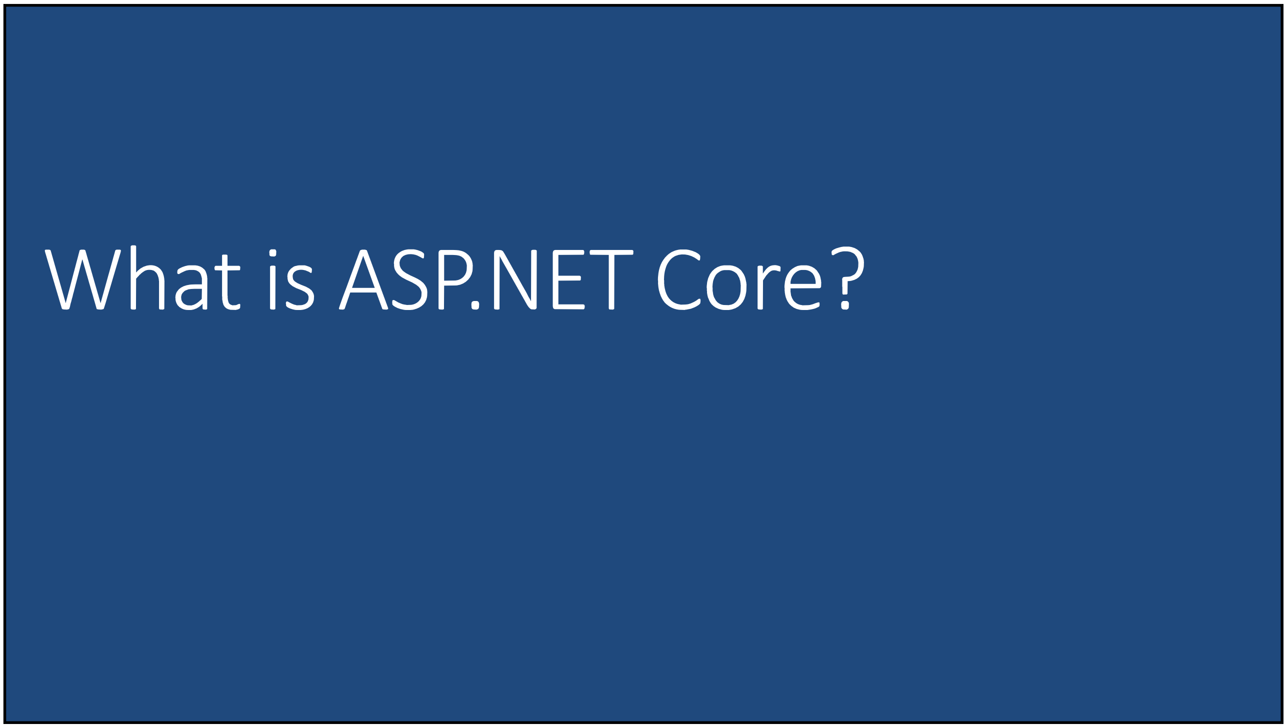
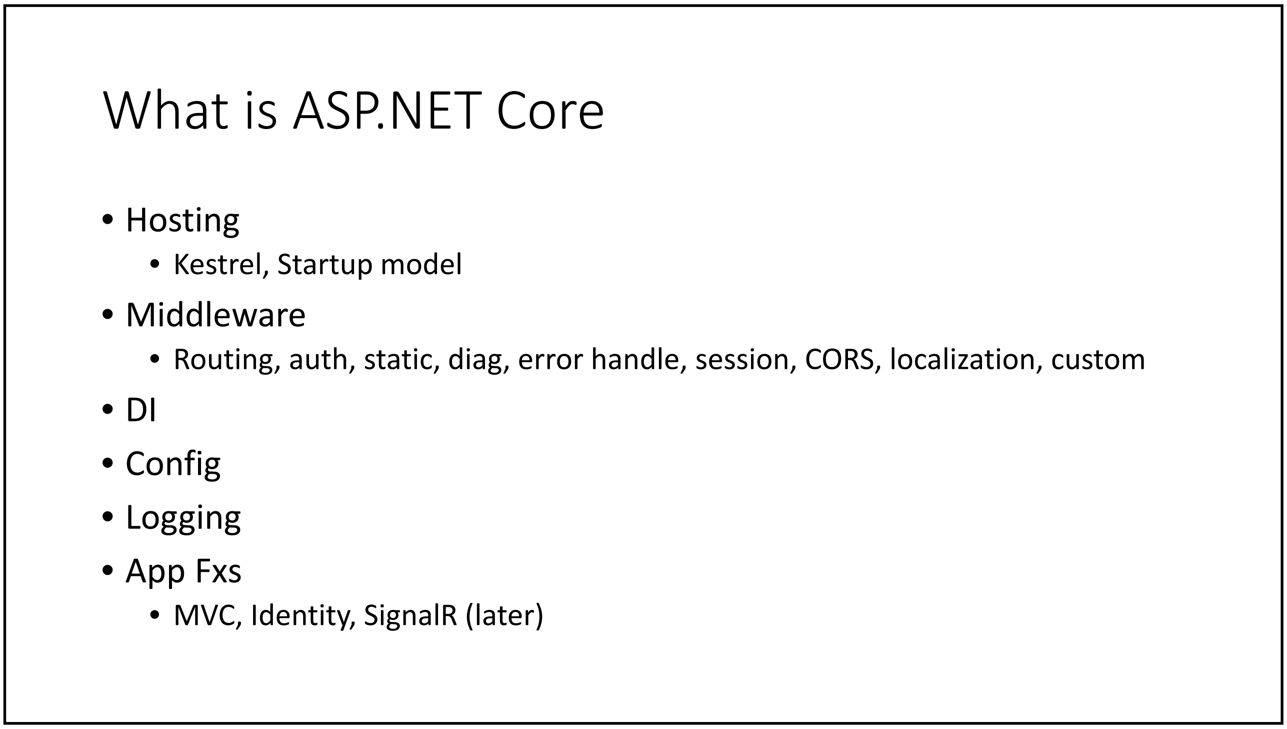
Microsoft
Visual Studio LIVE! EXPERT SOLUTIONS FOR .NET DEVELOPERS
San Diego
Code Again for the First Time!
Visual Studio 25 YEARS OF CODING INNOVATION

Build Data Driven Apps with ASP.NET Core

Rachel Appel
Sr Content Developer for Azure
Microsoft
rachelap@microsoft.com

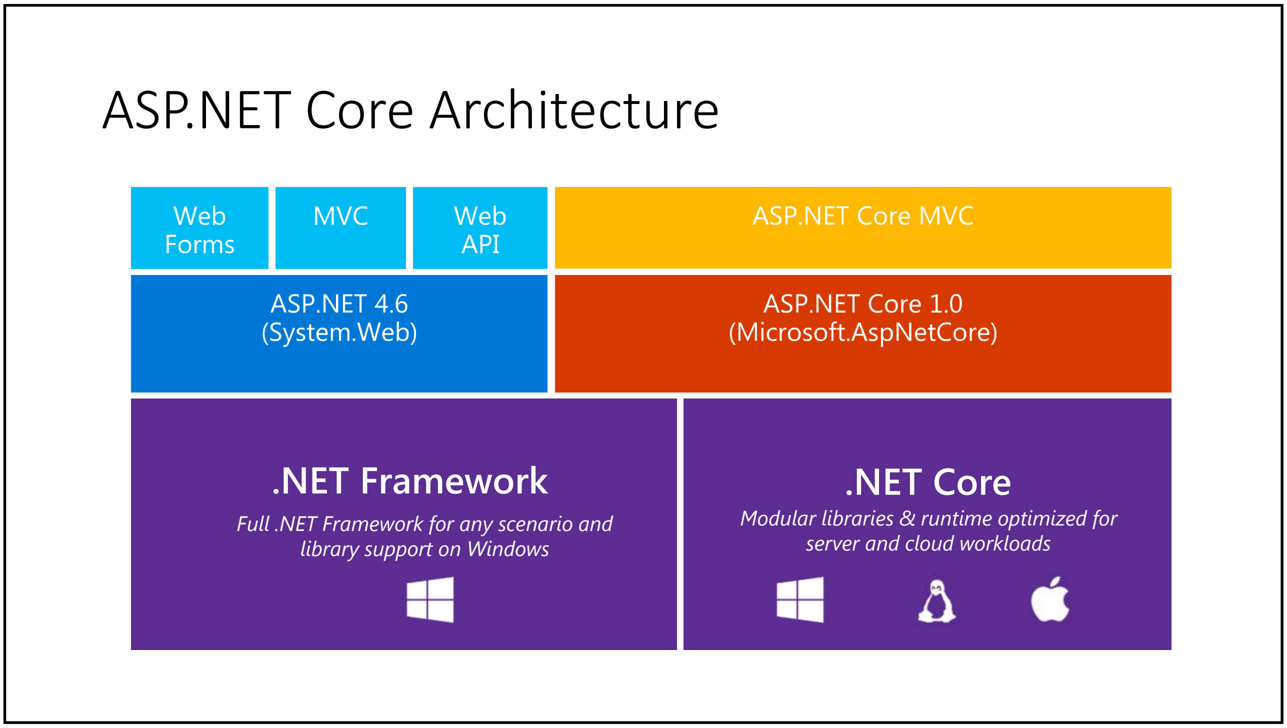


What is ASP.NET Core?



