How to integrate Vue and Asp.Net Core 3.1 app using VS Code

JANUARY 30, 2020 BY MEBAKAR1005 — LEAVE A COMMENT

In this tutorial, we are going to cover how to integrate ***Vue.js*** and ***Asp.Net Core 3.1*** application using ***Entity Framework Core*** in Visual Studio Code. We will use ***PostgreSql*** database in this tutorial. So, we are going to build a SPA (Single Page Application).

First-of-all, we will install all the dependencies and tools that we need in this project. And then we will see how to setup a new project using Vue and Asp.Net Core 3.1 in visual studio code. So, we will learn the following list in this tutorial:

1. Install all the dependencies and tools
2. How to install PostgreSql Database?

3. How to create a new project using dotnet CLI (Command Line Interface) in VS Code?
4. How to setup a database in Vue and Asp.Net (

And then finally, we will test our application and the

Setup a new project using Core 3.1 in visual studio code?

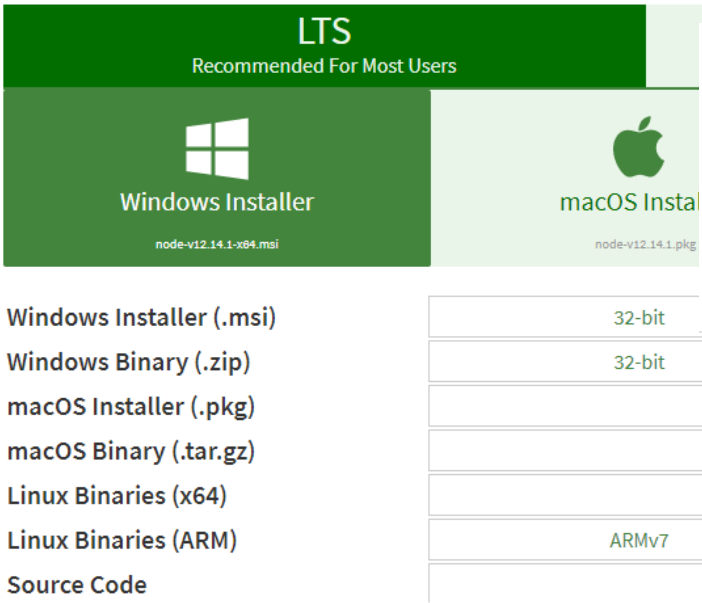
Let's start how to setup a new project in Vue.js and Asp.Net Core 3.1 application with Entity Framework Core using Visual Studio Code in step by step.

1- Install all the dependencies and tools

So, in this step, we will install all the dependencies and tools.

Install Node.js

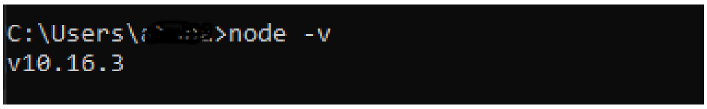
To install the Node.js, just click here (https://nodejs.org/en/download/) to go to the download page and then download the required version according to your machine and then install it on your machine.



Windows Installer (.msi)	32-bit	
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit	
macOS Binary (.tar.gz)	64-bit	
Linux Binaries (x64)	64-bit	
Linux Binaries (ARM)	ARMv7	ARMv8
Source Code	node-v12.14.1.tar.gz	

To test that node has been installed properly or not, go to command prompt and then run the below command and then you will see the output as you do see below in the screenshot.

