

ASP.NET CORE 3.1 CHEAT SHEET

www.ezzylearning.net

ASP.NET Core Project Structure

.csproj	The csproj file includes settings re NuGet package references etc.
Startup.cs	This file configure services require pipeline (Middleware) in this file.
Program.cs	Program.cs file contains code to b host encapsulates all of the app's Configuration etc.
appsettings.json	The appsettings.json file is an app settings such as database connect
launchSettings.json	This file holds project specific sett is configured to use to launch the should be used. This file is only us required when we publishing our
wwwroot	The wwwroot folder contains all t files, CSS files, Fonts etc. These fil request.
Dependencies	The Dependencies node in the AS installed server-side NuGet packa

ASP.NET Core Services Registration Lifeti

ASP.NET Core 3.1 Cheat Sheet

Why ASP.NET Core?

1. We can build web apps and services
2. We can build Internet of Things (IoT) apps
3. We can use our favorite development tool on Windows, macOS, Linux
4. We can deploy apps to any Platform (Windows, macOS, Linux)
5. We can deploy apps to the cloud or on-premises
6. We need to learn a unified object model for both web UI and web APIs
7. We can create fully testable apps
8. We can use Razor Pages to implement page focused apps
9. We can use Blazor to create single page apps (SPA) in C#
10. We can integrate it with modern client side frameworks e.g. Angular
11. We can use built in dependency injection
12. We can use a lightweight, high performance and modular request pipeline
13. We can host it on Kestrel, IIS, Nginx, Apache, Docket etc.
14. We can use side by side versioning

ASP.NET Core Application Types

App Type	Scenario	What to use?
Web App	New server-side web UI development	Razor Pages
Web App	Migrating existing ASP.NET MVC App	ASP.NET Core MVC
Web App	Client side web UI development	Blazor
Web API	RESTful HTTP services	ASP.NET Web APIs
Real-time Apps	Communicating between server and connected clients	SignalR

What is ASP.NET Core?

ASP.NET Core is a cross-platform, high-performance, open-source framework for building modern, cloud-enabled, Internet-connected apps. ASP.NET is a popular web-development framework for building web apps on the .NET platform.



ASP.NET Core Project Structure

.csproj	The csproj file includes settings related to targeted .NET Frameworks, project folders, NuGet package references etc.
Startup.cs	This file configure services required by the app. We also define the app request handling pipeline (Middleware) in this file.
Program.cs	Program.cs file contains code to build ASP.NET Core app host on application startup. The host encapsulates all of the app's resources, such as Logging, Dependency Injection, Configuration etc.
appsettings.json	The appsettings.json file is an application configuration file used to store configuration settings such as database connections strings, any application scope global variables.
launchSettings.json	This file holds project specific settings associated with each debug profile, Visual Studio is configured to use to launch the application, including any environment variables that should be used. This file is only used within the local development machine and not required when we publishing our asp.net core application to the production server.
wwwroot	The wwwroot folder contains all the static files in our project such as images, JavaScript files, CSS files, Fonts etc. These files are then served directly to clients over HTTP request.
Dependencies	The Dependencies node in the ASP.NET Core 3.1 solution explorer contain all the installed server-side NuGet packages, a list of Analyzers and SDKs used in the project.

ASP.NET Core Services Registration Lifetimes

Transient	Transient lifetime services are created each time they're requested from the service container and disposed at the end of the request. This means if a service is accessed multiple times within a same request, a new service object will be created every time.
Scoped	Scoped lifetime services are created once per client request (connection). This means if a service is accessed multiple times within a same request, the same service object will be reused in that particular request scope.
Singleton	Singleton lifetime services are created first time they are requested and then every subsequent request uses the same instance.