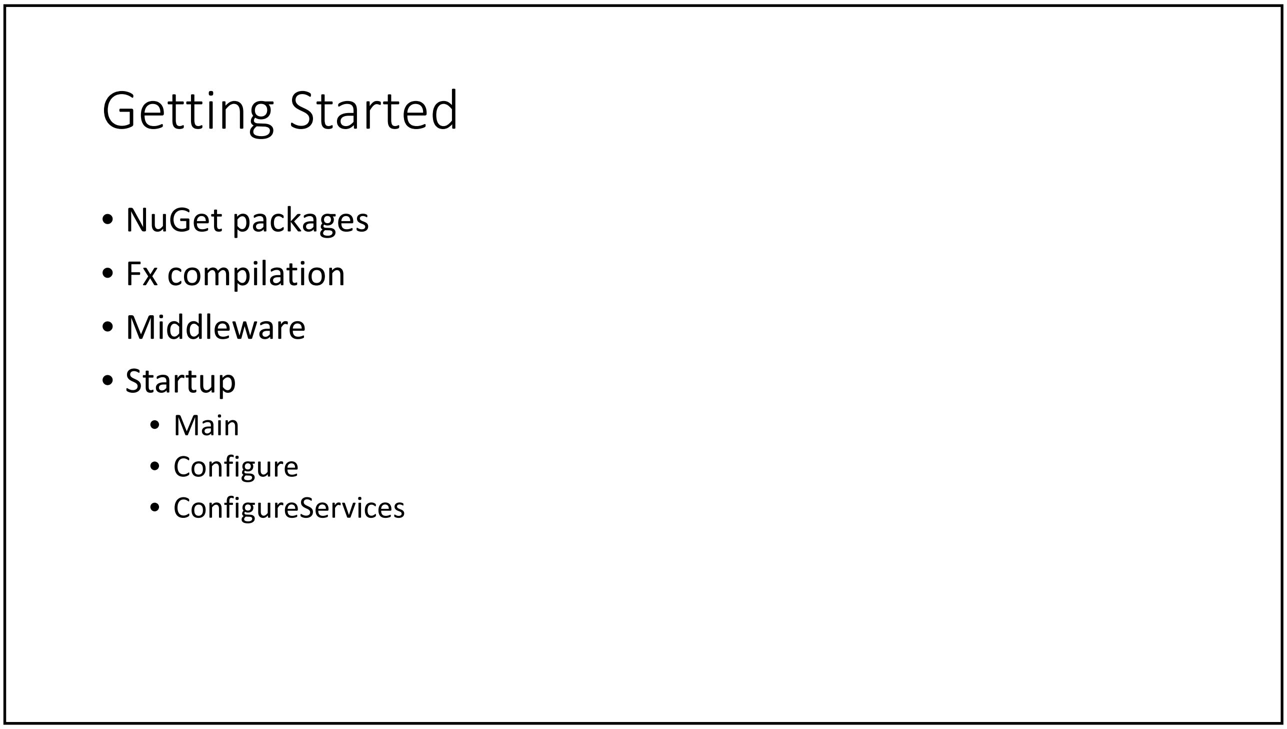
What is ASP.NET Core

- Hosting
    - Kestrel, Startup model
- Middleware
    - Routing, auth, static, diag, error handle, session, CORS, localization, custom
- DI
- Config
- Logging
- App Fxs
    - MVC, Identity, SignalR (later)



ASP.NET Core Architecture

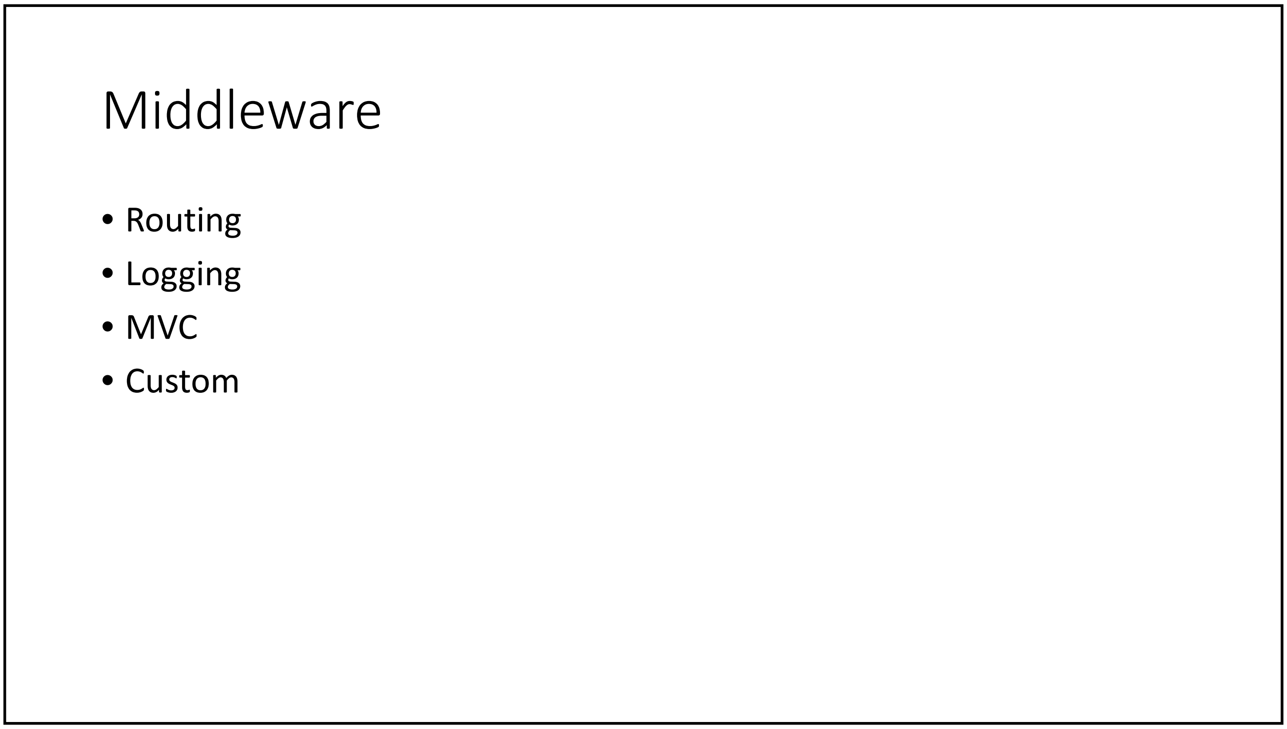
Web Forms
MVC
Web API
ASP.NET Core MVC
ASP.NET 4.6 (System.Web)
ASP.NET Core 1.0 (Microsoft.AspNetCore)
.NET Framework
Full .NET Framework for any scenario and library support on Windows
.NET Core
Modular libraries & runtime optimized for server and cloud workloads



Getting Started

- NuGet packages
- Fx compilation
- Middleware
- Startup
  - Main
  - Configure
  - ConfigureServices