=> node -v



```
C:\Users\[redacted]>node -v
v10.16.3
```

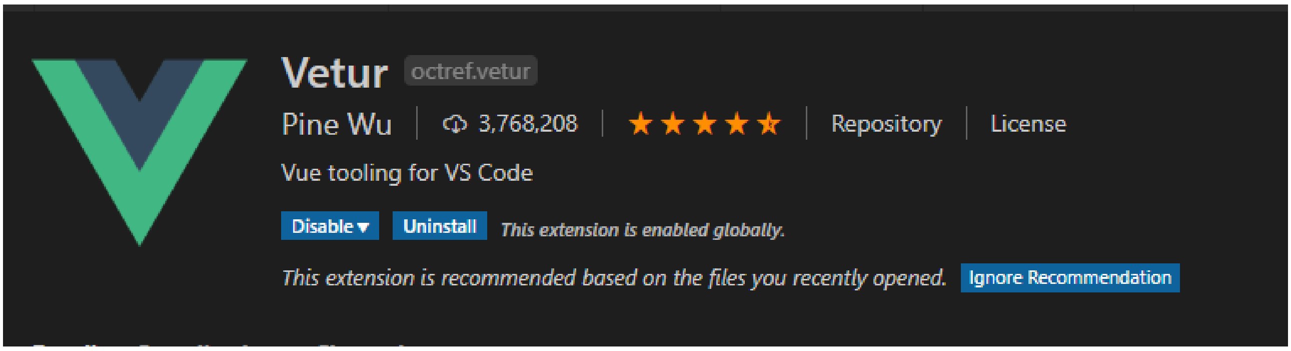
Install .Net Core 3.1

Now, we are going to install .Net Core 3.1 in our system. So, click here (https://dotnet.microsoft.com/download/dotnet-core/3.1) go to the download page and then download the required version according to your machine and then install it.

Note: – If you have already installed it, then you can skip this step.

Install Vetur

Vetur is a visual studio code extension. It will help us to build a Vue application like syntax highlighting, code snippets, intellisense and etc. To install it, just go to visual studio code and then go to extension menu from left sidebar and then search for Vetur and then install it.



Vetur octref.vetur

Pine Wu | 3,768,208 | ★★★★★ | Repository | License

Vue tooling for VS Code

This extension is enabled globally.

This extension is recommended based on the files you recently opened.

2: How to Install PostgreSQL Database?

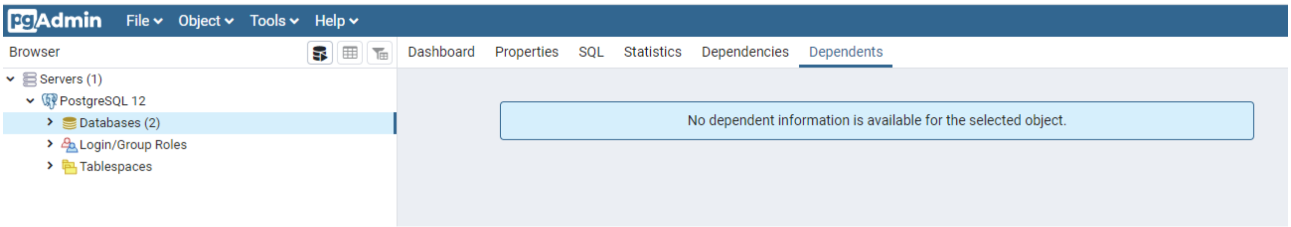
Now, in step, we will see how to install PostgreSQL. So, click here (https://www.enterprisedb.com/downloads/postgr download page and then install it on your machin the default options checked during the installation p

READ ASP.NET Core Interview Questions

Note: – Remember the port number and password that you have entered during the installation process and we will use it later.

After installation, we will see pgAdmin 4 App in our system as you do see below in the screenshot.

Note: – When we will run this app, it will ask the password.



No dependent information is available for the selected object.

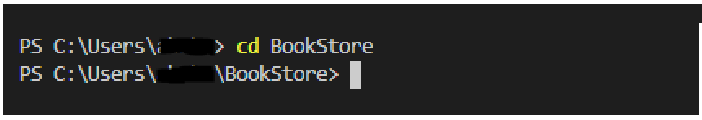
3: How to create a new project using dotnet CLI (Command Line Interface) in VS Code?

In this step, we will see how to create a new project in Vue and Asp.Net Core 3.1 using dotnet CLI (Command Line Interface) in Visual Studio Code. So, go to visual studio and create a new folder using this below command.

=> mkdir BookStore

And then go to inside this **BookStore** folder using the below command as you do see in the below screenshot.

