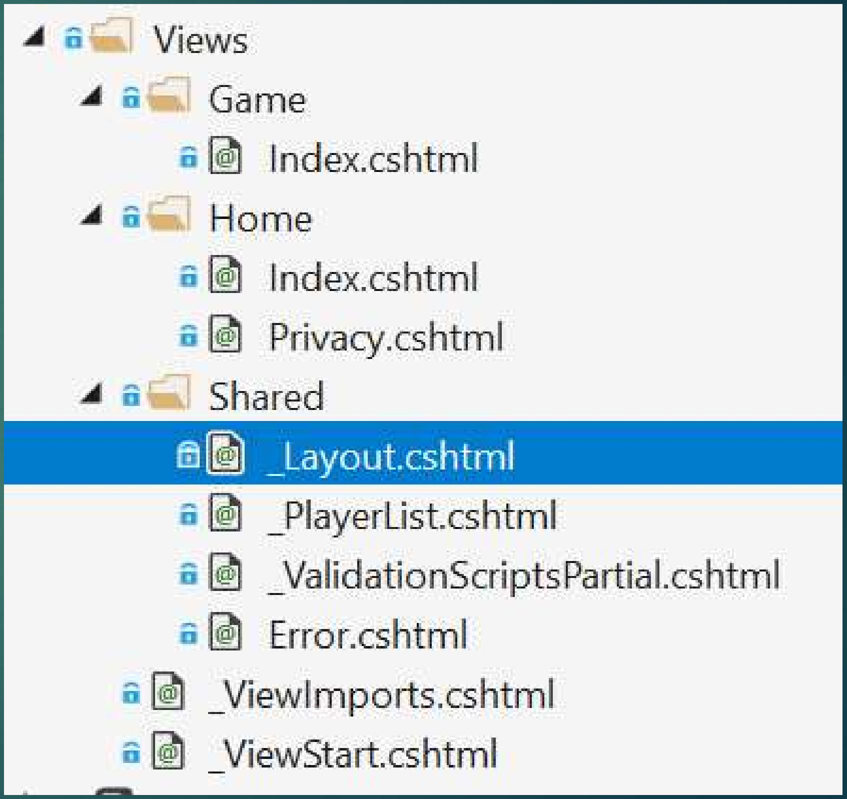
```
3 references
public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;

    0 references
    public HomeController(ILogger<HomeController> logger)
    {
        _logger = logger;
    }

    0 references
    public IActionResult Index()
    {
        return View();
    }

    0 references
    public IActionResult Privacy()
    {
        return View();
    }

    [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
    0 references
    public IActionResult Error()
    {
        return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
    }
}
```

Razor / Views

- Razor is a templating language
- Mix c# and HTML
- Can use a Razor Layout and partial Layouts
- Can use a data model
- @using declarations for namespaces
- Start expressions with @
- Wrap code-blocks with @{}

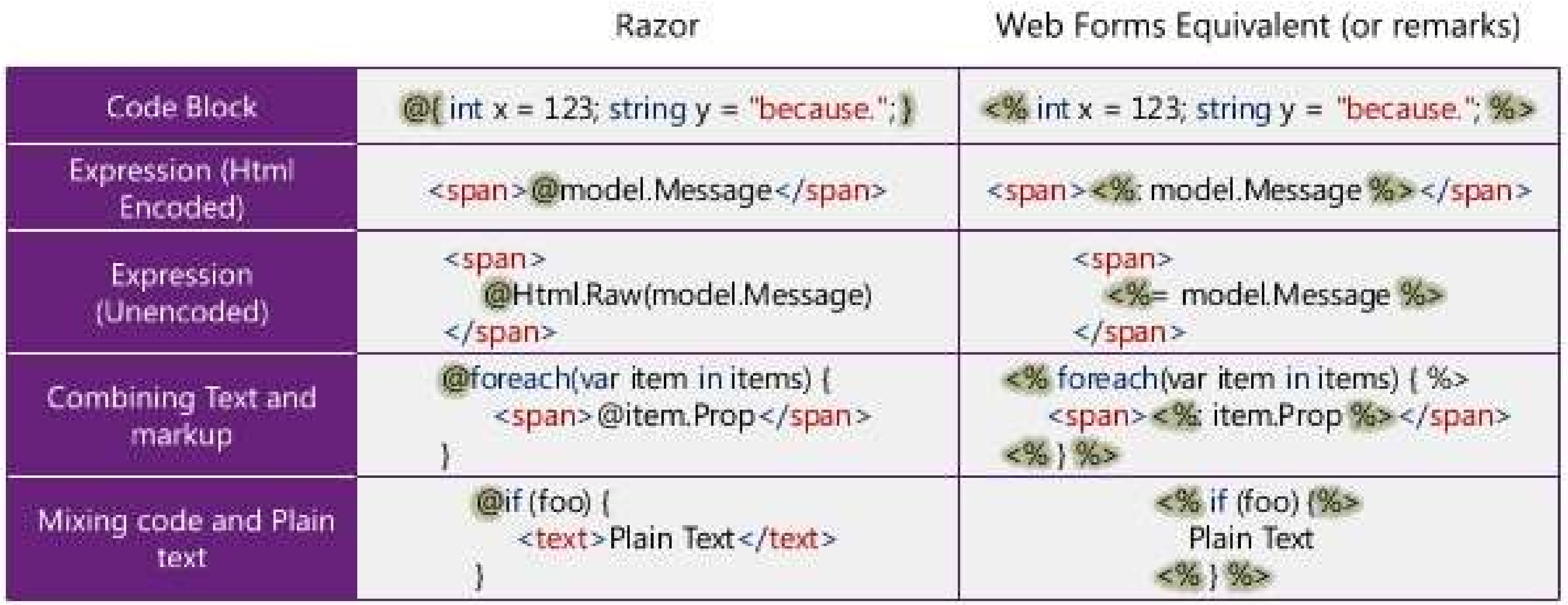


```
<div class="container">
    <main role="main" class="pb-3">
        @RenderBody()
    </main>
</div>
```



```
◢ Views
    ◢ Game
          Index.cshtml
    ◢ Home
          Index.cshtml
          Privacy.cshtml
    ◢ Shared
          _Layout.cshtml
          _PlayerList.cshtml
          _ValidationScriptsPartial.cshtml
          Error.cshtml
      _ViewImports.cshtml
      _ViewStart.cshtml
```