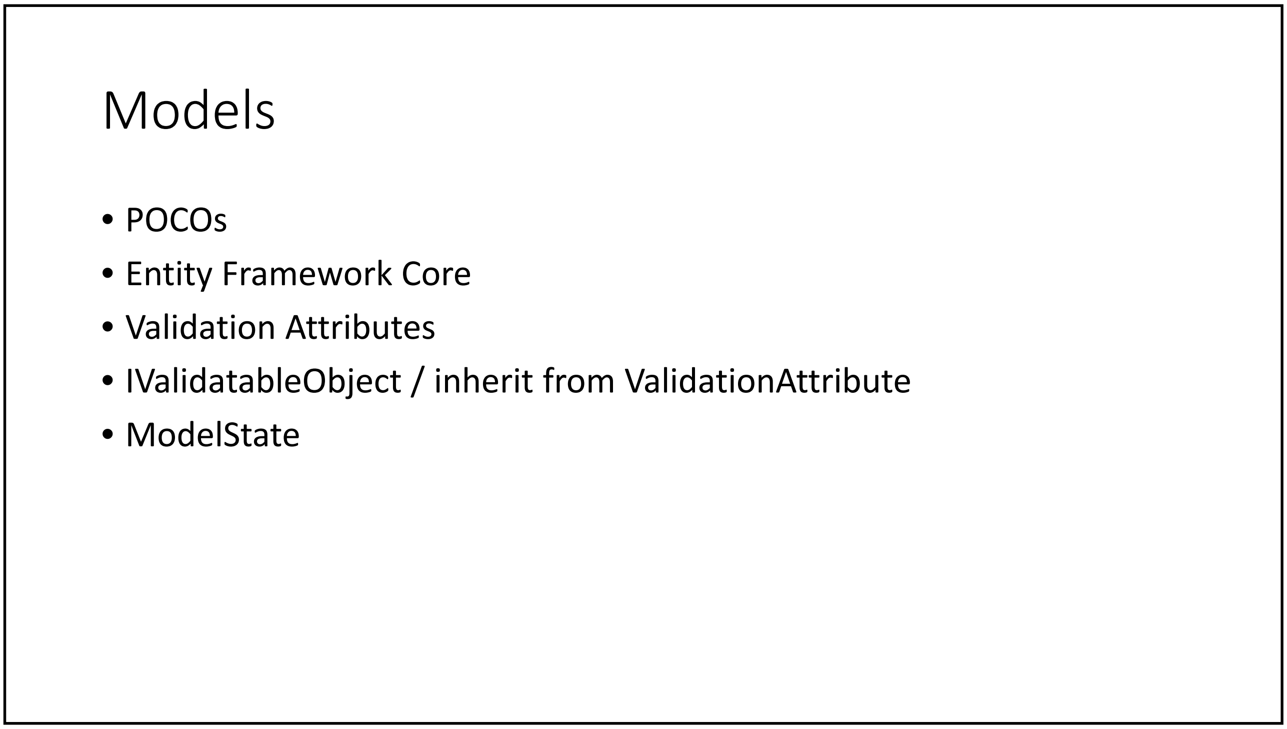
Middleware



Middleware

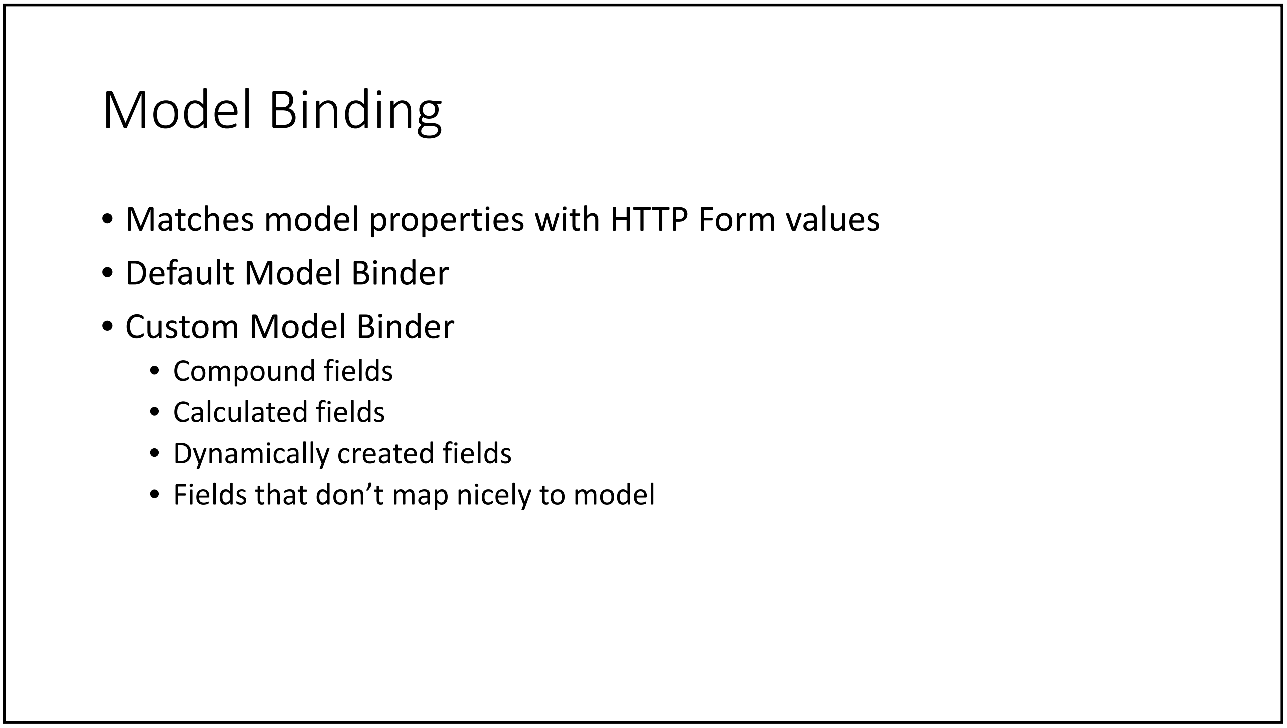
- Routing
- Logging
- MVC
- Custom

Models



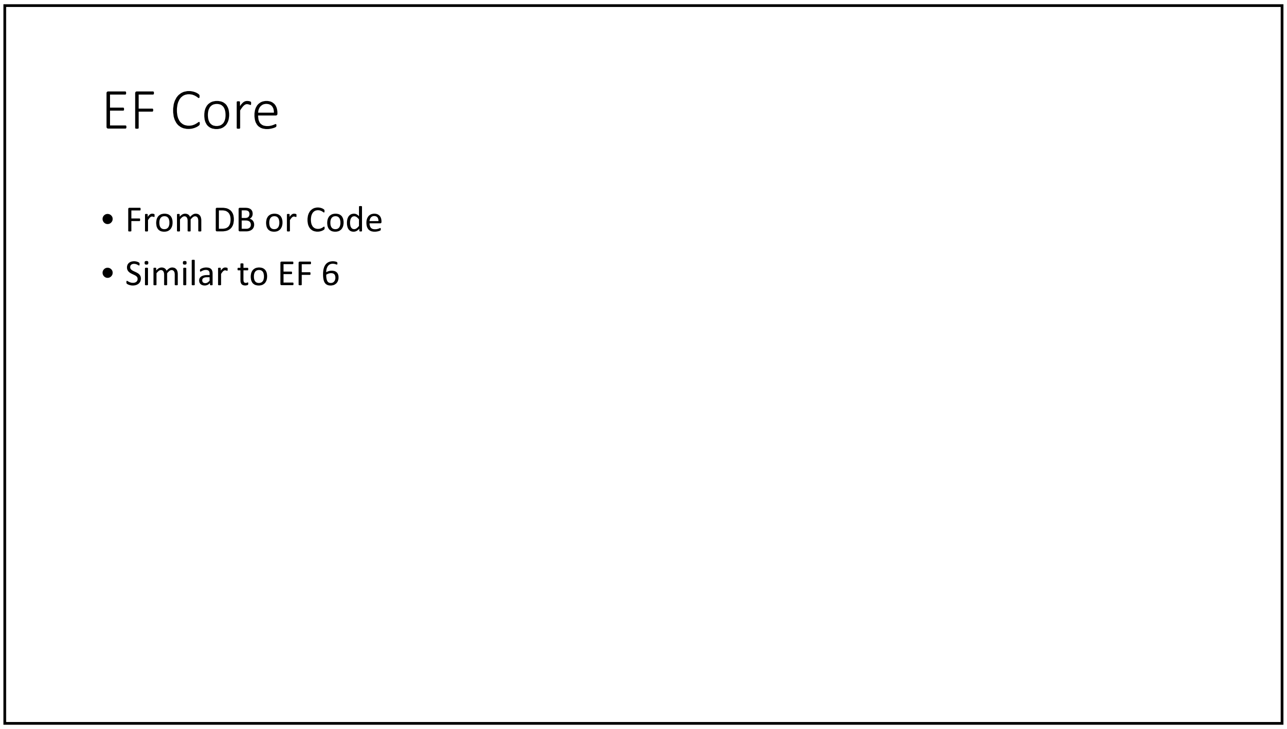
Models

- POCOs
- Entity Framework Core
- Validation Attributes
- IValidatableObject / inherit from ValidationAttribute
- ModelState