=> cd BookStore

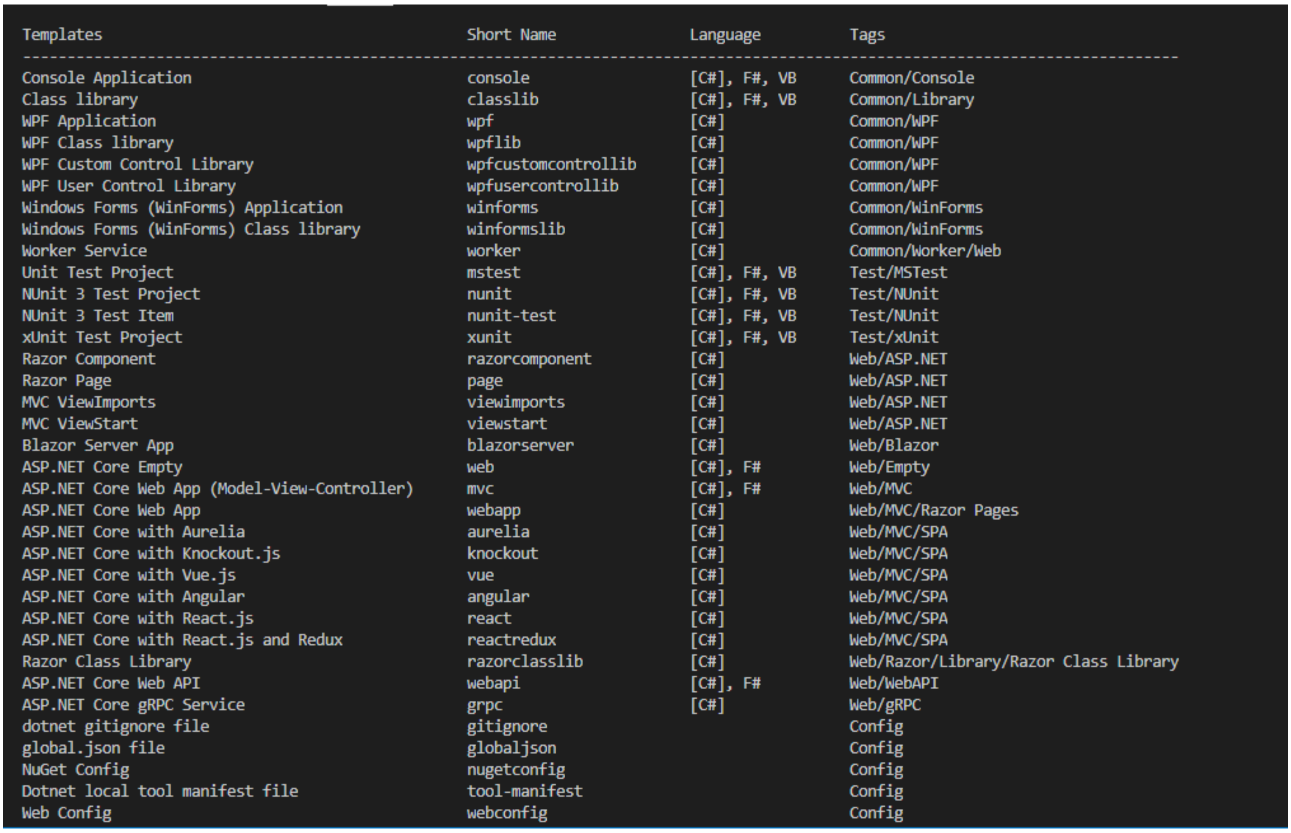


```
PS C:\Users\[redacted]> cd BookStore
PS C:\Users\[redacted]\BookStore>
```

Now, run the below command to download an Application templates.

=> dotnet new –install Microsoft.AspNetCore.SpaTer.plates..

