무료 전자 책

배우기

asp.net-core

Free unaffiliated eBook created from
Stack Overflow contributors.

#asp.net-core

1: asp.net-core

.NET Core Microsoft GitHub .NET . Windows, macOS Linux , / IoT .

.NET .

- : .
- : Windows, MacOS Linux . OS . (OS), CPU Microsoft, .
- : .
- : .NET Core .NET .NET Framework, Xamarin Mono .
- : .NET MIT Apache 2 . CC-BY . .NET Core .NET Foundation .
- Microsoft : .NET Core Microsoft, .NET Core Support .

RC1 *	1.0.0-rc1	2015-11-18
RC2 *	1.0.0-rc2	2016-05-16
1.0.0	1.0.0	2016-06-27
1.0.1	1.0.1	2016-09-13
1.1	1.1	2016-11-16

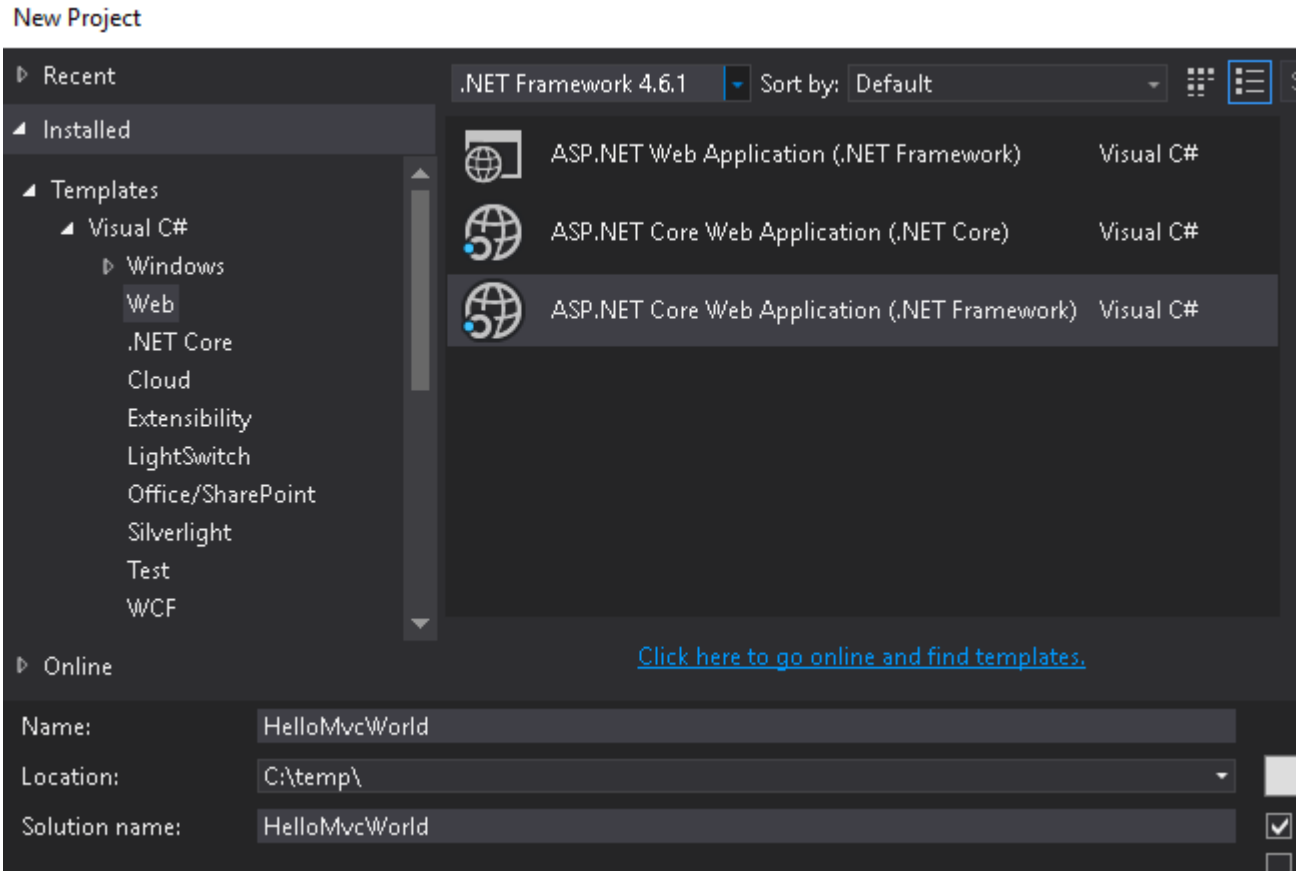
Examples

Visual Studio

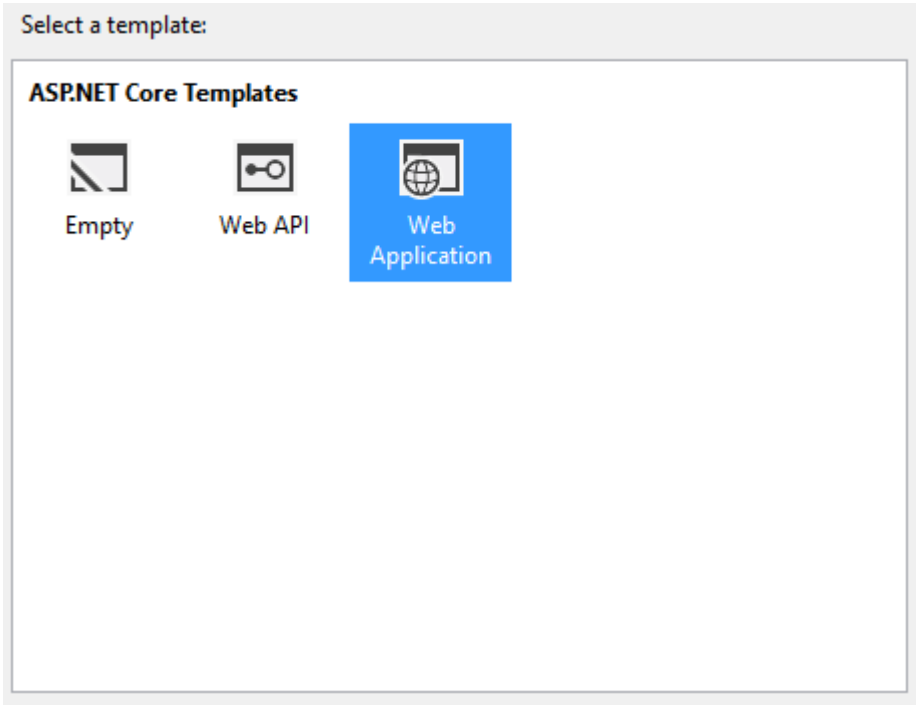
Visual Studio Visual Studio Community Edition . .

ASP.NET MVC .

1. **Visual Studio .**
2. **> .**
3. .
4. .
5. **: .NET Framework .**
6. .
7. .


New Project
Recent
Installed
Templates
Visual C#
Windows
Web
.NET Core
Cloud
Extensibility
LightSwitch
Office/SharePoint
Silverlight
Test
WCF
Online
.NET Framework 4.6.1
Sort by:
Default
ASP.NET Web Application (.NET Framework)
Visual C#
ASP.NET Core Web Application (.NET Core)
Visual C#
ASP.NET Core Web Application (.NET Framework)
Visual C#
Click here to go online and find templates.
Name:
HelloMvcWorld
Location:
C:\temp\
Solution name:
HelloMvcWorld

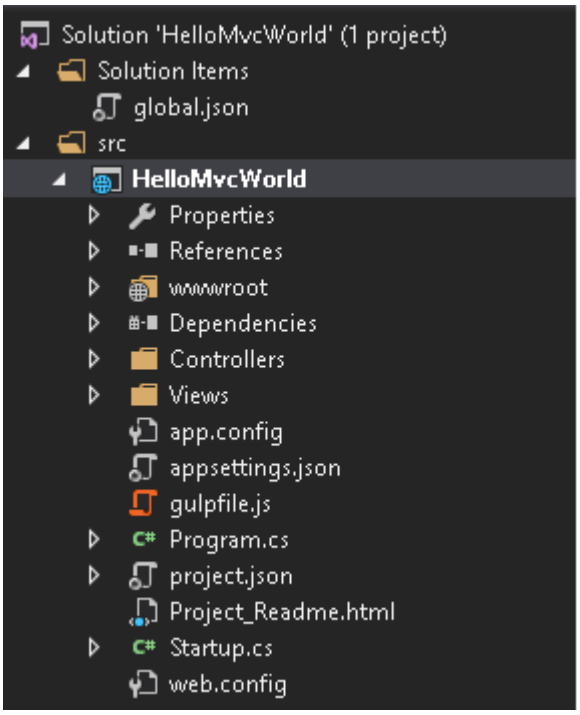
.


Select a template:
ASP.NET Core Templates
Empty
Web API
Web Application

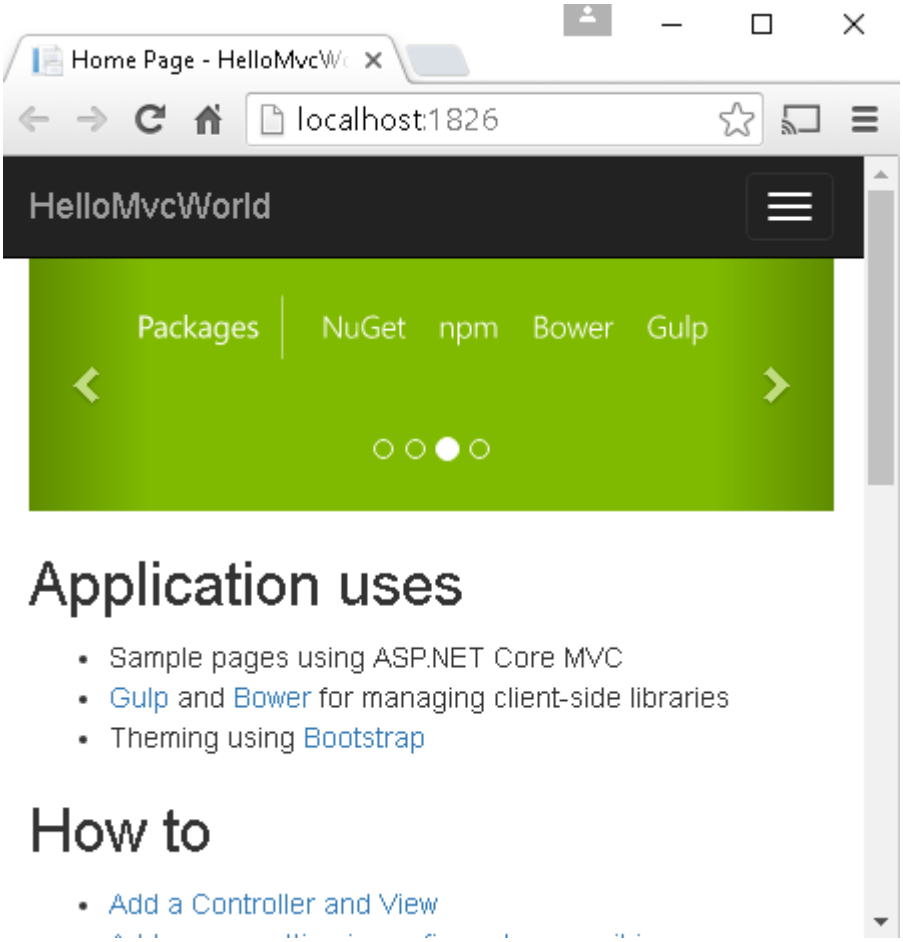
. , , .

.

.



F5 .



.

`dotnet` ASP.NET .

```
dotnet new web
```

```
dotnet restore
dotnet run
```

`dotnet new web` "" . `web` `dotnet` `ASP.NET Core Empty` . `dotnet new -all` . `console` , `classlib` , `mvc` `xunit` .

(`dotnet restore`) (`dotnet run`) .

(http : // localhost : 5000) .

ASP.NET Core MVC ASP.NET Core Web API

ASP.NET Core 1.0 MVC Web API ASP.NET Core MVC . MVC API , .

. , `Microsoft.AspNet.WebApi` API Web API 5.xx . `Microsoft.AspNetCore.Mvc` (`1.0.0`) . MVC . , .

API . API ? `Microsoft.AspNetCore.Mvc.Core` . MVC MVC .

MVC API . JSON CORS . ASP.NET 1.0 .json :

```
"Microsoft.AspNetCore.Mvc.Core": "1.0.0",
"Microsoft.AspNetCore.Mvc.Cors": "1.0.0",
"Microsoft.AspNetCore.Mvc.Formatters.Json": "1.0.0"
```

`AddMvcCore()` MVC .

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvcCore()
            .AddCors()
            .AddJsonFormatters();
}
```

`AddMvcCore` `IMvcCoreBuilder` . .

```
public void Configure(IApplicationBuilder app)
{
    app.UseCors(policy =>
    {
        policy.AllowAnyOrigin();
    });
    app.UseMvc();
}
```

" API `ApiController` . `Controller` . , JSON.NET .

`Controller` . `Controller` `Controller` . . `ApiController` .

```
/// <summary>
/// Base class for an API controller.
/// </summary>
```

```
[Controller]
public abstract class ApiController
{
    [ActionContext]
    public ActionContext ActionContext { get; set; }

    public HttpContext HttpContext => ActionContext?.HttpContext;

    public HttpRequest Request => ActionContext?.HttpContext?.Request;

    public HttpResponse Response => ActionContext?.HttpContext?.Response;

    public IServiceProvider Resolver => ActionContext?.HttpContext?.RequestServices;
}
```

`[Controller]` . `[ActionContext]` MVC `ActionContext` . `ActionContext` .

ASP.NET MVC `ControllerBase` . . .

ASP.NET Core MVC API . .

Visual Studio (cross plateform) aspnet

AspNetCore Mac, Linux, Window Docker .

1. Visual Studio .
2. C # extesnion
3. dot net core sdk . .

. . . Yeoman . Yeoman

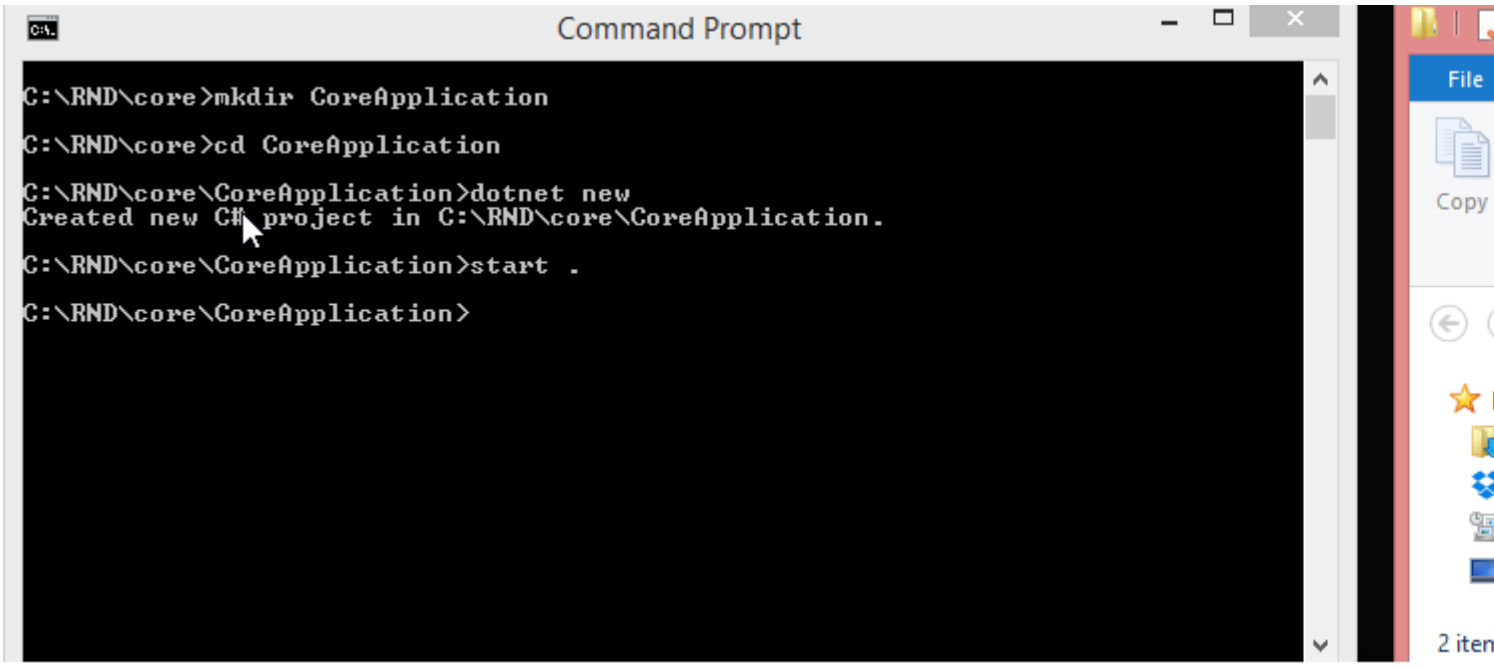
1. NPM . Node .
2. NPM Yeoman

   npm install -g yo
3. aspnet .

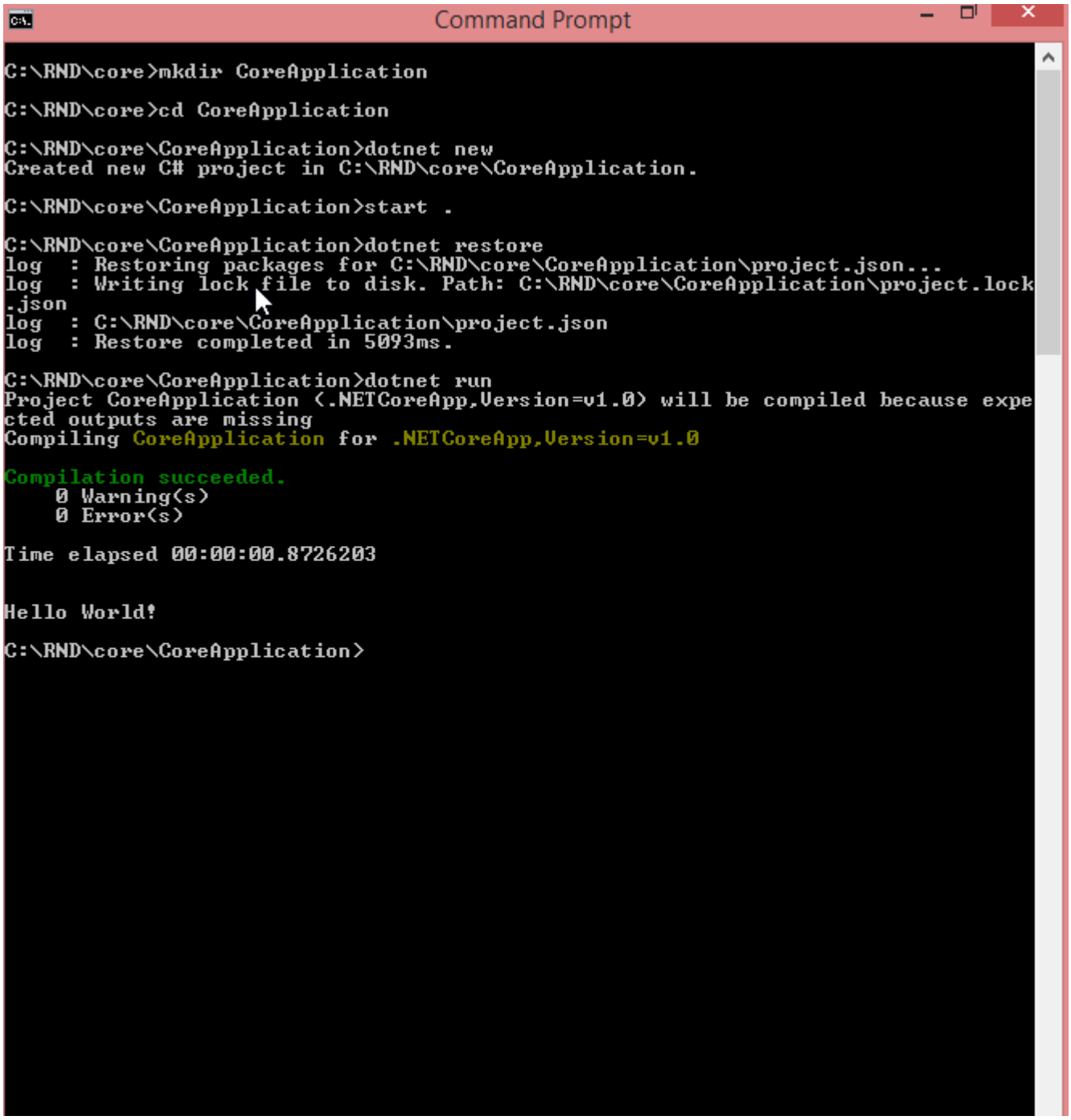
   npm install -g generator-aspnet

. DotNetCore Yo .

1. mkdir CoreApplication cd CoreApplication
2. dotnet



1. dotNet dotnet

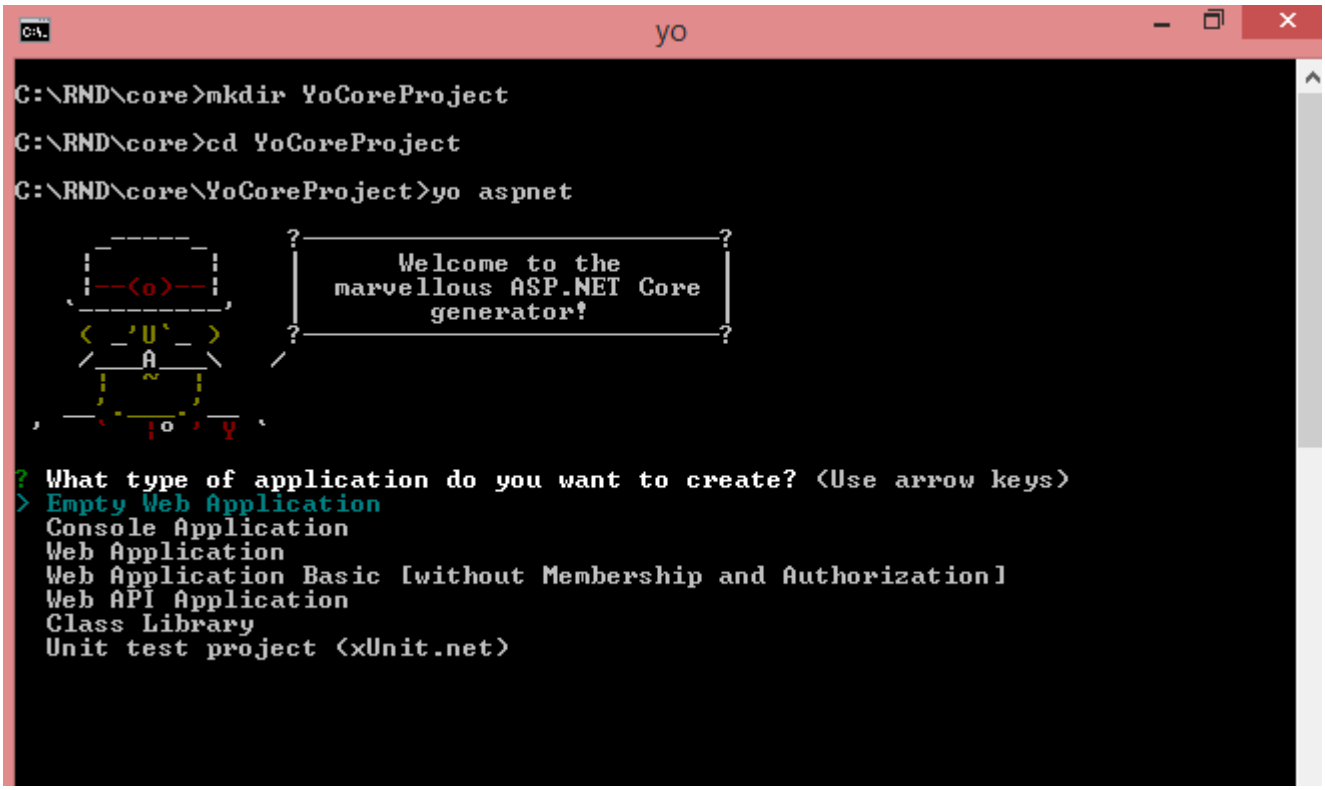


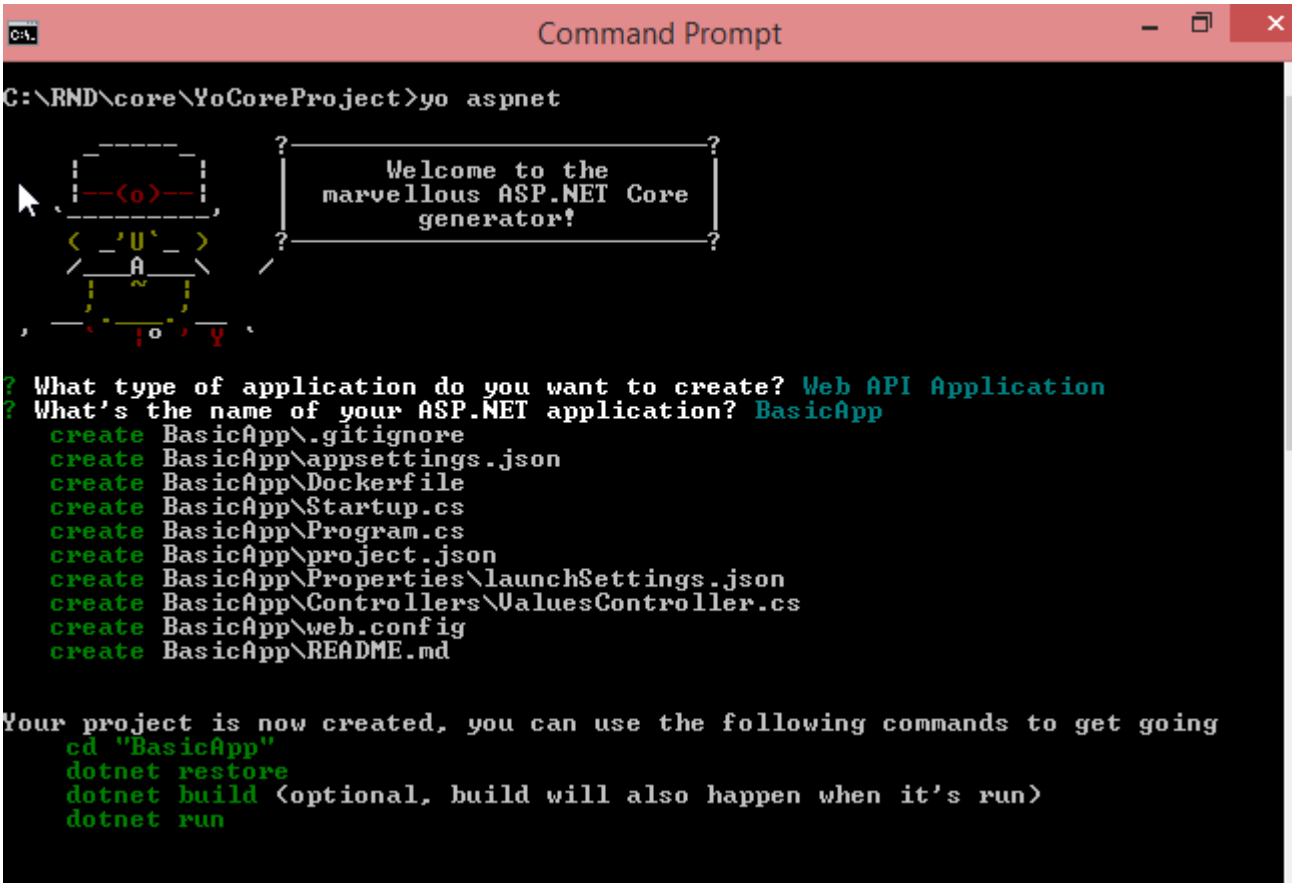
.

Yo

```
yo aspnet
```

Yeoman Project Type, Project Name .

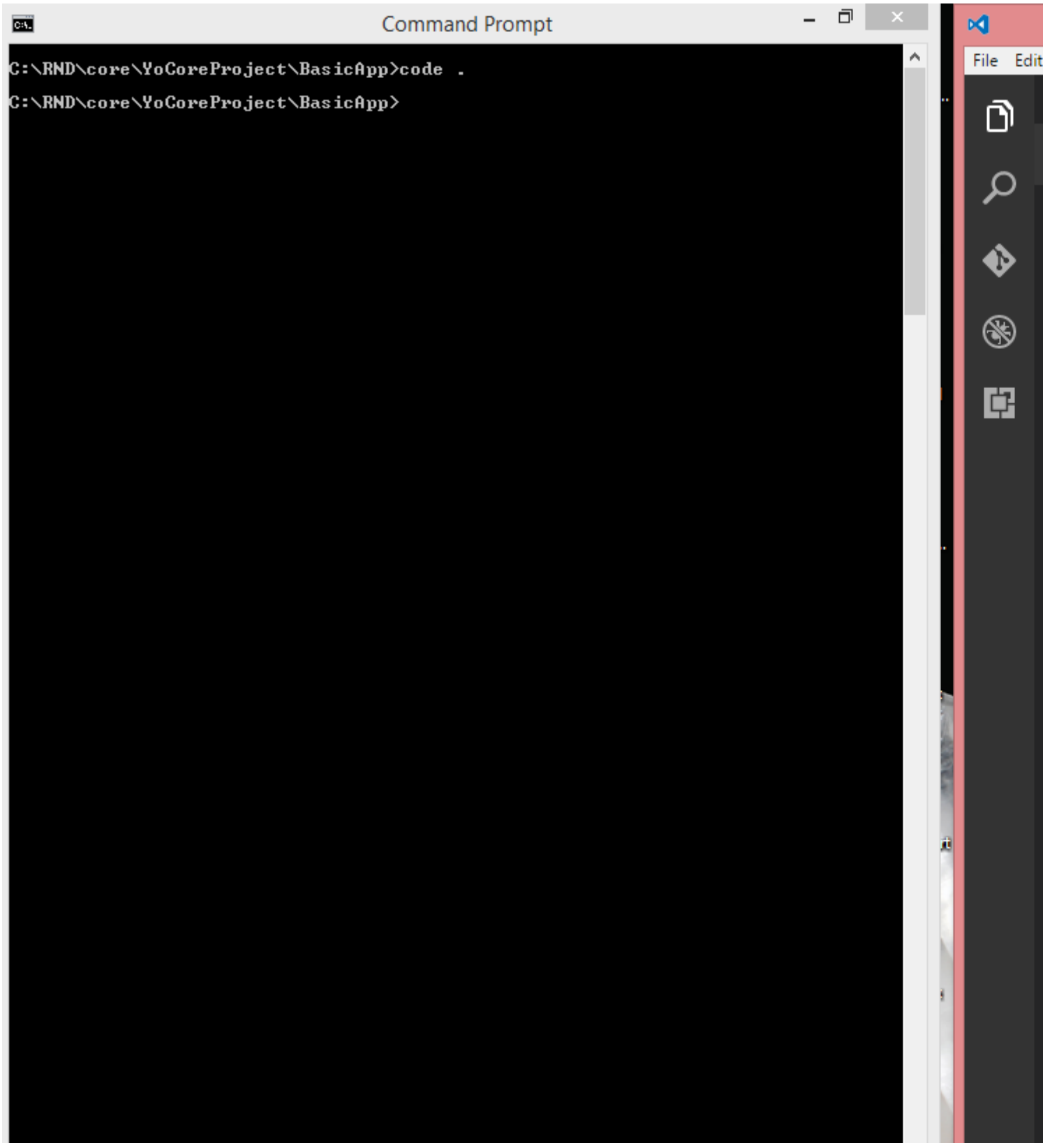




dotnet restore .

vs

.



. .

ASP.NET Core [Windows]

=> <=

ASP.NET Core `ASPNETCORE_ENVIRONMENT` . `Production` .