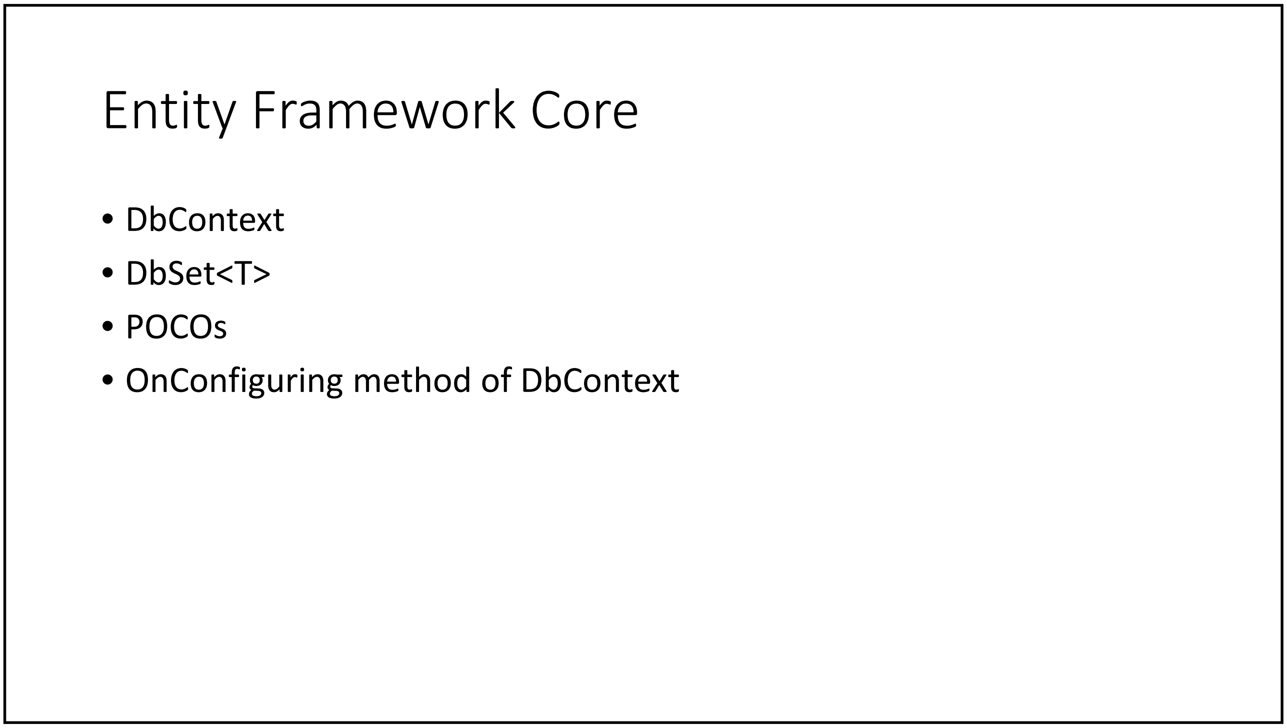
Model Binding

- Matches model properties with HTTP Form values
- Default Model Binder
- Custom Model Binder
  - Compound fields
  - Calculated fields
  - Dynamically created fields
  - Fields that don’t map nicely to model