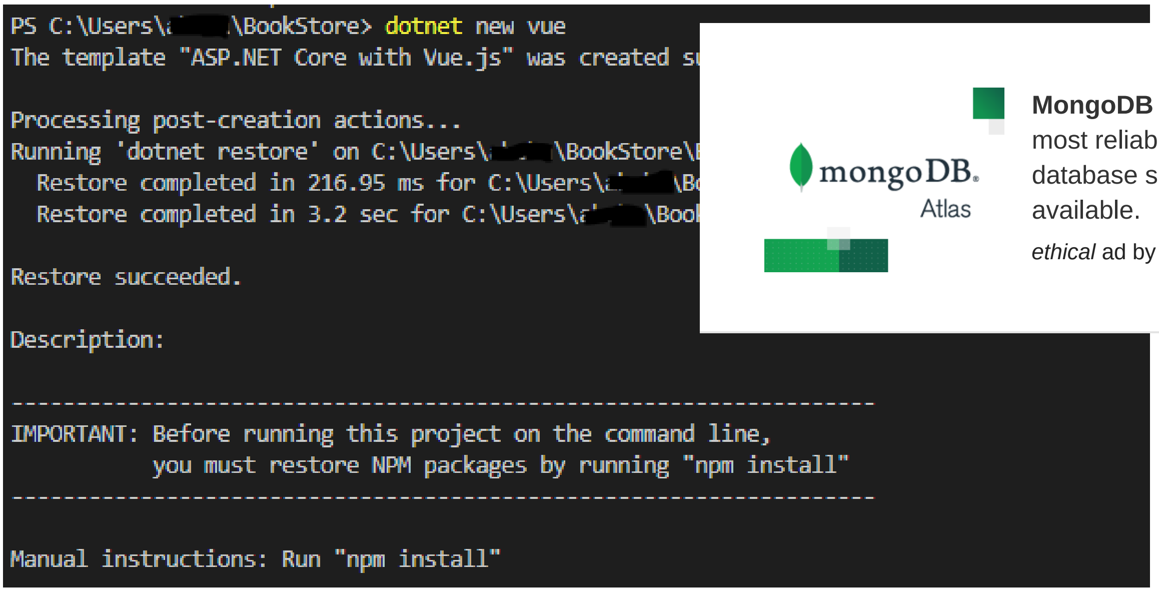
After running the above command, then you will see all the templates as you do see below in the screenshot.



```
Templates                                         Short Name               Language          Tags
-------------------------------------------------------------------------------------------------------------------------------------
Console Application                               console                  [C#], F#, VB      Common/Console
Class library                                     classlib                 [C#], F#, VB      Common/Library
WPF Application                                   wpf                      [C#]              Common/WPF
WPF Class library                                 wpflib                   [C#]              Common/WPF
WPF Custom Control Library                        wpfcustomcontrollib      [C#]              Common/WPF
WPF User Control Library                          wpfusercontrollib        [C#]              Common/WPF
Windows Forms (WinForms) Application              winforms                 [C#]              Common/WinForms
Windows Forms (WinForms) Class library            winformslib              [C#]              Common/WinForms
Worker Service                                    worker                   [C#]              Common/Worker/Web
Unit Test Project                                 mstest                   [C#], F#, VB      Test/MSTest
NUnit 3 Test Project                              nunit                    [C#], F#, VB      Test/NUnit
NUnit 3 Test Item                                 nunit-test               [C#], F#, VB      Test/NUnit
xUnit Test Project                                xunit                    [C#], F#, VB      Test/xUnit
Razor Component                                   razorcomponent           [C#]              Web/ASP.NET
Razor Page                                        page                     [C#]              Web/ASP.NET
MVC ViewImports                                   viewimports              [C#]              Web/ASP.NET
MVC ViewStart                                     viewstart                [C#]              Web/ASP.NET
Blazor Server App                                 blazorserver             [C#]              Web/Blazor
ASP.NET Core Empty                                web                      [C#], F#          Web/Empty
ASP.NET Core Web App (Model-View-Controller)      mvc                      [C#], F#          Web/MVC
ASP.NET Core Web App                              webapp                   [C#]              Web/MVC/Razor Pages
ASP.NET Core with Aurelia                         aurelia                  [C#]              Web/MVC/SPA
ASP.NET Core with Knockout.js                     knockout                 [C#]              Web/MVC/SPA
ASP.NET Core with Vue.js                          vue                      [C#]              Web/MVC/SPA
ASP.NET Core with Angular                         angular                  [C#]              Web/MVC/SPA
ASP.NET Core with React.js                        react                    [C#]              Web/MVC/SPA
ASP.NET Core with React.js and Redux              reactredux               [C#]              Web/MVC/SPA
Razor Class Library                               razorclasslib            [C#]              Web/Razor/Library/Razor Class Library
ASP.NET Core Web API                              webapi                   [C#], F#          Web/WebAPI
ASP.NET Core gRPC Service                         grpc                     [C#]              Web/gRPC
dotnet gitignore file                             gitignore                                  Config
global.json file                                  globaljson                                 Config
NuGet Config                                      nugetconfig                                Config
Dotnet local tool manifest file                   tool-manifest                              Config
Web Config                                        webconfig                                  Config
```

