ASP.NET Core Cheat Sheet

Minimal API · Controllers · DI · Middleware · Auth

worksheetdownload.com

SETUP & PROGRAM.CS

```
var builder =
WebApplication.CreateBuilder(args);
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
var app = builder.Build();
app.UseRouting(); app.MapControllers();
app.Run();
```

MINIMAL APIs

```
app.MapGet('/users', async (UserRepo
repo) =>
await repo.GetAllAsync());
app.MapPost('/users', async (User u,
UserRepo r) => {
await r.AddAsync(u);
return Results.Created($'/users/{u.Id}',
u);
});
app.MapPut('/users/{id}',
...).RequireAuthorization();
```

CONTROLLERS

```
[ApiController]
[Route("api/[controller]")]
public class UsersController :
ControllerBase {
[HttpGet] public async
Task<IActionResult> GetAll()
=> Ok(await _svc.GetAllAsync());
[HttpPost] public async
Task<IActionResult> Create(CreateUserDto
dto)
=> CreatedAtAction(nameof(GetById),
new{id=...}, result);
}
```

ROUTING

```
[Route("api/[controller]/[action]")]
[HttpGet("{id:int}")]
[HttpGet("{slug:regex(^[a-z-]+$)}")]
app.MapControllerRoute("default",
"{controller=Home}/{action=Index}")
app.MapGet("/", () => "Hello World!")
```

DEPENDENCY INJECTION

```
builder.Services.AddScoped<IUserService,
UserService>();
builder.Services.AddSingleton<ICache,
MemoryCache>();
builder.Services.AddTransient<IEmailSender,
SmtpSender>();
public class Ctrl(IUserService svc) {
... } // constructor inject
```

Lifetimes: Transient(new each time), Scoped(per request), Singleton(app lifetime)

MIDDLEWARE

```
app.Use(async (ctx, next) => {
// before
await next(ctx);
// after
});
app.UseExceptionHandler('/error');
app.UseHsts();
app.UseHttpsRedirection();
app.UseAuthentication();
app.UseAuthorization();
```

Order matters: HTTPS Routing Auth Endpoints

CONFIGURATION

```
var conn =
builder.Configuration.GetConnectionString("Default");
var secret =
builder.Configuration["Jwt:Secret"];
builder.Services.Configure<AppSettings>(
builder.Configuration.GetSection("App"));
// appsettings.json overridden by
appsettings.{env}.json
// Environment variables override config
files
```

MODEL BINDING

```
[FromRoute] int id
[FromQuery] string? filter
[FromBody] CreateUserDto dto
[FromHeader(Name="X-Api-Key")] string
apiKey
[FromForm] IFormFile file
[FromServices] IService svc  // DI in
action params
```

AUTH & AUTHORIZATION

```
builder.Services.AddAuthentication(JwtBearerDefaults.Authentic
.AddJwtBearer(opts => {
opts.TokenValidationParameters = ... });
[Authorize]  [Authorize(Roles="Admin")]
[AllowAnonymous]
builder.Services.AddAuthorization(opts
=>
opts.AddPolicy("Admin", p =>
p.RequireRole("Admin")));
```

FILTERS

```
[ServiceFilter(typeof(LoggingFilter))]
public class LoggingFilter :
IActionFilter {
public void
OnActionExecuting(ActionExecutingContext
ctx) {}
public void
OnActionExecuted(ActionExecutedContext
ctx) {}
}
```

Filter types: Authorization, Resource, Action, Exception, Result

LOGGING

```
private readonly
ILogger<UsersController> _logger;
_logger.LogInformation("Getting user
{Id}", id);
_logger.LogError(ex, "Failed to create
user");
// appsettings.json:
"Logging": { "LogLevel": { "Default":
"Information" } }
builder.Logging.AddConsole().AddDebug();
```

COMMON GOTCHAS

UseAuthentication() must come before UseAuthorization()
AddDbContext() is Scoped don't inject into Singleton
async void in controllers hides exceptions
Nullable annotations on DTOs prevent null surprises
CORS must be configured before UseRouting()