ASP.NET Core HTML Helpers

Html.ActionLink	Output HTML <a> element
Html.BeginForm	Output HTML <form> element
Html.EndForm	Output HTML </form> element
Html.CheckBox	Output HTML <input type='checkbox' /> element
Html.RadioButton	Output HTML <input type='radio' /> element
Html.TextBox	Output HTML <input type='text' /> element
Html.Password	Output HTML <input type='password' /> element
Html.Hidden	Output HTML <input type='hidden' /> element
Html.DropDownList	Output HTML <select> element
Html.ListBox	Output HTML <select multiple> element
Html.TextArea	Output HTML <textarea> element
Html.Encode	Encodes HTML contents
Html.AntiForgeryToken	Prevents Cross-Site Request Forgery (CSRF) attacks

ASP.NET Core Filter Types

Authorization	Authorization filters run first and are used to determine whether the user is authorized for the request. Authorization filters short-circuit the pipeline if the request is not authorized.
Resource	Resource filters run after authorization.
Action	Action filters run code immediately before and after an action method is called. They can change the arguments passed into an action. They can also change the result returned from the action. They are not supported in Razor Pages.
Exception	Exception filters apply global policies to unhandled exceptions that occur before the response body has been written to.
Result	Result filters run code immediately before and after the execution of action results. They run only when the action method has executed successfully. They are useful for logic that must surround view or formatter execution.



ASP.NET Core Built-in Tag Helpers

Anchor	Enhances the standard HTML anchor (<a ... >) tag by adding new attributes
Cache	Provides the ability to improve the performance of your ASP.NET Core app by caching its content to the internal ASP.NET Core cache provider.
Component	Renders a component from a page or view
Distributed Cache	Provides the ability to dramatically improve the performance of your ASP.NET Core app by caching its content to a distributed cache source.
Environment	Conditionally renders its enclosed content based on the current hosting environment.
Form	Generates the HTML <FORM> action attribute value for a MVC controller action or named route.
Form Action	The Form Action Tag Helper generates the form action attribute on the generated <button ...> or <input type="image" ...> tag
Image	The Image Tag Helper enhances the tag to provide cache-busting behavior for static image files.
Input	The Input Tag Helper binds an HTML <input> element to a model expression in your razor view.
Label	Generates the label caption and for attribute on a <label> element for an expression name
Link	The Link Tag Helper generates a link to a primary or fall back CSS file. Typically the primary CSS file is on a Content Delivery Network (CDN).
Partial	The Partial Tag Helper is used for rendering a partial view in Razor Pages and MVC apps.
Script	The Script Tag Helper generates a link to a primary or fall back script file. Typically the primary script file is on a Content Delivery Network (CDN).
Select	Generates select and associated option elements for properties of your model.
Textarea	Generates the id and name attributes, and the data validation attributes from the model for a <textarea> element.
Validation Message	Adds the HTML5 data-valmsg-for="property" attribute to the span element, which attaches the validation error messages on the input field of the specified model property. When a client side validation error occurs, jQuery displays the error message in the element. Validation also takes place on the server. Clients may have JavaScript disabled and some validation can only be done on the server side.
Validation Summary	The Validation Summary Tag Helper is used to display a summary of validation messages. It targets <div> elements with the asp-validation-summary attribute