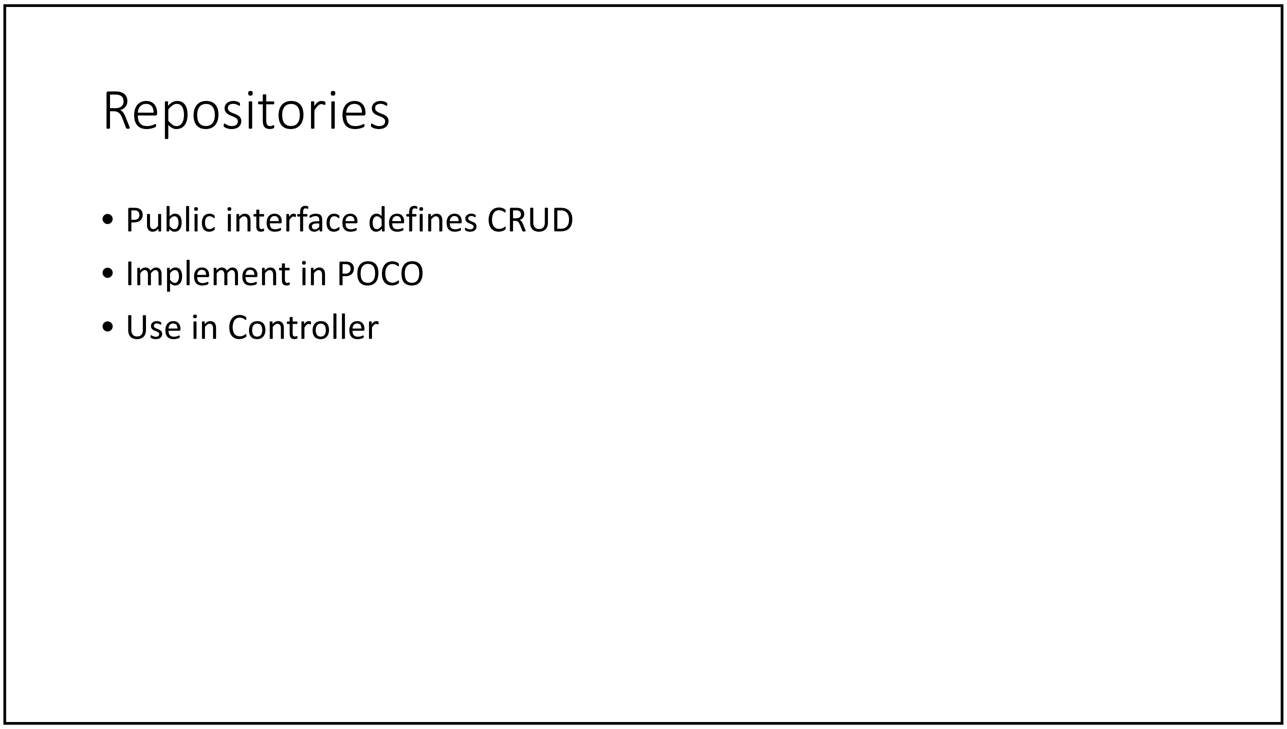
EF Core

- From DB or Code
- Similar to EF 6



Entity Framework Core

- DbContext
- DbSet<T>
- POCOs
- OnConfiguring method of DbContext



Repositories

- Public interface defines CRUD
- Implement in POCO
- Use in Controller

Controllers

Controllers

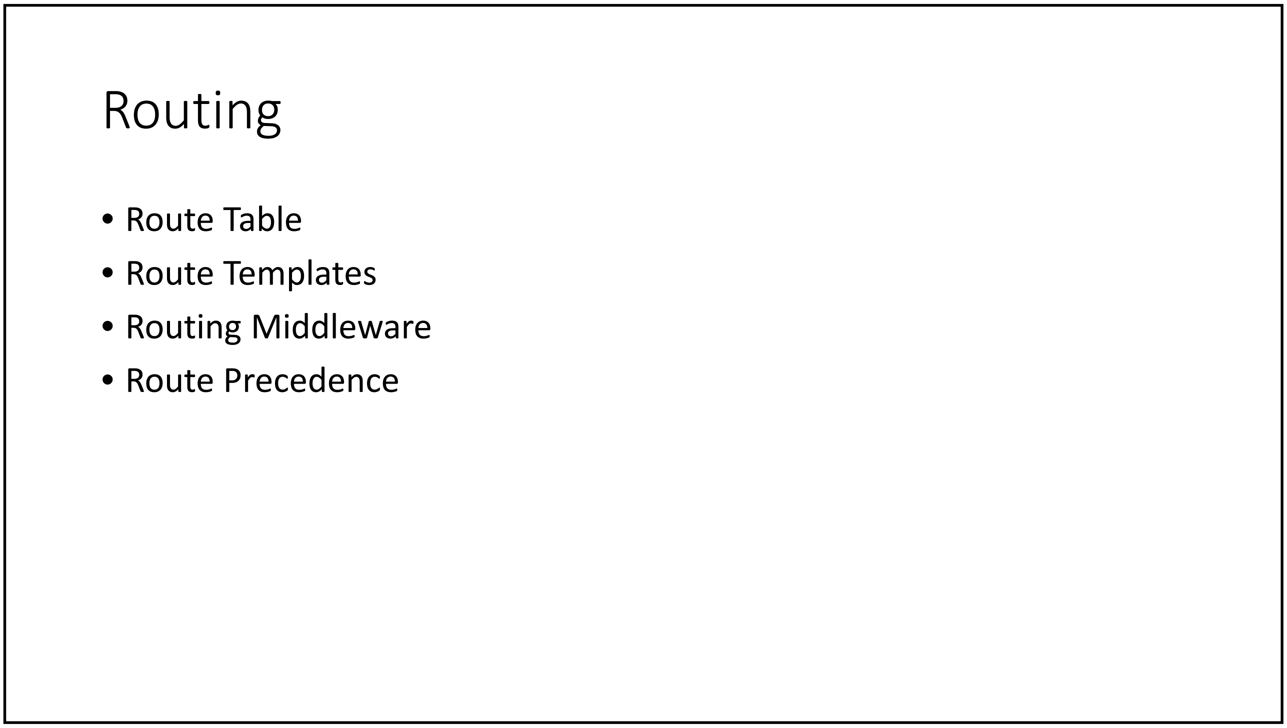
- HTTP POST Data & HTTP Verbs

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit(Product product)
{
    if (ModelState.IsValid)
    {
        db.Entry(product).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(product);
}
```

Controllers

- API vs. Standard

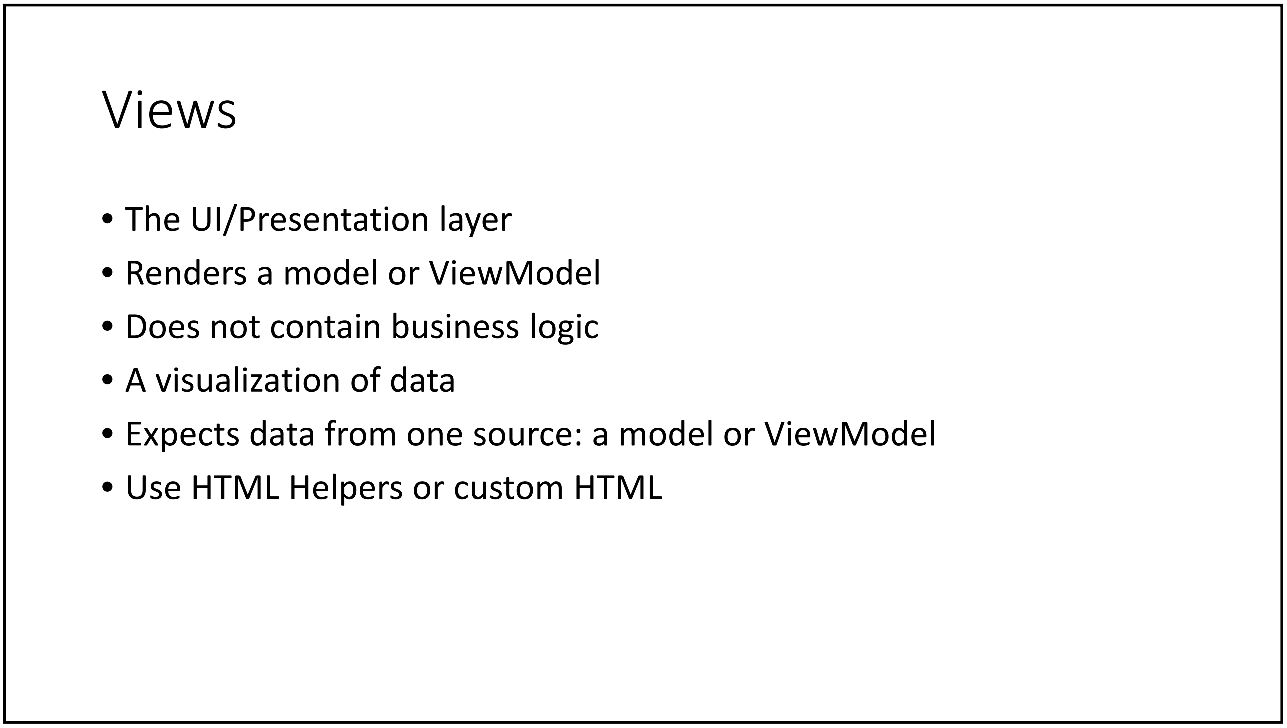
Routing



Routing

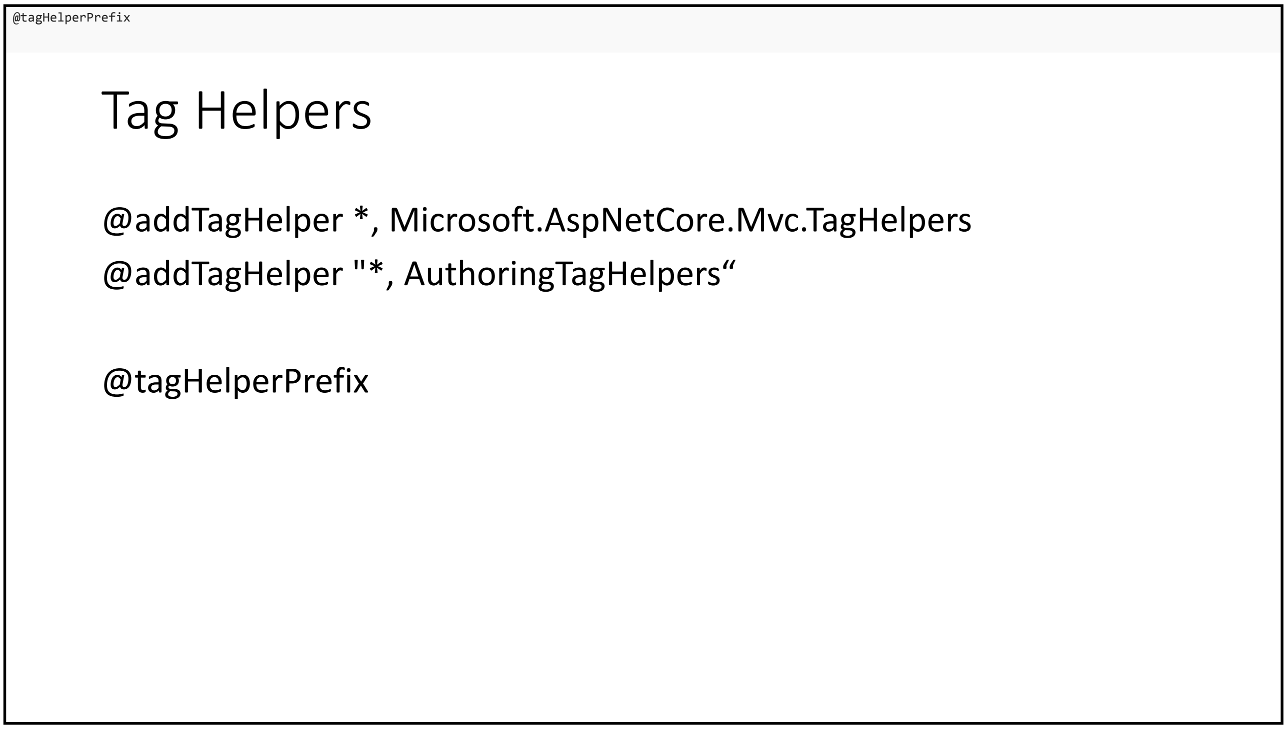
