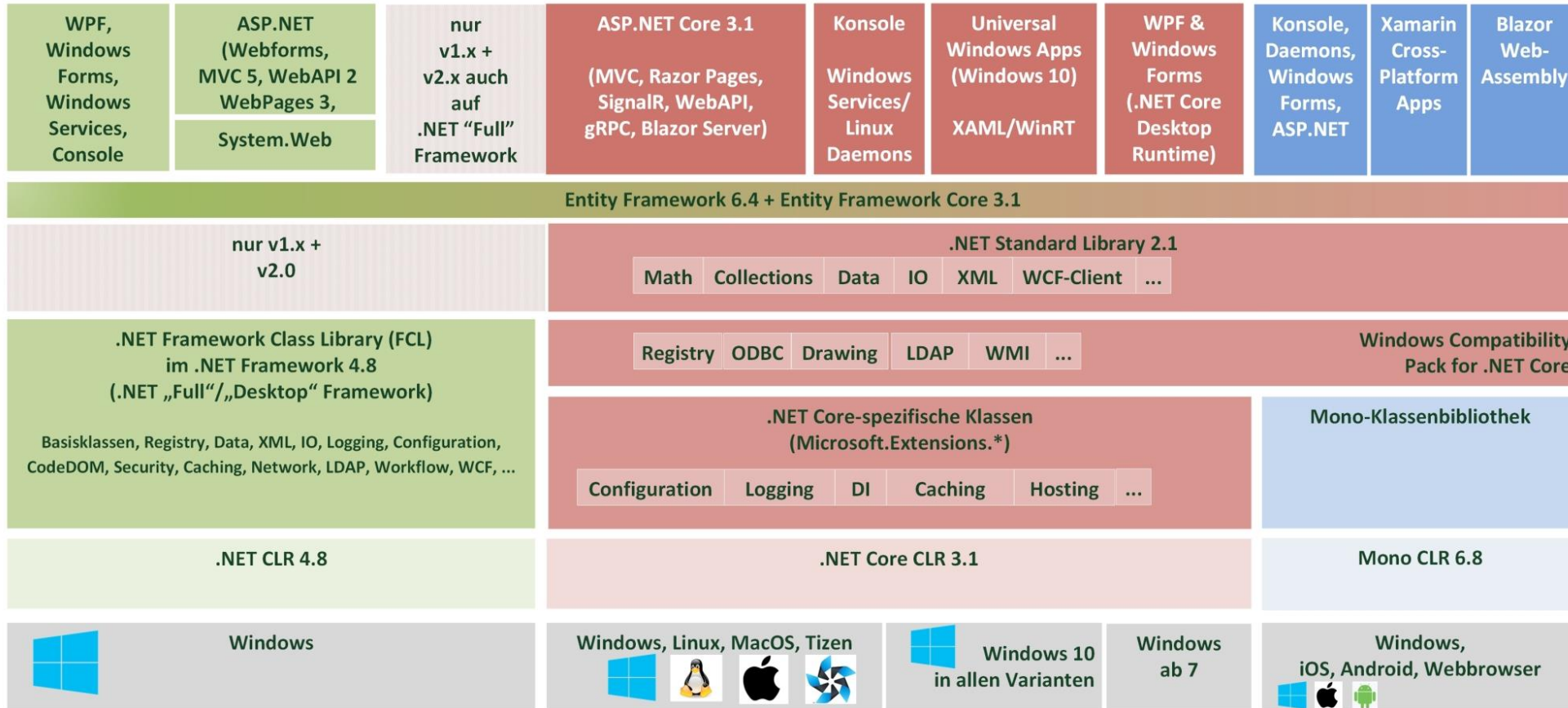


ASP.NET Core WebAPI

Anwendungsmodelle in .NET und .NET Core

Die .NET-Familie 2020

© Dr. Holger Schwichtenberg, www.IT-Visions.de, Stand 10.02.2020

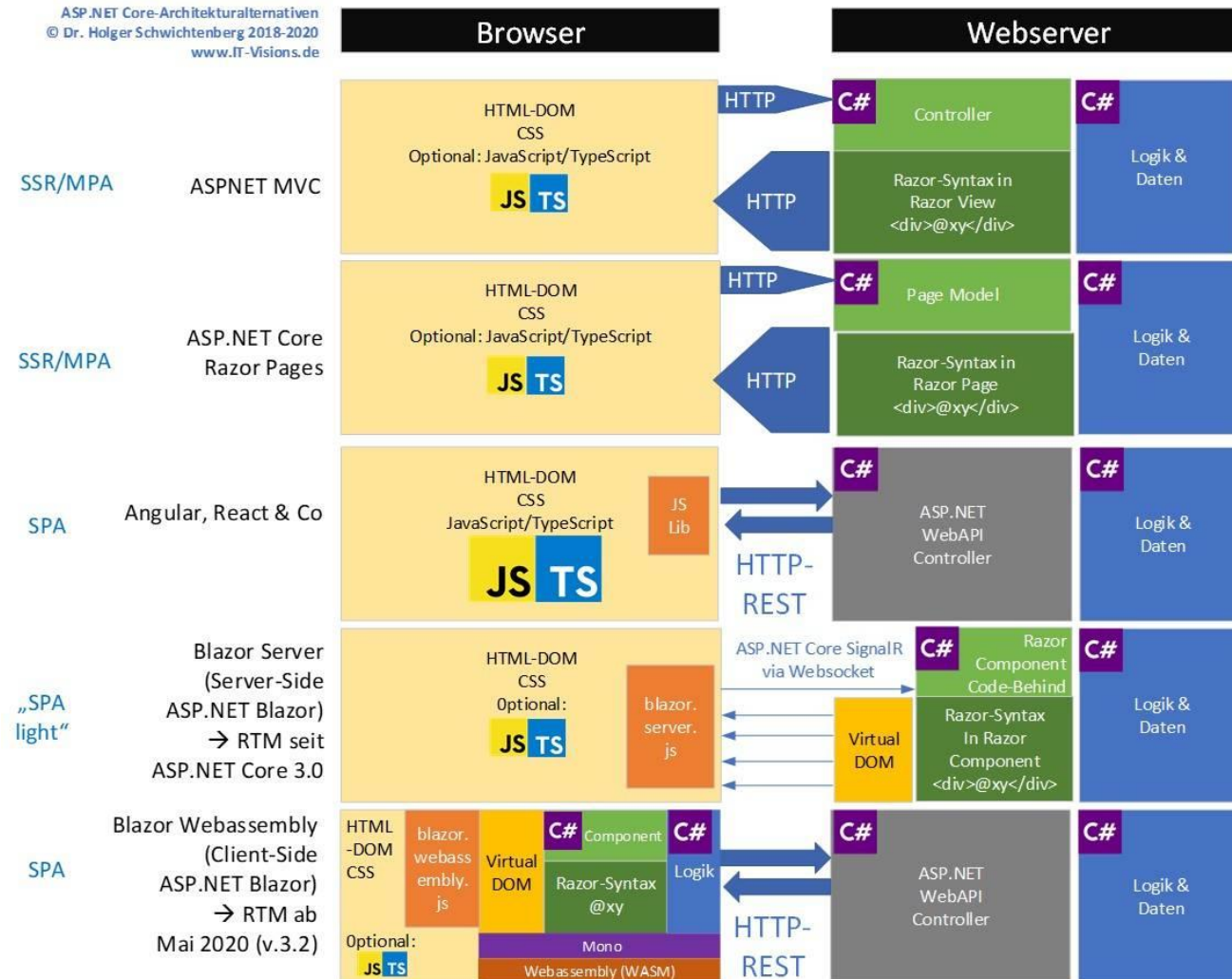


.NET Core Versionen

Version	Original Release Date	Latest Patch Version	Patch Release Date	Support Level	End of Support
.NET Core 3.1	December 3, 2019	3.1.0	3.1.0	LTS	
.NET Core 3.0	September 23, 2019	3.0.1	November 19, 2019	Maintenance	March 3, 2020