C# Razor Syntax Quick Reference



	Razor	Web Forms Equivalent (or remarks)
Code Block	`@{ int x = 123; string y = "because."; }`	`<% int x = 123; string y = "because."; %>`
Expression (Html Encoded)	`<span>@model.Message</span>`	`<span><%: model.Message %></span>`

Expression (Unencoded)	`<span>`
	`@Html.Raw(model.Message)`
	`</span>`
		`<span>`
		`<%= model.Message %>`
		`</span>`

Combining Text and markup	`@foreach(var item in items) {`
	`<span>@item.Prop</span>`
	`}`
		`<% foreach(var item in items) { %>`
		`<span><%: item.Prop %></span>`
		`<% } %>`

Mixing code and Plain text	`@if (foo) {`
	`<text>Plain Text</text>`
	`}`
		`<% if (foo) { %>`
		`Plain Text`
		`<% } %>`

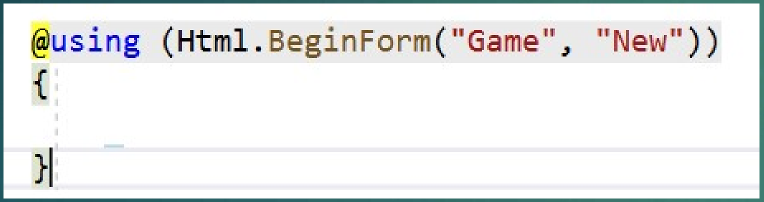
https://www.slideshare.net/RenierServen/aspnet-mvc-razor-syntax

TagHelpers

Tag Helpers are attached to HTML elements inside your Razor views **and** can help you write markup that is both cleaner **and** easier to read than the traditional **HTML Helpers**. **HTML Helpers**, on the other hand, are invoked as methods that are mixed with HTML inside your Razor views.

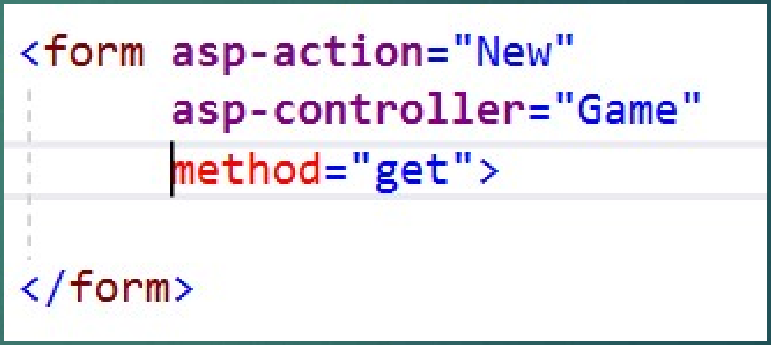
HtmlHelpers vs TagHelpers

HtmlHelpers



```
@using (Html.BeginForm("Game", "New"))
{

}
```

TagHelpers

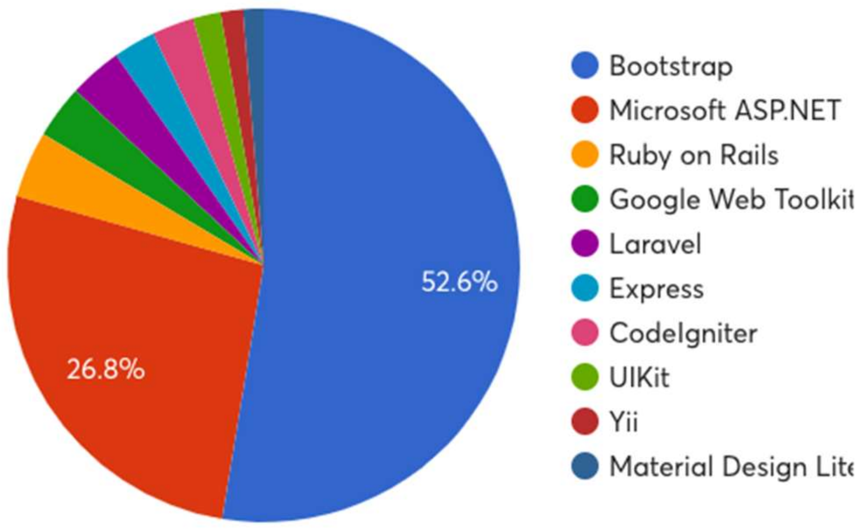


```
<form asp-action="New"
      asp-controller="Game"
      method="get">

</form>
```

Bootstrap

- Also known as Twitter Bootstrap
- Extremely popular basic ‘front-end framework’.
- CSS and JS (based on jQuery).
- Known for its Grid system
- Many components
- Many themes available

Front-end frameworks 2018



https://www.ostraining.com/blog/webdesign/bootstrap-popular/

Code time

Homework after session #3

- Add new web project to solution
- Prepare the game for the web
- Create a service to help load and save games
- Build a form that takes player name
- Build a controller that starts a new game and displays game state
- Add a view model, view and css to show a game state properly