- Route Table
- Route Templates
- Routing Middleware
- Route Precedence

Views



Views

- The UI/Presentation layer
- Renders a model or ViewModel
- Does not contain business logic
- A visualization of data
- Expects data from one source: a model or ViewModel
- Use HTML Helpers or custom HTML



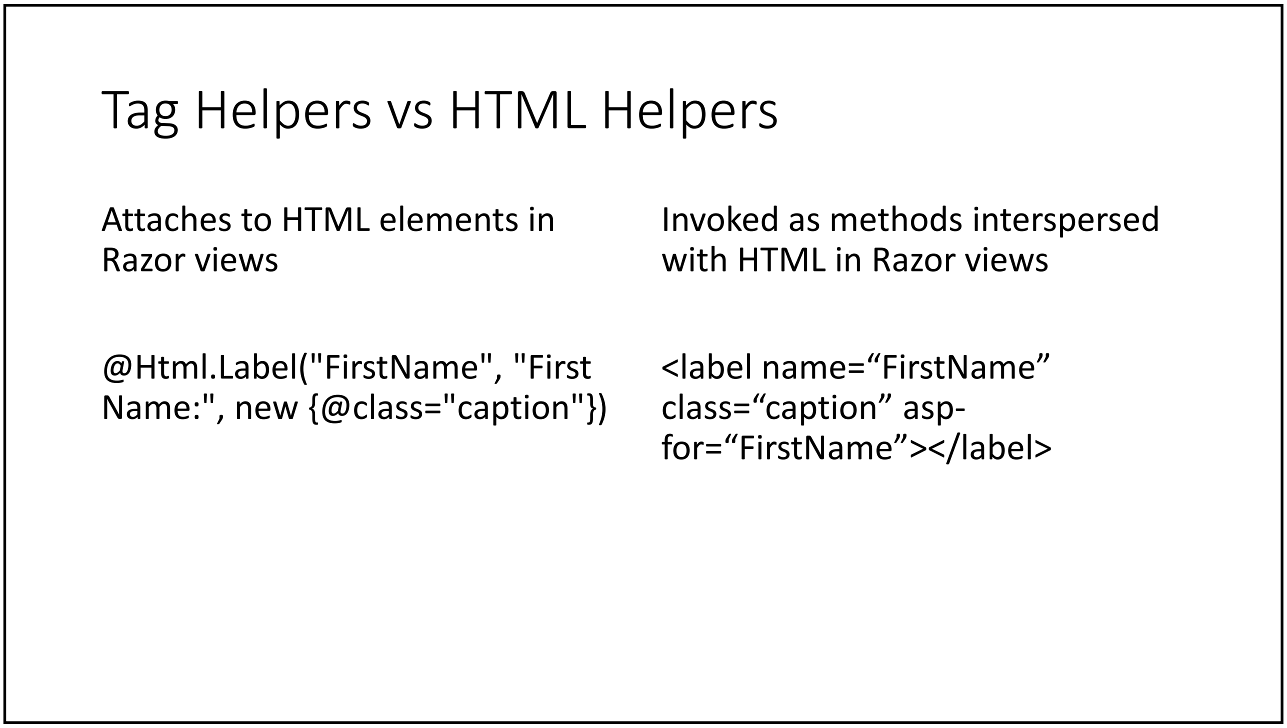
@tagHelperPrefix

Tag Helpers

@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers

@addTagHelper "*, AuthoringTagHelpers“

@tagHelperPrefix



Tag Helpers vs HTML Helpers

Attaches to HTML elements in Razor views	Invoked as methods interspersed with HTML in Razor views
@Html.Label("FirstName", "First Name:", new {@class="caption"})	<label name=“FirstName” class=“caption” asp-for=“FirstName”></label>