```
> dotnet run
Project TestApp (.NETCoreApp,Version=v1.0) was previously compiled. Skipping compilation.

Hosting environment: Production
Content root path: C:\Projects\TestApp
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
```

Windows

Windows setx.exe . .

```
>setx ASPNETCORE_ENVIRONMENT "Development"

SUCCESS: Specified value was saved.
```

. . /M ( ) .

```
>setx ASPNETCORE_ENVIRONMENT "Development" /M

SUCCESS: Specified value was saved.
```

PowerShell PowerShell . PowerShell `$Env:` .

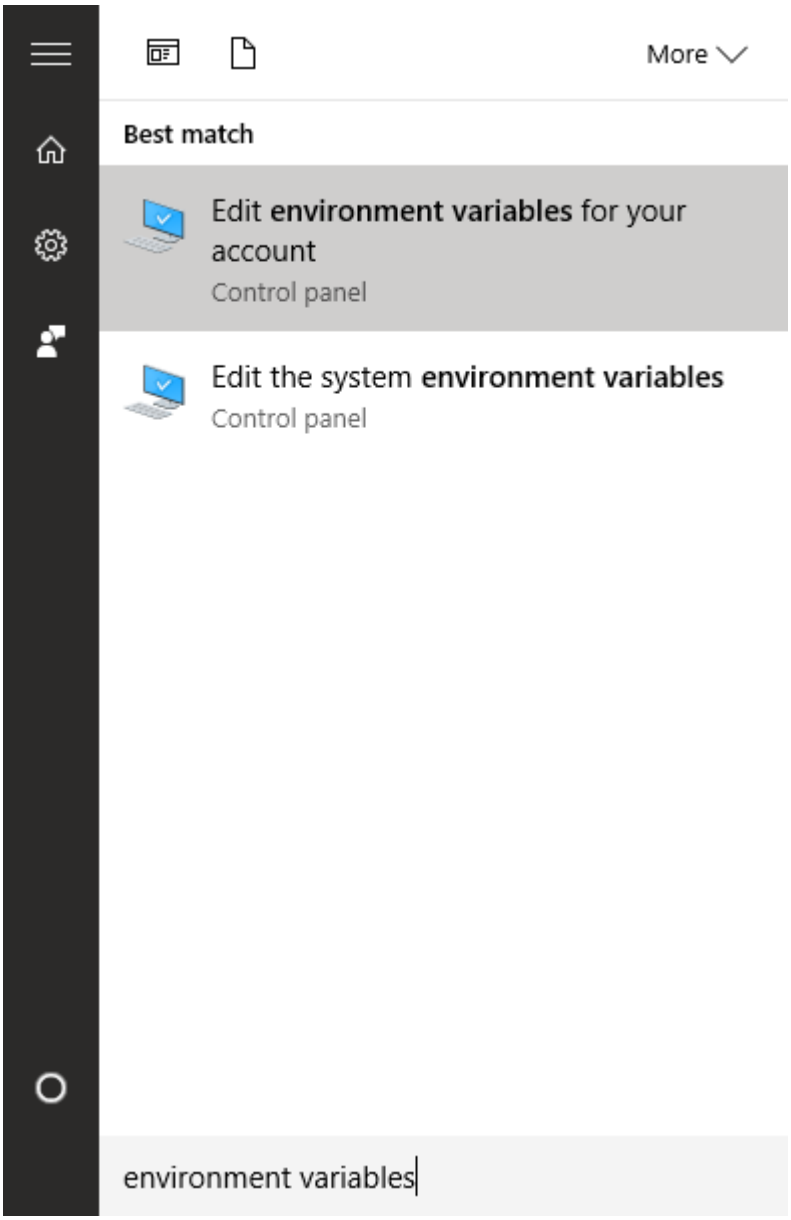
```
$Env:ASPNETCORE_ENVIRONMENT = "Development"
```

PowerShell . .

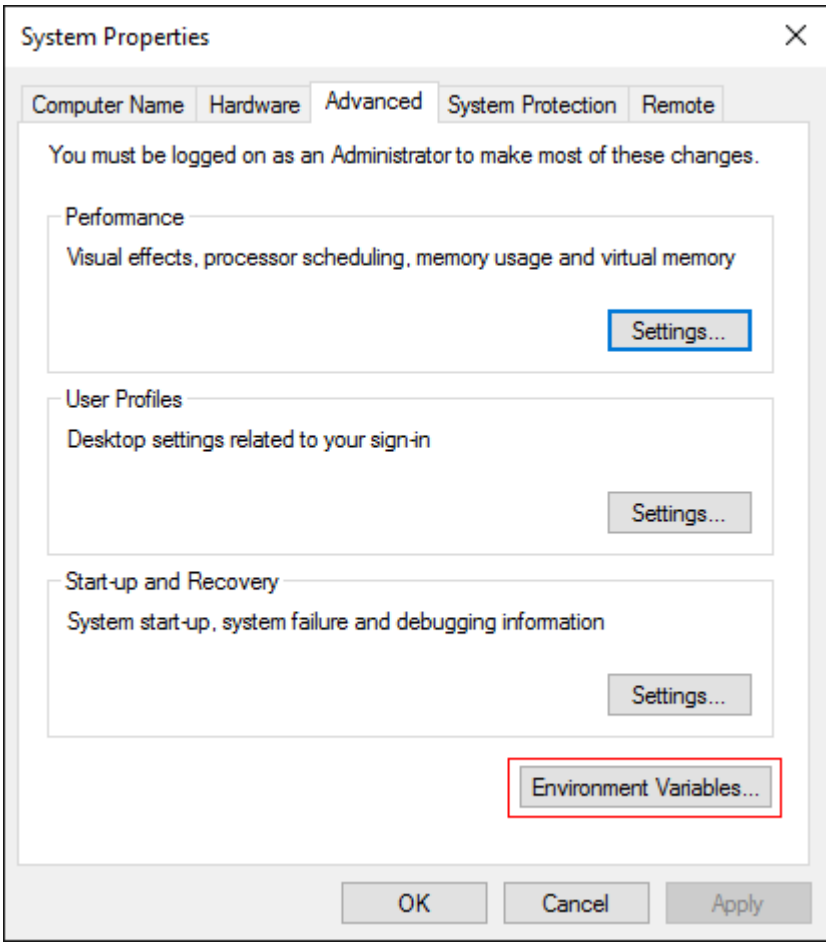
. PowerShell . .

```
[Environment]::SetEnvironmentVariable("ASPNETCORE_ENVIRONMENT", "Development", "User")
[Environment]::SetEnvironmentVariable("ASPNETCORE_ENVIRONMENT", "Development", "Machine")
```

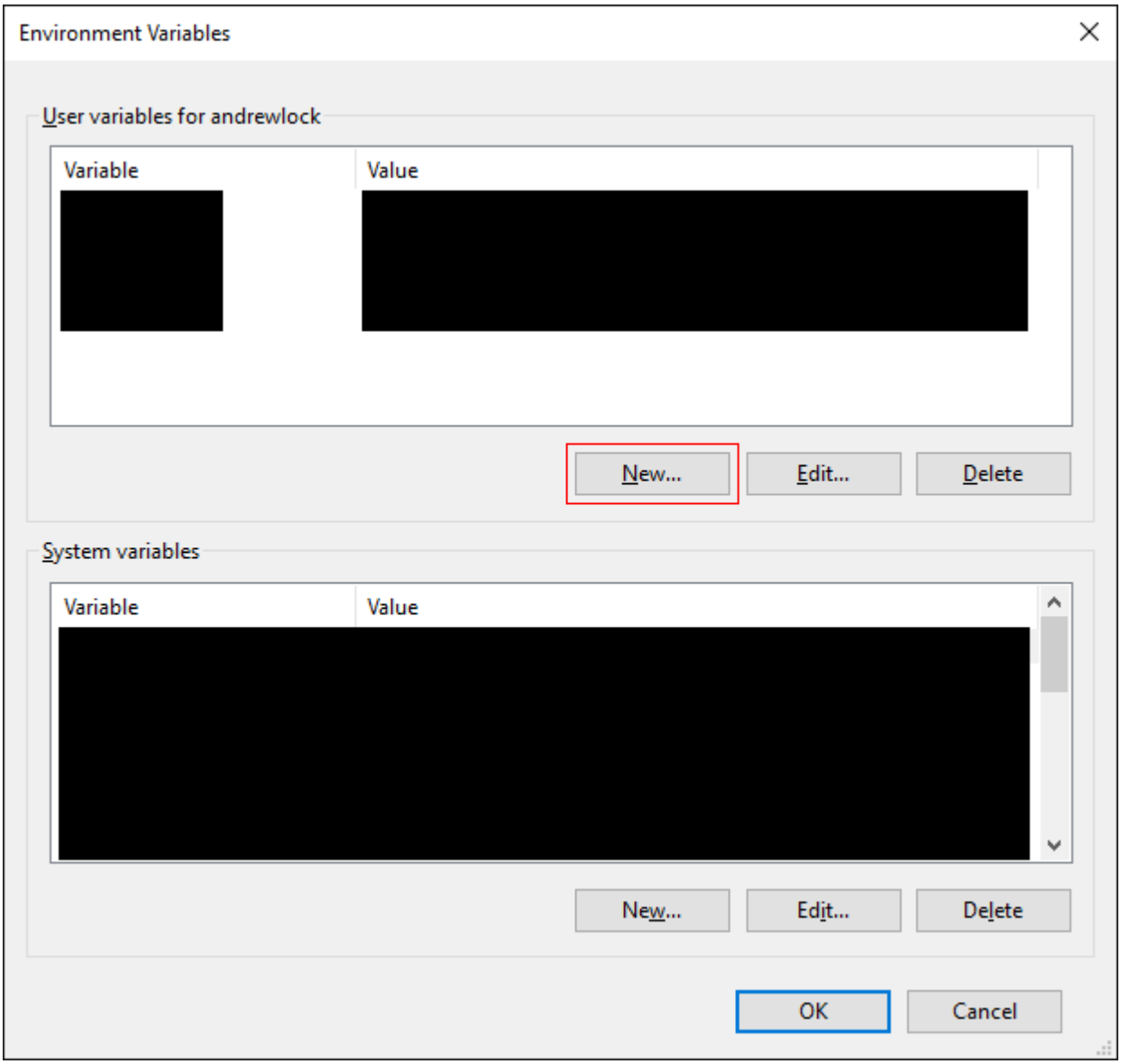
Windows ! Windows Windows `environment variables environment variables` *:*


More
Best match
Edit environment variables for your account
Control panel
Edit the system environment variables
Control panel
environment variables

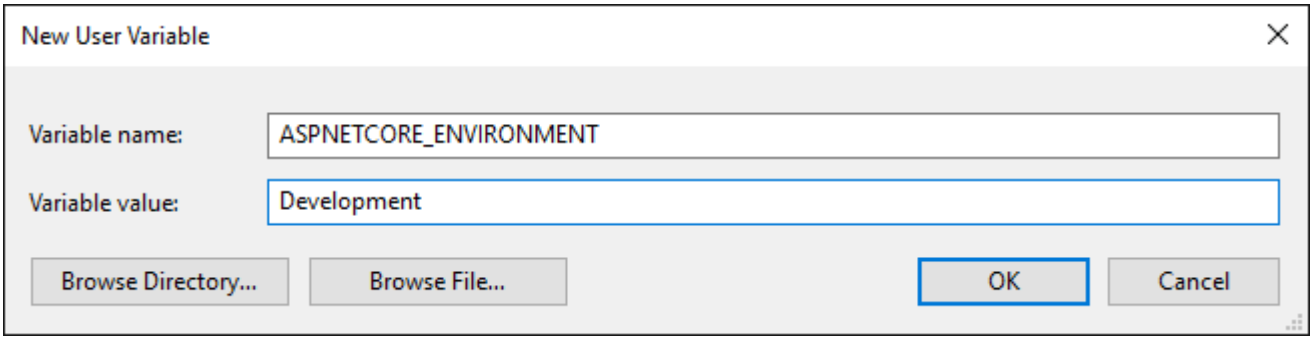
.


System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage and virtual memory
Settings...
User Profiles
Desktop settings related to your sign-in
Settings...
Start-up and Recovery
System start-up, system failure and debugging information
Settings...
Environment Variables...
OK
Cancel
Apply

.



`ASPNETCORE_ENVIRONMENT` New ... .



.

.

asp.net-core : https://riptutorial.com/ko/asp-net-core/topic/810/asp-net-core-

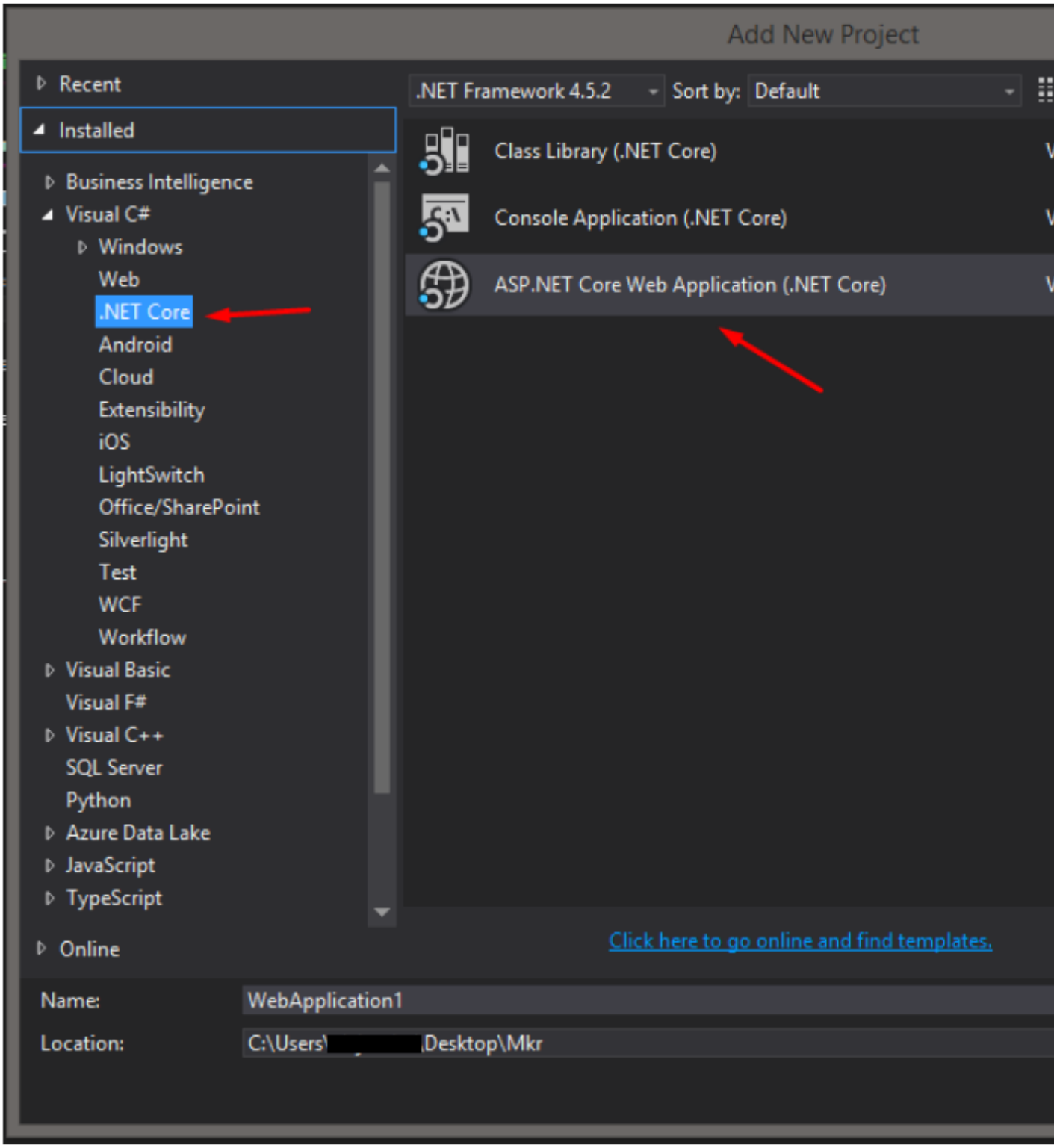
2: Angular2 .Net Core

Examples

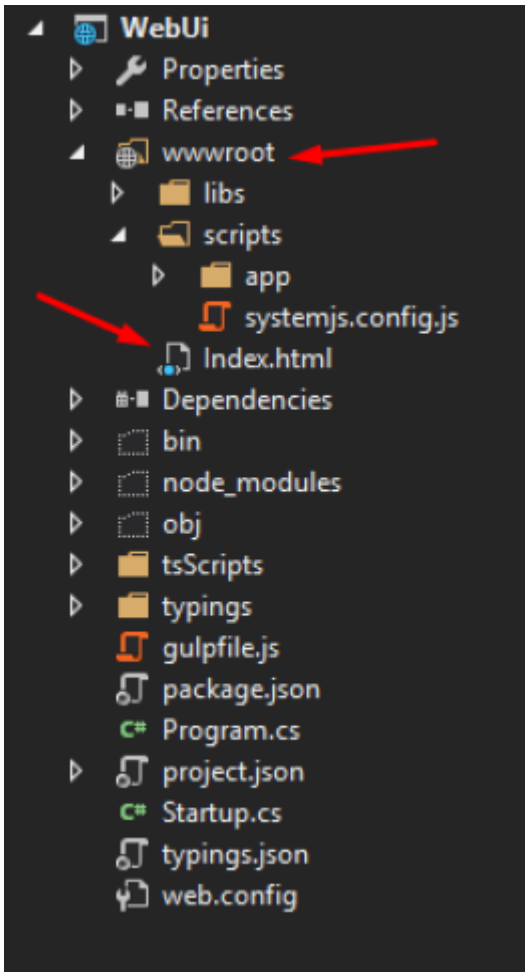
Angular 2 Hello World ! Visual Studio 2015 .Net Core

:

1. .Net :

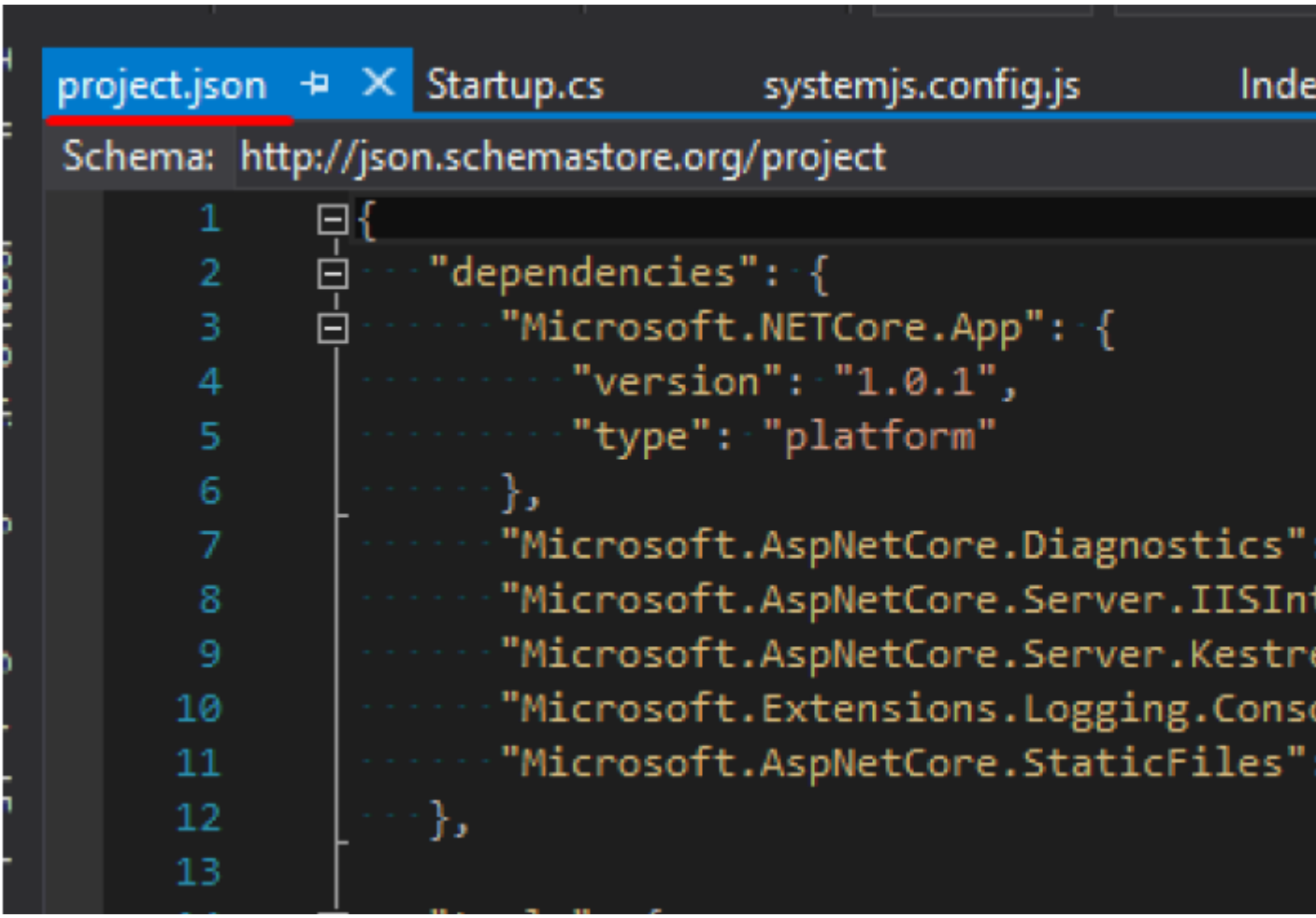


2. wwwroot Index.html html .



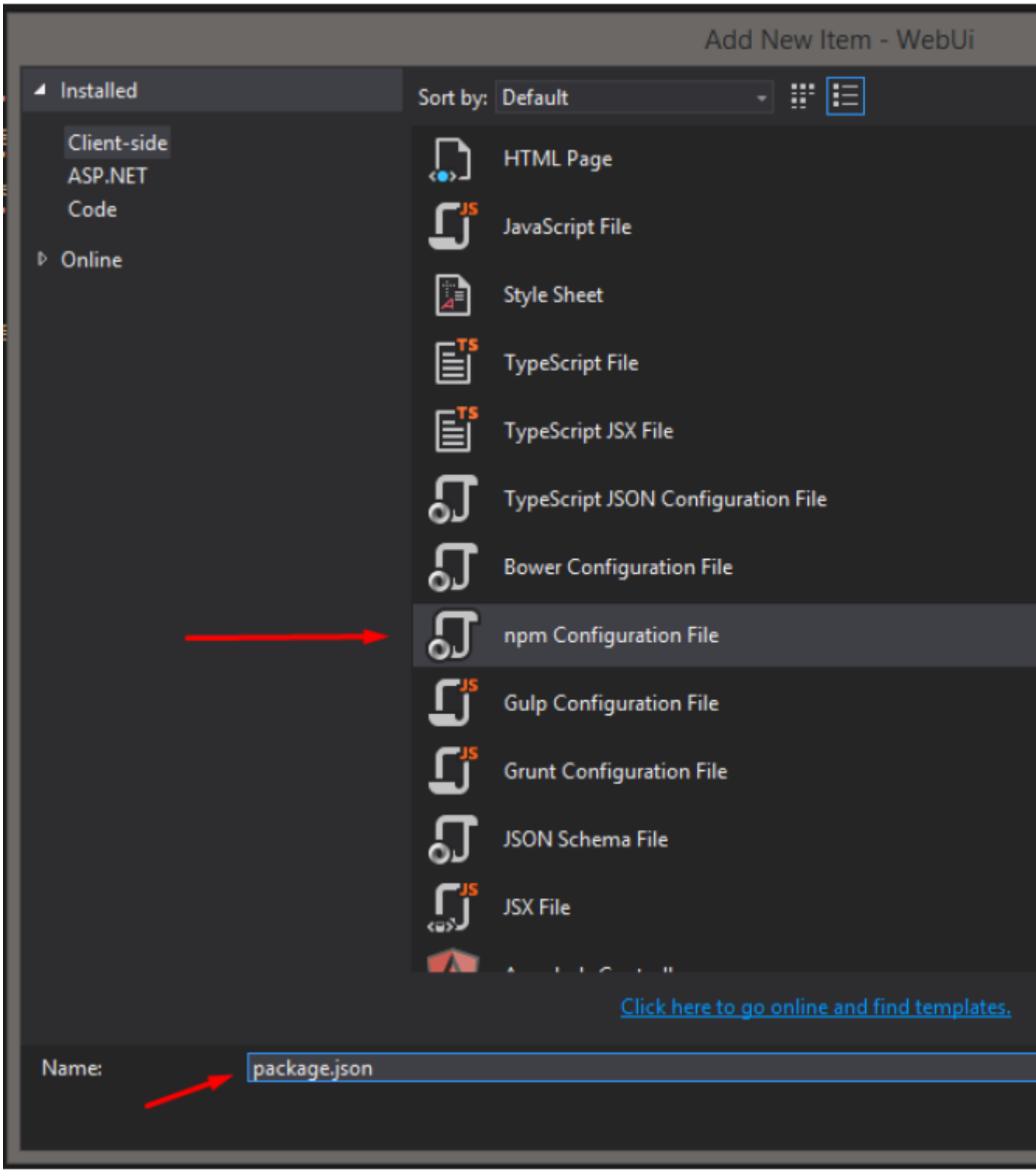
3. Startup.cs ("project.json" "Microsoft.AspNetCore.StaticFiles": "1.0.0").