.NET Core 2.2	December 4, 2018	2.2.8	November 19, 2019	Maintenance	December 23, 2019
.NET Core 2.1	May 30, 2018	2.1.14	November 19, 2019	LTS	August 21, 2021
.NET Core 2.0	August 14, 2017	2.0.9	July 10, 2018	EOL	October 1, 2018
.NET Core 1.1	November 16, 2016	1.1.13	May 14, 2019	EOL	June 27 2019
.NET Core 1.0	June 27, 2016	1.0.16	May 14, 2019	EOL	June 27 2019


Quelle: H. Schwichtenberg, <https://www.heise.de/hintergrund/Umstieg-auf-NET-Core-Migrieren-oder-nicht-migrieren-4628946.html?seite=2>

Architekturmodelle in ASP.NET Core



<https://docs.microsoft.com/en-us/aspnet/core/getting-started/?view=aspnetcore-5.0&tabs=windows>

Tutorial: Get started with ASP.NET Core

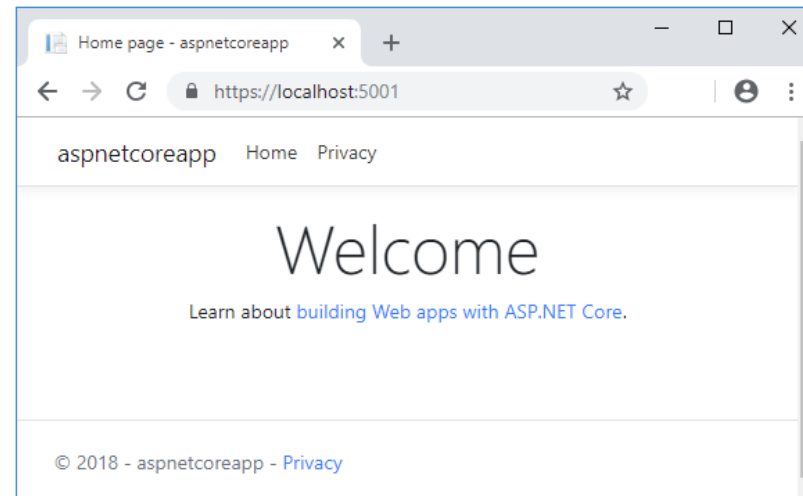
01/07/2020 • 2 minutes to read •  +8

This tutorial shows how to create and run an ASP.NET Core web app using the .NET Core CLI.

You'll learn how to:

- ✓ Create a web app project.
- ✓ Trust the development certificate.
- ✓ Run the app.
- ✓ Edit a Razor page.