ASP.NET Core Built-in Middleware

Middleware	Description	Order
Authentication	Provides authentication support	Before HttpContext.User is needed
Authorization	Provides authorization support	Immediately after the Authentication middleware
Cookie Policy	Tracks consent from users for storing personal information.	Before middleware that issues cookies such as Authentication, Session etc.
CORS	Configures Cross-Origin Resource Sharing	Before components that use CORS
Diagnostics	Multiple separate middleware that provide a developer exception page, status code pages etc.	Before components that generate errors. Used for logging exceptions or serving default error pages for new apps.
Forwarded Headers	Forwards proxied headers onto the current request	Before components that consume the updated fields such as Scheme, Host, Client IP etc.
Health Check	Checks the health of an ASP.NET Core app and its dependencies, such as checking database availability.	Terminal if a request matches a health check endpoint.
HTTP Method Override	Allows an incoming POST request to override the method.	Before components that consume the updated method.
HTTPS Redirection	Redirect all HTTP requests to HTTPS.	Before components that consume the URL.
HTTP Strict Transport Security (HSTS)	Security enhancement middleware that adds a special response header.	Before responses are sent and after components that modify requests. Examples: Forwarded Headers, URL Rewriting.
MVC	Processes requests with MVC/Razor Pages.	Terminal if a request matches a route.
OWIN	Interop with OWIN-based apps, servers, and middleware.	Terminal if the OWIN Middleware fully processes the request.
Response Caching	Provides support for caching responses.	Before components that require caching. UseCORS must come before UseResponseCaching.
Response Compression	Provides support for compressing responses.	Before components that require compression.
Request Localization	Provides localization support.	Before localization sensitive components.
Endpoint Routing	Defines and constrains request routes.	Terminal for matching routes.
SPA	Handles all requests from this point in the middleware chain by returning the default page for the Single Page Application (SPA)	Late in the chain, so that other middleware for serving static files, MVC actions, etc., takes precedence.
Session	Provides support for managing user sessions.	Before components that require Session.
Static Files	Provides support for serving static files and directory browsing.	Terminal if a request matches a file.
URL Rewrite	Provides support for rewriting URLs and redirecting requests.	Before components that consume the URL.
WebSockets	Enables the WebSockets protocol.	Before components that are required to accept WebSocket requests.

ASP.NET Core Built-in Data Annotation Attributes

Attribute	Description	Attribute	Description
AssociationAttribute	Specifies that an entity member represents a data relationship, such as a foreign key relationship.	RangeAttribute	Specifies the numeric range constraints for the value of a data field.
CompareAttribute	Provides an attribute that compares two properties.	RegularExpressionAttribute	Specifies that a data field value in ASP.NET Dynamic Data must match the specified regular expression.
ConcurrencyCheckAttribute	Specifies that a property participates in optimistic concurrency checks.	RequiredAttribute	Specifies that a data field value is required.
CreditCardAttribute	Specifies that a data field value is a credit card number.	ScaffoldColumnAttribute	Specifies whether a class or data column uses scaffolding.
CustomValidationAttribute	Specifies a custom validation method that is used to validate a property or class instance.	StringLengthAttribute	Specifies the minimum and maximum length of characters that are allowed in a data field.
DataTypeAttribute	Specifies the name of an additional type to associate with a data field.	TimestampAttribute	Specifies the data type of the column as a row version.
DisplayAttribute	Provides a general-purpose attribute that lets you specify localizable strings for types and members of entity partial classes.	UIHintAttribute	Specifies the template or user control that Dynamic Data uses to display a data field.
DisplayColumnAttribute	Specifies the column that is displayed in the referred table as a foreign-key column.	UrlAttribute	Provides URL validation.
DisplayFormatAttribute	Specifies how data fields are displayed and formatted by ASP.NET Dynamic Data.	ValidationAttribute	Serves as the base class for all validation attributes.
EditableAttribute	Indicates whether a data field is editable.	ValidationContext	Describes the context in which a validation check is performed.
EmailAddressAttribute	Validates an email address.	ValidationException	Represents the exception that occurs during validation of a data field when the ValidationAttribute class is used.
EnumDataTypeAttribute	Enables a .NET Framework enumeration to be mapped to a data column.	ValidationResult	Represents a container for the results of a validation request.
FileExtensionsAttribute	Validates file name extensions.	Validator	Defines a helper class that can be used to validate objects, properties, and methods when it is included in their associated ValidationAttribute attributes.
FilterUIHintAttribute	Represents an attribute that is used to specify the filtering behavior for a column.	<div> www.ezzylearning.net     </div>	
KeyAttribute	Denotes one or more properties that uniquely identify an entity.		
MaxLengthAttribute	Specifies the maximum length of array or string data allowed in a property.		
MetadataTypeAttribute	Specifies the metadata class to associate with a data model class.		
MinLengthAttribute	Specifies the minimum length of array or string data allowed in a property.		
PhoneAttribute	Specifies that a data field value is a well-formed phone number.		