```
// This method gets called by the runtime. Use this method to configure the
-references
public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILo
{
    app.UseDefaultFiles();
    app.UseStaticFiles();
}
```



4. NPN :

- WebUi NPN (package.json) .


Add New Item - WebUi
Installed
Client-side
ASP.NET
Code
Online
Sort by: Default
HTML Page
JavaScript File
Style Sheet
TypeScript File
TypeScript JSX File
TypeScript JSON Configuration File
Bower Configuration File
npm Configuration File
Gulp Configuration File
Grunt Configuration File
JSON Schema File
JSX File
Click here to go online and find templates.
Name:
package.json

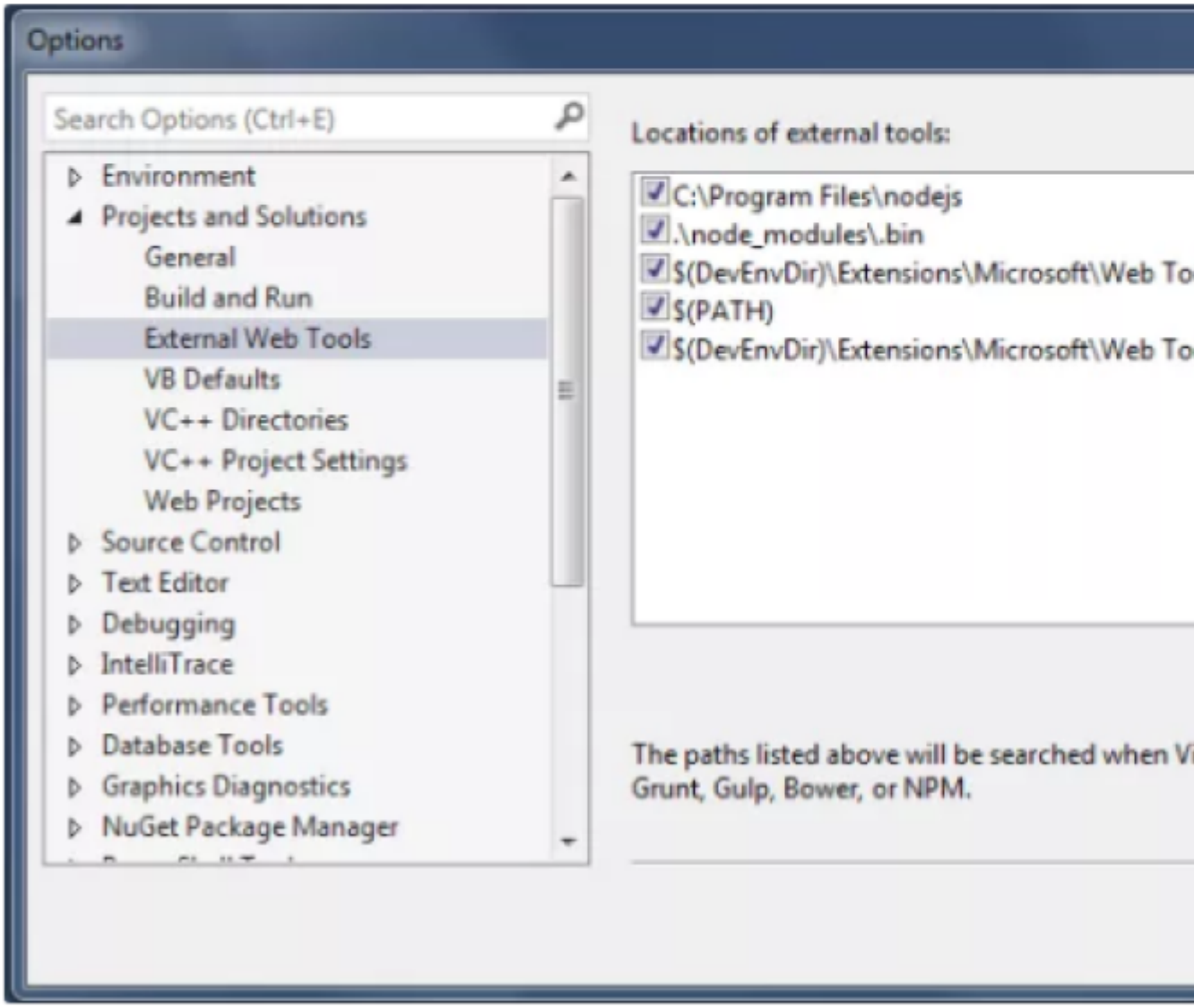
- .

: Visual Studio (.) Visual Studio , js Visual Studio .

. js : https://nodejs.org/es/download/

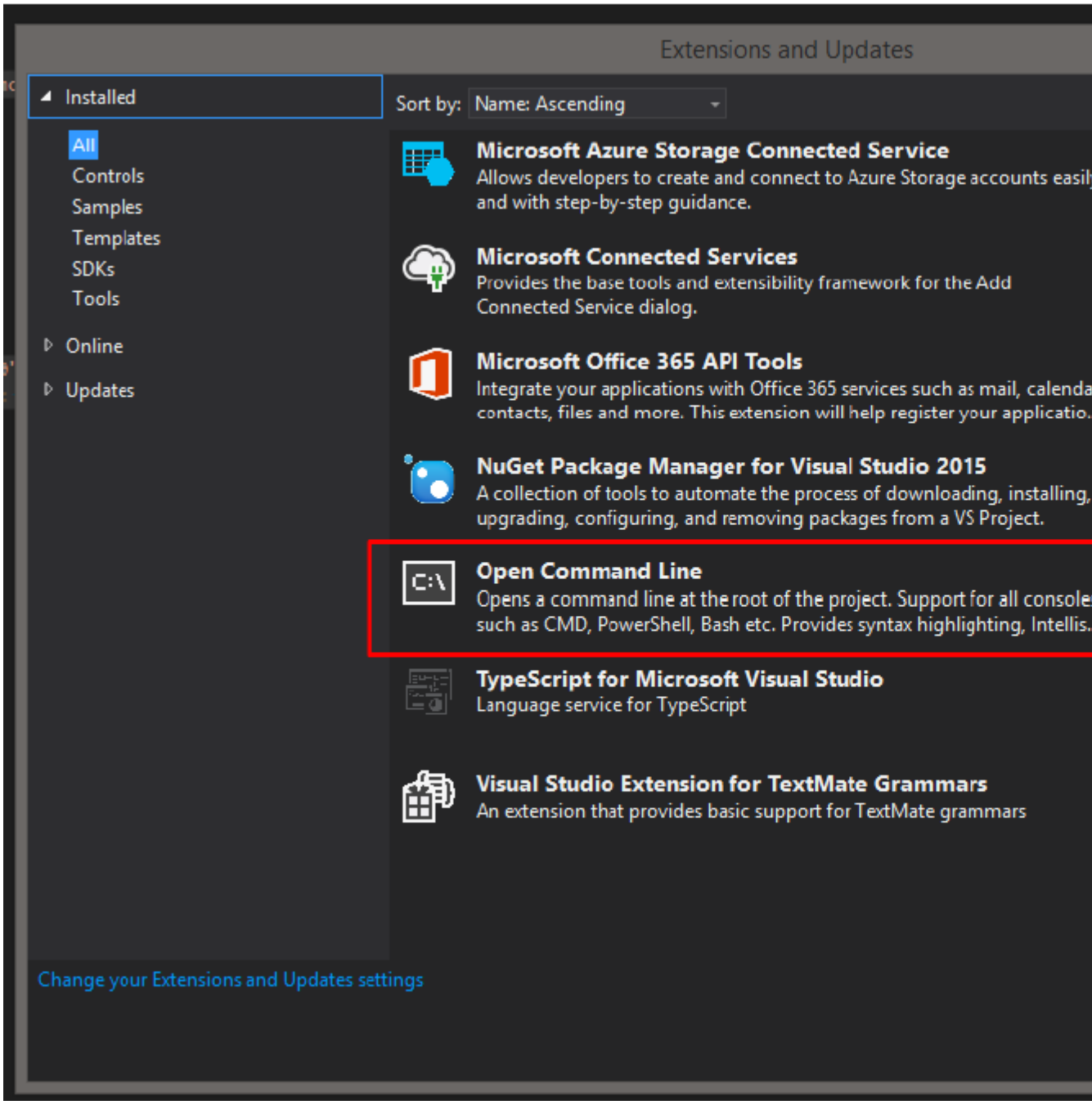
ii. Visual Studio : https://ryanhayes.net/synchronize-node-js-install-version-with-visual-studio-2015/ :

1. First, **find the Node.js installation you already have** and u
   By default, Node.js 0.12.7 installs to "C:\Program Files\nod
2. Once you've got that all copied out to your clipboard, got t
   Studio 2015.
3. In this dialog, go to **Projects and Solutions > External Web**
   that manages all of the 3rd party tools used within VS. Th
   pointed to.
4. **Add an entry at the top to the path to the node.js directory**
   use that version instead.



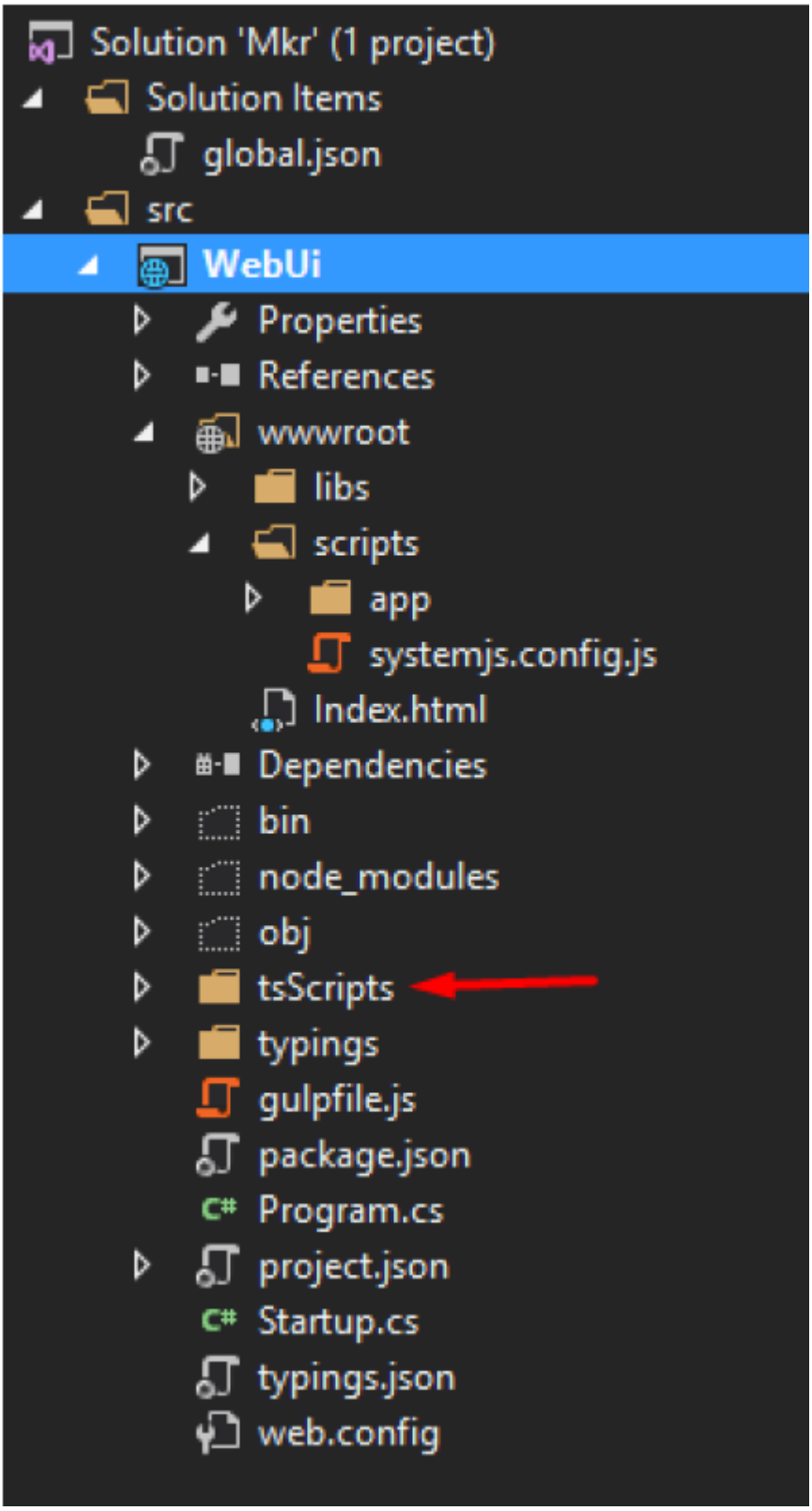
iii. ( ) package.json , package.json "npm install" .

: Visual Studio " " .

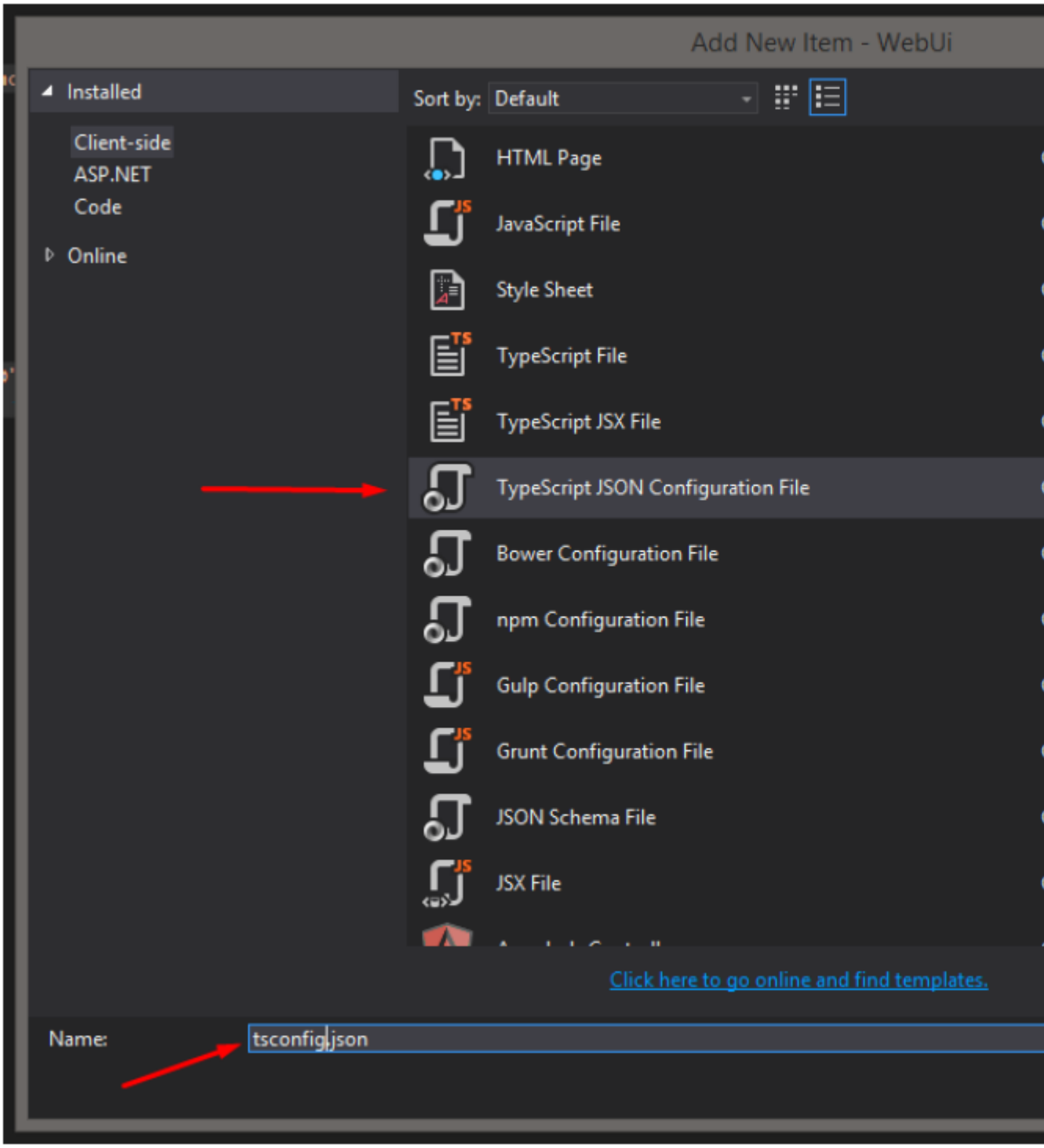


5. :

- WebUi TsScript (TypeScript JS). JS gulp wwwroot foder .).

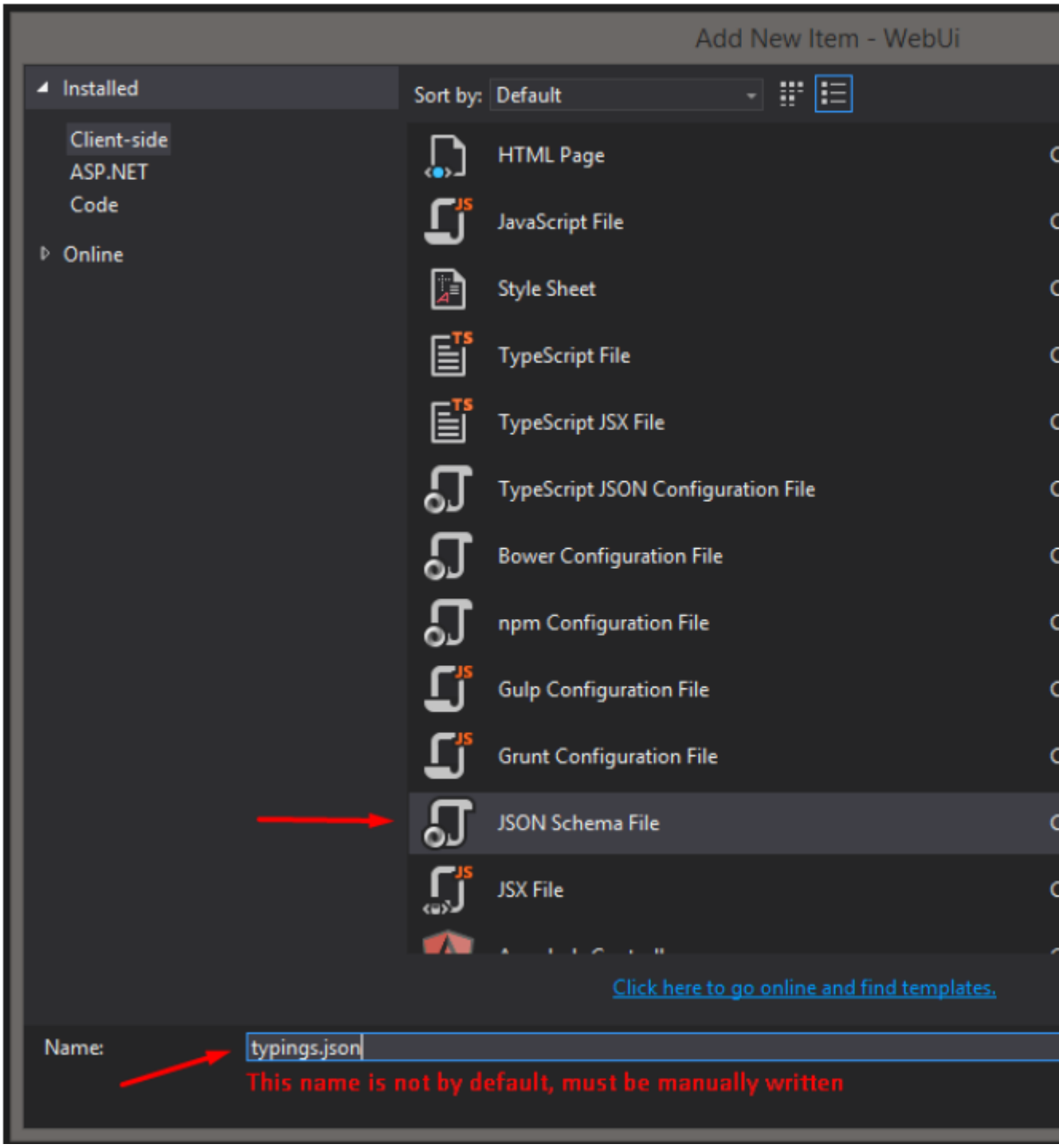


- "TypeScript JSON "(tsconfig.json) .

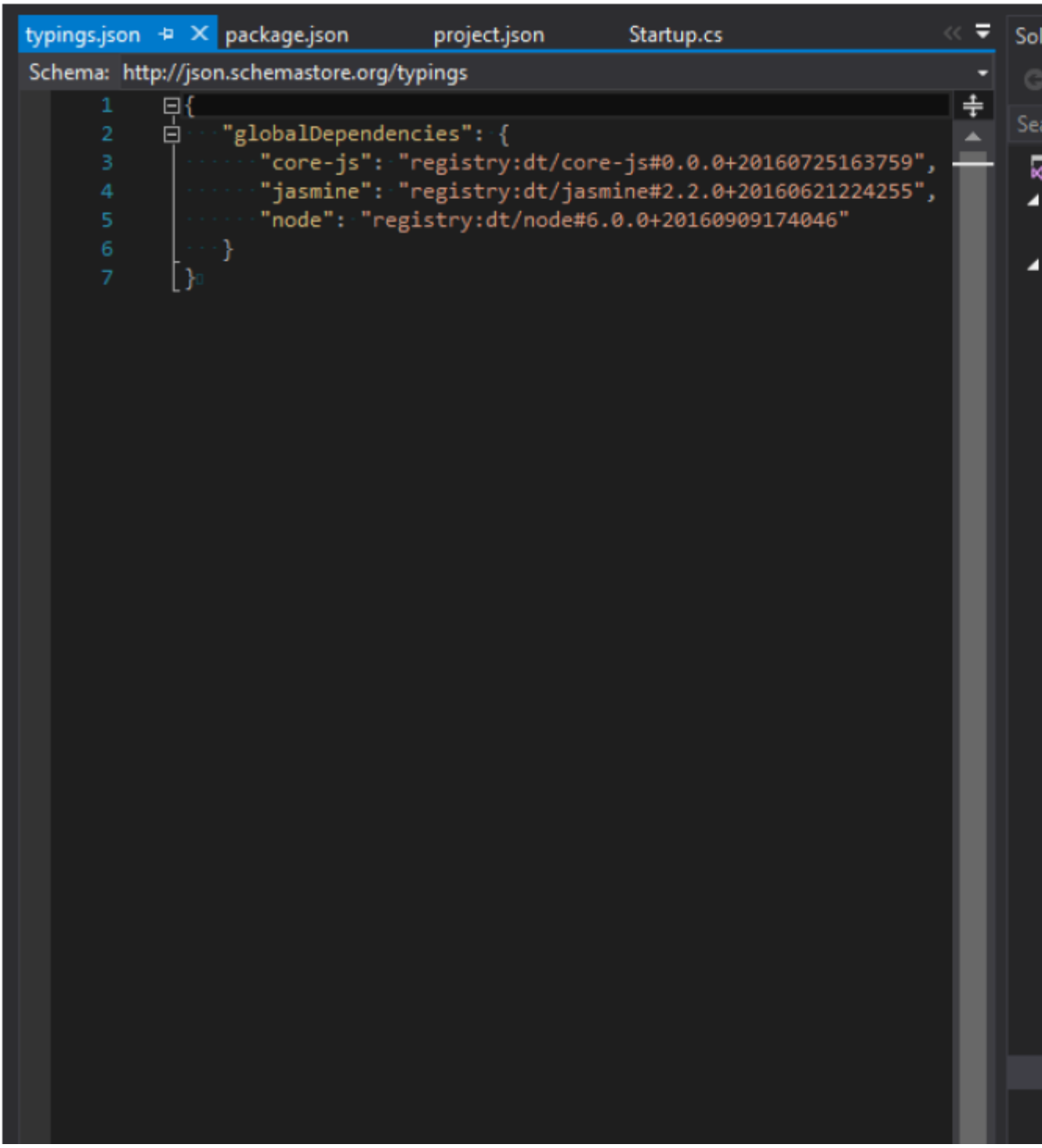

Add New Item - WebUi
Installed
Client-side
ASP.NET
Code
Online
Sort by: Default
HTML Page
JavaScript File
Style Sheet
TypeScript File
TypeScript JSX File
TypeScript JSON Configuration File
Bower Configuration File
npm Configuration File
Gulp Configuration File
Grunt Configuration File
JSON Schema File
JSX File
Click here to go online and find templates.
Name:
tsconfig.json

.

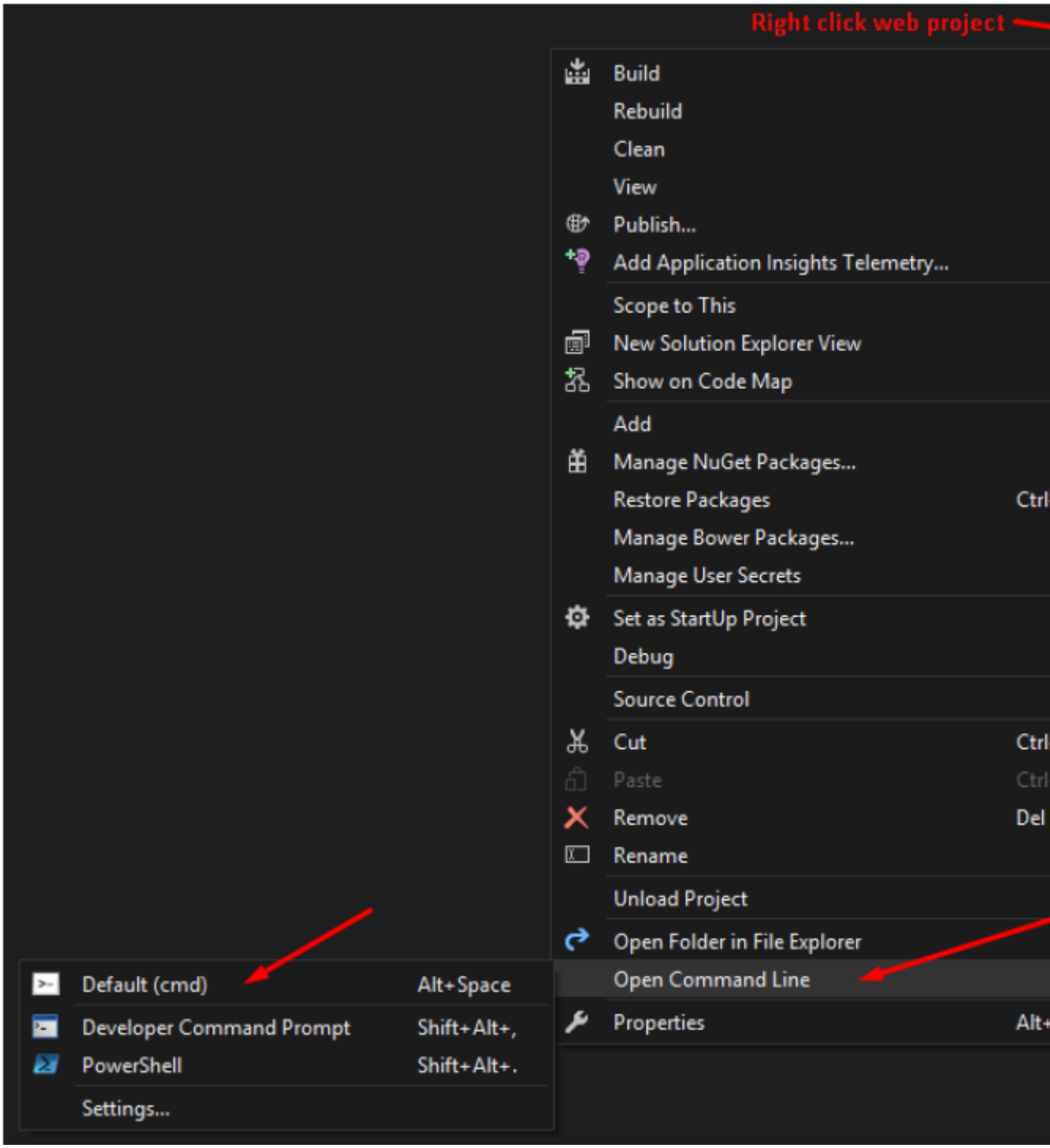
- WebUi typings.json .

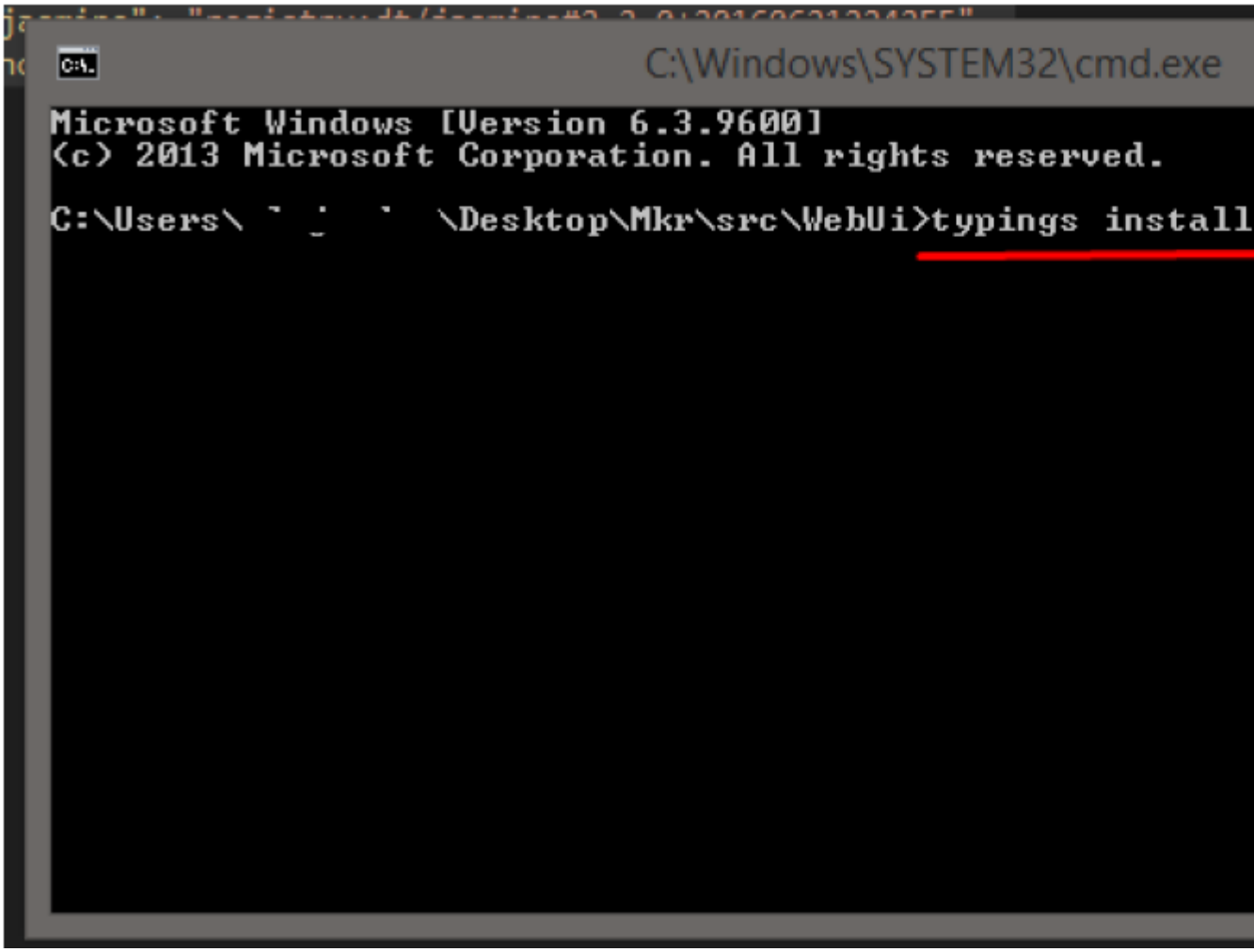

Add New Item - WebUi
Installed
Client-side
ASP.NET
Code
Online
Sort by: Default
HTML Page
JavaScript File
Style Sheet
TypeScript File
TypeScript JSX File
TypeScript JSON Configuration File
Bower Configuration File
npm Configuration File
Gulp Configuration File
Grunt Configuration File
JSON Schema File
JSX File
Click here to go online and find templates.
Name:
typings.json
This name is not by default, must be manually written

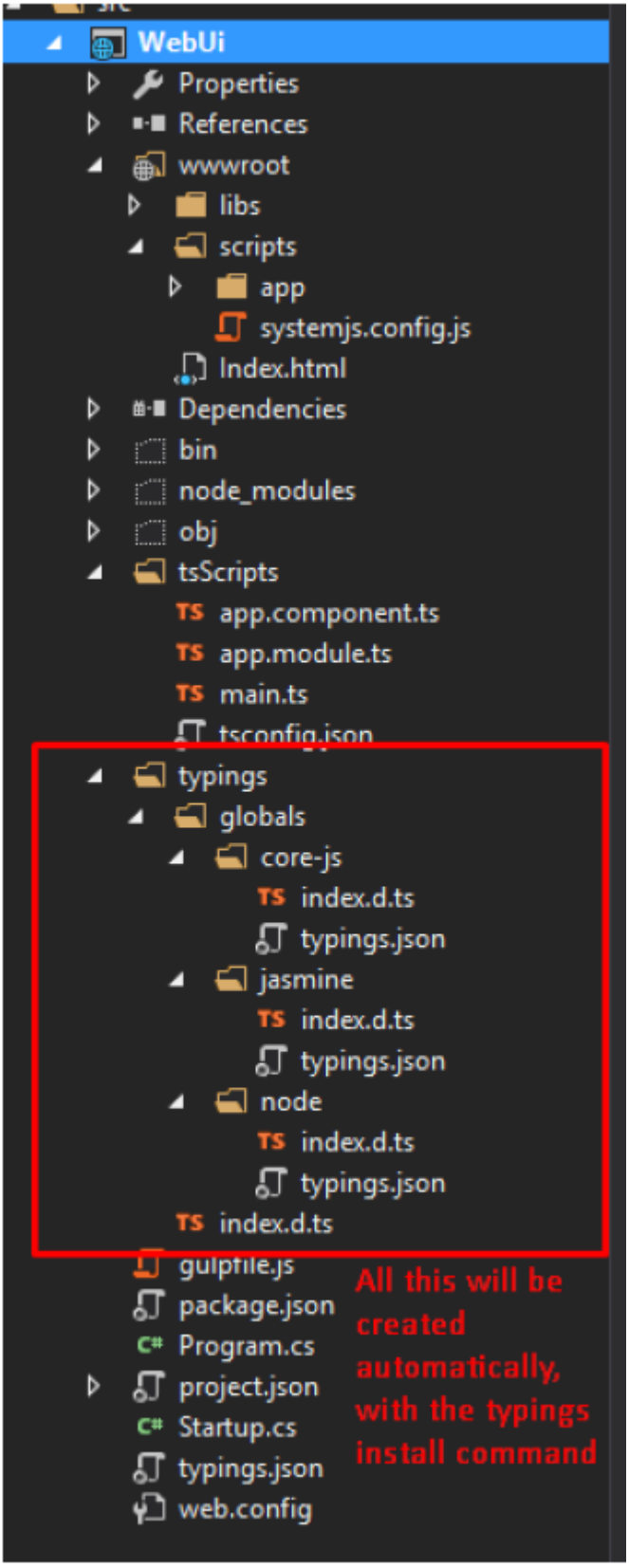
.



- Web Project " " (4, iii " ").

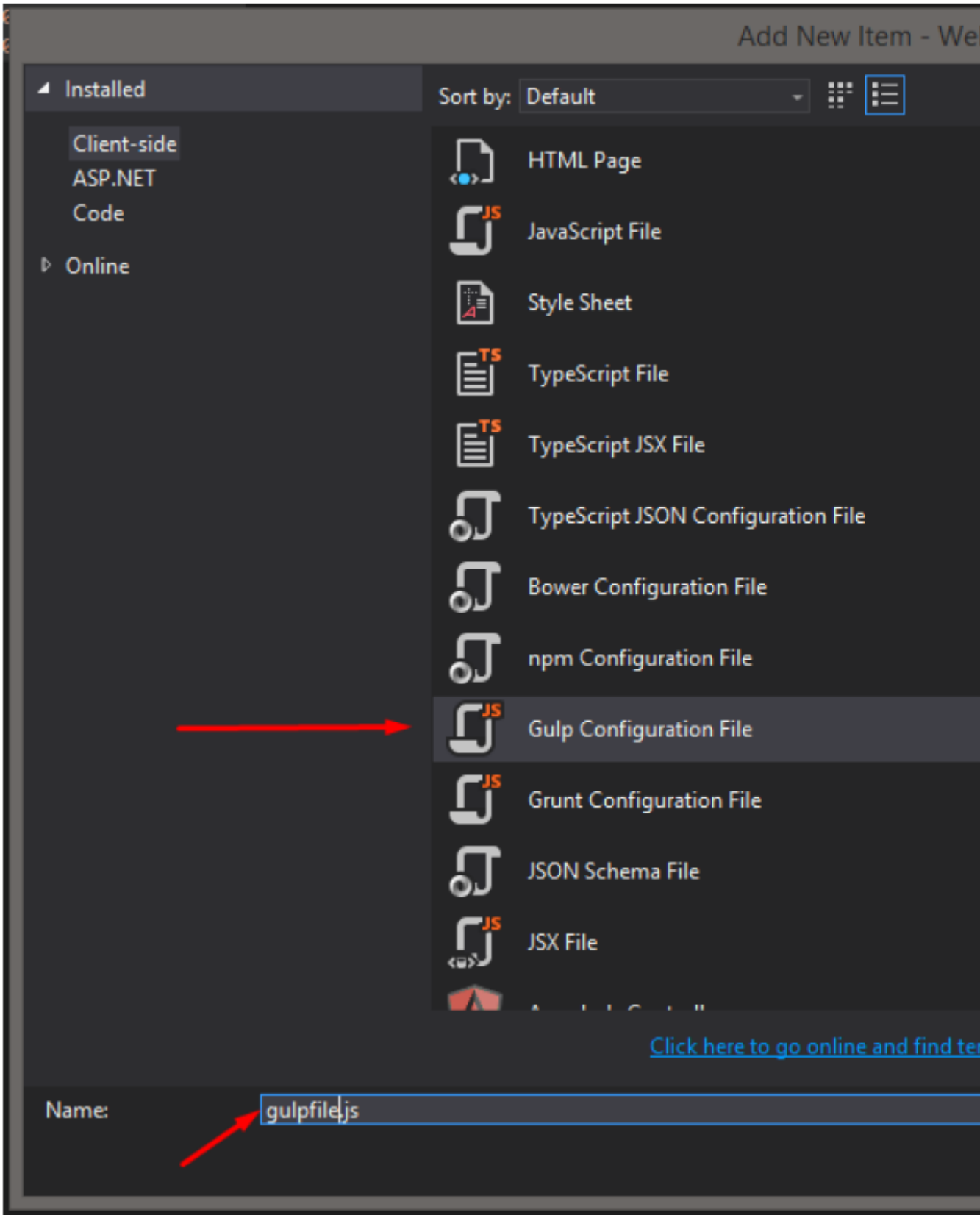

Right click web project
Build
Rebuild
Clean
View
Publish...
Add Application Insights Telemetry...
Scope to This
New Solution Explorer View
Show on Code Map
Add
Manage NuGet Packages...
Restore Packages
Manage Bower Packages...
Manage User Secrets
Set as StartUp Project
Debug
Source Control
Cut
Paste
Remove
Rename
Unload Project
Open Folder in File Explorer
Open Command Line
Properties
Default (cmd) Alt+Space
Developer Command Prompt Shift+Alt+,
PowerShell Shift+Alt+.
Settings...


C:\Windows\SYSTEM32\cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
C:\Users\ ` _ ` \Desktop\Mkr\src\WebUi>typings install

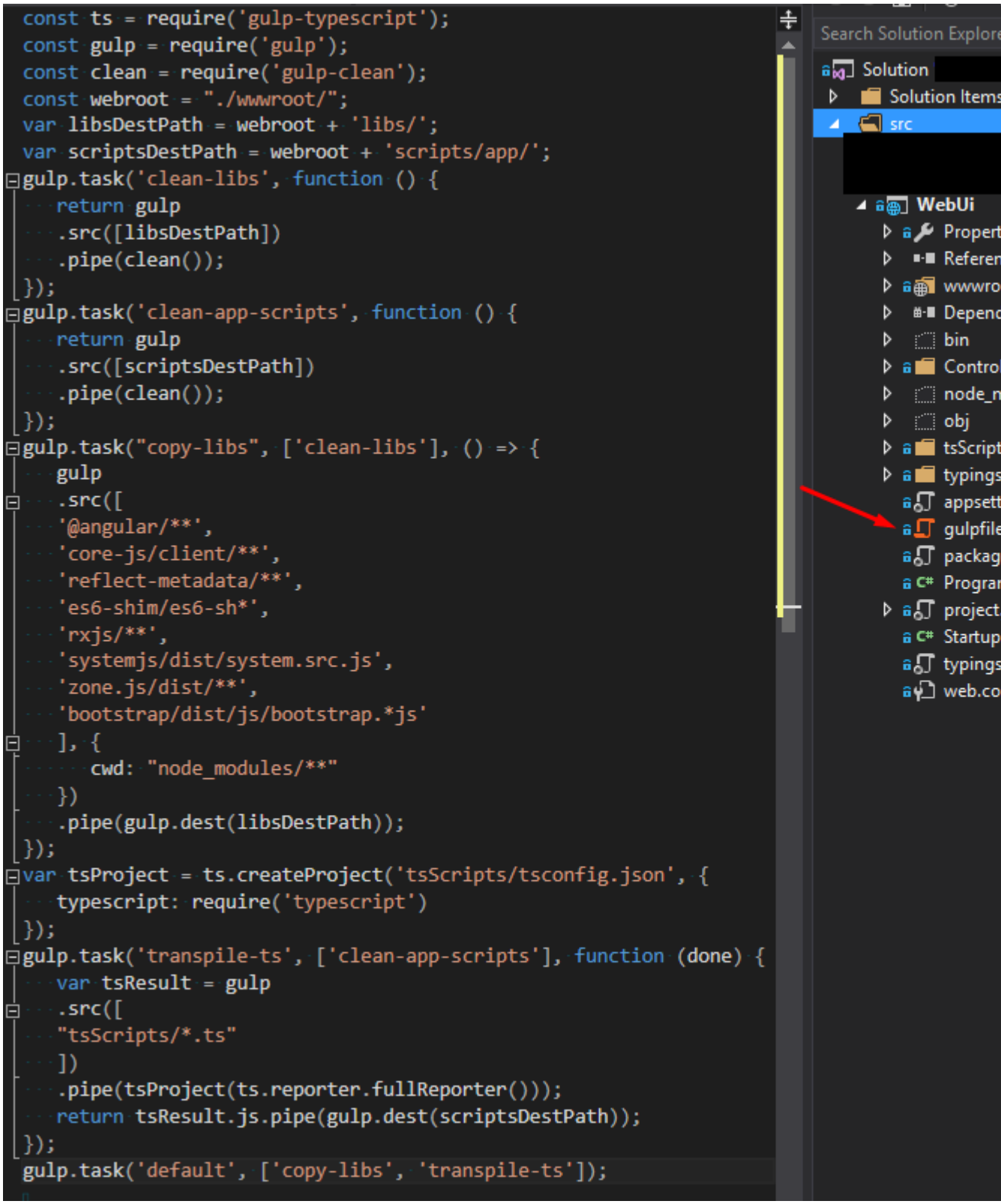


6. .

- "Gulp Configuration File"(gulpfile.js) .

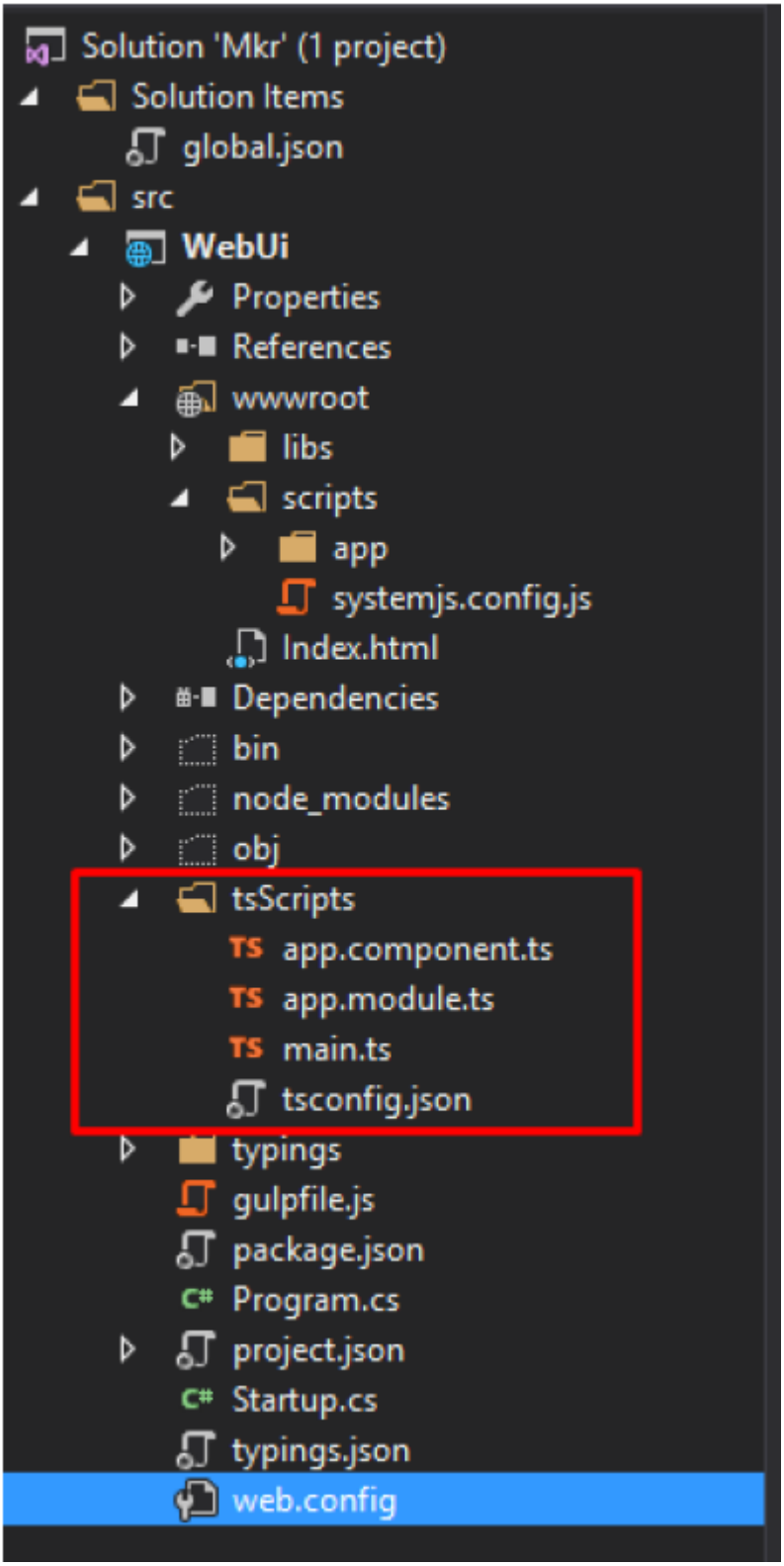

Add New Item - We
Installed
Client-side
ASP.NET
Code
Online
Sort by: Default
HTML Page
JavaScript File
Style Sheet
TypeScript File
TypeScript JSX File
TypeScript JSON Configuration File
Bower Configuration File
npm Configuration File
Gulp Configuration File
Grunt Configuration File
JSON Schema File
JSX File
Click here to go online and find te
Name:
gulpfile.js

- :