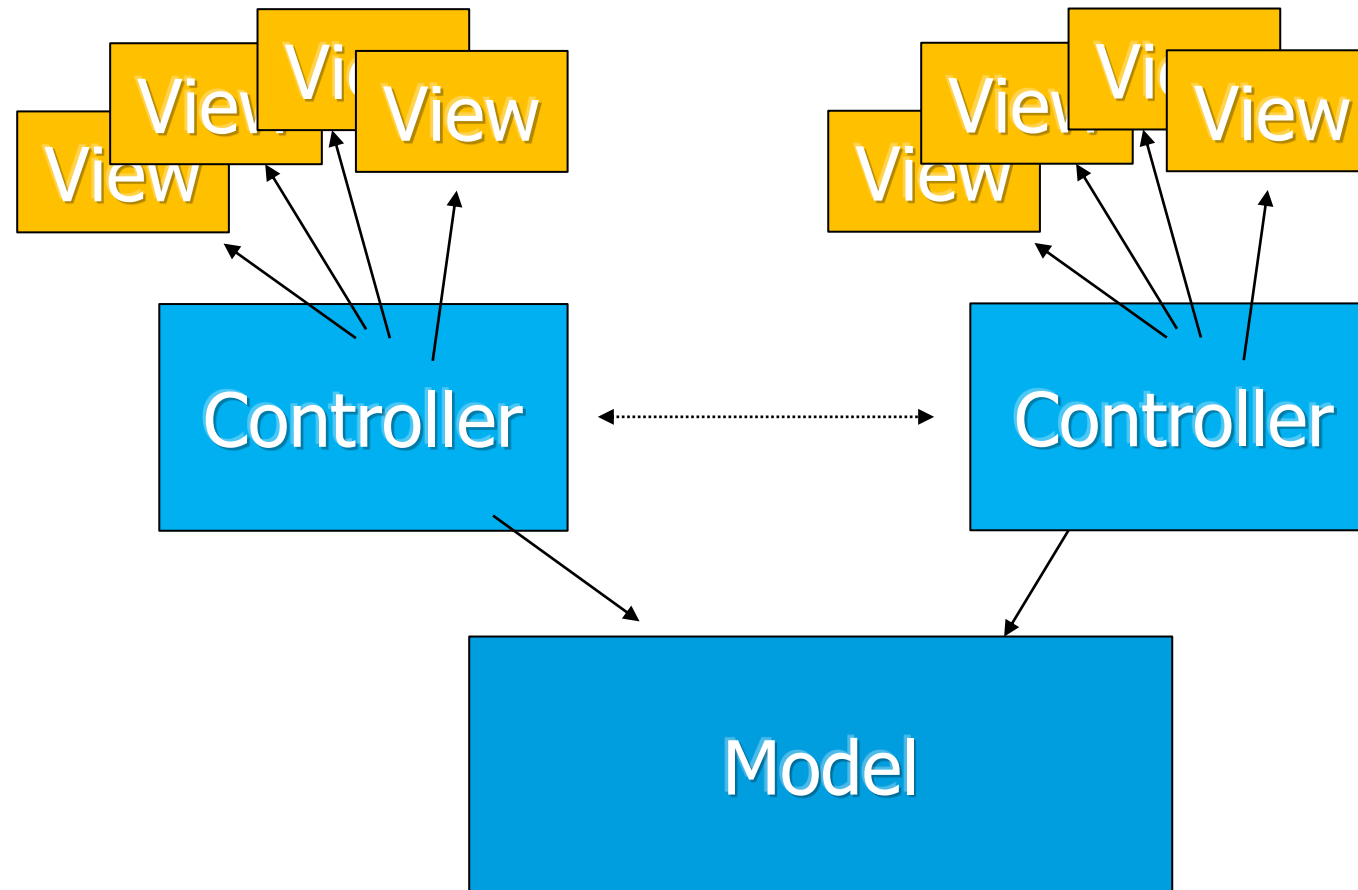
At the end, you'll have a working web app running on your local machine.



Was ist MVC?

- Model
 - Beinhaltet Geschäftslogik, Datenmodell
- View
 - Definiert, was der Browser anzeigen soll
- Controller
 - Steuert die Bedienlogik, wählt die Views aus und versorgt sie mit Informationen

Model, Controller, Views



- Controller geben keine Views zurück, sondern Daten und Statuscodes
- Daten je nach Bedarf
 - Text
 - JSON
 - XML
 - andere Formate

- Standard-Setup über CreateDefaultBuilder

[https://docs.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.webhost.createdefaultbuilder?view=aspnetcore-3.1#Microsoft.AspNetCore.WebHost.CreateDefaultBuilder\(System.String\)](https://docs.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.webhost.createdefaultbuilder?view=aspnetcore-3.1#Microsoft.AspNetCore.WebHost.CreateDefaultBuilder(System.String))

