Asp.Net Core tutorial| Asp.Net Core | Asp.Net Core MVC Tutorial |ASP.NET Core MVC Tutorial for beginners

<https://www.youtube.com/playlist?list=PLaFzfwmPR7_LTXu0Vz9Zz_Y0OMMC7ArHZ>

(#1) Asp.Net Core MVC 3.1 course overview | Asp.Net Core tutorial

<https://www.youtube.com/watch?v=CP-zbZA5LAc&list=PLaFzfwmPR7_LTXu0Vz9Zz_Y0OMMC7ArHZ&index=1>

Details:

- Step by step tutorial

- Basic to advanced concept

- Designed for beginner & experienced professionals

- Live examples with theory

- Live application

Prerequisite:

- Basic knowledge of C#, HTML

- That’s it 😊

Who can learn from this tutorial:

- Beginners

- Intermediate

- Expert

Main topics:

- Dot Net Core

- Asp.Net Core MVC

- Entity framework Core

- Identity

- SQL Server, My SQL etc.

- And some other topics

# (#2) What is ASP.NET Core | Asp.Net Core | Asp.Net Core tutorial

# <https://www.youtube.com/watch?v=-lgNclGseLM&list=PLaFzfwmPR7_LTXu0Vz9Zz_Y0OMMC7ArHZ&index=2>

# Agenda:

# - What is Dot Net Core

# - Where can we develop application using .Net Core

# - What can we develop with .Net Core

# - Features of .Net Core

# - Language support

# - Package Management

# - Community

# What is .Net Core?

# Dot Net Core is an open-source and cross-platform framework which is used for building several type of applications.

# Dot Net Core is supported by Microsoft.

# Where can we develop applications?

# - Windows

# - Mac

# - Linux

# - Docker (We can use dot net core image on docker also)

# What can we develop?

# - Web (Asp.Net MVC Core, Web API, Razor Pages, Microservices)

# - Mobile

# - Console

# - Desktop

# - IOT (Internet of things)

# - ML (Machine learning)

# - Gaming

# - Cloud

# Features of Dot Net Core:

# - Free (No fees or license is required even for commercial applications also)

# - Open Source (Available on GitHub <https://github.com/dotnet> )

# - Cross platform

# - Supported by Microsoft

# - CLI support

# Language support:

# - C#

# - F#

# - Visual Basic (VB)

# Package Management:

# - Nuget

# Dot Net Core Community:

# - All major community platform has a special community for Dot Net Core

# - <https://dotnet.microsoft.com/platform/community>

# (#3) Setting up dot net core machine development | Asp.Net Core tutorial | Asp.Net Core 3.1 tutorial

# <https://www.youtube.com/watch?v=bFsu3nup00Y&list=PLaFzfwmPR7_LTXu0Vz9Zz_Y0OMMC7ArHZ&index=3>

# Agenda:

# - How to setup development machine for dot net core applications development

# - Install SDK & Editor

# - Verify the installation

# Tools:

# - Machine (Windows, Mac, Linux)

# - Editor (Recommended Visual Studio, VS Code)

# - Dot Net Core SDK

# Editor:

# - Visual studio (<https://visualstudio.microsoft.com/> )

# - VS Code (<https://code.visualstudio.com/> )

# - Atom

# - Subline

# - Brackets

# - Etc...

# Dot Net Core SDK:

# - Download and install Dot Net Core SDK

# - <https://dotnet.microsoft.com/download>

# Verify Dot Net Core installation:

# - Open your operating system’s commandline

# - Type command: *dotnet --version*

# - You will see the latest version of installed Dot Net Core

# (#4) How to create asp.net core mvc web application (using Visual Studio 2019 & CLI) | .Net Core 3

# <https://www.youtube.com/watch?v=_AlcrY5YEww&list=PLaFzfwmPR7_LTXu0Vz9Zz_Y0OMMC7ArHZ&index=4>.

# Agenda:

# - How to create first Asp.Net MVC Core project using visual studio

# - How to create first Asp.Net MVC Core project using dotnet CLI

# - Run the project using Visual studio & dotnet CLI

# How to create first Asp.Net MVC Core project and run it using visual studio

# 1)Click “Create a new project”

# 2)Select Asp.Net Core Web Application

# 3)Name it as you wish, choose project’s location, click “Create” button

# 4)Choose a template “Web Application (Model-View-Controller)”

# 5)Uncheck “Configure for HTTPS” (for this lesson)

# 6)Click “Create” button

# 7)After creating your project, you can run it

# How to create first Asp.Net MVC Core project and run it using dotnet CLI

# 1)Open your operating system’s commandline from your folder

# 2)Type command: *dotnet new*

# 3)You will see “Templates” that you want to create with your project

# 4)For ASP.NET Core Web App (Model-View-Controller) we will use template “mvc” - very short name

# 5)Type command: *dotnet new mvc --name FirstMVCApp* – it will create your new ASP.NET Core Web App (Model-View-Controller) project with name “FirstMVCApp”