```
const ts = require('gulp-typescript');
const gulp = require('gulp');
const clean = require('gulp-clean');
const webroot = "./wwwroot/";
var libsDestPath = webroot + 'libs/';
var scriptsDestPath = webroot + 'scripts/app/';
gulp.task('clean-libs', function () {
    return gulp
    .src([libsDestPath])
    .pipe(clean());
});
gulp.task('clean-app-scripts', function () {
    return gulp
    .src([scriptsDestPath])
    .pipe(clean());
});
gulp.task("copy-libs", ['clean-libs'], () => {
    gulp
    .src([
    '@angular/**',
    'core-js/client/**',
    'reflect-metadata/**',
    'es6-shim/es6-sh*',
    'rxjs/**',
    'systemjs/dist/system.src.js',
    'zone.js/dist/**',
    'bootstrap/dist/js/bootstrap.*js'
    ], {
        cwd: "node_modules/**"
    })
    .pipe(gulp.dest(libsDestPath));
});
var tsProject = ts.createProject('tsScripts/tsconfig.json', {
    typescript: require('typescript')
});
gulp.task('transpile-ts', ['clean-app-scripts'], function (done) {
    var tsResult = gulp
    .src([
    "tsScripts/*.ts"
    ])
    .pipe(tsProject(ts.reporter.fullReporter()));
    return tsResult.js.pipe(gulp.dest(scriptsDestPath));
});
gulp.task('default', ['copy-libs', 'transpile-ts']);
```


7. "tsScripts" Angular 2 :



app.component.ts

```
import { Component } from '@angular/core';

@Component({
    selector: 'my-app',
    template: `<h1>Hello {{name}}</h1>`
})
export class AppComponent { name = 'Angular'; }
```

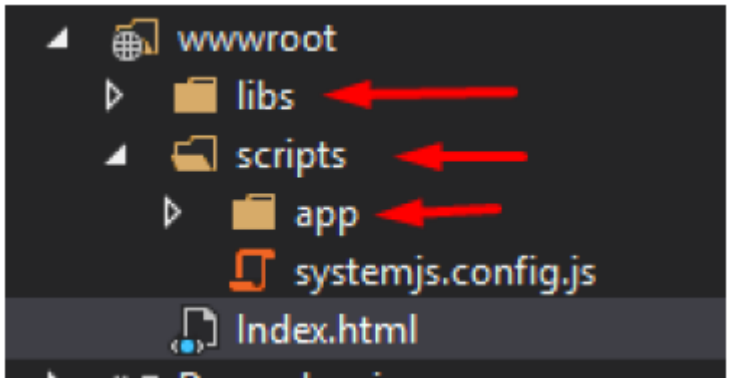
app.module.ts

```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent }  from './app.component';

@NgModule({
    imports: [BrowserModule],
    declarations: [AppComponent],
    bootstrap: [AppComponent]
})
export class AppModule { }
```

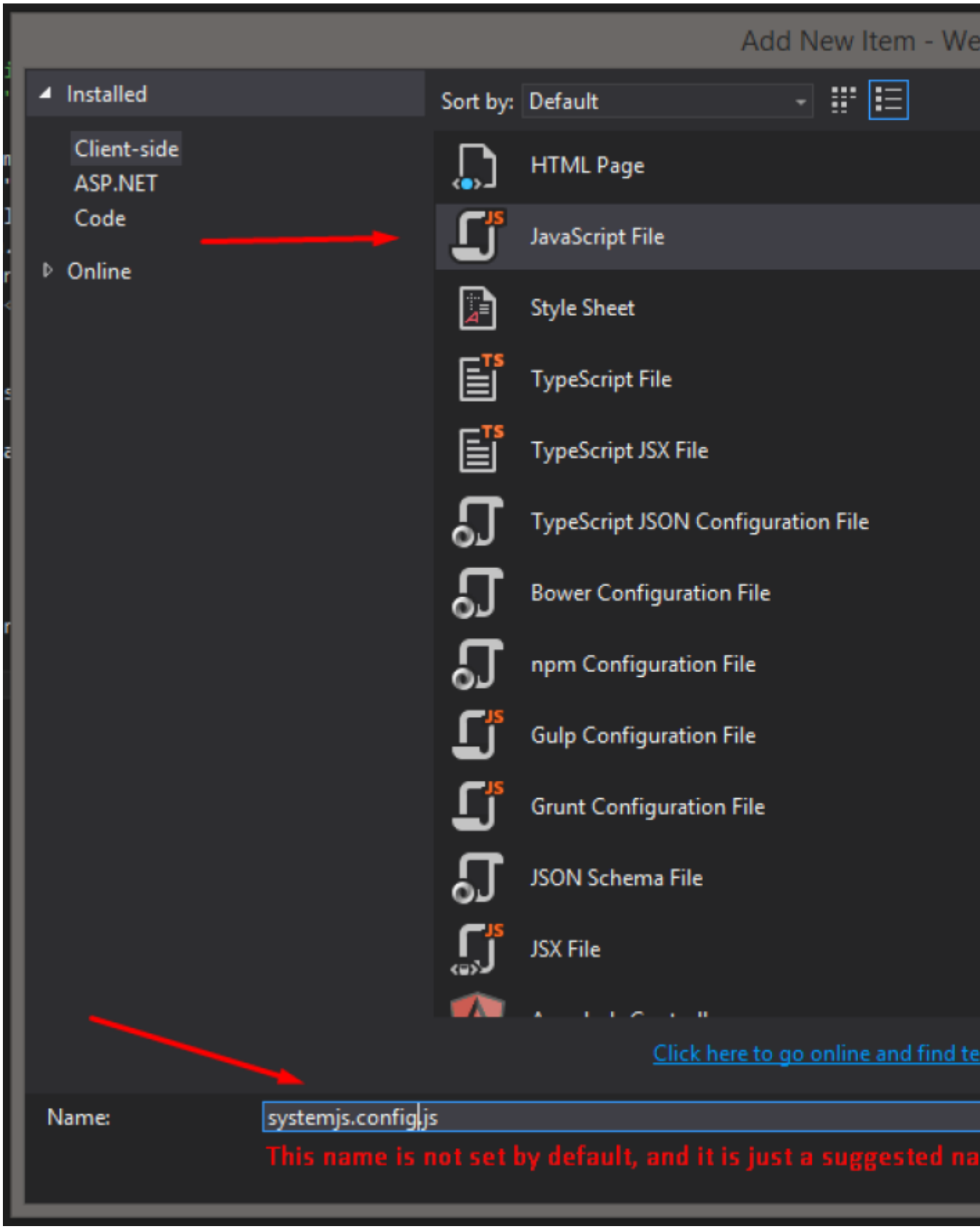
main.ts