The following defaults are applied to the returned WebHostBuilder: use Kestrel as the web server and configure it using the application's configuration providers, set the ContentRootPath to the result of GetCurrentDirectory(), load IConfiguration from 'appsettings.json' and 'appsettings.[EnvironmentName].json', load IConfiguration from User Secrets when EnvironmentName is 'Development' using the entry assembly, load IConfiguration from environment variables, load IConfiguration from supplied command line args, configure the ILoggerFactory to log to the console and debug output, adds the HostFiltering middleware, adds the ForwardedHeaders middleware if ASPNETCORE_FORWARDEDHEADERS_ENABLED=true, and enable IIS integration.

- Startup-Klasse

```
public class Program
{
    public static void Main(string[] args)
    {
        CreateHostBuilder(args).Build().Run();
    }

    public static IHostBuilder CreateHostBuilder(string[] args) =>
        Host.CreateDefaultBuilder(args)
            .ConfigureWebHostDefaults(webBuilder =>
            {
                webBuilder.UseStartup<Startup>();
            });
}
```

- Konfiguration
- Dependency-Injection
- Aufruf der gewünschten Module

```
public class Startup
{
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    public IConfiguration Configuration { get; }

    // This method gets called by the runtime. Use this method to add services to the con
    public void ConfigureServices(IServiceCollection services) {...}

    // This method gets called by the runtime. Use this method to configure the HTTP requ
    public void Configure(IApplicationBuilder app, IHostingEnvironment env) {...}
}
```

- In ConfigureServices:

```
services.AddControllers();
```

- In Configure:

```
app.UseRouting();  
app.UseEndpoints(endpoints =>  
{  
    endpoints.MapControllers();  
});
```

Controller-Klasse

- Klasse als POCO oder von **ControllerBase** (nicht Controller!) ableiten
- **RouteAttribute** auf Controller-Klasse setzen
- **ApiControllerAttribute** vereinfacht einige Prüfungen und Deklarationen
<https://docs.microsoft.com/de-de/aspnet/core/web-api/?view=aspnetcore-5.0>
- **HttpGetAttribute**, **HttpPostAttribute** etc. auf Methoden setzen
- **ApiConventionTypeAttribute** auf Assembly-Ebene setzen

ApiControllerAttribute

- The ApiController attribute is commonly coupled with the ControllerBase class to enable REST-specific behavior for controllers, and it allows us to build HTTP APIs. First of all, it provides implicit model state validation, which means that we do not need to explicitly check the ModelState.IsValid attribute in each action. Secondly, it also implicitly defines the model binding attributes, which means that we do not need to specify the [FromBody], ...

- <https://docs.microsoft.com/en-us/aspnet/core/web-api/action-return-types?view=aspnetcore-3.1>
- Spezifischer Typ
- IActionResult
- ActionResult<T>
- Synchron
- Asynchron (immer mit Task<> als Rückgabetyt)

- Ok
- NotFound
- BadRequest
- PhysicalFile
- TryUpdateModelAsync
- TryValidateModel
- CreatedAtAction (bei Posts)

<https://docs.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-3.1>

MIME-Types

- Einige Result-Typen benötigen die Angabe eines Content-Types
- z. B.

MIME-Typ	Dateiendung	Bedeutung
application/pdf	*.pdf	Adobe PDF-Dateien
application/xml	*.xml	XML-Dateien
image/jpeg	*.jpeg *.jpg	JPEG-Dateien
text/xml	*.xml	XML-Dateien

siehe: <http://de.selfhtml.org/diverses/mimetypen.htm>

Attribute zur Steuerung

- FromQuery, FromBody, FromHeader, FromForm ...
- Route
- Bind
- HttpGet, HttpPost...
- Consumes, Produces (lässt sich durch API-Conventions ersetzen)

<https://docs.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-3.1#attributes>

<https://docs.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-3.1#binding-source-parameter-inference>

- allgemein:
- <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/routing?view=aspnetcore-3.1>
- Web-API:
- <https://docs.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-3.1#attributes>