# 6)If you have VS Code then you can open your project’s folder with VC Code and run it from terminal by next command: *dotnet run*

# (#5) What is MVC pattern (model view controller architecture) in .net core | Asp.Net Core Tutorial

# <https://www.youtube.com/watch?v=zBYxYgDWh4w&list=PLaFzfwmPR7_LTXu0Vz9Zz_Y0OMMC7ArHZ&index=5>

# Agenda:

# - What is MVC pattern

# - Role of each component

# - How MVC pattern works in Asp.Net Core MVC

# - Details about each component

# - Benefits of using MVC pattern

# What is MVC pattern?

# - MVC stands for Model – View – Controller

# - MVC is an architectural design pattern. It means this design pattern is used at the architecture level of an application. MVC is not property of .net, it can be used with any

# languages. MVC and .net Core MVC both are different things.

# - Model, View & Controller are the main components of mvc pattern.

# Role of Model, View & Controller?

# - Model is responsible for data.

# - View is responsible for UI (User Interface).

# - Controller is responsible for the flow of application by accepting user input. What is flow of application? Flow of application is that how our website behaves by some user’s actions.

# 

# How MVC works in Asp.Net Core

# 

# Understanding Controller?

# - A controller is a .cs (for C#) file which has some methods called action methods.

# - When a request comes on controller, it actually hits an action method.

# - Now everything depends on Action method what to return from it. It may return only view, only data, or both of them.

# Understanding Model?

# - A model in Asp.Net Core MVC is a simple class which has some properties.

# - Model is used to pass data from controller (action method) to view and vice versa i.e. View to controller (action method).

# - Model is also used for server side data validation. And with some techniques we can generate client side validation also

# - It is not mandatory that each action method will return some model.

# Understanding View?

# - A view is the combination of HTML and C# (or F#, VB).

# - Hence for C# application the extension of a view is .cshtml

# - Whatever you see on your browser is a view.

# - It is not mandatory that each action method will return a view.

# - When a view (containing C#) gets rendered on browser then all its C# is converted into HTML. It means on a browser we will only see HTML and data.

# Benefits of MVC:

# - Separation of concern.

# - Each component has a specific job hence it is easy to debug the code.

# (#6) Convert console application to web application in asp.net core | WebHostBuilder in .net core

# <https://www.youtube.com/watch?v=9QQSPFtZ6Uk&list=PLaFzfwmPR7_LTXu0Vz9Zz_Y0OMMC7ArHZ&index=6>

# Agenda:

# - How to convert console application into web application

# - How to setup host builder for web application

# - What is Host

# - What default features are available in Host builder

# - CreateDefaultBuilder & ConfigureWebHostDefaults methods

# DotNet Core Application

# - By default every web application is a console application in dotnet core

# Steps to convert console app into web app:

# - Update SDK in csproj file (add .Web)

# - Update output type in csproj(delete <OutputType>Exe</OutputType>)

# - Add new WebHostBuilder

# - Add routing

# - Setup default route

# What is Host

# - A Host is an object that encapsulates the resources of an application.

# For example: Dependency Injection;

# Logging;

# Configuration;

# IHostedService implementation

# Converting console application into web application

# Program.cs

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Hosting;

using System;

namespace MyFirstConsoleApp

{

class Program

{

static void Main(string[] args)

{

CreateHostBuilder(args).Build().Run();

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args).ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

}

# }

# Startup.cs

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

using Microsoft.Extensions.DependencyInjection;

namespace MyFirstConsoleApp

{

public class Startup

{

public void ConfigurationServices(IServiceCollection service)

{

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

app.UseRouting();

app.UseEndpoints(endpoints =>

{

endpoints.MapGet("/", async context =>

{

await context.Response.WriteAsync("Hello from webgentle application");

});

});