```
import { platformBrowserDynamic } from '@angular/platform-br
import { AppModule }              from './app.module';

platformBrowserDynamic().bootstrapModule(AppModule);
```

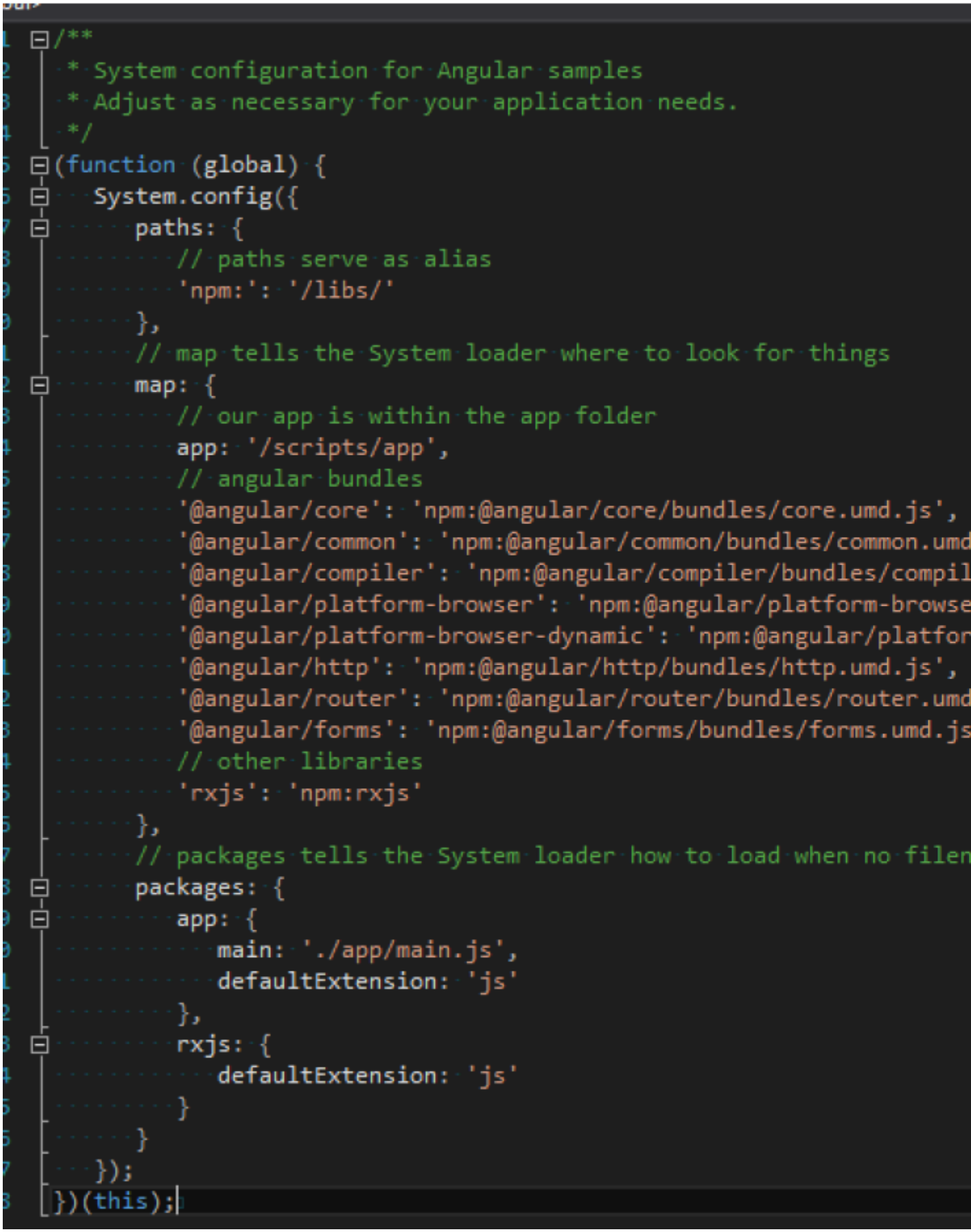
8. wwwroot .



9. ( ) systemjs.config.js .


Add New Item - We
Installed
Client-side
ASP.NET
Code
Online
Sort by: Default
HTML Page
JavaScript File
Style Sheet
TypeScript File
TypeScript JSX File
TypeScript JSON Configuration File
Bower Configuration File
npm Configuration File
Gulp Configuration File
Grunt Configuration File
JSON Schema File
JSX File
Click here to go online and find te
Name:
systemjs.config.js
This name is not set by default, and it is just a suggested na

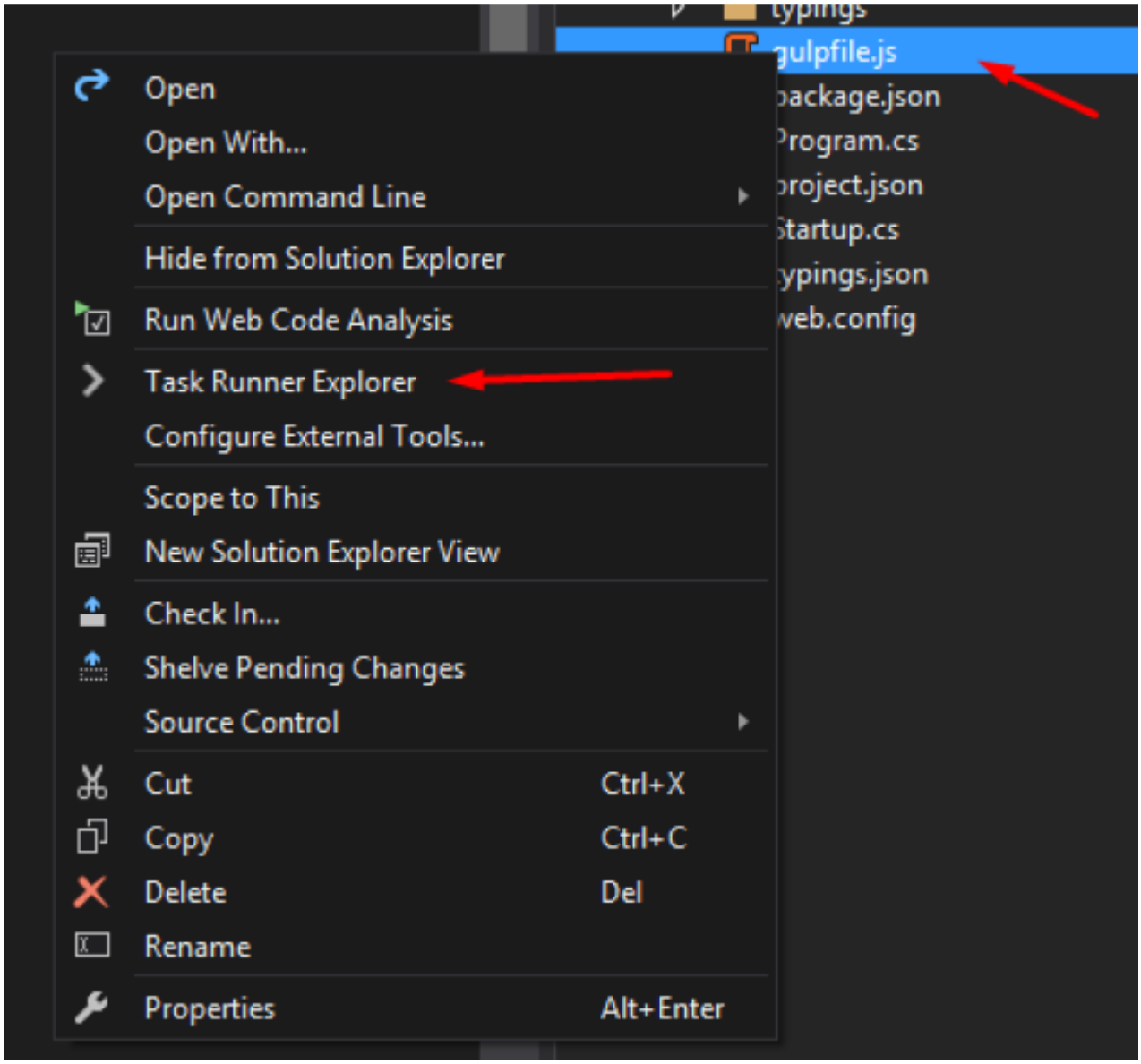
.



```
/**
 * System configuration for Angular samples
 * Adjust as necessary for your application needs.
 */
(function (global) {
    System.config({
        paths: {
            // paths serve as alias
            'npm:': '/libs/'
        },
        // map tells the System loader where to look for things
        map: {
            // our app is within the app folder
            app: '/scripts/app',
            // angular bundles
            '@angular/core': 'npm:@angular/core/bundles/core.umd.js',
            '@angular/common': 'npm:@angular/common/bundles/common.umd
            '@angular/compiler': 'npm:@angular/compiler/bundles/compil
            '@angular/platform-browser': 'npm:@angular/platform-browse
            '@angular/platform-browser-dynamic': 'npm:@angular/platfor
            '@angular/http': 'npm:@angular/http/bundles/http.umd.js',
            '@angular/router': 'npm:@angular/router/bundles/router.umd
            '@angular/forms': 'npm:@angular/forms/bundles/forms.umd.js
            // other libraries
            'rxjs': 'npm:rxjs'
        },
        // packages tells the System loader how to load when no filen
        packages: {
            app: {
                main: './app/main.js',
                defaultExtension: 'js'
            },
            rxjs: {
                defaultExtension: 'js'
            }
        }
    });
})(this);
```

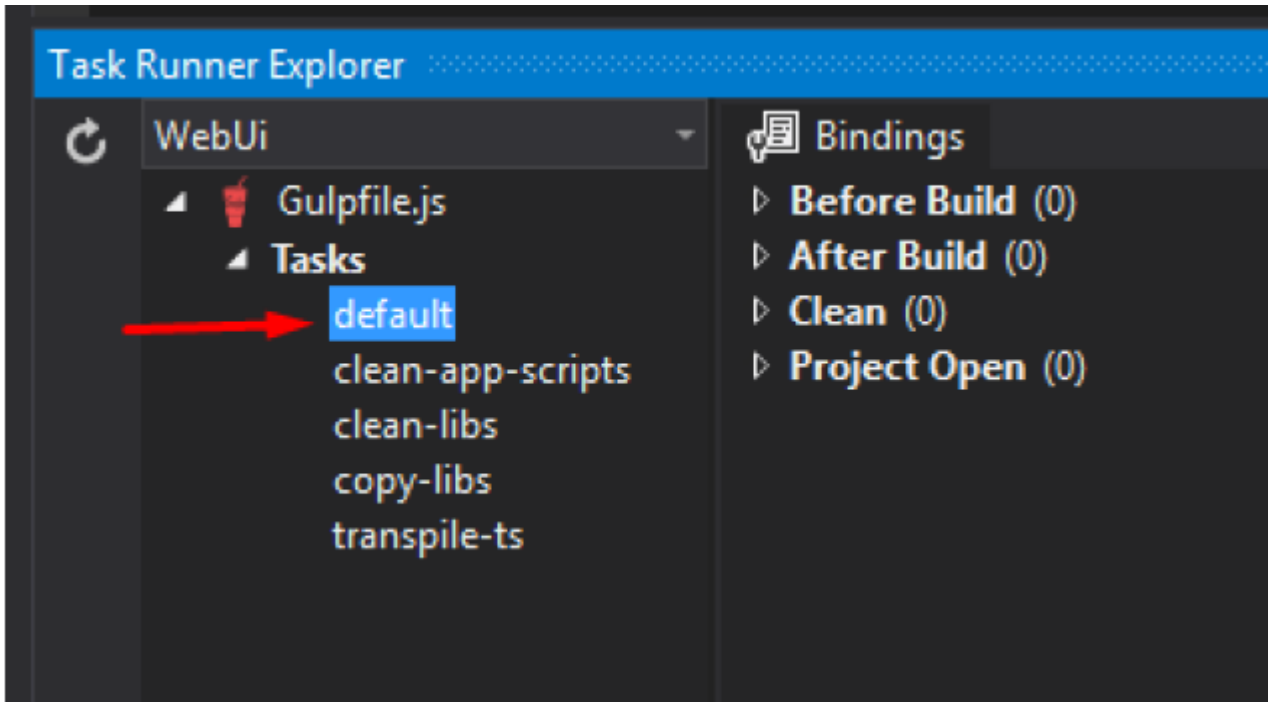
10. Gulp Task wwwroot .

    - gulpfile.js .
    -

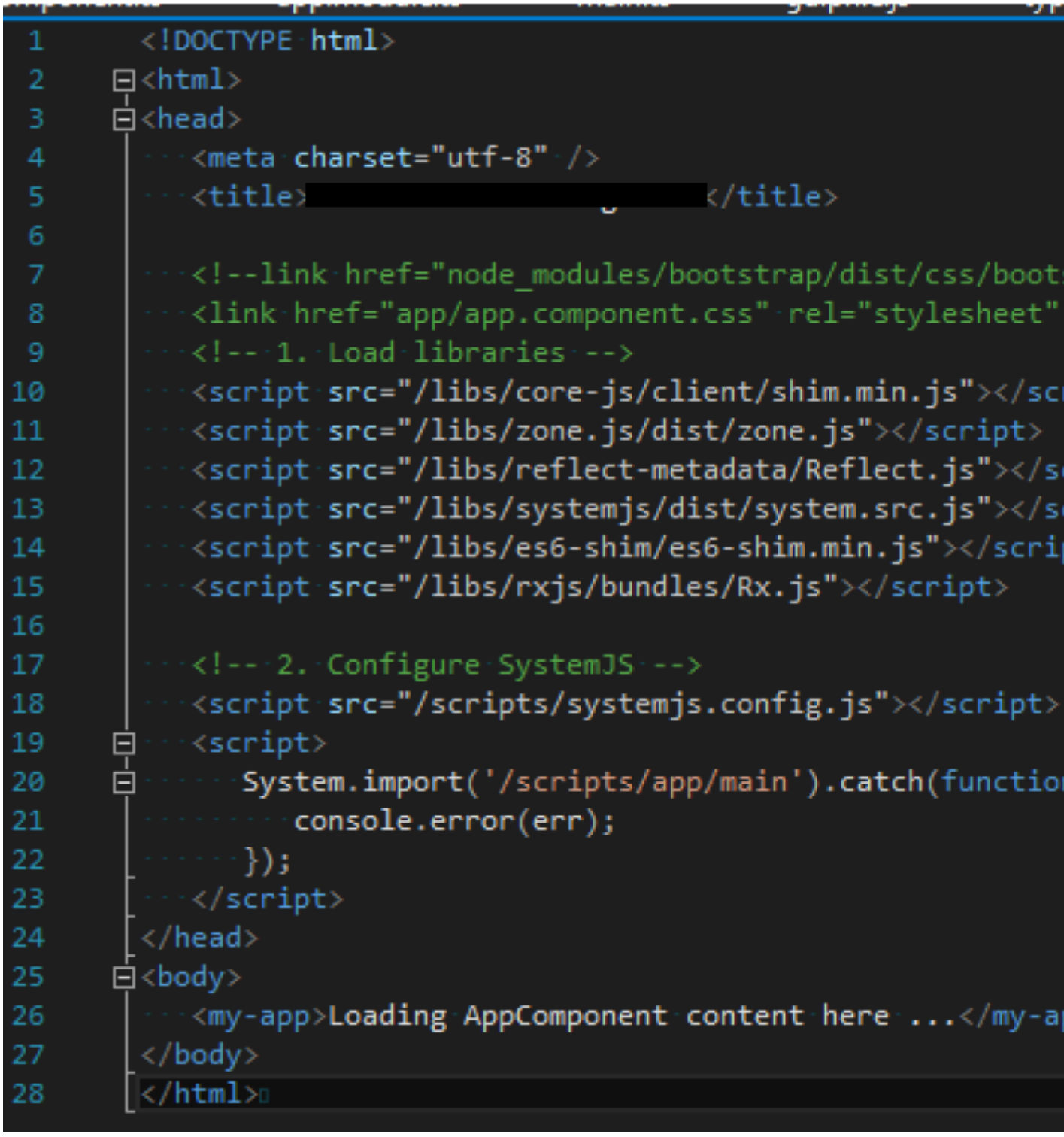

gulpfile.js
Open
Open With...
Open Command Line
Hide from Solution Explorer
Run Web Code Analysis
Task Runner Explorer
Configure External Tools...
Scope to This
New Solution Explorer View
Check In...
Shelve Pending Changes
Source Control
Cut Ctrl+X
Copy Ctrl+C
Delete Del
Rename
Properties Alt+Enter

. (" ") .

- "" "" ( ).


Task Runner Explorer
WebUi
Gulpfile.js
Tasks
default
clean-app-scripts
clean-libs
copy-libs
transpile-ts
Bindings
Before Build (0)
After Build (0)
Clean (0)
Project Open (0)

11. Index.html .



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>[redacted]</title>

    <!--link href="node_modules/bootstrap/dist/css/boot
    <link href="app/app.component.css" rel="stylesheet"
    <!-- 1. Load libraries -->
    <script src="/libs/core-js/client/shim.min.js"></sc
    <script src="/libs/zone.js/dist/zone.js"></script>
    <script src="/libs/reflect-metadata/Reflect.js"></s
    <script src="/libs/systemjs/dist/system.src.js"></s
    <script src="/libs/es6-shim/es6-shim.min.js"></scri
    <script src="/libs/rxjs/bundles/Rx.js"></script>

    <!-- 2. Configure SystemJS -->
    <script src="/scripts/systemjs.config.js"></script>
    <script>
        System.import('/scripts/app/main').catch(functio
            console.error(err);
        });
    </script>
</head>
<body>
    <my-app>Loading AppComponent content here ...</my-a
</body>
</html>
```

12. .

:

- typescript (: "TypeScript Virtual Project") Visual Studio TypeScript "package.json" . : https://www.microsoft.com/en-us/download/details.aspx?id=48593

:

- Deborah Kurata Pluralsight "Angular 2 : Getting Started" :

https://www.pluralsight.com/courses/angular-2getting-started-update

- 2 :

  https://angular.io/

- Mithun Pattankar :

  http://www.mithunvp.com/angular-2-in-asp-net-5-typescript-visual-studio-2015/

  http://www.mithunvp.com/using-angular-2-asp-net-mvc-5-visual-studio/

.NET Core Angular 2 (0.8.3)

.NET Core Angular 2 (0.8.3).

```
Error locating module for declaration
        SilentError: No module files found
```

```
No app module found. Please add your new Class to your component.
        Identical ClientApp/app/app.module.ts
```

[]

1. app.module.client.ts app.client.module.ts .
2. app.client.module.ts . 3 "..." .

   . *[...sharedConfig.declarations, <MyComponent>]*

3. Open boot-client.ts : app.client.module .

   . *import { AppModule } from './app/app.client.module';*

4. . `ng g component my-component`

[]

Angular CLI app.module.ts declarations . (sharedConfig.declarations), .

[]

- https://github.com/angular/angular-cli/issues/2962
- https://www.udemy.com/aspnet-core-angular/learn/v4/t/lecture/6848548 (3.33 Bryan Garzon)

Angular2 .Net Core : https://riptutorial.com/ko/asp-net-core/topic/9352/angular2---net-core

3: ASP.NET -

ASP.Net Core . . . , . , .

:

- http://www.sulhome.com/blog/10/log-asp-net-core-request-and-response-using-middleware
- http://dotnetliberty.com/index.php/2016/01/07/logging-asp-net-5-requests-using-middleware/
- ASP.NET 1.0 HTTP

Examples

```
using Microsoft.AspNetCore.Http;
using System;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNet.Http.Internal;
using Microsoft.AspNetCore.Http.Internal;

  public class LoggerMiddleware
    {
        private readonly RequestDelegate _next;

        public LoggerMiddleware(RequestDelegate next)
        {
            _next = next;
        }

        public async Task Invoke(HttpContext context)
        {
            using (MemoryStream requestBodyStream = new MemoryStream())
            {
                using (MemoryStream responseBodyStream = new MemoryStream())
                {
                    Stream originalRequestBody = context.Request.Body;
                    context.Request.EnableRewind();
                    Stream originalResponseBody = context.Response.Body;

                    try
                    {
                        await context.Request.Body.CopyToAsync(requestBodyStream);
                        requestBodyStream.Seek(0, SeekOrigin.Begin);

                        string requestBodyText = new
StreamReader(requestBodyStream).ReadToEnd();

                        requestBodyStream.Seek(0, SeekOrigin.Begin);
                        context.Request.Body = requestBodyStream;

                        string responseBody = "";


                        context.Response.Body = responseBodyStream;
```

```
                        Stopwatch watch = Stopwatch.StartNew();
                        await _next(context);
                        watch.Stop();

                        responseBodyStream.Seek(0, SeekOrigin.Begin);
                        responseBody = new StreamReader(responseBodyStream).ReadToEnd();
                        AuditLogger.LogToAudit(context.Request.Host.Host,
                            context.Request.Path, context.Request.QueryString.ToString(),
context.Connection.RemoteIpAddress.MapToIPv4().ToString(),
                            string.Join(",", context.Request.Headers.Select(he => he.Key +
":[" + he.Value + "]").ToList()),
                            requestBodyText, responseBody, DateTime.Now,
watch.ElapsedMilliseconds);

                        responseBodyStream.Seek(0, SeekOrigin.Begin);

                        await responseBodyStream.CopyToAsync(originalResponseBody);
                    }
                    catch (Exception ex)
                    {
                        ExceptionLogger.LogToDatabse(ex);
                        byte[] data = System.Text.Encoding.UTF8.GetBytes("Unhandled Error
occured, the error has been logged and the persons concerned are notified!! Please, try again
in a while.");
                        originalResponseBody.Write(data, 0, data.Length);
                    }
                    finally
                    {
                        context.Request.Body = originalRequestBody;
                        context.Response.Body = originalResponseBody;
                    }
                }
            }
        }
    }
```

ASP.NET - : https://riptutorial.com/ko/asp-net-core/topic/9510/asp-net----------

4: ASP.NET 1.0

ASP.NET Core 1.0

Examples

1) , project.json - `"Microsoft.AspNetCore.Session": "1.1.0",`

2) `startup.cs` `AddSession()` `AddDistributedMemoryCache()` `ConfigureServices` .

```
services.AddDistributedMemoryCache(); //This way ASP.NET Core will use a Memory Cache to store
session variables
services.AddSession(options =>
        {
            options.IdleTimeout = TimeSpan.FromDays(1); // It depends on user requirements.
            options.CookieName = ".My.Session"; // Give a cookie name for session which will
be visible in request payloads.
        });
```

3) `UseSession()` .

```
app.UseSession(); //make sure add this line before UseMvc()
```

4) Controller Session .

```
using Microsoft.AspNetCore.Http;

public class HomeController : Controller
{
    public IActionResult Index()
    {
        HttpContext.Session.SetString("SessionVariable1", "Testing123");
        return View();
    }

    public IActionResult About()
    {
        ViewBag.Message = HttpContext.Session.GetString("SessionVariable1");

        return View();
    }
}
```

5. cors .
   AllowCredentials
   WithArigins *AllowAllOrigins* .

ASP.NET 1.0 : https://riptutorial.com/ko/asp-net-core/topic/8067/asp-net--1-0-

5: JavascriptServices

:

`JavaScriptServices` ASP.NET . Angular 2 / React / Knockout / Webpack JavaScript .

Examples

asp.net-core webpack-dev-middleware

`Webpack` . `webpack-dev-middleware` . .

NuGet

Microsoft.AspNetCore.SpaServices

npm

npm install --save-dev aspnet-webpack, webpack-dev-middleware, webpack-dev-server

`Startup Configure`

```
if (env.IsDevelopment())
{
     app.UseWebpackDevMiddleware(new WebpackDevMiddlewareOptions()
     {
         ConfigFile = "webpack.config.js" //this is defualt value
     });
}
```

(HMR)

, . .

`webpack-dev-middleware` :

```
npm install --save-dev webpack-hot-middleware
```

`UseWebpackDevMiddleware` :

```
app.UseWebpackDevMiddleware(new WebpackDevMiddlewareOptions()
{
```

```
    ConfigFile = "webpack.config.js", //this is defualt value
    HotModuleReplacement = true,
    ReactHotModuleReplacement = true, //for React only
});
```

.

HMR Angular 2, React, Knockout Vue .

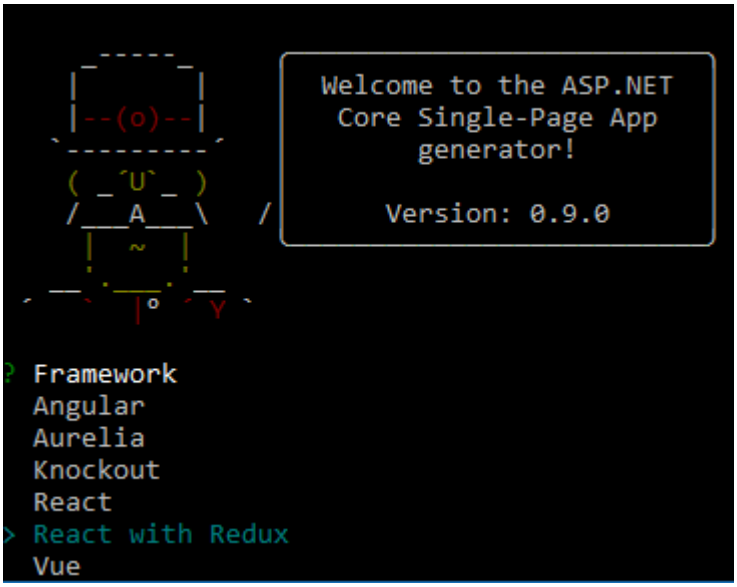
asp.net

`Yeoman aspnetcore-spa` generator `aspnetcore-spa` asp.net .

webpack, dev , .

```
npm install -g yo generator-aspnetcore-spa
cd newproject
yo aspnetcore-spa
```

.



JavascriptServices : https://riptutorial.com/ko/asp-net-core/topic/9621/javascriptservices-

6: MailKit .NET

.Net Core .Net `System.Net.Mail` . MailKit (nuget) .

Examples

```
Install-Package MailKit
```

```
using MailKit.Net.Smtp;
using MimeKit;
using MimeKit.Text;
using System.Threading.Tasks;

namespace Project.Services
{
    /// Using a static class to store sensitive credentials
    /// for simplicity. Ideally these should be stored in
    /// configuration files
    public static class Constants
    {
        public static string SenderName => "<sender_name>";
        public static string SenderEmail => "<sender_email>";
        public static string EmailPassword => "email_password";
        public static string SmtpHost => "<smtp_host>";
        public static int SmtpPort => "smtp_port";
    }
    public class EmailService : IEmailSender
    {
        public Task SendEmailAsync(string recipientEmail, string subject, string message)
        {
            MimeMessage mimeMessage = new MimeMessage();
            mimeMessage.From.Add(new MailboxAddress(Constants.SenderName,
Constants.SenderEmail));
            mimeMessage.To.Add(new MailboxAddress("", recipientEmail));
            mimeMessage.Subject = subject;

            mimeMessage.Body = new TextPart(TextFormat.Html)
            {
                Text = message,
            };

            using (var client = new SmtpClient())
            {

                client.ServerCertificateValidationCallback = (s, c, h, e) => true;

                client.Connect(Constants.SmtpHost, Constants.SmtpPort, false);

                client.AuthenticationMechanisms.Remove("XOAUTH2");

                // Note: only needed if the SMTP server requires authentication
                client.Authenticate(Constants.SenderEmail, Constants.EmailPassword);

                client.Send(mimeMessage);

                client.Disconnect(true);
```

```
                return Task.FromResult(0);
            }
        }
    }

}
```

MailKit .NET : https://riptutorial.com/ko/asp-net-core/topic/8831/mailkit---net-----

7: project.json

Project json asp.net . Microsoft msbuild csproj .

Examples

NETStandard 1.6 .

```
{
  "version": "1.0.0",
  "dependencies": {
    "NETStandard.Library": "1.6.1", //nuget dependency
   },
  "frameworks": { //frameworks the library is build for
    "netstandard1.6": {}
  },
  "buildOptions": {
    "debugType": "portable"
  }
}
```

json :

Microsoft github

```
{
"name": String, //The name of the project, used for the assembly name as well as the name of
the package. The top level folder name is used if this property is not specified.
"version": String, //The Semver version of the project, also used for the NuGet package.
"description": String, //A longer description of the project. Used in the assembly properties.
"copyright": String, //The copyright information for the project. Used in the assembly
properties.
"title": String, //The friendly name of the project, can contain spaces and special characters
not allowed when using the `name` property. Used in the assembly properties.
"entryPoint": String, //The entrypoint method for the project. `Main` by default.
"testRunner": String, //The name of the test runner, such as NUnit or xUnit, to use with this
project. Setting this also marks the project as a test project.
"authors": String[], // An array of strings with the names of the authors of the project.
"language": String, //The (human) language of the project. Corresponds to the "neutral-
language" compiler argument.
"embedInteropTypes": Boolean, //`true` to embed COM interop types in the assembly; otherwise,
`false`.
"preprocess": String or String[], //Specifies which files are included in preprocessing.
"shared": String or String[], //Specifies which files are shared, this is used for library
export.
"dependencies": Object { //project and nuget dependencies
    version: String, //Specifies the version or version range of the dependency. Use the \*
wildcard to specify a floating dependency version.
    type: String, //type of dependency: build
    target: String, //Restricts the dependency to match only a `project` or a `package`.
    include: String,
    exclude: String,
    suppressParent: String
},
```

```
"tools": Object, //An object that defines package dependencies that are used as tools for the
current project, not as references. Packages defined here are available in scripts that run
during the build process, but they are not accessible to the code in the project itself. Tools
can for example include code generators or post-build tools that perform tasks related to
packing.
"scripts": Object, // commandline scripts: precompile, postcompile, prepublish & postpublish
"buildOptions": Object {
    "define": String[], //A list of defines such as "DEBUG" or "TRACE" that can be used in
conditional compilation in the code.
    "nowarn": String[], //A list of warnings to ignore.
    "additionalArguments": String[], //A list of extra arguments that will be passed to the
compiler.
    "warningsAsErrors": Boolean,
    "allowUnsafe": Boolean,
    "emitEntryPoint": Boolean,
    "optimize": Boolean,
    "platform": String,
    "languageVersion": String,
    "keyFile": String,
    "delaySign": Boolean,
    "publicSign": Boolean,
    "debugType": String,
    "xmlDoc": Boolean,
    "preserveCompilationContext": Boolean,
    "outputName": String,
    "compilerName": String,
    "compile": Object {
        "include": String or String[],
        "exclude": String or String[],
        "includeFiles": String or String[],
        "excludeFiles": String or String[],
        "builtIns": Object,
        "mappings": Object
    },
    "embed": Object {
        "include": String or String[],
        "exclude": String or String[],
        "includeFiles": String or String[],
        "excludeFiles": String or String[],
        "builtIns": Object,
        "mappings": Object
    },
    "copyToOutput": Object {
        "include": String or String[],
        "exclude": String or String[],
        "includeFiles": String or String[],
        "excludeFiles": String or String[],
        "builtIns": Object,
        "mappings": Object
    }
},
"publishOptions": Object {
    "include": String or String[],
    "exclude": String or String[],
    "includeFiles": String or String[],
    "excludeFiles": String or String[],
    "builtIns": Object,
    "mappings": Object
},
"runtimeOptions": Object {
    "configProperties": Object {
```

```
        "System.GC.Server": Boolean,
        "System.GC.Concurrent": Boolean,
        "System.GC.RetainVM": Boolean,
        "System.Threading.ThreadPool.MinThreads": Integer,
        "System.Threading.ThreadPool.MaxThreads": Integer
    },
    "framework": Object {
        "name": String,
        "version": String,
    },
    "applyPatches": Boolean
},
"packOptions": Object {
    "summary": String,
    "tags": String[],
    "owners": String[],
    "releaseNotes": String,
    "iconUrl": String,
    "projectUrl": String,
    "licenseUrl": String,
    "requireLicenseAcceptance": Boolean,
    "repository": Object {
        "type": String,
        "url": String
    },
    "files": Object {
        "include": String or String[],
        "exclude": String or String[],
        "includeFiles": String or String[],
        "excludeFiles": String or String[],
        "builtIns": Object,
        "mappings": Object
    }
},
"analyzerOptions": Object {
    "languageId": String
},
"configurations": Object,
"frameworks": Object {
    "dependencies": Object {
        version: String,
        type: String,
        target: String,
        include: String,
        exclude: String,
        suppressParent: String
    },
    "frameworkAssemblies": Object,
    "wrappedProject": String,
    "bin": Object {
        assembly: String
    }
},
"runtimes": Object,
"userSecretsId": String
}
```

. NETCore 1.1

```
{
```

```
  "version": "1.0.0",
  "buildOptions": {
    "emitEntryPoint": true // make sure entry point is emitted.
  },
  "dependencies": {
  },
  "tools": {
  },
  "frameworks": {
    "netcoreapp1.1": { // run as console app
      "dependencies": {
        "Microsoft.NETCore.App": {
          "type": "platform",
          "version": "1.1.0"
        }
      },
      "imports": "dnxcore50"
    }
  },
}
```

project.json : https://riptutorial.com/ko/asp-net-core/topic/9364/project-json

8:

Examples

.

