ASP.NET Core Cheat Sheet

.NET 10 (LTS) · C# 14 · Blazor, Minimal APIs, MVC, Razor Pages · quick reference for irurueta docs

dotnet CLI

```
dotnet new web -n MyApp
dotnet new blazor -n MyApp
dotnet run
dotnet watch
dotnet publish -c Release
dotnet add package
Microsoft.EntityFrameworkCore
```

Minimal Program.cs

```
var builder =
WebApplication.CreateBuilder(args);
builder.Services.AddScoped<IProductRepo, Repo>
();
builder.Services.AddValidation();

var app = builder.Build();
app.UseExceptionHandler();
app.UseHttpsRedirection();
app.MapStaticAssets();

app.MapGet("/", () => "Hi").WithName("root");
app.Run();
```

Middleware order

`ExceptionHandler → HSTS → HttpsRedirection → StaticAssets → Routing → RateLimiter → Cors → Authentication → Authorization → OutputCache → SessionAntiforgery → Endpoints`

```
app.Use(async (ctx, next) => {
  // before
  await next(ctx);
  // after
});
```

DI lifetimes

Lifetime	Created
`Singleton`	once, app lifetime
`Scoped`	once per request
`Transient`	every resolution

```
builder.Services.AddSingleton<IClock, Clock>();
builder.Services.AddKeyedScoped<ICache, Redis>
("r");
```

Options pattern

```
builder.Services.Configure<SmtpOptions>(
  builder.Configuration.GetSection("Smtp"));
// consume
public class Mailer(IOptions<SmtpOptions> opt)
{ }
```

Route templates

```
app.MapGet("/products/{id:int}",
  (int id) => $"product {id}");
app.MapGet("/search/{*path}", (string path) =>
path);
```

Minimal API CRUD + filter

```
var products = app.MapGroup("/products");
products.MapGet("/{id:int}", async (int id, Db
db)
  => await db.Products.FindAsync(id)
     is { } p ? TypedResults.Ok(p)
              : TypedResults.NotFound());
products.MapPost("/", (Product p, Db db) => {
    db.Products.Add(p);
    return
TypedResults.Created($"/products/{p.Id}", p);
  })
  .AddEndpointFilter(async (ctx, next) => {
    // validation / logging filter
    return await next(ctx);
  });
```

MVC controller

```
[ApiController, Route("api/[controller]")]
public class ProductsController(Db db) :
ControllerBase {
  [HttpGet("{id:int}")]
  public async Task<IActionResult> Get(int id)
=>
    await db.Products.FindAsync(id) is { } p
      ? Ok(p) : NotFound();
}
```

Razor Page PageModel

```
public class IndexModel(Db db) : PageModel {
  public List<Product> Items { get; set; } =
[];
  public async Task OnGetAsync() =>
    Items = await db.Products.ToListAsync();
}
@* Index.cshtml *@
@page
@model IndexModel
```

Tag Helpers

```
<form asp-action="Create" asp-
controller="Product">
  <input asp-for="Name" />
  <span asp-validation-for="Name"></span>
  <a asp-route-id="@p.Id">Details</a>
</form>
```

Blazor component

```
@rendermode InteractiveServer
<EditForm Model="model" OnValidSubmit="Save">
  <DataAnnotationsValidator />
  <InputText @bind-Value="model.Name" />
  <button type="submit">Save</button>
</EditForm>
@code {
  [PersistentState] private Model model { get;
set; } = new();
  private void Save() { }
}
```

SignalR hub

```
public class ChatHub : Hub {
  public async Task Send(string user, string
msg) =>
    await Clients.All.SendAsync("Receive",
user, msg);
}
app.MapHub<ChatHub>("/chatHub");
```

EF Core

```
builder.Services.AddDbContext<Db>(o =>
  o.UseNpgsql(connStr));

var items = await db.Products
  .Where(p => p.Price > 10)
  .AsNoTracking().ToListAsync();

dotnet ef migrations add InitialCreate
dotnet ef database update
```

Auth + policy

```
builder.Services.AddAuthentication()
  .AddJwtBearer();
builder.Services.AddAuthorizationBuilder()
  .AddPolicy("Admin", p =>
    p.RequireRole("Administrator"));

[Authorize(Policy = "Admin")]
public IActionResult Secret() => Ok();
```

Error handling / caching

```
builder.Services.AddProblemDetails();
app.UseExceptionHandler();

builder.Services.AddOutputCache();
app.MapGet("/data", Handler).CacheOutput();

builder.Services.AddRateLimiter(o =>
  o.AddFixedWindowLimiter("fixed", opt => {
}));

builder.Services.AddHybridCache();
```

Integration test

```
public class ApiTests(
  WebApplicationFactory<Program> factory)
  :
IClassFixture<WebApplicationFactory<Program>> {
  [Fact]
  public async Task Get_Returns200() {
    var client = factory.CreateClient();
    var r = await
client.GetAsync("/products/1");
    r.EnsureSuccessStatusCode();
  }
}
```

Publish / Docker

```
dotnet publish -c Release -o out
# Dockerfile
FROM mcr.microsoft.com/dotnet/aspnet:10.0
COPY out/ .
ENTRYPOINT ["dotnet", "MyApp.dll"]
```