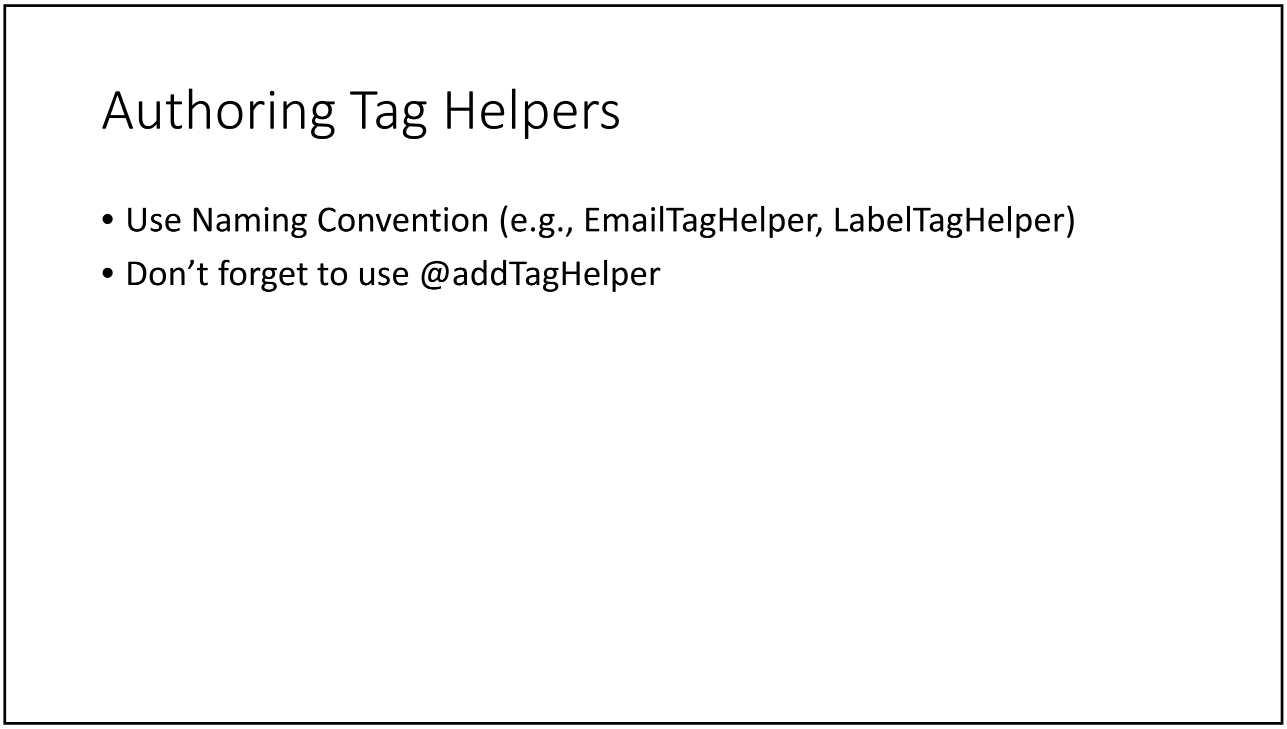
Tag Helpers vs Web Server Controls

- Trivial lifecycle
- No DOM
- No Browser detection
- Apply multiple Tag Helpers to same object

- Non trivial lifecycle
- Access DOM via ClientID
- Browser detection
- One Control per object

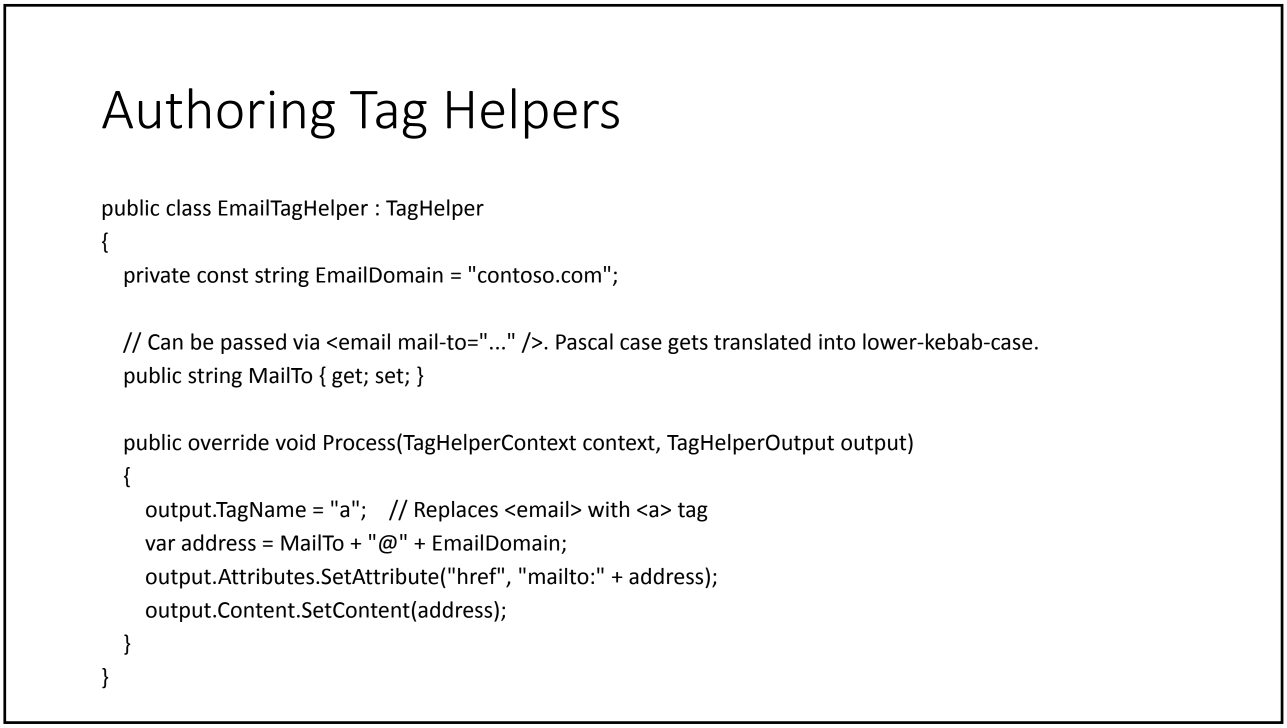
Authoring Tag helpers

1. Create class
2. Inherit from TagHelper
3. Override Process/ProcessAsync method



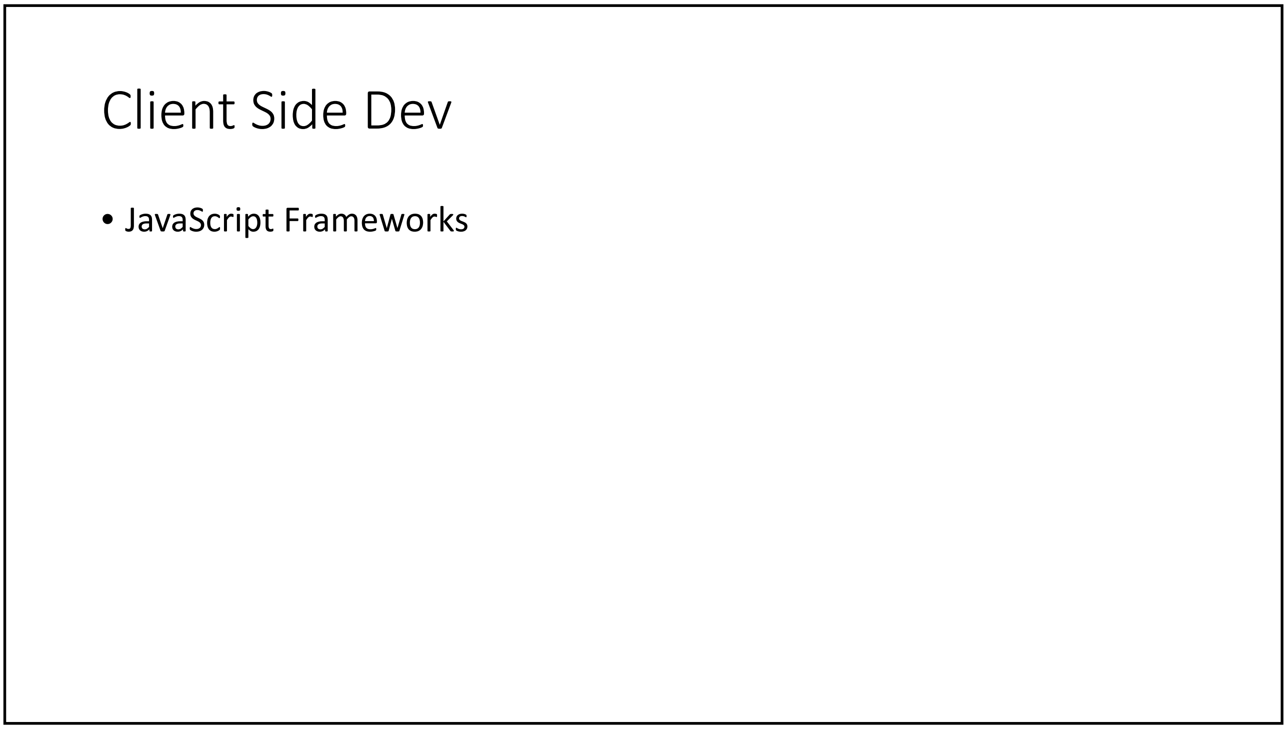
Authoring Tag Helpers

- Use Naming Convention (e.g., EmailTagHelper, LabelTagHelper)
- Don’t forget to use @addTagHelper



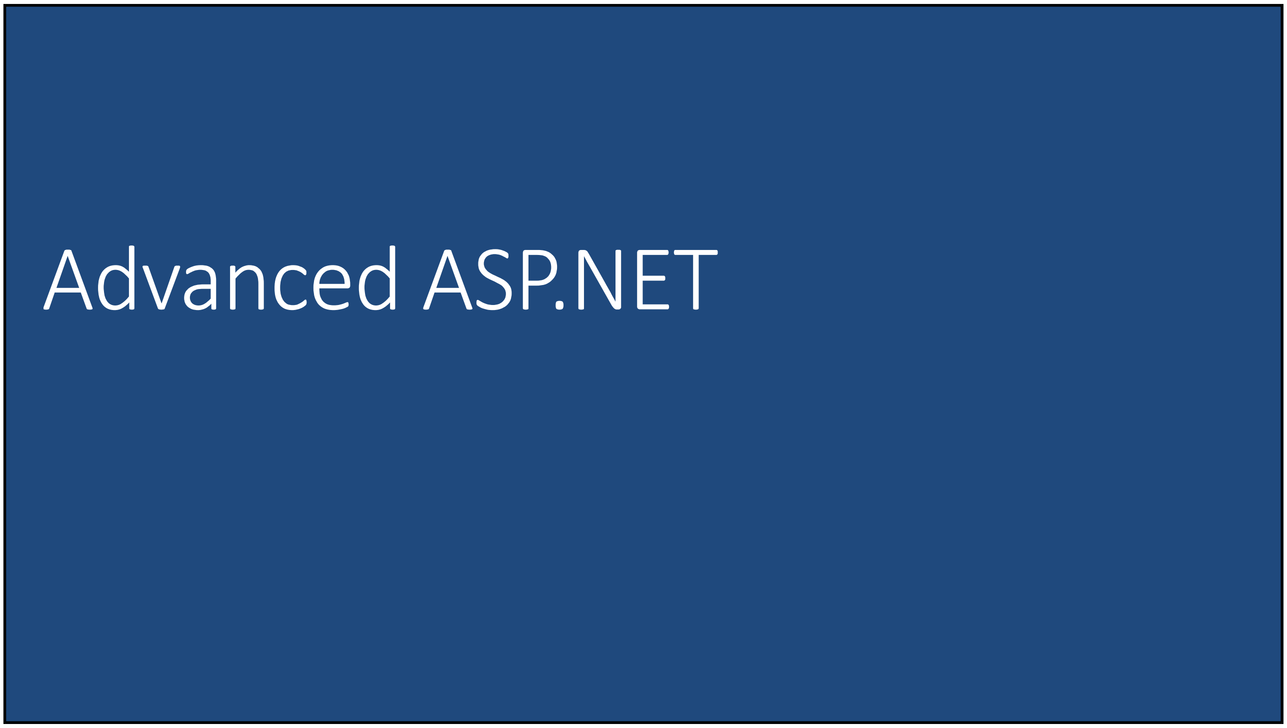
Authoring Tag Helpers