}

}

# }

# Used:

# - IHostBuilder

# -ConfigureWebHostDefaults

# - IServiceCollection

# - IApplicationBuilder

# - IWebHostEnvironment

# - app.UseRouting(), .UseEndpoints

# - .MapGet

# (#7) Setup application on GitHub repository | ASP.NET Core Tutorial | .Net Core 3.1 MVC Tutorial

# <https://www.youtube.com/watch?v=lQMiMyMQHVg&list=PLaFzfwmPR7_LTXu0Vz9Zz_Y0OMMC7ArHZ&index=7>

# Agenda:

# - Create a blank asp.net core application

# - Add remote repo (GitHub)

# - Push code from local repo to remote repo

# Used:

# - View -> Team explorer -> Home -> Changes

# - .gitignore

# - commits

# (#8) Middleware in Asp.net Core | app.Use(), app.Next(), app.Map() | Http Pipeline | ASP.NET Core

# <https://www.youtube.com/watch?v=10AWqnAph2g&list=PLaFzfwmPR7_LTXu0Vz9Zz_Y0OMMC7ArHZ&index=8>

# Agenda:

# - What is Middleware.

# - How Middleware works.

# - Where is Middleware available in our app

# - How to add new middleware

# - Order of Middleware

# - Use(), Next() & Map() method

# DotNet Core Application Http Pipeline

# 

# Middleware

# - Asp.Net Core create an HTTP application pipeline that processes the request.

# - This Http Pipeline is configures in Configure method of Startup.cs

# - All the request to the application goes through the HTTP pipeline

# - A middleware is a piece of code (component) which is used in Http pipeline.

# - In an application we use multiple middleware.

# - Middleware has access to all the request and response.

# DotNet Core Application Pipeline

# 

# Used:

# - app.Use();

# - await next();

# - endpoints.MapGet();

# - endpoints.Map();

# - app.UseRouting(); - is used before middlewares

# (#9) launchsettings.json in Asp.Net Core | Asp.Net Core tutorial

# <https://www.youtube.com/watch?v=5A68_ArtOJA&list=PLaFzfwmPR7_LTXu0Vz9Zz_Y0OMMC7ArHZ&index=9>

# Agenda:

# - What is *launchSettings.json* file.

# - What all is available in this file.

# - How to set default port for the application

# What is *launchSetting.json* file

- The settings that are present within this file are going to be used when we run the .NET core application either from Visual Studio or by using .NET Core CLI.

- The most important point that you need to keep in mind is this launchSettings.json file is only used within the local development machine. That means this file is not required when we publishing our asp.net core application to the production server.

- If you have certain settings and you want your application to use such settings when you publish and deploy your asp.net core application to the production server, then you need to store such settings in the appsettings.json file. Generally, in the ASP.NET Core application, the configuration settings are going to be stored in the appsettings.json file. In our upcoming article, we will discuss the [appsettings.json file](https://dotnettutorials.net/lesson/asp-net-core-appsettings-json-file/)in detail.

# What all is available in this file

{

"iisSettings": {

"windowsAuthentication": false,

"anonymousAuthentication": true,

"iisExpress": {

"applicationUrl": "http://localhost:50314",

"sslPort": 0

}

},

"profiles": {

"IIS Express": {

"commandName": "IISExpress",

"launchBrowser": true,

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

},

"WebGentle.BookStore": {

"commandName": "Project",

"launchBrowser": true,

"applicationUrl": "http://localhost:5000",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

}

}

# }

# 

# 

# How to set default port for the application

# - IIS Express

{

"iisSettings": {

"windowsAuthentication": false,

"anonymousAuthentication": true,

"iisExpress": {

"applicationUrl": "http://localhost:50314", -> 50314 -> 50315

"sslPort": 0

}

},

- Kestrel

"WebGentle.BookStore": {

"commandName": "Project",

"launchBrowser": true,

"applicationUrl": "http://localhost:5000", -> 5000 -> 5001

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

}

"profiles": {

"IIS Express": {

"commandName": "IISExpress",

"launchBrowser": true, true -> false (browser will not be launched)

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

},

# (#10) environment variables in asp.net core | Asp.Net Core tutorial for beginners

# <https://www.youtube.com/watch?v=nQBNkISSE_4&list=PLaFzfwmPR7_LTXu0Vz9Zz_Y0OMMC7ArHZ&index=10>

# Agenda:

# - What are the environment variables.

# - How to set environment variables.

# - How to access environment variables.

# - Use of environment variables.

# What are the environment variables

# - ASP.NET Core uses an environment variable called ASPNETCORE\_ENVIRONMENT to indicate the runtime environment. The value of this variable can be anything as per your need but typically it can be Development, Staging, or Production.

# 

# How to set environment variables

# *launchSetting.json*

"profiles": {

"IIS Express": {

"commandName": "IISExpress",

"launchBrowser": true, true -> false (browser will not be launched)

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development" -> Development -> Production

}

},

# How to access environment variables

# 

# Use of environment variables

# 

# Used:

# - IsEnvironment(“”);

# - IsDevelopment();

# - IsProduction();

# - IsStaging();

# \*ASP.NET Core Understanding the middleware pipeline, Startup configuration - Application Life Cycle

# <https://www.youtube.com/watch?v=2SRUc7zZiyw>

# What is the Web Application?

# - Web Application is an application that handles HTTP requests and process it and returns HTTP response for the browser to interpret and renders the HTML browser.

# Application Life Cycle of ASP.NET Core application

# 

# How does it work

# Program.cs

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Hosting;

using System;

namespace MyFirstConsoleApp

{

class Program

{

static void Main(string[] args)

{

var builder = CreateHostBuilder(args);

var webhost = builder.Build(); -> in Startup.cs ConfigureServices method (C# reflection is used)

webhost.Run(); -> It is Kestrel Server, and it says “I will run”. So it goes to Confiugre method with all middlewares in Startup.cs

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args).ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>(); <- WebHostBuilder says that “I want to use Startup.cs to do the configuration”

});

}

# }