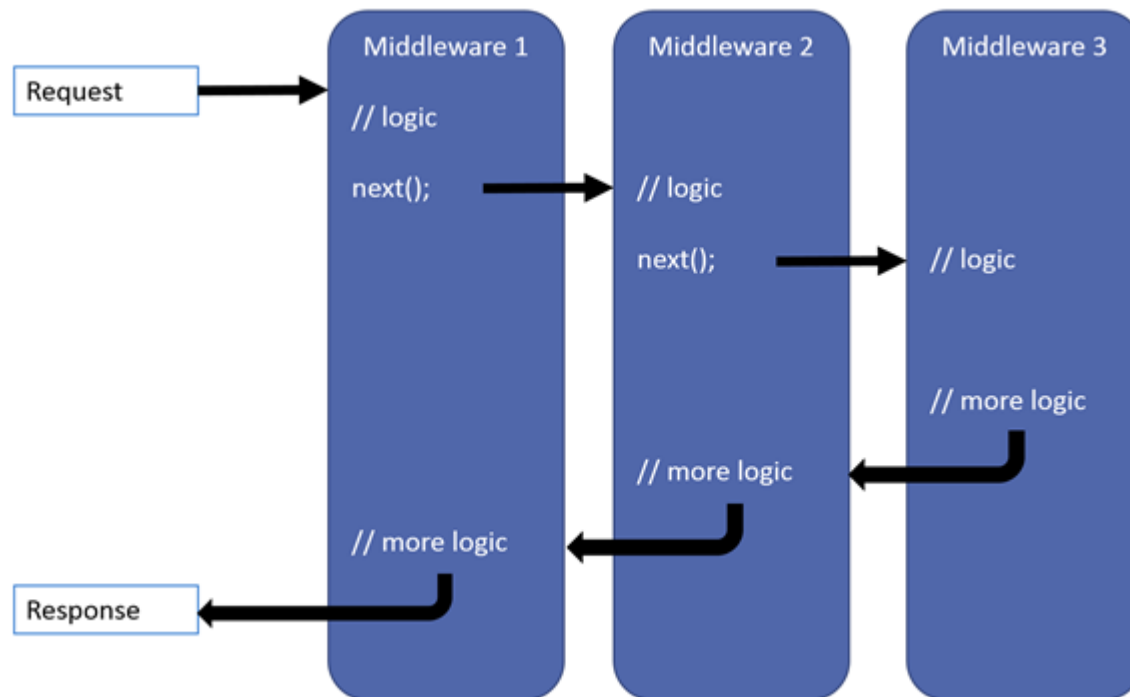
Route-Inline-Constraints (Auswahl)

Name	Beispiel	Beschreibung
bool	{n:bool}	A Boolean value
int	{n:int}	An Int32 value
minlength	{n:minlength(2)}	A String value with at least two characters
length	{n:length(2)} {n:length(2,4)}	A String value with exactly two characters A String value with two, three, or four characters
range	{n:range(1,3)}	The Int64 value 1, 2, or 3
alpha	{n:alpha}	A String value containing only the A–Z and a–z characters
regex	{n:regex (^a+\$)}	A String value containing only one or more 'a' characters (a Regex match for the ^a+\$ pattern)

Einrichtungen in Startup-Klasse

Middleware-Pipeline

- Ziel: nur das einbinden, was tatsächlich benötigt wird



Quelle: <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware/?view=aspnetcore-5.0>

Dependency Injection

- siehe <https://docs.microsoft.com/de-de/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-3.1>
- Container-Aufbau in *ConfigureServices(IServiceCollection services)*
- diverse Methoden zum Hinzufügen von Objekten, Factories etc.
- Drei verschiedene Instanzierungsvarianten
 - Singleton
 - Scoped
 - Transient
- Vorgefertigte Methoden wie AddMvc, AddDbContext, AddLogging etc.