ASP.NET Core Configuration Providers

Configuration Source	Description	Class or Package
Setting Files e.g. appsettings.json	These providers load configuration from different type of files <ul style="list-style-type: none">INI Configuration providerJSON Configuration providerXML Configuration provider	<ul style="list-style-type: none">IniConfigurationProviderJsonConfigurationProviderXmlConfigurationProvider
Environment Variables	This provider load configuration from environment variable key-value pairs	EnvironmentVariablesConfigurationProvider
Azure Key Vault	This provider load app configuration values from Azure Key Vault secrets	Microsoft.Extensions.Configuration.AzureKeyVault
Azure App Configuration	Uses Azure App Configuration to centralize storage and management of application settings for an ASP.NET Core app	Microsoft.Azure.AppConfiguration.AspNetCore
Command-line arguments	Loads configuration from command-line argument key-value pairs	CommandLineConfigurationProvider
Custom Providers	A basic configuration provider that reads configuration key-value pairs from custom sources e.g. reading settings from database using Entity Framework	ConfigurationProvider
Directory Files	Uses a directory's files as configuration key-value pairs	KeyPerFileConfigurationProvider
In-Memory .NET Objects	Uses an in-memory collection as configuration key-value pairs.	MemoryConfigurationProvider

ASP.NET Core Directive Attributes for Razor Components

Directive	Description	Example
@attributes	Allows a component to render non-declared attributes	<input id="useAttributesDict" @attributes="InputAttributes" />
@bind	Provides data binding support	<input @bind="startDate" @bind:format="yyyy-MM-dd" />
@on{EVENT}	Provides event handling features	<button class="btn btn-primary" @onclick="UpdateHeading"> Update </button>
@on{EVENT}:preventDefault	Prevents the default action for the event in Razor Components	<input @onkeypress:preventDefault="shouldPreventDefault" />
@on{EVENT}:stopPropagation	Stops event propagation for the event	<div @onclick="OnSelectChildDiv" @onclick:stopPropagation="stopPropagation"> Child div that stops propagation when selected. </div>
@key	Causes the components diffing algorithm to guarantee preservation of elements or components based on the key's value	<div @key="currentPerson">Contents</div>
@ref	Provide a way to reference a component instance so that you can issue commands to that instance.	<MyLoginDialog @ref="loginDialog" ... />

ASP.NET Core Razor Directives

Directive	Description	Example
@attribute	Adds the given attribute to the class of the generated page or view	@attribute [Authorize]
@code	Enables a Razor component to add C# members (fields, properties, and methods) to a component	@code { // C# members (fields, properties, and methods) }
@functions	Enables adding C# members (fields, properties, and methods) to the generated class	@functions { // C# members (fields, properties, and methods) }
@implements	Implements an interface for the generated class	@implements IDisposable
@inherits	Provides full control of the class the view inherits	@inherits LoginRazorPage<LoginViewModel>
@inject	Enables the Razor Page to inject a service from the service container into a view	@inject IConfiguration Configuration
@layout	Specifies a layout for a Razor component. Layout components are used to avoid code duplication and inconsistency	@layout MasterLayout
@model	Specifies the type of the model passed to a view or page	@model LoginViewModel
@namespace	Sets the namespace of the class of the generated Razor page, MVC view, or Razor component	@namespace Your.Namespace.Here
@page	The @page directive has different effects depending on the type of the file where it appears. <ul style="list-style-type: none">• In a .cshtml file indicates that the file is a Razor Page.• Specifies that a Razor component should handle requests directly.	@page
@section	Used in conjunction with MVC and Razor Pages layouts to enable views or pages to render content in different parts of the HTML page	@section Scripts { <script type="text/javascript" src="~/scripts/main.js"></script> }
@using	Adds the C# using directive to the generated view	@using System.IO

www.ezzylearning.net

An online tutorial website that provides a comprehensive array of tutorials and code samples on a variety of topics ranging from classic Windows based Application development to the latest in ASP.NET Core, ASP.NET MVC, AngularJS, C#, VB.NET, AJAX, JQuery, WCF, LINQ, Android development.