Kestrel .

1. `ASPNETCORE_URLS` .

   Windows

   ```
   SET ASPNETCORE_URLS=https://0.0.0.0:5001
   ```

   OS X

   ```
   export ASPNETCORE_URLS=https://0.0.0.0:5001
   ```

2. `--server.urls`

   ```
   dotnet run --server.urls=http://0.0.0.0:5001
   ```

3. `UseUrls()`

   ```
   var builder = new WebHostBuilder()
                 .UseKestrel()
                 .UseUrls("http://0.0.0.0:5001")
   ```

4. `server.urls` .

hosting.json .

```
Add `hosting.json` with the following content to you project:

    {
       "server.urls": "http://<ip address>:<port>"
    }
```

:

- IP4 IP6 5000 .

  ```
  "server.urls": "http://*:5000"
  ```

  ```
  "server.urls": "http://::5000;http://0.0.0.0:5000"
  ```

- IP4 5000 :

```
"server.urls": "http://0.0.0.0:5000"
```

`http://*:5000;http://::5000` , `http://::5000;http://*:5000` ,
`http://*:5000;http://0.0.0.0:5000` IP6 :: IP4 0.0.0.0
`http://*:5000;http://0.0.0.0:5000` `http://*:5000;http://0.0.0.0:5000`

`project.json` `publishOptions`

```
"publishOptions": {
"include": [
    "hosting.json",
    ...
  ]
}
```

`.UseConfiguration(config)` `.UseConfiguration(config)` .

```
public static void Main(string[] args)
{
    var config = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("hosting.json", optional: true)
        .Build();

    var host = new WebHostBuilder()
        .UseConfiguration(config)
        .UseKestrel()
        .UseContentRoot(Directory.GetCurrentDirectory())
        .UseIISIntegration()
        .UseStartup<Startup>()
        .Build();

    host.Run();
}
```

: https://riptutorial.com/ko/asp-net-core/topic/2262/--

9: (CORS)

AJAX . . .

Cross Origin Resource Sharing (CORS) W3C . CORS . CORS JSONP .

Examples

CORS

`Configure IApplicationBuilder UseCors()` CORS .

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();
    services.AddCors();
}

public void Configure(IApplicationBuilder app)
{
    // Other middleware..

    app.UseCors(builder =>
    {
        builder.AllowAnyOrigin()
               .AllowAnyHeader()
               .AllowAnyMethod();
    });

    // Other middleware..

    app.UseMvc();
}
```

CORS

CORS `ConfigureServices AddCors` .

```
services.AddCors(cors => cors.AddPolicy("AllowAll", policy =>
{
    policy.AllowAnyOrigin()
            .AllowAnyMethod()
            .AllowAnyHeader();
}));
```

.

```
[EnableCors("AllowAll")]
public class HomeController : Controller
{
   // ...
}
```

CORS

.

```
app.UseCors(builder =>
{
    builder.WithOrigins("http://localhost:5000", "http://myproductionapp.com")
           .WithMethods("GET", "POST", "HEAD")
           .WithHeaders("accept", "content-type", "origin")
           .SetPreflightMaxAge(TimeSpan.FromDays(7));
});
```

`GET` , `POST` `HEAD` `http://localhost:5000` `http://myproductionapp.com` `accept` , `content-type` `origin` HTTP . `SetPreflightMaxAge` (`OPTIONS`) .

CORS

MVC CORS ConfigureServices AddCors `CorsAuthorizationFilterFactory` `CorsAuthorizationFilterFactory`

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Cors.Internal;
...
public void ConfigureServices(IServiceCollection services) {
    // Add AllowAll policy just like in single controller example.
    services.AddCors(options => {
        options.AddPolicy("AllowAll",
           builder => {
                builder.AllowAnyOrigin()
                       .AllowAnyMethod()
                       .AllowAnyHeader();
        });
    });

    // Add framework services.
    services.AddMvc();

    services.Configure<MvcOptions>(options => {
        options.Filters.Add(new CorsAuthorizationFilterFactory("AllowAll"));
    });
}

public void Configure(IApplicationBuilder app) {
    app.useMvc();
    // For content not managed within MVC. You may want to set the Cors middleware
    // to use the same policy.
    app.UseCors("AllowAll");
}
```

CORS .

(CORS) : https://riptutorial.com/ko/asp-net-core/topic/2556/----cors-

10:

Asp.net . .

- IConfiguration
- string this[string key] { get; set; }
- IEnumerable<IConfigurationSection> GetChildren();
- IConfigurationSection GetSection(string key);

Examples

IOptions<TOptions> IServiceCollection.Configure<TOptions> .

, IConfigurationRoot .

Startup.cs .

```
 Configuration = builder.Build();
```

Configuration IConfigurationRoot , ConfigureServices , Startup.cs Singleton .

```
 public void ConfigureServices(IServiceCollection services)
 {
    services.AddSingleton<IConfigurationRoot>(provider => Configuration);
```

, /

```
  public MyController(IConfigurationRoot config){
      var setting1= config.GetValue<string>("Setting1")
  }
```

.

, (project.json).

```
"Microsoft.Extensions.Configuration.EnvironmentVariables": "1.0.0",
"Microsoft.Extensions.Configuration.Json": "1.0.0",
```

ConfigurationBuilder API ConfigurationBuilder Startup.cs .

1. ConfigurationBuilder .
2.
3. appsettings.json .
4. appsettings.environementName.json .
5. .

```
public Startup(IHostingEnvironment env)
{
```

```
    var builder = new ConfigurationBuilder()
        .SetBasePath(env.ContentRootPath)
        .AddJsonFile("appsettings.json", optional: true, reloadOnChange: true)
        .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional: true)
        .AddEnvironmentVariables();

    Configuration = builder.Build();
}
```

.

indexer . : .

```
Configuration["AzureLogger:ConnectionString"]
```

AzureLogger ConnectionString .

.AddEnvironmentVariables() ConfigurationBuilder .

APPSETTING_ : .

: :

```
APPSETTING_Security:Authentication:UserName = a_user_name
APPSETTING_Security:Authentication:Password = a_user_password
```

json :

```
{
    "Security" : {
       "Authentication" : {
           "UserName" : "a_user_name",
           "Password" : "a_user_password"
        }
    }
}
```

** Azure Service . . Azure AppSettings .

```
Security:Authentication:UserName         a_user_name
Security:Authentication:Password         a_user_password
```

, .

asp.net section StorageOptions poco . , Microsoft.Extensions.Options.ConfigurationExtensions Configure<TOptions>(IConfiguration config) Configure<TOptions>(IConfiguration config) StorageOptions . Configure<TOptions>(IConfiguration config) .

```
services.Configure<StorageOptions>(Configuration.GetSection("Storage"));
```

Dictionary<string,string> .

```
.AddInMemoryCollection(new Dictionary<string, string>
{
    ["akey"] = "a value"
})
```

/ .

: https://riptutorial.com/ko/asp-net-core/topic/8660/

11:

Examples

. :

- ViewComponent .
-

, .

.

```
public class MyCustomViewComponent : ViewComponent
{
    public async Task<IViewComponentResult> InvokeAsync(string param1, int param2)
    {
        //some business logic

        //renders ~/Views/Shared/Components/MyCustom/Default.cshtml
        return View(new MyCustomModel{ ... });
    }
}

@*View file located in ~/Views/Shared/Components/MyCustom/Default.cshtml*@
@model WebApplication1.Models.MyCustomModel
<p>Hello @Model.UserName!</p>
```

(`ViewComponentResult`) .

```
@await Component.InvokeAsync("MyCustom", new {param1 = "foo", param2 = 42})
```

_LoginPartial.cshtml . .

LoginPartial . (2) .

```
public class LoginViewComponent : ViewComponent
{
    private readonly SignInManager<ApplicationUser> signInManager;
    private readonly UserManager<ApplicationUser> userManager;

    public LoginViewComponent(SignInManager<ApplicationUser> signInManager,
UserManager<ApplicationUser> userManager)
    {
        this.signInManager = signInManager;
        this.userManager = userManager;
    }

    public async Task<IViewComponentResult> InvokeAsync()
    {
        if (signInManager.IsSignedIn(this.User as ClaimsPrincipal))
        {
            return View("SignedIn", await userManager.GetUserAsync(this.User as
```

```
ClaimsPrincipal));
        }
        return View("SignedOut");
    }
}
```

SignedIn (~ / Views / Shared / Components / Login / SignedIn.cshtml)

```
@model WebApplication1.Models.ApplicationUser

<form asp-area="" asp-controller="Account" asp-action="LogOff" method="post" id="logoutForm"
class="navbar-right">
    <ul class="nav navbar-nav navbar-right">
        <li>
            <a asp-area="" asp-controller="Manage" asp-action="Index" title="Manage">Hello
@Model.UserName!</a>
        </li>
        <li>
            <button type="submit" class="btn btn-link navbar-btn navbar-link">Log off</button>
        </li>
    </ul>
</form>
```

SignedOut (~ / Views / Shared / Components / Login / SignedOut.cshtml)

```
<ul class="nav navbar-nav navbar-right">
    <li><a asp-area="" asp-controller="Account" asp-action="Register">Register</a></li>
    <li><a asp-area="" asp-controller="Account" asp-action="Login">Log in</a></li>
</ul>
```

_Layout.cshtml

```
@await Component.InvokeAsync("Login")
```

`Controller` `ViewComponent()` .

```
public IActionResult GetMyComponent()
{
    return ViewComponent("Login", new { param1 = "foo", param2 = 42 });
}
```

POCO `ViewComponentResult` . .

```
public IActionResult GetMyComponent()
{
    return new ViewComponentResult
    {
        ViewComponentName = "Login",
        Arguments = new { param1 = "foo", param2 = 42 }
    };
}
```

: https://riptutorial.com/ko/asp-net-core/topic/3248/-

12:

Examples

asp.net `AuthorizeAttribute` `AuthorizeAttribute`

```
[Authorize]
public class SomeController : Controller
{
    public IActionResult Get()
    {
    }

    public IActionResult Post()
    {
    }
}
```

.

.

```
public class SomeController : Controller
{
    public IActionResult Get()
    {
    }

    [Authorize]
    public IActionResult Post()
    {
    }
}
```

`AllowAnonymousAttribute`

```
[Authorize]
public class SomeController: Controller
{
    public IActionResult Get()
    {
    }

    [AllowAnonymous]
    public IActionResult Post()
    {
    }
}
```

`Post` . `AllowAnonymous` `AllowAnonymous` `AuthorizeAttribute` .

.

```
services.AddMvc(config =>
{
    var policy = new AuthorizationPolicyBuilder()
        .RequireAuthenticatedUser()
        .Build();
    config.Filters.Add(new AuthorizeFilter(policy));
})
```

. / `Authorize` / `AllowAnonymous` .

: https://riptutorial.com/ko/asp-net-core/topic/6914/-

13:

Examples

```
app.UseMvc(routes =>
{
    routes.MapRoute(
    name: "default",
    template: "{controller=Home}/{action=Index}/{id?}");
});
```

/Home/Index , /Home/Index/123 /

.

en-US, de-DE, zh-CHT, zh-Hant / .

```
public class LocaleConstraint : IRouteConstraint
{
    private static readonly Regex LocalePattern = new Regex(@"^[a-z]{2}(-[a-z]{2,4})?$",
                                    RegexOptions.Compiled | RegexOptions.IgnoreCase);

    public bool Match(HttpContext httpContext, IRouter route, string routeKey,
                        RouteValueDictionary values, RouteDirection routeDirection)
    {
        if (!values.ContainsKey(routeKey))
            return false;

        string locale = values[routeKey] as string;
        if (string.IsNullOrWhiteSpace(locale))
            return false;

        return LocalePattern.IsMatch(locale);
    }
}
```

.

```
services.Configure<RouteOptions>(options =>
{
    options.ConstraintMap.Add("locale", typeof(LocaleConstraint));
});
```

.

```
[Route("api/{culture:locale}/[controller]")]
public class ProductController : Controller { }
```

```
[HttpGet("api/{culture:locale}/[controller]/{productId}"]
public Task<IActionResult> GetProductAsync(string productId) { }
```

```
app.UseMvc(routes =>
{
    routes.MapRoute(
        name: "default",
        template: "api/{culture:locale}/{controller}/{id?}");
    routes.MapRoute(
        name: "default",
        template: "api/{controller}/{id?}");
});
```

: https://riptutorial.com/ko/asp-net-core/topic/2863/

14:

Examples

.

```
public class MyModel
{
    public int id { get; set; }

    //sets the FirstName to be required, and no longer than 100 characters
    [Required]
    [StringLength(100)]
    public string FirstName { get; set; }
}
```

.

- `[CreditCard]` : .
- `[Compare]` : .
- `[EmailAddress]` : .
- `[Phone]` : .
- `[Range]` : .
- `[RegularExpression]` : .
- `[Required]` : .
- `[StringLength]` : .
- `[Url]` : URL .

ValidationAttribute . .

```
public class OddNumberAttribute : ValidationAttribute
{
    protected override ValidationResult IsValid(object value, ValidationContext
validationContext)
    {
        try
        {
            var number = (int) value;
            if (number % 2 == 1)
                return ValidationResult.Success;
            else
                return new ValidationResult("Only odd numbers are valid.");
        }
        catch (Exception)
        {
            return new ValidationResult("Not a number.");
        }
    }
}
```

```
public class MyModel
```

```
{
    [OddNumber]
    public int Number { get; set; }
}
```

: https://riptutorial.com/ko/asp-net-core/topic/4625/

15:

. .

Examples

ExceptionHandler JSON

.

```
public class ErrorDto
{
   public int Code { get; set; }
   public string Message { get; set; }

   // other fields

   public override string ToString()
   {
      return JsonConvert.SerializeObject(this);
   }
}
```

ExceptionHandler . .

```
app.UseExceptionHandler(errorApp =>
    {
        errorApp.Run(async context =>
        {
            context.Response.StatusCode = 500; // or another Status
            context.Response.ContentType = "application/json";

            var error = context.Features.Get<IExceptionHandlerFeature>();
            if (error != null)
            {
                var ex = error.Error;

                await context.Response.WriteAsync(new ErrorDto()
                {
                    Code = <your custom code based on Exception Type>,
                    Message = ex.Message // or your custom message

                    ... // other custom data
                }.ToString(), Encoding.UTF8);
            }
        });
    });
```

ContentType

HttpContext.Response.OnStarting . Invoke .

```
public async Task Invoke(HttpContext context)
{
    context.Response.OnStarting((state) =>
    {
        if (context.Response.StatusCode == (int)HttpStatusCode.OK)
        {
           if (context.Request.Path.Value.EndsWith(".map"))
           {
             context.Response.ContentType = "application/json";
           }
        }
        return Task.FromResult(0);
    }, null);

    await nextMiddleware.Invoke(context);
}
```

:

HttpContext.Items . . .

`HttpContext.Items` `IDictionary<object, object>` .

- `HttpRequest` .
- .

.

, Items .

```
app.Use(async (context, next) =>
{
    // perform some verification
    context.Items["isVerified"] = true;
    await next.Invoke();
});
```

.

```
app.Run(async (context) =>
{
    await context.Response.WriteAsync("Verified request? " + context.Items["isVerified"]);
});
```

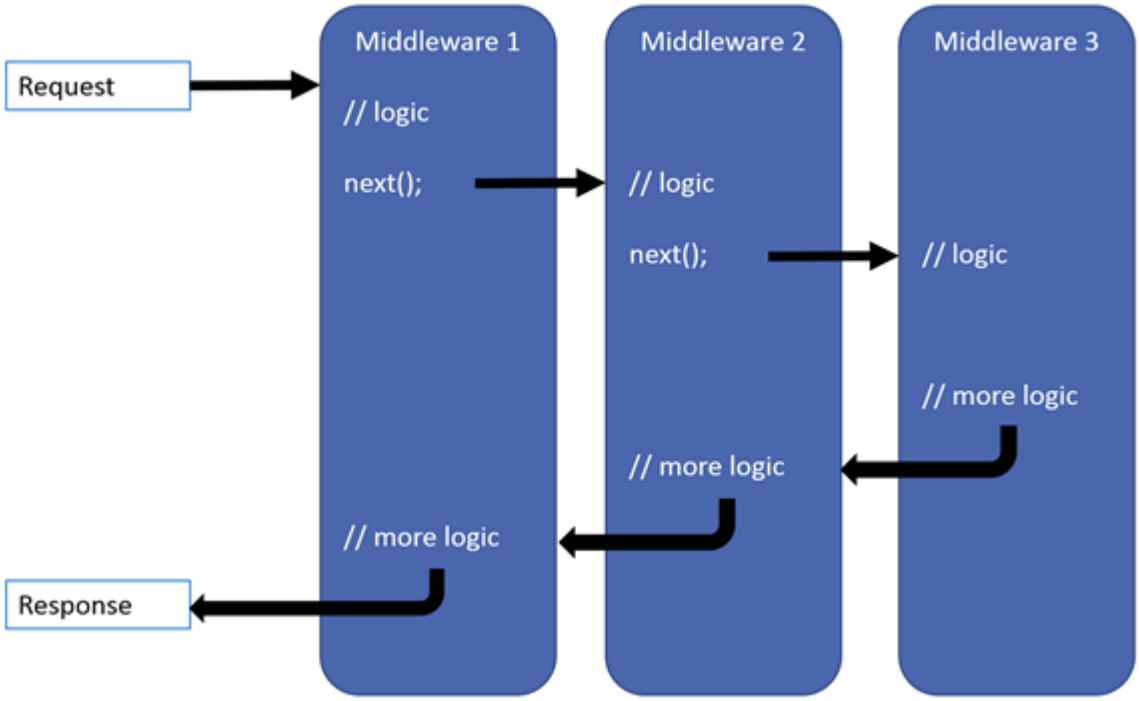
”

. . .

```
app.Run(async context =>
{
    await context.Response.WriteAsync("Hello from " + _environment);
});
```

.

```
app.Use(async (context, next) =>
{
    //action before next delegate
    await next.Invoke(); //call next middleware
    //action after called middleware
});
```



Ilustration :

MapWhen

. .

```
private static void HandleBranch(IApplicationBuilder app)
{
    app.Run(async context =>
    {
        await context.Response.WriteAsync("Condition is fulfilled");
    });
}

public void ConfigureMapWhen(IApplicationBuilder app)
{
    app.MapWhen(context => {
        return context.Request.Query.ContainsKey("somekey");
    }, HandleBranch);
}
```

MapWhen . .

```
private static void HandleMapTest(IApplicationBuilder app)
{
    app.Run(async context =>
    {
        await context.Response.WriteAsync("Map Test Successful");
```

```
    });
}

public void ConfigureMapping(IApplicationBuilder app)
{
    app.Map("/maptest", HandleMapTest);

}
```

ASP.net

: https://riptutorial.com/ko/asp-net-core/topic/1479/

16:

Examples

unt

ASP.NET Gulp Grunt . . . .

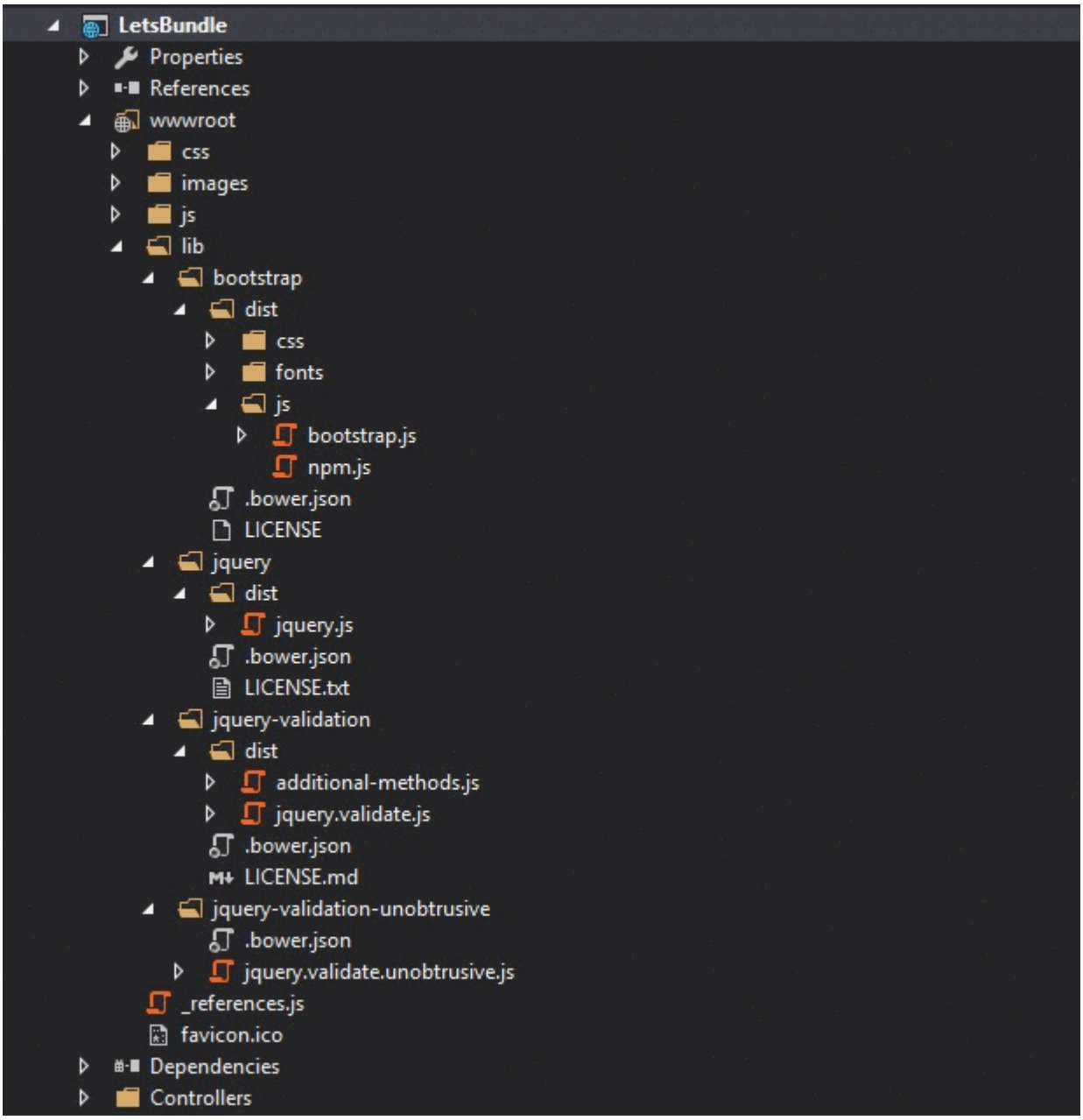
`gulpfile.js` Gulp ASP.NET .

```
// Defining dependencies
var gulp = require("gulp"),
    rimraf = require("rimraf"),
    concat = require("gulp-concat"),
    cssmin = require("gulp-cssmin"),
    uglify = require("gulp-uglify");

// Define web root
var webroot = "./wwwroot/"

// Defining paths
var paths = {
    js: webroot + "js/**/*.js",
    minJs: webroot + "js/**/*.min.js",
    css: webroot + "css/**/*.css",
    minCss: webroot + "css/**/*.min.css",
    concatJsDest: webroot + "js/site.min.js",
    concatCssDest: webroot + "css/site.min.css"
};

// Bundling (via concat()) and minifying (via uglify()) Javascript
gulp.task("min:js", function () {
    return gulp.src([paths.js, "!" + paths.minJs], { base: "." })
        .pipe(concat(paths.concatJsDest))
        .pipe(uglify())
        .pipe(gulp.dest("."));
});

// Bundling (via concat()) and minifying (via cssmin()) Javascript
gulp.task("min:css", function () {
    return gulp.src([paths.css, "!" + paths.minCss])
        .pipe(concat(paths.concatCssDest))
        .pipe(cssmin())
        .pipe(gulp.dest("."));
});
```

Javascript CSS globbing .

Visual Studio Bundler Minifier Extension . .

.



. `bundleconfig.json` `bundleconfig.json` :

```
[
  {
    "outputFileName": "wwwroot/app/bundle.js",
    "inputFiles": [
      "wwwroot/lib/jquery/dist/jquery.js",
      "wwwroot/lib/bootstrap/dist/js/bootstrap.js",
      "wwwroot/lib/jquery-validation/dist/jquery.validate.js",
      "wwwroot/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.js"
    ]
  }
]
```

: .

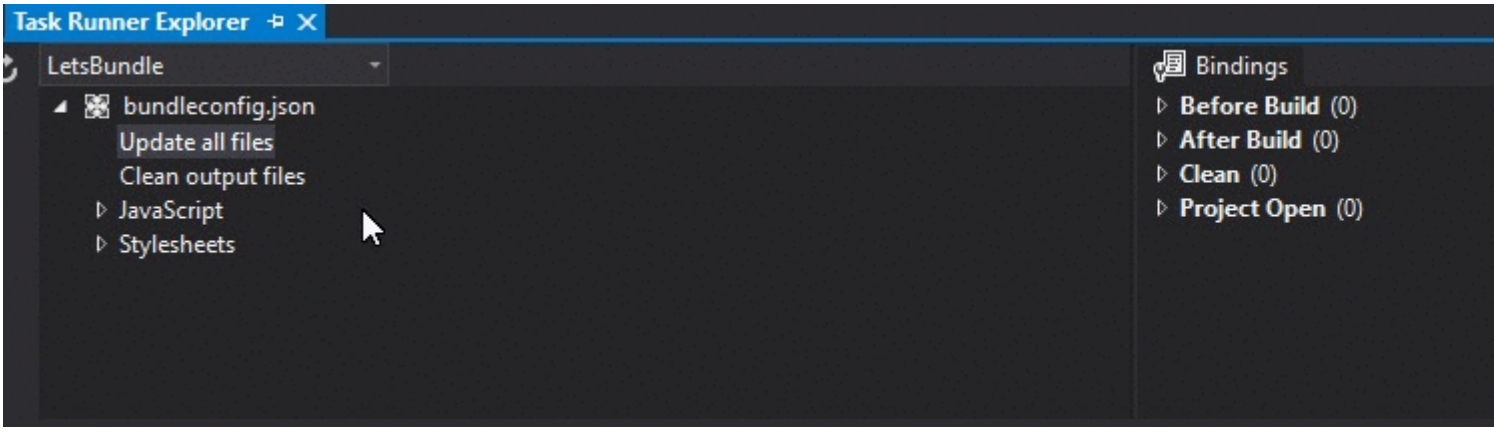
bundleconfig.json . **`minify`** .

```
[
  {
    "outputFileName": "wwwroot/app/bundle.js",
    "inputFiles": [
      "wwwroot/lib/jquery/dist/jquery.js",
      "wwwroot/lib/bootstrap/dist/js/bootstrap.js",
      "wwwroot/lib/jquery-validation/dist/jquery.validate.js",
      "wwwroot/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.js"
    ],
    "minify": {
      "enabled": true
    }
  }
]
```

, .