- <https://docs.microsoft.com/de-de/aspnet/core/fundamentals/configuration/?view=aspnetcore-3.1>
- Verschiedene Quellen
 - JSON-Dateien
 - XML
 - Systemvariablen
 - Standardquellen:
 - appsettings.json
 - appsettings.[EnvironmentName].json
 - User Secrets für "Development"
 - Environment-Variablen (Windows)
 - Kommandozeilenparameter

- <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/logging/?view=aspnetcore-3.1>
- Einbau über Dependency Injection (ILogger<T>)
- Build-In-Provider:
 - Console, Debug, EventLog usw.
- Beliebig erweiterbar
 - z. B. Serilog-File-LoggingNuGet: Serilog.Extensions.Logging.File
 - <https://github.com/serilog/serilog-extensions-logging-file>
 - <https://andrewlock.net/creating-a-rolling-file-logging-provider-for-asp-net-core-2-0/>
- Konfigurierbar über Konfigurationsdatei

- <https://docs.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-3.1>
- NSwag / NSwag-Studio
- <https://docs.microsoft.com/en-us/aspnet/core/tutorials/getting-started-with-nswag?view=aspnetcore-3.1&tabs=visual-studio>
- Analyzer
 - Erzeugt Warnungen für fehlenden Open-API-Informationen
- <https://docs.microsoft.com/en-us/aspnet/core/web-api/advanced/analyzers?view=aspnetcore-3.1&tabs=visual-studio>

Data-Annotations und Validierungen

- Attribute aus System.ComponentModel, System.ComponentModel.DataAnnotations und System.Web.Mvc
- Model- oder ViewModel-Klassen können über Attribute Darstellung und Validierung steuern
- Vorteile:
 - deklarative Steuerung der Validierung in den Datenklassen
 - Deklaration von Standard-Ansichten
- Nachteile:
 - Aufgabenbereiche der Darstellungsschicht werden teilweise in die Models verlagert

- ModelState gibt Auskunft über Validierungsfehler
- Abfragen:
 - `ModelState.IsValid`
 - `ModelState.IsValidField("Vorname")`
 - `ModelState["Vorname"].Errors`
- Setzen
 - `ModelState.AddModelError("", "Oh weh - alles falsch...");`
 - `ModelState.AddModelError("Vorname", "ungültiger Vorname");`
 - `ModelState["Vorname"].Errors.Add("Nicht die schon wieder");`
- Zurücksetzen
 - `ModelState.Clear()`

Validierung über Attribute

Attribut	Verwendung
Required	Eingabe darf nicht leer sein
StringLength	Maximale und minimale Eingabelänge
RegularExpression	RegEx-Ausdruck für die Prüfung
Range	Wertebereich für numerische Daten
Eigene Attribute	Von ValidationAttribute ableiten und IsValid überschreiben

Eigene Validierungsattribute

```
[AttributeUsage(AttributeTargets.Property | AttributeTargets.Field)]  
public class EvenNumberValidationAttribute: ValidationAttribute  
{  
    public override bool IsValid(object value)  
    {  
        if (value is int)  
        {  
            int number = (int)value;  
            return (number % 2) == 0;  
        }  
        return false;  
    }  
}
```

Verschiedenes

Static Files, Directory browsing

- <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/static-files?view=aspnetcore-3.1>

- <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/http-requests?view=aspnetcore-3.1>
- HttpClient nicht selbst instanzieren, sondern HttpClientFactory verwenden!
- Polly:
<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/http-requests?view=aspnetcore-3.1#use-polly-based-handlers>

File upload / download

- HTML:

```
<a href="..." download="name">...</a>
```

- Controller:

```
public FileResult Get(string name)
{
    name = Path.GetFileName(name);
    return File(System.IO.File.OpenRead(pfad), mimetype [, name]);
}
```