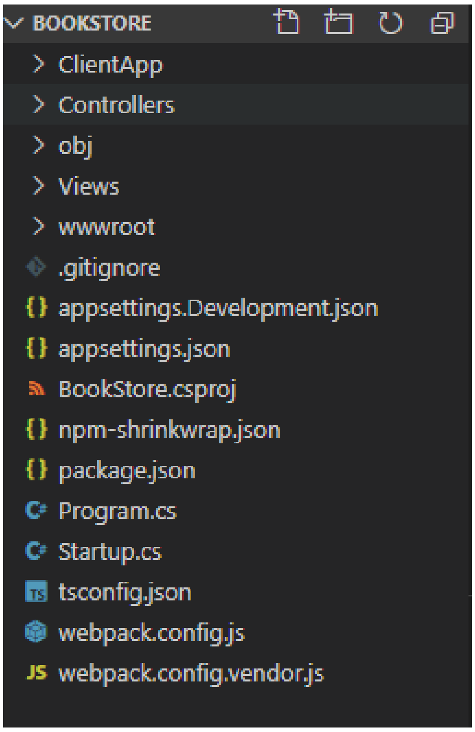
Now, run the below command to create a new project within your specific directory. It will name the application based on the folder that you are inside as you do see below in the screenshot.

=> dotnet new vue



```
PS C:\Users\[redacted]\BookStore> dotnet new vue
The template "ASP.NET Core with Vue.js" was created su

Processing post-creation actions...
Running 'dotnet restore' on C:\Users\[redacted]\BookStore\
  Restore completed in 216.95 ms for C:\Users\[redacted]\Bo
  Restore completed in 3.2 sec for C:\Users\[redacted]\Bool

Restore succeeded.

Description:

-------------------------------------------------------------------
IMPORTANT: Before running this project on the command line,
           you must restore NPM packages by running "npm install"
-------------------------------------------------------------------

Manual instructions: Run "npm install"
```

Now, open the project directory within the VS Code and then you will see the project folder structure as you do see below in the screenshot.