.

- > .
- bundleconfig.json bundleconfig.json .
- .



.

dotnet bundle

ASP.NET Core RTM ASP.NET BundlerMinifier.Core BundlerMinifier.Core BundlerMinifier.Core BundlerMinifier.Core .

BundlerMinifier.Core

project.json tools BundlerMinifier.Core .

```
"tools": {
  "BundlerMinifier.Core": "2.0.238",
  "Microsoft.AspNetCore.Razor.Tools": "1.0.0-preview2-final",
  "Microsoft.AspNetCore.Server.IISIntegration.Tools": "1.0.0-preview2-final"
}
```

bundleconfig.json **bundleconfig.json** . .

```
[
  {
    "outputFileName": "wwwroot/css/site.min.css",
    "inputFiles": [
      "wwwroot/css/site.css"
    ]
  },
  {
    "outputFileName": "wwwroot/js/site.min.js",
    "inputFiles": [
      "wwwroot/js/site.js"
    ],
    "minify": {
      "enabled": true,
      "renameLocals": true
    },
    "sourceMap": false
  },
  {
    "outputFileName": "wwwroot/js/semantic.validation.min.js",
    "inputFiles": [
      "wwwroot/js/semantic.validation.js"
    ],
    "minify": {
      "enabled": true,
      "renameLocals": true
    }
  }
]
```

/

.

```
dotnet bundle
```

Bundling Minification `project.json` `dotnet bundle` .

```
"scripts": {
  "precompile": [
    "dotnet bundle"
  ]
}
```

.

- **dotnet** - `bundleconfig.json` bundle `bundleconfig.json` .
- **dotnet bundle clean** - .
- **dotnet bundle watch** - `bundleconfig.json` `dotnet bundle` `bundleconfig.json` `dotnet bundle` watcher .
- **dotnet bundle help** - .

: https://riptutorial.com/ko/asp-net-core/topic/4051/--

17:

Examples

NLog Logger

NLog.Extensions.Logging Microsoft .NET ASP.NET NLog . .

ILoggerFactory ILogger ILogger . T .

```
public class TodoController : Controller
{
    private readonly ILogger _logger;

    public TodoController(ILogger<TodoController> logger)
    {
     _logger = logger;
    }
}
```

ASP.NET 1.0 Serilog

1) project.json .

```
"Serilog": "2.2.0",
"Serilog.Extensions.Logging": "1.2.0",
"Serilog.Sinks.RollingFile": "2.0.0",
"Serilog.Sinks.File": "3.0.0"
```

2) Startup.cs -

```
Log.Logger = new LoggerConfiguration()
    .MinimumLevel.Debug()
    .WriteTo.RollingFile(Path.Combine(env.ContentRootPath, "Serilog-{Date}.txt"))
    .CreateLogger();
```

3) Startup Configure -

```
loggerFactory.AddSerilog();
```

4) ILogger .

```
public class HomeController : Controller
{
    ILogger<HomeController> _logger = null;
    public HomeController(ILogger<HomeController> logger)
    {
        _logger = logger;
    }
```

5) -

```
try
{
    throw new Exception("Serilog Testing");
}
catch (System.Exception ex)
{
    this._logger.LogError(ex.Message);
}
```

: https://riptutorial.com/ko/asp-net-core/topic/1946/-

18:

- `@inject<NameOfService><Identifier>`
- `@<Identifier>.Foo()`
- @inject <> <>

Examples

@inject

ASP.NET `@inject` .

```
@inject <type> <name>
```

View View .

```
@inject YourWidgetServiceClass WidgetService

<!-- This would call the service, which is already populated and output the results -->
There are <b>@WidgetService.GetWidgetCount()</b> Widgets here.
```

`Startup.cs ConfigureServices()` .

```
public void ConfigureServices(IServiceCollection services)
{
     // Other stuff omitted for brevity

     services.AddTransient<IWidgetService, WidgetService>();
}
```

: https://riptutorial.com/ko/asp-net-core/topic/4284/--

19:

AspNetCoreRateLimit IP ID API MVC ASP.NET .

Examples

IP

IpRateLimit IP IP , 15 . API URL HTTP .

NuGet :

```
Install-Package AspNetCoreRateLimit
```

Startup.cs :

```
public void ConfigureServices(IServiceCollection services)
{
    // needed to load configuration from appsettings.json
    services.AddOptions();

    // needed to store rate limit counters and ip rules
    services.AddMemoryCache();

    //load general configuration from appsettings.json
    services.Configure<IpRateLimitOptions>(Configuration.GetSection("IpRateLimiting"));

    //load ip rules from appsettings.json
    services.Configure<IpRateLimitPolicies>(Configuration.GetSection("IpRateLimitPolicies"));

    // inject counter and rules stores
    services.AddSingleton<IIpPolicyStore, MemoryCacheIpPolicyStore>();
    services.AddSingleton<IRateLimitCounterStore, MemoryCacheRateLimitCounterStore>();

    // Add framework services.
    services.AddMvc();
}

public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory
loggerFactory)
{
    loggerFactory.AddConsole(Configuration.GetSection("Logging"));
    loggerFactory.AddDebug();

    app.UseIpRateLimiting();

    app.UseMvc();
}
```

loggerFactory .

`IDistributedCache` SQLServer `IDistributedCache` kestrel . .

```
    // inject counter and rules distributed cache stores
    services.AddSingleton<IIpPolicyStore, DistributedCacheIpPolicyStore>();
    services.AddSingleton<IRateLimitCounterStore,DistributedCacheRateLimitCounterStore>();
```

appsettings.json :

```
  "IpRateLimiting": {
    "EnableEndpointRateLimiting": false,
    "StackBlockedRequests": false,
    "RealIpHeader": "X-Real-IP",
    "ClientIdHeader": "X-ClientId",
    "HttpStatusCode": 429,
    "IpWhitelist": [ "127.0.0.1", "::1/10", "192.168.0.0/24" ],
    "EndpointWhitelist": [ "get:/api/license", "*:/api/status" ],
    "ClientWhitelist": [ "dev-id-1", "dev-id-2" ],
    "GeneralRules": [
      {
        "Endpoint": "*",
        "Period": "1s",
        "Limit": 2
      },
      {
        "Endpoint": "*",
        "Period": "15m",
        "Limit": 100
      },
      {
        "Endpoint": "*",
        "Period": "12h",
        "Limit": 1000
      },
      {
        "Endpoint": "*",
        "Period": "7d",
        "Limit": 10000
      }
    ]
  }
```

`EnableEndpointRateLimiting false` `*` . 5 HTTP .

`EnableEndpointRateLimiting true` `{HTTP_Verb}{PATH}` . , `*:/api/values` 5 `GET /api/values` 5 `PUT /api/values` 5 .

`StackBlockedRequests false` . 3 . `StackBlockedRequests true` `true` .

`RealIpHeader` Kestrel IP . `X-Real-IP` .

`ClientIdHeader` ID . ID ClientWhitelist .

IP **appsettings.json** :

```
  "IpRateLimitPolicies": {
     "IpRules": [
       {
         "Ip": "84.247.85.224",
         "Rules": [
```

```
          {
            "Endpoint": "*",
            "Period": "1s",
            "Limit": 10
          },
          {
            "Endpoint": "*",
            "Period": "15m",
            "Limit": 200
          }
        ]
      },
      {
        "Ip": "192.168.3.22/25",
        "Rules": [
          {
            "Endpoint": "*",
            "Period": "1s",
            "Limit": 5
          },
          {
            "Endpoint": "*",
            "Period": "15m",
            "Limit": 150
          },
          {
            "Endpoint": "*",
            "Period": "12h",
            "Limit": 500
          }
        ]
      }
    ]
  }
```

IP IP v4 v6 "192.168.0.0/24", "fe80 :: / 10" "192.168.0.0-192.168.0.255" .

, .

`{HTTP_Verb}:{PATH}` , HTTP .

`{INT}{PERIOD_TYPE}` . : `s, m, h, d` .

`{LONG}` .

:

2 .

```
{
 "Endpoint": "*",
 "Period": "1s",
 "Limit": 2
}
```

IP api / value 3 GET . PUT api / values . `{HTTP_Verb}{PATH}` .

HTTP `/api/values` 15 5 .

```
{
 "Endpoint": "*:/api/values",
 "Period": "15m",
 "Limit": 5
}
```

`/api/values` GET 5 .

```
{
 "Endpoint": "get:/api/values",
 "Period": "1h",
 "Limit": 5
}
```

IP API / 6 GET . GET API / / 1 .

HTTP IpRateLimitMiddleware .

- IP, ID, HTTP URL . `IpRateLimitMiddleware.SetIdentity` .
- IP, ID URL .
- IP .
- IP
- ,

.

```
Status Code: 429
Retry-After: 58
Content: API calls quota exceeded! maximum admitted 2 per 1m.
```

. `HttpStatusCode` `QuotaExceededMessage` . `IpRateLimitMiddleware.ReturnQuotaExceededResponse` . `Retry-After` .

X-Rate-Limit .

```
X-Rate-Limit-Limit: the rate limit period (eg. 1m, 12h, 1d)
X-Rate-Limit-Remaining: number of request remaining
X-Rate-Limit-Reset: UTC date time when the limits resets
```

`Microsoft.Extensions.Logging.ILogger` . `IpRateLimitMiddleware.LogBlockedRequest` . .

```
info: AspNetCoreRateLimit.IpRateLimitMiddleware[0]
      Request get:/api/values from IP 84.247.85.224 has been blocked, quota 2/1m exceeded by
3. Blocked by rule *:/api/value, TraceIdentifier 0HKTLISQQVV9D.
```

`appsettings.json` IP `MemoryCacheClientPolicyStore` `DistributedCacheIpPolicyStore` . IP IP .

```
public class IpRateLimitController : Controller
{
    private readonly IpRateLimitOptions _options;
    private readonly IIpPolicyStore _ipPolicyStore;

    public IpRateLimitController(IOptions<IpRateLimitOptions> optionsAccessor, IIpPolicyStore
ipPolicyStore)
    {
        _options = optionsAccessor.Value;
        _ipPolicyStore = ipPolicyStore;
    }

    [HttpGet]
    public IpRateLimitPolicies Get()
    {
        return _ipPolicyStore.Get(_options.IpPolicyPrefix);
    }

    [HttpPost]
    public void Post()
    {
        var pol = _ipPolicyStore.Get(_options.IpPolicyPrefix);

        pol.IpRules.Add(new IpRateLimitPolicy
        {
            Ip = "8.8.4.4",
            Rules = new List<RateLimitRule>(new RateLimitRule[] {
                new RateLimitRule {
                    Endpoint = "*:/api/testupdate",
                    Limit = 100,
                    Period = "1d" }
            })
        });

        _ipPolicyStore.Set(_options.IpPolicyPrefix, pol);
    }
}
```

IP .

ID

ClientRateLimit , 15 . API URL HTTP .

NuGet :

```
Install-Package AspNetCoreRateLimit
```

Startup.cs :

```
public void ConfigureServices(IServiceCollection services)
{
    // needed to load configuration from appsettings.json
    services.AddOptions();

    // needed to store rate limit counters and ip rules
    services.AddMemoryCache();
```

```
    //load general configuration from appsettings.json

services.Configure<ClientRateLimitOptions>(Configuration.GetSection("ClientRateLimiting"));

    //load client rules from appsettings.json

services.Configure<ClientRateLimitPolicies>(Configuration.GetSection("ClientRateLimitPolicies"));


    // inject counter and rules stores
    services.AddSingleton<IClientPolicyStore, MemoryCacheClientPolicyStore>();
    services.AddSingleton<IRateLimitCounterStore, MemoryCacheRateLimitCounterStore>();

    // Add framework services.
    services.AddMvc();
}

public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory
loggerFactory)
{
    loggerFactory.AddConsole(Configuration.GetSection("Logging"));
    loggerFactory.AddDebug();

    app.UseClientRateLimiting();

    app.UseMvc();
}
```

loggerFactory .

`IDistributedCache` SQLServer `IDistributedCache` kestrel . .

```
    // inject counter and rules distributed cache stores
    services.AddSingleton<IClientPolicyStore, DistributedCacheClientPolicyStore>();
    services.AddSingleton<IRateLimitCounterStore,DistributedCacheRateLimitCounterStore>();
```

appsettings.json :

```
  "ClientRateLimiting": {
    "EnableEndpointRateLimiting": false,
    "StackBlockedRequests": false,
    "ClientIdHeader": "X-ClientId",
    "HttpStatusCode": 429,
    "EndpointWhitelist": [ "get:/api/license", "*:/api/status" ],
    "ClientWhitelist": [ "dev-id-1", "dev-id-2" ],
    "GeneralRules": [
      {
        "Endpoint": "*",
        "Period": "1s",
        "Limit": 2
      },
      {
        "Endpoint": "*",
        "Period": "15m",
        "Limit": 100
      },
      {
        "Endpoint": "*",
```

```
        "Period": "12h",
        "Limit": 1000
      },
      {
        "Endpoint": "*",
        "Period": "7d",
        "Limit": 10000
      }
    ]
  }
```

`EnableEndpointRateLimiting` `false` `*` . 5 HTTP .

`EnableEndpointRateLimiting` `true` `{HTTP_Verb}{PATH}` . , `*:/api/values` 5 `GET /api/values` 5 `PUT /api/values` 5 .

`StackBlockedRequests` `false` . 3 . `StackBlockedRequests` `true` `true` .

`ClientIdHeader` ID . ID ClientWhitelist .

appsettings.json :

```
"ClientRateLimitPolicies": {
   "ClientRules": [
     {
       "ClientId": "client-id-1",
       "Rules": [
         {
           "Endpoint": "*",
           "Period": "1s",
           "Limit": 10
         },
         {
           "Endpoint": "*",
           "Period": "15m",
           "Limit": 200
         }
       ]
     },
     {
       "Client": "client-id-2",
       "Rules": [
         {
           "Endpoint": "*",
           "Period": "1s",
           "Limit": 5
         },
         {
           "Endpoint": "*",
           "Period": "15m",
           "Limit": 150
         },
         {
           "Endpoint": "*",
           "Period": "12h",
           "Limit": 500
         }
       ]
```

```
      }
    ]
  }
```

, .

`{HTTP_Verb}:{PATH}` , HTTP .

`{INT}{PERIOD_TYPE}` . : `s, m, h, d` .

`{LONG}` .

:

2 .

```
{
 "Endpoint": "*",
 "Period": "1s",
 "Limit": 2
}
```

API / 3 GET . PUT api / values . `{HTTP_Verb}{PATH}` .

HTTP `/api/values` 15 5 .

```
{
 "Endpoint": "*:/api/values",
 "Period": "15m",
 "Limit": 5
}
```

`/api/values` GET 5 .

```
{
 "Endpoint": "get:/api/values",
 "Period": "1h",
 "Limit": 5
}
```

API / 6 . GET api / values / 1 .

HTTP ClientRateLimitMiddleware .

- ID, HTTP URL . `ClientRateLimitMiddleware.SetIdentity` .
- ID URL .
- .
- .
- ,

.

```
Status Code: 429
Retry-After: 58
Content: API calls quota exceeded! maximum admitted 2 per 1m.
```

. HttpStatusCode QuotaExceededMessage . ClientRateLimitMiddleware.ReturnQuotaExceededResponse
. Retry-After .

X-Rate-Limit .

```
X-Rate-Limit-Limit: the rate limit period (eg. 1m, 12h, 1d)
X-Rate-Limit-Remaining: number of request remaining
X-Rate-Limit-Reset: UTC date time when the limits resets
```

Microsoft.Extensions.Logging.ILogger . ClientRateLimitMiddleware.LogBlockedRequest . .

```
info: AspNetCoreRateLimit.ClientRateLimitMiddleware[0]
      Request get:/api/values from ClientId client-id-1 has been blocked, quota 2/1m exceeded
by 3. Blocked by rule *:/api/value, TraceIdentifier 0HKTLISQQVV9D.
```

appsettings.json MemoryCacheClientPolicyStore DistributedCacheClientPolicyStore . .

```
public class ClientRateLimitController : Controller
{
    private readonly ClientRateLimitOptions _options;
    private readonly IClientPolicyStore _clientPolicyStore;

    public ClientRateLimitController(IOptions<ClientRateLimitOptions> optionsAccessor,
IClientPolicyStore clientPolicyStore)
    {
        _options = optionsAccessor.Value;
        _clientPolicyStore = clientPolicyStore;
    }

    [HttpGet]
    public ClientRateLimitPolicy Get()
    {
        return _clientPolicyStore.Get($"{_options.ClientPolicyPrefix}_cl-key-1");
    }

    [HttpPost]
    public void Post()
    {
        var id = $"{_options.ClientPolicyPrefix}_cl-key-1";
        var clPolicy = _clientPolicyStore.Get(id);
        clPolicy.Rules.Add(new RateLimitRule
        {
            Endpoint = "*/api/testpolicyupdate",
            Period = "1h",
            Limit = 100
        });
        _clientPolicyStore.Set(id, clPolicy);
    }
}
```

.

: https://riptutorial.com/ko/asp-net-core/topic/5240/-

20:

Examples

appsetting

appsettings. {EnvironmentName} .json .

- appsettings.Development.json
- appsettings.Staging.json
- appsettings.Production.json

project.json "publishOptions" "include" . .

```
"publishOptions": {
  "include": [
    "appsettings.Development.json",
    "appsettings.Staging.json",
    "appsettings.Production.json"
    ...
  ]
}
```

. Startup .

```
.AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional: true);
```

:

```
var builder = new ConfigurationBuilder()
            .SetBasePath(env.ContentRootPath)
            .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional: true)
            .AddEnvironmentVariables();
```

/

`IHostingEnvironment` `IHostingEnvironment` .

- :

  ```
  env.EnvironmentName
  ```

- `Development` , `Staging` , `Production` HostingEnvironmentExtensions .

  ```
  env.IsDevelopment()
  env.IsStaging()
  env.IsProduction()
  ```

-

(HostingEnvironmentExtensions :

```
env.IsEnvironment("environmentname")
```

- :

```
env.EnvironmentName == "Development"
```

Dependency Injection `Startup` .

`public void Configure(IApplicationBuilder app)` `public void ConfigureServices(IServiceCollection services)` `Configure{EnvironmentName}` `Configure{EnvironmentName}Services` .

/ `Startup` `if/else` .

```
public class Startup
{
    public void ConfigureServices(IServiceCollection services) { }
    public void ConfigureStaggingServices(IServiceCollection services) { }
    public void ConfigureProductionServices(IServiceCollection services) { }

    public void Configure(IApplicationBuilder app) { }
    public void ConfigureStagging(IApplicationBuilder app) { }
    public void ConfigureProduction(IApplicationBuilder app) { }
}
```

`Configure{Environmentname}` `Configure{Environmentname}Services` `Configure` `ConfigureServices` .

`Startup` .`ASPNETCORE_ENVIRONMENT` `Production` `StartupProduction` `Stagging` `Development` `Startup` .

:

```
public class Startup
{
    public Startup(IHostingEnvironment hostEnv)
    {
        // Set up configuration sources.
        var builder = new ConfigurationBuilder()
            .SetBasePath(hostEnv.ContentRootPath)
            .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)
            .AddJsonFile($"appsettings.{hostEnv.EnvironmentName}.json", optional: true,
reloadOnChange: true);

        if (hostEnv.IsDevelopment())
        {
            // This will push telemetry data through Application Insights pipeline faster,
allowing you to view results immediately.
            builder.AddApplicationInsightsSettings(developerMode: true);
        }

        builder.AddEnvironmentVariables();
        Configuration = builder.Build();
    }

    public IConfigurationRoot Configuration { get; set; }
```

```
    // This method gets called by the runtime. Use this method to add services to the
container
    public static void RegisterCommonServices(IServiceCollection services)
    {
        services.AddScoped<ICommonService, CommonService>();
        services.AddScoped<ICommonRepository, CommonRepository>();
    }

    public void ConfigureServices(IServiceCollection services)
    {
        RegisterCommonServices(services);

        services.AddOptions();
        services.AddMvc();
    }

    public void ConfigureDevelopment(IApplicationBuilder app, IHostingEnvironment env,
ILoggerFactory loggerFactory)
    {
        loggerFactory.AddConsole(Configuration.GetSection("Logging"));
        loggerFactory.AddDebug();

        app.UseBrowserLink();
        app.UseDeveloperExceptionPage();

        app.UseApplicationInsightsRequestTelemetry();
        app.UseApplicationInsightsExceptionTelemetry();
        app.UseStaticFiles();
        app.UseMvc();
    }

    // No console Logger and debugging tools in this configuration
    public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory
loggerFactory)
    {
        loggerFactory.AddDebug();

        app.UseApplicationInsightsRequestTelemetry();
        app.UseApplicationInsightsExceptionTelemetry();
        app.UseStaticFiles();
        app.UseMvc();
    }
}
```

. .

```
<environment names="Development">
    <h1>This is heading for development environment</h1>
</environment>
<environment names="Staging,Production">
    <h1>This is heading for Staging or production environment</h1>
</environment>
```

Environment `names` .

```
Development
```

```
SET ASPNETCORE_ENVIRONMENT=Development
```

Asp.Net Core .

1. (= .
2. .

PowerShell

PowerShell setx.exe .

1. PowerShell
2. .

   setx ASPNETCORE_ENVIRONMENT ""

   setx ASPNETCORE_ENVIRONMENT ""

3. PowerShell

web.config ASPNETCORE_ENVIRONMENT

ASPNETCORE_ENVIRONMENT web.config web.config .

```
<aspNetCore processPath=".\WebApplication.exe" arguments="" stdoutLogEnabled="false"
stdoutLogFile=".\logs\stdout" forwardWindowsAuthToken="false">
  <environmentVariables>
    <environmentVariable name="ASPNETCORE_ENVIRONMENT" value="Development" />
  </environmentVariables>
</aspNetCore>
```

: https://riptutorial.com/ko/asp-net-core/topic/2292/--

21:

Examples

ASP.NET `UseStatusCodePages` `UseStatusCodePagesWithRedirects` `UseStatusCodePagesWithRedirects` .

- UseStatusCodePages 400 599 StatusCodePages . :

```
app.UseStatusCodePages(async context => {
  //context.HttpContext.Response.StatusCode contains the status code

  // your redirect logic

});
```

- UseStatusCodePagesWithRedirects StatusCodePages . URL . '{0}' . '~' URL PathBase URL . ~ / errors / < > (: 403 ~ / errors / 403).

```
app.UseStatusCodePagesWithRedirects("~/errors/{0}");
```

ASP.NET

UseExceptionHandler . (Stack Trace), (Inner Exception) . . .

```
app.UseExceptionHandler(
 options => {
    options.Run(
    async context =>
    {
      context.Response.StatusCode = (int)HttpStatusCode.InternalServerError;
      context.Response.ContentType = "text/html";
      var ex = context.Features.Get<IExceptionHandlerFeature>();
      if (ex != null)
      {
        var err = $"<h1>Error: {ex.Error.Message}</h1>{ex.Error.StackTrace }";
        await context.Response.WriteAsync(err).ConfigureAwait(false);
      }
    });
 }
);
```

startup.cs configure () .

: https://riptutorial.com/ko/asp-net-core/topic/6581/-

22:

Aspnet Dependency Injection . .

- IServiceCollection.Add(ServiceDescriptor item);
- IServiceCollection.AddScoped(Type serviceType);
- IServiceCollection.AddScoped(Type serviceType, Type implementationType);
- IServiceCollection.AddScoped(Type serviceType, Func<IServiceProvider, object> implementationFactory);
- IServiceCollection.AddScoped<TService>()
- IServiceCollection.AddScoped<TService>(Func<IServiceProvider, TService> implementationFactory)
- IServiceCollection.AddScoped<TService, TImplementation>()
- IServiceCollection.AddScoped<TService, TImplementation>(Func<IServiceProvider, TImplementation> implementationFactory)
- IServiceCollection.AddSingleton(Type serviceType);
- IServiceCollection.AddSingleton(Type serviceType, Func<IServiceProvider, object> implementationFactory);
- IServiceCollection.AddSingleton(Type serviceType, Type implementationType);
- IServiceCollection.AddSingleton(Type serviceType, object implementationInstance);
- IServiceCollection.AddSingleton<TService>()
- IServiceCollection.AddSingleton<TService>(Func<IServiceProvider, TService> implementationFactory)
- IServiceCollection.AddSingleton<TService>(TService implementationInstance)
- IServiceCollection.AddSingleton<TService, TImplementation>()
- IServiceCollection.AddSingleton<TService, TImplementation>(Func<IServiceProvider, TImplementation> implementationFactory)
- IServiceCollection.AddTransient(Type serviceType);
- IServiceCollection.AddTransient(Type serviceType, Func<IServiceProvider, object> implementationFactory);
- IServiceCollection.AddTransient(Type serviceType, Type implementationType);
- IServiceCollection.AddTransient<TService>()
- IServiceCollection.AddTransient<TService>(Func<IServiceProvider, TService> implementationFactory)
- IServiceCollection.AddTransient<TService, TImplementation>()
- IServiceCollection.AddTransient<TService, TImplementation>(Func<IServiceProvider, TImplementation> implementationFactory)
- IServiceProvider.GetService(Type serviceType)
- IServiceProvider.GetService<T>()
- IServiceProvider.GetServices(Type serviceType)
- IServiceProvider.GetServices<T>()

IServiceProvider .

```
using Microsoft.Extensions.DependencyInjection;
```

Examples

. :

ITestService.cs

```
public interface ITestService
{
```

```
    int GenerateRandom();
}
```

TestService.cs

```
public class TestService : ITestService
{
    public int GenerateRandom()
    {
        return 4;
    }
}
```

Startup.cs (ConfigureServices)

```
public void ConfigureServices(IServiceCollection services)
{
    // ...

    services.AddTransient<ITestService, TestService>();
}
```

HomeController.cs

```
using Microsoft.Extensions.DependencyInjection;

namespace Core.Controllers
{
    public class HomeController : Controller
    {
        public HomeController(ITestService service)
        {
            int rnd = service.GenerateRandom();
        }
    }
}
```

Builtin .

```
public void ConfigureServices(IServiceCollection services)
    {
        // ...

        services.AddTransient<ITestService, TestService>();
        // or
        services.AddScoped<ITestService, TestService>();
        // or
        services.AddSingleton<ITestService, TestService>();
        // or
        services.AddSingleton<ITestService>(new TestService());
    }
```

- **AddTransient** : .

- **AddScoped** :
- **AddSingleton** :
- **AddSingleton ()** : .

.

```
 services.TryAddEnumerable(ServiceDescriptor.Transient<ITestService, TestServiceImpl1>());
 services.TryAddEnumerable(ServiceDescriptor.Transient<ITestService, TestServiceImpl2>());
```

.

```
public class HomeController : Controller
{
    public HomeController(IEnumerable<ITestService> services)
    {
        // do something with services.
    }
}
```

.

```
services.Add(ServiceDescriptor.Singleton(typeof(IKeyValueStore<>), typeof(KeyValueStore<>)));
```

.

```
public class HomeController : Controller
{
    public HomeController(IKeyValueStore<UserSettings> userSettings)
    {
        // do something with services.
    }
}
```

.

```
// ...
using System;
using Microsoft.Extensions.DependencyInjection;

namespace Core.Controllers
{
    public class HomeController : Controller
    {
        public HomeController(ITestService service)
        {
            int rnd = service.GenerateRandom();
        }
    }
}
```

`FromServicesAttribute` Controller Action .

```
[HttpGet]
public async Task<IActionResult> GetAllAsync([FromServices]IProductService products)
{
     return Ok(await products.GetAllAsync());
}
```

`[FromServices]` "Property Injection" "Method Injection" . (ASP.NET Core DI).

. ASP.NET Core MVC `Microsoft.AspNetCore.Mvc.Core` .

ASP.NET MVC GitHub ([FromServices]) .

> @rynowak :
>
> > @ :
> >
> > .
>
> . "[FromServices] "[FromServices] " . .
>
> [FromServices] . .
>
> / cc @ danroth27 - docs
>
> [FromServices] DI (Autofac) .

:

- .NET Core Dependency Injection `[FromServices]` .
- . .
- `Microsoft.AspNetCore.Mvc.Core` ASP.NET Core MVC (.NET Framework .NET Core) .
- IoC (Autofac, Unity) .

/

Microsoft ASP.NET .

.

```
public class MySettings
{
    public string Value1 { get; set; }
    public string Value2 { get; set; }
}
```

`appsettings.json` .

```
{
  "mysettings" : {
    "value1": "Hello",
    "value2": "World"
  }
}
```

Startup . .

1. appsettings.json "mysettings" .