File upload HTML

```
<form method="post" enctype="multipart/form-data">  
  <input type="file" multiple="multiple" accept="image/*" asp-for="FileToUpload" />  
</form>
```

[BindProperty]

```
public IEnumerable< IFormFile> FileToUpload { get; set; }  
public async Task OnPostAsync()  
{  
    foreach (var item in FileToUpload)  
    {  
        var stream = item.OpenReadStream();  
        var buffer = new byte[item.Length];  
        var n = await stream.ReadAsync(buffer, 0, buffer.Length);  
        await System.IO.File.WriteAllBytesAsync(pfad, buffer);  
    }  
}
```

Fehlerbehandlung

- <https://docs.microsoft.com/en-us/aspnet/core/web-api/handle-errors?view=aspnetcore-3.1>
- Vorgefertigte Behandlung in Startup-Template passt nicht -> austauschen
- Dann geht Swagger aber nicht mehr ->
`[ApiExplorerSettings(IgnoreApi =true)]`

Health Checks

Health checks

- <https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/health-checks?view=aspnetcore-2.2>
- [Beat Pulse abgelöst durch Xabaril Enterprise HealthChecks for ASP.NET Core Diagnostics Package](#)
- [Nuget-Pakete für verschiedene Anwendungen:
<https://github.com/Xabaril/AspNetCore.Diagnostics.HealthChecks>](#)

Health checks einrichten

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddHealthChecks()
        .AddCheck("self", c => HealthCheckResult.Healthy())
        .AddSqlServer(connString)
        .AddMongoDb(new MongoClientSettings() {...})
        .AddRedis("192.168.0.123:6379");
}
```

```
app.UseHealthChecks("/health", new HealthCheckOptions
{
    Predicate = registration => true, // registration.Name=="self"
    ResponseWriter=UIResponseWriter.WriteHealthCheckUIResponse
});
```

Health checks Monitor

♥ Health Checks status Refresh every seconds [Change](#)

[↗](#) [✕](#)

Name	Health	On state from	Last execution
- Fu_Check	✓	Healthy a minute ago	24.4.2019, 10:26:47

Name	Health	Description	Duration
self	✓ Healthy		00:00:00.0000012
sqlserver	✓ Healthy		00:00:00.0003614
mongodb	✓ Healthy		00:00:00.0023677
redis	✓ Healthy		00:00:00.0015128

```
services.AddHealthChecksUI();
```

```
app.UseHealthChecksUI(setup =>  
{  
    setup.UIPath = "/ui";  
});
```

<https://www.youtube.com/watch?v=kzRKGCmGbqo>

Host / Installation

Host / Server

- <https://docs.microsoft.com/de-de/aspnet/core/fundamentals/host/generic-host?view=aspnetcore-3.1>
- <https://docs.microsoft.com/de-de/aspnet/core/fundamentals/host/web-host?view=aspnetcore-3.1>
- <https://docs.microsoft.com/de-de/aspnet/core/fundamentals/servers/?view=aspnetcore-3.1&tabs=windows>
- Bei IIS wichtig: Install the .NET Core Hosting Bundle:
<https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/iis/?view=aspnetcore-5.0#install-the-net-core-hosting-bundle>

Versionierung

- <https://github.com/Microsoft/aspnet-api-versioning/wiki>
- <https://www.infoworld.com/article/3562355/how-to-use-api-versioning-in-aspnet-core.html>
- <https://www.c-sharpcorner.com/article/maintaine-multiple-versions-of-api-code-base-using/>

OAuth

- <https://docs.microsoft.com/de-de/aspnet/core/security/authentication/social/?view=aspnetcore-5.0&tabs=visual-studio>
- oder eigener Token-Provider

Docker Container

HTTPS

- s. <https://docs.microsoft.com/en-us/aspnet/core/security/docker-https?view=aspnetcore-6.0>
- mit docker-compose:
<https://docs.microsoft.com/en-us/aspnet/core/security/docker-compose-https?view=aspnetcore-6.0>