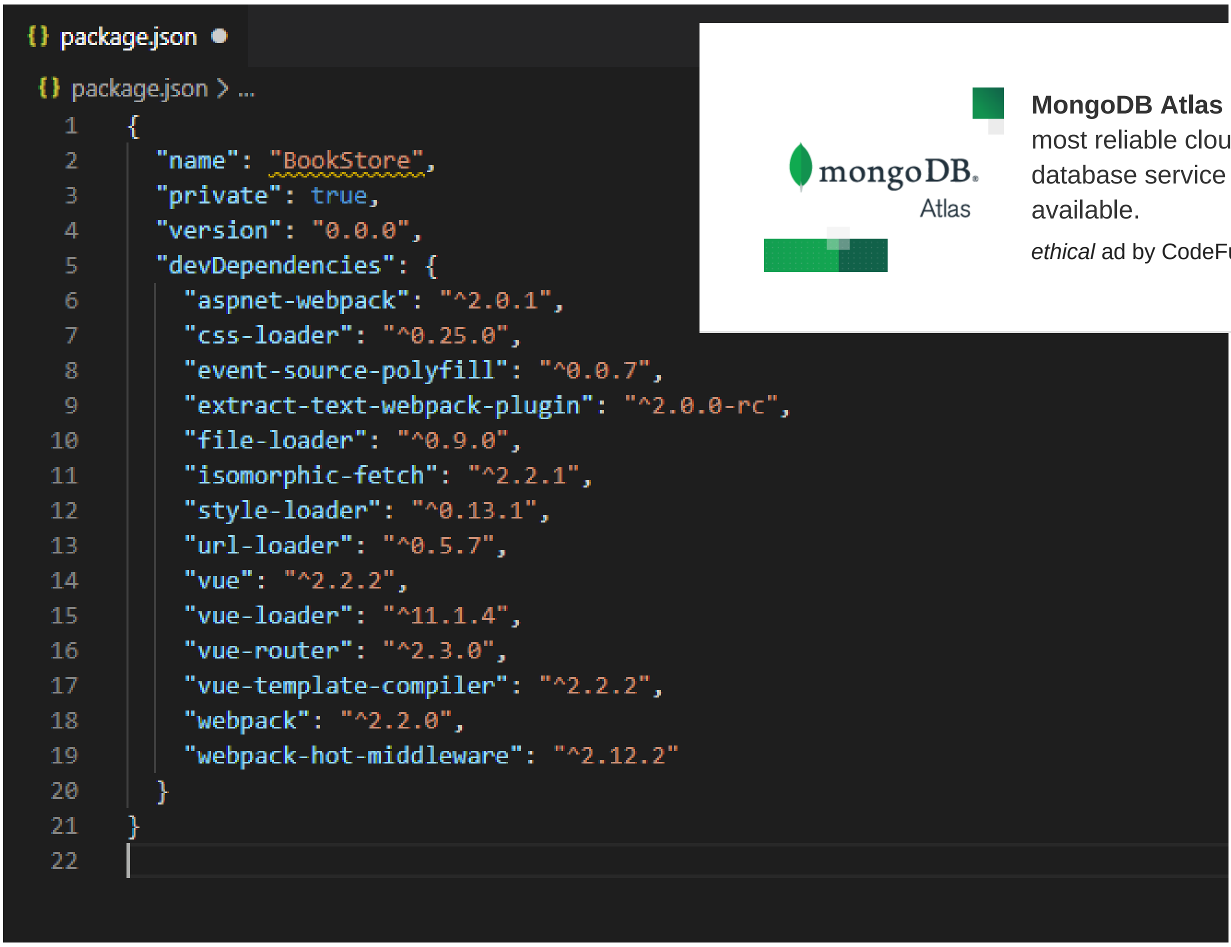


```
BOOKSTORE
  > ClientApp
  > Controllers
  > obj
  > Views
  > wwwroot
  .gitignore
  appsettings.Development.json
  appsettings.json
  BookStore.csproj
  npm-shrinkwrap.json
  package.json
  Program.cs
  Startup.cs
  tsconfig.json
  webpack.config.js
  webpack.config.vendor.js
```

Now, in this project we will not use typescript. So, we will remove all the dependencies of typescript from this project. To do this, go to **package.json** file and then remove these below packages. And then you will see the file as you do see below in the screenshot.

```
"@types/webpack-env": "^1.13.0",
"awesome-typescript-loader": "^3.0.0",
"bootstrap": "^3.3.6",
"jquery": "^3.1.1",
"typescript": "^2.2.1",
"vue-property-decorator": "^5.0.1"
```

removed packages - vue and asp.net core 3.1 integration hosted with ❤ by **GitHub** view raw



```
{
  "name": "BookStore",
  "private": true,
  "version": "0.0.0",
  "devDependencies": {
    "aspnet-webpack": "^2.0.1",
    "css-loader": "^0.25.0",
    "event-source-polyfill": "^0.0.7",
    "extract-text-webpack-plugin": "^2.0.0-rc",
    "file-loader": "^0.9.0",
    "isomorphic-fetch": "^2.2.1",
    "style-loader": "^0.13.1",
    "url-loader": "^0.5.7",
    "vue": "^2.2.2",
    "vue-loader": "^11.1.4",
    "vue-router": "^2.3.0",
    "vue-template-compiler": "^2.2.2",
    "webpack": "^2.2.0",
    "webpack-hot-middleware": "^2.12.2"
  }
}
```