```
public class EmailTagHelper : TagHelper
{
    private const string EmailDomain = "contoso.com";

    // Can be passed via <email mail-to="..." />. Pascal case gets translated into lower-kebab-case.
    public string MailTo { get; set; }

    public override void Process(TagHelperContext context, TagHelperOutput output)
    {
        output.TagName = "a";    // Replaces <email> with <a> tag
        var address = MailTo + "@" + EmailDomain;
        output.Attributes.SetAttribute("href", "mailto:" + address);
        output.Content.SetContent(address);
    }
}
```

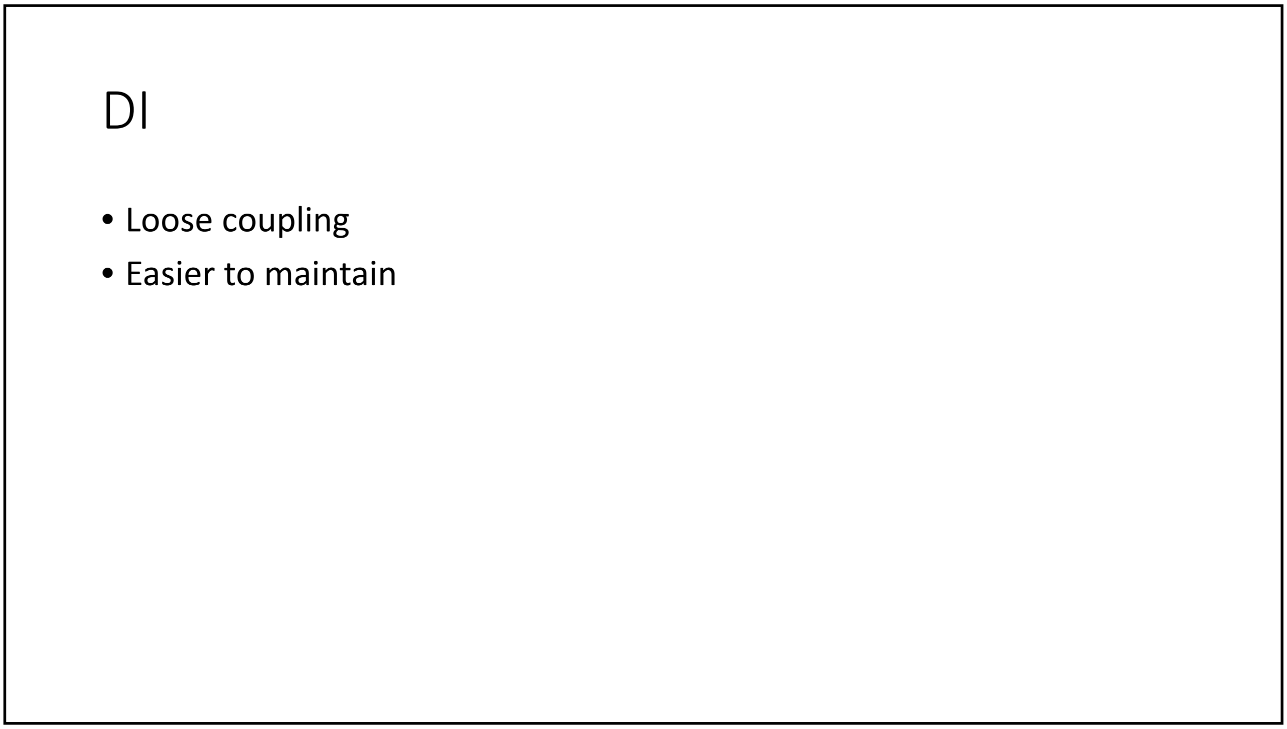


Client Side Dev

- JavaScript Frameworks

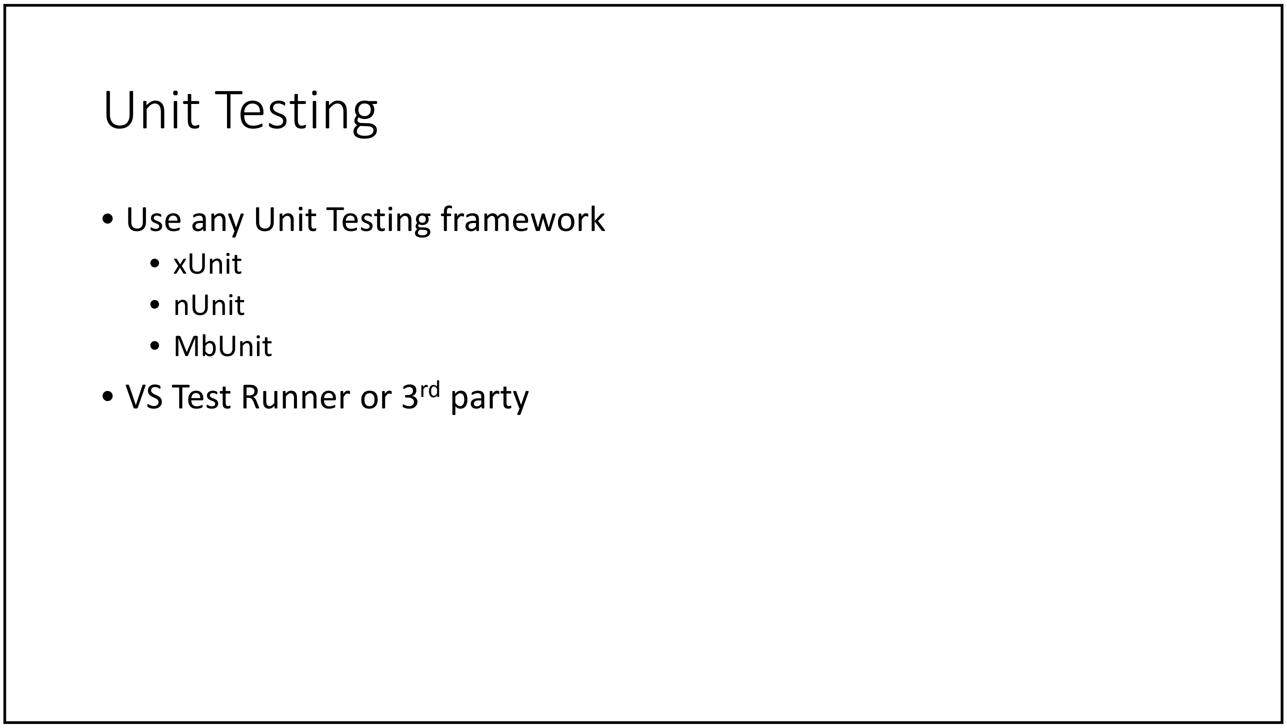


Advanced ASP.NET



DI

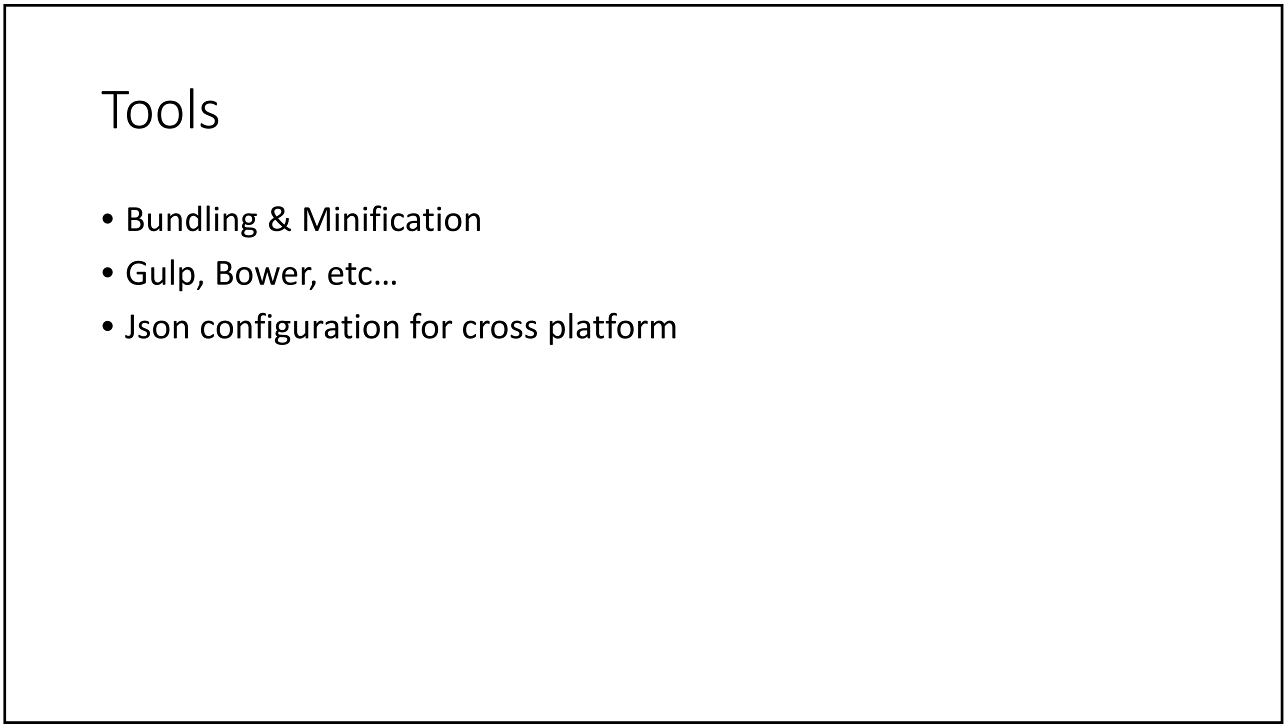
- Loose coupling
- Easier to maintain



Unit Testing

- Use any Unit Testing framework
  - xUnit
  - nUnit
  - MbUnit
- VS Test Runner or 3rd party

Tools & Deployment



Tools

- Bundling & Minification
- Gulp, Bower, etc…
- Json configuration for cross platform

Deployment

- Publish
  - Azure
  - Host
- FTP
- CI