   ```
   services.Configure<MySettings>(Configuration.GetSection("mysettings"));
   ```

2. .

   ```
   services.Configure<MySettings>(new MySettings
   {
       Value1 = "Hello",
       Value2 = Configuration["mysettings:value2"]
   });
   ```

   appsettings.json : . value2 mysettings mysettings:value2 .

IOptions<T> .

```
public class MyService : IMyService
{
    private readonly MySettings settings;

    public MyService(IOptions<MySettings> mysettings)
    {
        this.settings = mySettings.Value;
    }
}
```

IOptions<T> IOptions<T> T .

/

.

app.ApplicationServices.GetService<AppDbContext>()
app.ApplicationServices.GetService<AppDbContext>() Cannot access a disposed object in ASP.NET Core when injecting DbContext .

.

```
public Configure(IApplicationBuilder app)
{
    // serviceProvider is app.ApplicationServices from Configure(IApplicationBuilder app)
method
```

```
    using (var serviceScope =
app.ApplicationServices.GetRequiredService<IServiceScopeFactory>().CreateScope())
    {
        var db = serviceScope.ServiceProvider.GetService<AppDbContext>();

        if (await db.Database.EnsureCreatedAsync())
        {
            await SeedDatabase(db);
        }
    }
}
```

Entity Framework MusicStore .

Dependency Injection , ViewComponents TagHelpers

, ViewComponents TagHelpers . AutoFac 3 Inversion of Control (IoC) .

ASP.NET MVC IoC `Startup.cs` (GitHub ControllersFromService)

```
public void ConfigureServices(IServiceCollection services)
{
    var builder = services
        .AddMvc()
        .ConfigureApplicationPartManager(manager => manager.ApplicationParts.Clear())
        .AddApplicationPart(typeof(TimeScheduleController).GetTypeInfo().Assembly)
        .ConfigureApplicationPartManager(manager =>
        {
            manager.ApplicationParts.Add(new TypesPart(
              typeof(AnotherController),
              typeof(ComponentFromServicesViewComponent),
              typeof(InServicesTagHelper)));

            manager.FeatureProviders.Add(new AssemblyMetadataReferenceFeatureProvider());
        })
        .AddControllersAsServices()
        .AddViewComponentsAsServices()
        .AddTagHelpersAsServices();

    services.AddTransient<QueryValueService>();
    services.AddTransient<ValueService>();
    services.AddSingleton<IHttpContextAccessor, HttpContextAccessor>();
}
```

Plain Dependency Injection (Startup.cs)

kestrel `WebHostBuilder` Microsoft.Extensions.DependencyInjection nuget (:).

```
internal class Program
{
    public static void Main(string[] args)
    {
        var services = new ServiceCollection(); //Creates the service registry
        services.AddTransient<IMyInterface, MyClass>(); //Add registration of IMyInterface
(should create an new instance of MyClass every time)
        var serviceProvider = services.BuildServiceProvider(); //Build dependencies into an
```

```
IOC container
        var implementation = serviceProvider.GetService<IMyInterface>(); //Gets a dependency

        //serviceProvider.GetService<ServiceDependingOnIMyInterface>(); //Would throw an error
since ServiceDependingOnIMyInterface is not registered
        var manualyInstaniate = new ServiceDependingOnIMyInterface(implementation);

        services.AddTransient<ServiceDependingOnIMyInterface>();
        var spWithService = services.BuildServiceProvider(); //Generaly its bad practise to
rebuild the container because its heavey and promotes use of anti-pattern.
        spWithService.GetService<ServiceDependingOnIMyInterface>(); //only now i can resolve
    }
}

interface IMyInterface
{
}

class MyClass : IMyInterface
{
}

class ServiceDependingOnIMyInterface
{
    private readonly IMyInterface _dependency;

    public ServiceDependingOnIMyInterface(IMyInterface dependency)
    {
        _dependency = dependency;
    }
}
```

Microsoft.Extensions.DependencyInjection

IServiceCollection

Microsoft DI nuget IOC `IServiceCollection` . Collection : `ServiceCollection` :

```
var services = new ServiceCollection();
```

`IServiceCollection IList<ServiceDescriptor>, ICollection<ServiceDescriptor>, IEnumerable<ServiceDescriptor>, IEnumerable`

`ServiceDescriptor` .

```
services.AddTransient<Class>(); //add registration that is always recreated
services.AddSingleton<Class>(); // add registration that is only created once and then re-used
services.AddTransient<Abstract, Implementation>(); //specify implementation for interface
services.AddTransient<Interface>(serviceProvider=> new
Class(serviceProvider.GetService<IDependency>())); //specify your own resolve function/
factory method.
services.AddMvc(); //extension method by the MVC nuget package, to add a whole bunch of
registrations.
// etc..
```

```
//when not using an extension method:
services.Add(new ServiceDescriptor(typeof(Interface), typeof(Class)));
```

IServiceProvider

serviceprovider ' ' `services.BuildServiceProvider()` `services.BuildServiceProvider()` .BuildServiceProvider `services.BuildServiceProvider()` (mehtod).

```
var provider = new ServiceProvider( services, false); //false is if it should validate scopes
```

`IServiceCollection` `ServiceDescriptor` `Func<ServiceProvider, object>` . object , Singleton .

`ServiceTable` `ConcurrentDictionary` `ServiceType` .

`ConcurrentDictionary<Type, Func<ServiceProvider, object>>` . .

```
  var serviceProvider = new ConcurrentDictionary<Type, Func<ServiceProvider, object>>();
  var factoryMethod = serviceProvider[typeof(MyService)];
  var myServiceInstance = factoryMethod(serviceProvider)
```

!

`ConcurrentDictionary` `ServiceProvider` `ServiceTable` .

: https://riptutorial.com/ko/asp-net-core/topic/1949/-

23:

. *Aspnet Core* .

.

. 3 , ( ), Redis Sql Server.

Examples

ASP.NET InMemory

ASP.NET `project.json` .

```
  "Microsoft.Extensions.Caching.Memory": "1.0.0-rc2-final",
```

Startup ConfigureServices (Microsoft.Extensions.Caching.Memory)

```
services.AddMemoryCache();
```

(: Controller) `IMemoryCache` .

```
private IMemoryCache _memoryCache;
public HomeController(IMemoryCache memoryCache)
{
    _memoryCache = memoryCache;
}
```

Get `null` .

```
    // try to get the cached item; null if not found
    // greeting = _memoryCache.Get(cacheKey) as string;

    // alternately, TryGet returns true if the cache entry was found
    if(!_memoryCache.TryGetValue(cacheKey, out greeting))
```

Set . Set , `MemoryCacheEntryOptions` . `MemoryCacheEntryOptions` , , . -

```
_memoryCache.Set(cacheKey, greeting,
                new MemoryCacheEntryOptions()
                .SetAbsoluteExpiration(TimeSpan.FromMinutes(1)));
```

.

-
- SQL

Redis .

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDistributedRedisCache(options =>
    {
        options.Configuration = "ServerAdress";
        options.InstanceName = "InstanceName";
    });
}
```

IDistributedCache .

```
public class BooksController {
    private IDistributedCache distributedCache;

    public BooksController(IDistributedCache distributedCache) {
        this.distributedCache = distributedCache;
    }

    [HttpGet]
    public async Task<Books[]> GetAllBooks() {
        var serialized = this.distributedCache.GetStringAsync($"allbooks");
        Books[] books = null;
        if (string.IsNullOrEmpty(serialized)) {
            books = await Books.FetchAllAsync();
            this.distributedCache.SetStringAsync($"allbooks",
JsonConvert.SerializeObject(books));
        } else {
            books = JsonConvert.DeserializeObject<Books[]>(serialized);
        }
        return books;
    }
}
```

: https://riptutorial.com/ko/asp-net-core/topic/8090/

24:

ASP	
ASP	asp-action .
ASP - * . 8 .	

Examples

-

```
<form asp-action="create" asp-controller="Home">
     <!--Your form elements goes here-->
</form>
```

-

```
<form asp-action="create"
      asp-controller="Home"
      asp-route-returnurl="dashboard"
      asp-route-from="google">
     <!--Your form elements goes here-->
</form>
```

.

```
<form action="/Home/create?returnurl=dashboard&amp;from=google" method="post">
    <!--Your form elements goes here-->
</form>
```

.

```
public class CreateProduct
{
   public string Name { set; get; }
}
```

.

```
@model CreateProduct
<form asp-action="create" asp-controller="Home" >

    <input type="text" asp-for="Name"/>
    <input type="submit"/>

</form>
```

.

```
<form action="/Home/create" method="post">

    <input type="text" id="Name" name="Name" value="" />
    <input type="submit"/>
    <input name="__RequestVerificationToken" type="hidden" value="ThisWillBeAUniqueToken" />

</form>
```

Name .

```
public IActionResult Create()
{
  var vm = new CreateProduct { Name="IPhone"};
  return View(vm);
}
```

`CreateProduct` HttpPost / `name` .

.

```
public class CreateProduct
{
   public IEnumerable<SelectListItem> Categories { set; get; }
   public int SelectedCategory { set; get; }
}
```

GET , Categories .

```
public IActionResult Create()
{
    var vm = new CreateProduct();
    vm.Categories = new List<SelectListItem>
    {
        new SelectListItem {Text = "Books", Value = "1"},
        new SelectListItem {Text = "Furniture", Value = "2"}
    };
    return View(vm);
}
```

```
@model CreateProduct
```

```
<form asp-action="create" asp-controller="Home">
    <select asp-for="SelectedCategory" asp-items="@Model.Categories">
        <option>Select one</option>
    </select>
    <input type="submit"/>
</form>
```

(/)

```
<form action="/Home/create" method="post">
```

```
    <select data-val="true" id="SelectedCategory" name="SelectedCategory">
        <option>Select one</option>
        <option value="1">Shyju</option>
        <option value="2">Sean</option>
    </select>
    <input type="submit"/>
</form>
```

HttpPost .

```
[HttpPost]
public ActionResult Create(CreateProduct model)
{
  //check model.SelectedCategory value
  / /to do : return something
}
```

`SelectedCategory` .

```
public IActionResult Create()
{
    var vm = new CreateProduct();
    vm.Categories = new List<SelectListItem>
    {
        new SelectListItem {Text = "Books", Value = "1"},
        new SelectListItem {Text = "Furniture", Value = "2"},
        new SelectListItem {Text = "Music", Value = "3"}
    };
    vm.SelectedCategory = 2;
    return View(vm);
}
```

/ ListBox

`asp-for` .

```
public class CreateProduct
{
   public IEnumerable<SelectListItem> Categories { set; get; }
   public int[] SelectedCategories { set; get; }
}
```

```
@model CreateProduct
```

```
<form asp-action="create" asp-controller="Home" >
    <select asp-for="SelectedCategories" asp-items="@Model.Categories">
        <option>Select one</option>
    </select>
    <input type="submit"/>
</form>
```

`multiple` SELECT .

```
<form action="/Home/create" method="post">
```

```
     <select id="SelectedCategories" multiple="multiple" name="SelectedCategories">
        <option>Select one</option>
        <option value="1">Shyju</option>
       <option value="2">Sean</option>
     </select>
    <input type="submit"/>
</form>
```

ITagHelper TagHelper .

- TagHelper HTML . WidgetTagHelper <widget> .
- [HtmlTargetElement] [HtmlTargetElement] .
- . public string Title {get; set;} <widget title="my title"> public string Title {get; set;}
- C # . [HtmlTargetElement] WidgetBoxTagHelper Razor <widget-box></widget-box> .
- Process ProcessAsync . .

_ViewImports.cshtml ( ).

```
@addTagHelper *, MyAssembly
```

.

```
<widget-box title="My Title">This is my content: @ViewData["Message"]</widget-box>
```

.

```
<div class="widget-box">
    <div class="widget-header">My Title</div>
    <div class="widget-body">This is my content: some message</div>
</div>
```

.

```
[HtmlTargetElement("widget-box")]
public class WidgetTagHelper : TagHelper
{
    public string Title { get; set; }

    public override async Task ProcessAsync(TagHelperContext context, TagHelperOutput output)
    {
        var outerTag = new TagBuilder("div");
        outerTag.Attributes.Add("class", output.TagName);
        output.MergeAttributes(outerTag);
        output.TagName = outerTag.TagName;

        //Create the header
        var header = new TagBuilder("div");
        header.Attributes.Add("class", "widget-header");
        header.InnerHtml.Append(this.Title);
        output.PreContent.SetHtmlContent(header);
```

```
        //Create the body and replace original tag helper content
        var body = new TagBuilder("div");
        body.Attributes.Add("class", "widget-body");
        var originalContents = await output.GetChildContentAsync();
        body.InnerHtml.Append(originalContents.GetContent());
        output.Content.SetHtmlContent(body);
    }
}
```

`label` . MVC `Html.LabelFor` .

.

```
public class FormViewModel
{
    public string Name { get; set; }
}
```

HTML `label asp-for` `asp-for` .

```
<form>
    <label asp-for="Name"></label>
    <input asp-for="Name" type="text" />
</form>
```

MVC .

```
<form>
    @Html.LabelFor(x => x.Name)
    @Html.TextBoxFor(x => x.Name)
</form>
```

HTML .

```
<form>
    <label for="Name">Name</label>
    <input name="Name" id="Name" type="text" value="">
</form>
```

href MVC .

```
<a asp-controller="Products" asp-action="Index">Login</a>
```

. asp-route- .

```
<a asp-controller="Products" asp-action="Details" asp-route-id="@Model.ProductId">
   View Details
</a>
```

: https://riptutorial.com/ko/asp-net-core/topic/2665/-

25:

Examples

JSON

ASP.NET / . . .json ASP.NET .

Microsoft.EntityFrameworkCore.InMemory .

:

1. DigitalShop , .
2. .
3. . .

project.json dependencies .

```
"Microsoft.EntityFrameworkCore": "1.0.0",
"Microsoft.EntityFrameworkCore.SqlServer": "1.0.0",
"Microsoft.EntityFrameworkCore.InMemory": "1.0.0"
```

Startup.cs Startup.cs : (using)

Startup.cs

```
namespace DigitalShop
{
    public class Startup
    {
        public static string UiCulture;
        public static string CultureDirection;
        public static IStringLocalizer _e; // This is how we access language strings

        public static IConfiguration LocalConfig;

        public Startup(IHostingEnvironment env)
        {
            var builder = new ConfigurationBuilder()
                .SetBasePath(env.ContentRootPath)
                .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true) //
this is where we store apps configuration including language
                .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional: true)
                .AddEnvironmentVariables();

            Configuration = builder.Build();
            LocalConfig = Configuration;
        }

        public IConfigurationRoot Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the
container.
```

```
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddMvc().AddViewLocalization().AddDataAnnotationsLocalization();

            // IoC Container
            // Add application services.
            services.AddTransient<EFStringLocalizerFactory>();
            services.AddSingleton<IConfiguration>(Configuration);
        }

        // This method gets called by the runtime. Use this method to configure the HTTP
request pipeline.
        public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory
loggerFactory, EFStringLocalizerFactory localizerFactory)
        {
            _e = localizerFactory.Create(null);

            // a list of all available languages
            var supportedCultures = new List<CultureInfo>
            {
                new CultureInfo("en-US"),
                new CultureInfo("fa-IR")
            };

            var requestLocalizationOptions = new RequestLocalizationOptions
            {
                SupportedCultures = supportedCultures,
                SupportedUICultures = supportedCultures,
            };
            requestLocalizationOptions.RequestCultureProviders.Insert(0, new
JsonRequestCultureProvider());
            app.UseRequestLocalization(requestLocalizationOptions);

            app.UseStaticFiles();

            app.UseMvc(routes =>
            {
                routes.MapRoute(
                    name: "default",
                    template: "{controller=Home}/{action=Index}/{id?}");
            });
        }
    }

    public class JsonRequestCultureProvider : RequestCultureProvider
    {
        public override Task<ProviderCultureResult> DetermineProviderCultureResult(HttpContext
httpContext)
        {
            if (httpContext == null)
            {
                throw new ArgumentNullException(nameof(httpContext));
            }

            var config = Startup.LocalConfig;

            string culture = config["AppOptions:Culture"];
            string uiCulture = config["AppOptions:UICulture"];
            string culturedirection = config["AppOptions:CultureDirection"];

            culture = culture ?? "fa-IR"; // Use the value defined in config files or the
```

```
default value
            uiCulture = uiCulture ?? culture;

            Startup.UiCulture = uiCulture;

            culturedirection = culturedirection ?? "rlt"; // rtl is set to be the default
value in case culturedirection is null
            Startup.CultureDirection = culturedirection;

            return Task.FromResult(new ProviderCultureResult(culture, uiCulture));
        }
    }
}
```

public static . .

Startup json builder . . appsettings.json . Visual Studio 2015 . (Logging)

appsettings.json

```
{
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Debug",
      "System": "Information",
      "Microsoft": "Information"
    }
  },
  "AppOptions": {
    "Culture": "en-US", // fa-IR for Persian
    "UICulture": "en-US", // same as above
    "CultureDirection": "ltr" // rtl for Persian/Arabic/Hebrew
  }
}
```

.

Models , Services Languages . Models Localization .

Services EFLocalization .cs EFLocalization . (using)

EFLocalization.cs

```
namespace DigitalShop.Services
{
    public class EFStringLocalizerFactory : IStringLocalizerFactory
    {
        private readonly LocalizationDbContext _db;

        public EFStringLocalizerFactory()
        {
            _db = new LocalizationDbContext();
            // Here we define all available languages to the app
            // available languages are those that have a json and cs file in
            // the Languages folder
            _db.AddRange(
```

```
                new Culture
                {
                    Name = "en-US",
                    Resources = en_US.GetList()
                },
                new Culture
                {
                    Name = "fa-IR",
                    Resources = fa_IR.GetList()
                }
            );
            _db.SaveChanges();
        }

        public IStringLocalizer Create(Type resourceSource)
        {
            return new EFStringLocalizer(_db);
        }

        public IStringLocalizer Create(string baseName, string location)
        {
            return new EFStringLocalizer(_db);
        }
    }

    public class EFStringLocalizer : IStringLocalizer
    {
        private readonly LocalizationDbContext _db;

        public EFStringLocalizer(LocalizationDbContext db)
        {
            _db = db;
        }

        public LocalizedString this[string name]
        {
            get
            {
                var value = GetString(name);
                return new LocalizedString(name, value ?? name, resourceNotFound: value ==
null);
            }
        }

        public LocalizedString this[string name, params object[] arguments]
        {
            get
            {
                var format = GetString(name);
                var value = string.Format(format ?? name, arguments);
                return new LocalizedString(name, value, resourceNotFound: format == null);
            }
        }

        public IStringLocalizer WithCulture(CultureInfo culture)
        {
            CultureInfo.DefaultThreadCurrentCulture = culture;
            return new EFStringLocalizer(_db);
        }

        public IEnumerable<LocalizedString> GetAllStrings(bool includeAncestorCultures)
```

```
        {
            return _db.Resources
                .Include(r => r.Culture)
                .Where(r => r.Culture.Name == CultureInfo.CurrentCulture.Name)
                .Select(r => new LocalizedString(r.Key, r.Value, true));
        }

        private string GetString(string name)
        {
            return _db.Resources
                .Include(r => r.Culture)
                .Where(r => r.Culture.Name == CultureInfo.CurrentCulture.Name)
                .FirstOrDefault(r => r.Key == name)?.Value;
        }
    }

    public class EFStringLocalizer<T> : IStringLocalizer<T>
    {
        private readonly LocalizationDbContext _db;

        public EFStringLocalizer(LocalizationDbContext db)
        {
            _db = db;
        }

        public LocalizedString this[string name]
        {
            get
            {
                var value = GetString(name);
                return new LocalizedString(name, value ?? name, resourceNotFound: value ==
null);
            }
        }

        public LocalizedString this[string name, params object[] arguments]
        {
            get
            {
                var format = GetString(name);
                var value = string.Format(format ?? name, arguments);
                return new LocalizedString(name, value, resourceNotFound: format == null);
            }
        }

        public IStringLocalizer WithCulture(CultureInfo culture)
        {
            CultureInfo.DefaultThreadCurrentCulture = culture;
            return new EFStringLocalizer(_db);
        }

        public IEnumerable<LocalizedString> GetAllStrings(bool includeAncestorCultures)
        {
            return _db.Resources
                .Include(r => r.Culture)
                .Where(r => r.Culture.Name == CultureInfo.CurrentCulture.Name)
                .Select(r => new LocalizedString(r.Key, r.Value, true));
        }

        private string GetString(string name)
        {
```

```
            return _db.Resources
                .Include(r => r.Culture)
                .Where(r => r.Culture.Name == CultureInfo.CurrentCulture.Name)
                .FirstOrDefault(r => r.Key == name)?.Value;
        }
    }
}
```

Entity Framework Core `IStringLocalizerFactory` . `EFStringLocalizerFactory` `EFStringLocalizerFactory` . .

`Models/Localization` .

Culture.cs

```
namespace DigitalShop.Models.Localization
{
    public class Culture
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public virtual List<Resource> Resources { get; set; }
    }
}
```

Resource.cs

```
namespace DigitalShop.Models.Localization
{
    public class Resource
    {
        public int Id { get; set; }
        public string Key { get; set; }
        public string Value { get; set; }
        public virtual Culture Culture { get; set; }
    }
}
```

LocalizationDbContext.cs

```
namespace DigitalShop.Models.Localization
{
    public class LocalizationDbContext : DbContext
    {
        public DbSet<Culture> Cultures { get; set; }
        public DbSet<Resource> Resources { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseInMemoryDatabase();
        }
    }
}
```

, `DBContext` EF `DBContext` .

. - JSON .

. . . - JSON JSON .cs . JSON deserialize `GetList` . `EFStringLocalizerFactory` .

`Languages` .

en-US.cs

```
namespace DigitalShop.Languages
{
    public static class en_US
    {
        public static List<Resource> GetList()
        {
            var jsonSerializerSettings = new JsonSerializerSettings();
            jsonSerializerSettings.MissingMemberHandling = MissingMemberHandling.Ignore;
            return
JsonConvert.DeserializeObject<List<Resource>>(File.ReadAllText("Languages/en-US.json"),
jsonSerializerSettings);
        }
    }
}
```

en-US.json

```
[
  {
    "Key": "Welcome",
    "Value": "Welcome"
  },
  {
    "Key": "Hello",
    "Value": "Hello"
  },
]
```

fa-IR.cs

```
public static class fa_IR
{
    public static List<Resource> GetList()
    {
        var jsonSerializerSettings = new JsonSerializerSettings();
        jsonSerializerSettings.MissingMemberHandling = MissingMemberHandling.Ignore;
        return JsonConvert.DeserializeObject<List<Resource>>(File.ReadAllText("Languages/fa-
IR.json", Encoding.UTF8), jsonSerializerSettings);
    }
}
```

fa-IR.json

```
[
  {
    "Key": "Welcome",
    "Value": "دیدمآ شوخ"
 },
```

```
 {
  "Key": "Hello",
    "Value": "سلام"
 },
]
```

. (.cs .cshtml) (-) .

.cs (.cs):

```
// Returns "Welcome" for en-US and "خوش آمدی" for fa-IR
var welcome = Startup._e["Welcome"];
```

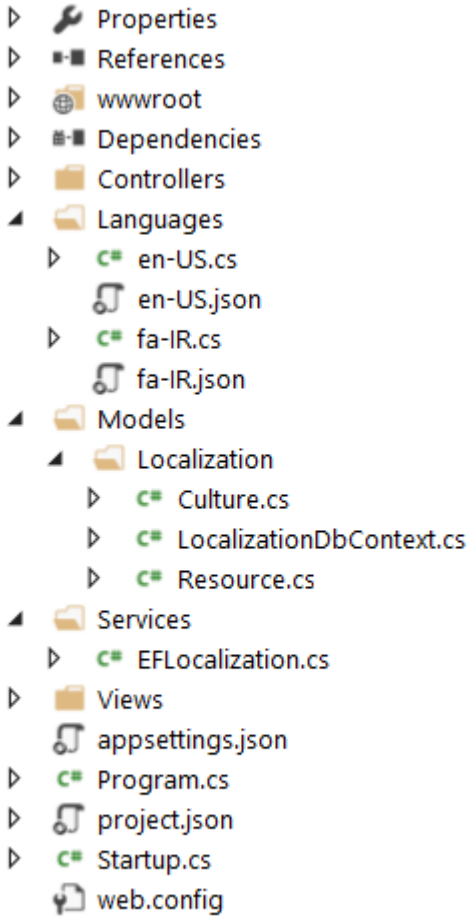
(.cshtml)

```
<h1>@Startup._e["Welcome"]</h1>
```

:

- Key JSON , , (Startup._e["How are you"] How are you
- .json , . () . .
- appsettings.json .
- **appsettings.json** / . , / .

.



URL

Request Localization query, cookie `Accept-Language` culture . `/api/en-US/products` .

.

```
public class UrlRequestCultureProvider : RequestCultureProvider
{
    private static readonly Regex LocalePattern = new Regex(@"^[a-z]{2}(-[a-z]{2,4})?$",
                                                            RegexOptions.IgnoreCase);

    public override Task<ProviderCultureResult> DetermineProviderCultureResult(HttpContext
httpContext)
    {
        if (httpContext == null)
        {
            throw new ArgumentNullException(nameof(httpContext));
        }

        var url = httpContext.Request.Path;

        // Right now it's not possible to use httpContext.GetRouteData()
        // since it uses IRoutingFeature placed in httpContext.Features when
        // Routing Middleware registers. It's not set when the Localization Middleware
        // is called, so this example simply assumes the locale will always
        // be located in the second segment of a path, like in /api/en-US/products
        var parts = httpContext.Request.Path.Value.Split('/');
```

```
        if (parts.Length < 3)
        {
            return Task.FromResult<ProviderCultureResult>(null);
        }

        if (!LocalePattern.IsMatch(parts[2]))
        {
            return Task.FromResult<ProviderCultureResult>(null);
        }

        var culture = parts[2];
        return Task.FromResult(new ProviderCultureResult(culture));
    }
}
```

```
var localizationOptions = new RequestLocalizationOptions
{
    SupportedCultures = new List<CultureInfo>
    {
        new CultureInfo("de-DE"),
        new CultureInfo("en-US"),
        new CultureInfo("en-GB")
    },
    SupportedUICultures = new List<CultureInfo>
    {
        new CultureInfo("de-DE"),
        new CultureInfo("en-US"),
        new CultureInfo("en-GB")
    },
    DefaultRequestCulture = new RequestCulture("en-US")
};

// Adding our UrlRequestCultureProvider as first object in the list
localizationOptions.RequestCultureProviders.Insert(0, new UrlRequestCultureProvider
{
    Options = localizationOptions
});

app.UseRequestLocalization(localizationOptions);
```

. .

```
app.UseMvc(routes =>
{
    routes.MapRoute(
        name: "default",
        template: "api/{culture::regex(^[a-z]{{2}}-[A-Za-z]{{4}}$)}}/{controller}/{id?}");
    routes.MapRoute(
        name: "default",
        template: "api/{controller}/{id?}");
});
```

: https://riptutorial.com/ko/asp-net-core/topic/2869/

S. No		Contributors
1	asp.net-core	Alex Logan, Alexan, Ashish Rajput, Ashley Medway, Bogdan Stefanjuk, BrunoLM, ChadT, Community, gbellmann, Henk Mollema, Nate Barbettini, Rion Williams, Shog9, Shyju, Svek, Tseng, VSG24, Zach Becknell
2	Angular2 .Net Core	Alejandro Tobón, Sentient Entities
3	ASP.NET -	Gubr
4	ASP.NET 1.0	ravindra, Sanket
5	JavascriptServices	hmnzr
6	MailKit .NET	Ankit
7	project.json	Joel Harkes
8		Set
9	(CORS)	Henk Mollema, Sanket, Saqib Rokadia, Tseng
10		Cyprien Autexier, Jayantha Lal Sirisena
11		Daniel J.G.
12		gilmishal, RamenChef
13		ChadT, Tseng
14		Alex Logan, Ralf Bönning
15		Ali, Piotrek, Set, VSG24, Zach Becknell
16		Rion Williams, Zach Becknell
17		Dmitry, Sanket, Set, Tseng
18		Alex Logan, Rion Williams
19		Stefan P.
20		dotnetom, Johnny, Robert Paulsen, Sanket, Set, Tseng
21		Sanket, Set
22		Alexan, BrunoLM, Cyprien Autexier, Dan Soper, Darren Evans,

		gilmishal, Gurgen Hakobyan, Jayantha Lal Sirisena, Joel Harkes, mazzt, Tseng, Zach Becknell
23		Cyprien Autexier, Sanket
24		Ali, Daniel J.G., dotnetom, Shyju, tmg, Zach Becknell
25		Tseng, VSG24, Zach Becknell