ASP.NET Core Cheat Sheet

A quick reference guide for ASP.NET Core, covering essential concepts, libraries, and tools for building modern web applications.

Core Concepts

Project Structure

Program.cs: Entry point of the application. Configures the host and startup.
Startup.cs: Configures services and the request pipeline.
appsettings.json: Configuration settings for different environments.
Controllers: Handle incoming HTTP requests.
Models: Represent data structures.

Views: (MVC) Represent the user interface.
wwwroot: Static files like CSS, JavaScript, and images.
.csproj: Project file containing dependencies and build configuration.

Dependency Injection

`services.AddSingleton<TInterface, TImplementation>();`	Registers a service as a singleton (one instance per application).
`services.AddScoped<TInterface, TImplementation>();`	Registers a service as scoped (one instance per request).
`services.AddTransient<TInterface, TImplementation>();`	Registers a service as transient (a new instance every time it's requested).
`[FromServices] TService service`	Injecting services into controller actions.

Middleware

Middleware components form the request pipeline. They handle requests and responses.

`app.UseMiddleware<MyMiddleware>();` - Adds custom middleware to the pipeline.

`app.UseRouting();` - Adds route matching to the pipeline.

`app.UseAuthentication();` - Enables authentication.

`app.UseAuthorization();` - Enables authorization.

`app.UseEndpoints(endpoints => { ... });` - Configures endpoint routing.

Configuration

Configuration Sources

ASP.NET Core supports various configuration sources:

- `appsettings.json`
- `appsettings.{Environment}.json`
- Environment variables
- Command-line arguments
- User secrets (for development)

Accessing Configuration

`IConfiguration configuration;`	Inject `IConfiguration` into your classes.
`configuration["Section:Key"]`	Accessing configuration values.
`configuration.GetSection("Section").Get<MyOptions>()`	Binding configuration sections to objects.
`services.Configure<MyOptions>(configuration.GetSection("Section"));`	Configuring options using the Options pattern.

Options Pattern

The Options pattern provides a way to access configuration values in a strongly-typed manner.

1. Define an options class:

```
public class MyOptions
{
    public string Option1 { get; set; }
    public int Option2 { get; set; }
}
```

2. Configure the options in `Startup.cs`:

```
services.Configure<MyOptions>
(configuration.GetSection("MySection"));
```

3. Inject `IOptions<MyOptions>` into your class:

```
public class MyClass
{
    private readonly MyOptions _options;

    public MyClass(IOptions<MyOptions>
options)
    {
        _options = options.Value;
    }
}
```

Routing and Controllers

Routing

Routing is responsible for mapping incoming requests to controller actions.

Attribute Routing:

```
[Route("api/[controller]")]
public class MyController : ControllerBase
{
    [HttpGet("items/{id}")]
    public IActionResult GetItem(int id) { ...
}
}
```

Conventional Routing:
Configured in `Startup.cs` using `app.UseEndpoints()`.

Controllers and Actions

`[ApiController]`	Attribute to enable API-specific behaviors.
`IActionResult`	Return type for controller actions (allows returning different HTTP status codes).
`Ok(value)`	Returns a 200 OK result with a value.
`BadRequest(error)`	Returns a 400 Bad Request result with an error.
`NotFound()`	Returns a 404 Not Found result.

Model Binding

Model binding automatically maps incoming request data to action parameters.

```
public IActionResult Create([FromBody] MyModel
model) { ... }
```

`[FromBody]` - Binds data from the request body.
`[FromQuery]` - Binds data from the query string.
`[FromRoute]` - Binds data from the route.
`[FromHeader]` - Binds data from the request headers.

Data Access

Entity Framework Core

Entity Framework Core (EF Core) is an ORM for .NET Core.

1. Install the `Microsoft.EntityFrameworkCore` NuGet package.
2. Define your data models as C# classes.
3. Create a DbContext class that represents your database session.
4. Configure EF Core in `Startup.cs`:

```
services.AddDbContext<MyDbContext>(options
=>

options.UseSqlServer(Configuration.GetConn
ectionString("DefaultConnection")));
```

DbContext

`DbSet<MyEntity>`	Represents a collection of entities in the database.
`context.SaveChanges()`	Saves changes to the database.
`context.MyEntities.Add(entity)`	Adds a new entity to the database.
`context.MyEntities.Remove(entity)`	Removes an entity from the database.
`context.MyEntities.FindAsync(id)`	Finds an entity by its primary key.

LINQ Queries

EF Core uses LINQ (Language Integrated Query) to query the database.

```
var items = context.MyEntities
    .Where(i => i.Property == value)
    .OrderBy(i => i.Name)
    .ToList();
```

- `Where()` - Filters the results.
- `OrderBy()` - Sorts the results.
- `ToListAsync()` - Executes the query asynchronously and returns a list.