Now, go to **tsconfig.json** file and then delete it. Then go to **webpack.config.js** file as you do see below file.

```
const path = require('path');
const webpack = require('webpack');
const ExtractTextPlugin = require('extract-text-webpack-plugin');
const CheckerPlugin = require('awesome-typescript-loader').CheckerPlugin;
const bundleOutputDir = './wwwroot/dist';

module.exports = (env) => {
    const isDevBuild = !(env && env.prod);

    return [{
        stats: { modules: false },
        context: __dirname,
        resolve: { extensions: [ '.js', '.ts' ] },
        entry: { 'main': './ClientApp/boot.ts' },
        module: {
            rules: [
                { test: /\.vue\.html$/, include: /ClientApp/, loader: 'vue-loader', option
                { test: /\.ts$/, include: /ClientApp/, use: 'awesome-typescript-loader?sil
                { test: /\.css$/, use: isDevBuild ? [ 'style-loader', 'css-loader' ] : Ext
                { test: /\.(png|jpg|jpeg|gif|svg)$/, use: 'url-loader?limit=25000' }
            ]
        },
23         output: {
24             path: path.join(__dirname, bundleO
25             filename: '[name].js',
26             publicPath: 'dist/'
27         },
28         plugins: [
29             new CheckerPlugin(),
30             new webpack.DefinePlugin({
31                 'process.env': {
32                     NODE_ENV: JSON.stringify(isDevBuild ? 'development' : 'production')
33                 }
34             }),
35             new webpack.DllReferencePlugin({
36                 context: __dirname,
37                 manifest: require('./wwwroot/dist/vendor-manifest.json')
38             })
39         ].concat(isDevBuild ? [
40             // Plugins that apply in development builds only
41             new webpack.SourceMapDevToolPlugin({
42                 filename: '[file].map', // Remove this line if you prefer inline source ma
43                 moduleFilenameTemplate: path.relative(bundleOutputDir, '[resourcePath]') /
44             })
45         ] : [
46             // Plugins that apply in production builds only
47             new webpack.optimize.UglifyJsPlugin(),
48             new ExtractTextPlugin('site.css')
49         ])
50     }];
51 };
```

webpack.config.js before update hosted with ❤ by **GitHub** **view raw**

Now, make some changes to remove typescript. First-of-all, remove this below line of code from line # 4

=> `const CheckerPlugin = require('awesome-typescript-loader').CheckerPlugin;`

Now, update line # 13 and 14 as you do see below

=> `resolve: { extensions: [ '.js' ] },`

=> `entry: { 'main': './ClientApp/boot.js' },`

Now, update the code of lines from 17 to 20 as you do see below

```
{ test: /\.vue$/, include: /ClientApp/, loader: 'vue-loader' },
2   { test: /\.css$/, use: isDevBuild ? [ 'style-loader', 'css-loader' ] : ExtractTextPlugin.ex
3   { test: /\.(png|jpg|jpeg|gif|svg)$/, use: 'url-
```

webpack.config.js updated - vue and asp.net core integration ho

Now, go to line # 29 and remove it. Then you will s
the file code.

```
1   const path = require('path');
2   const webpack = require('webpack');
3   const ExtractTextPlugin = require('extract-text-webpack-plugin');
4   const bundleOutputDir = './wwwroot/dist';
5
6   module.exports = (env) => {
7       const isDevBuild = !(env && env.prod);
8
9       return [{
10          stats: { modules: false },
11          context: __dirname,
12          resolve: { extensions: [ '.js' ] },
13          entry: { 'main': './ClientApp/boot.js' },
14          module: {
15              rules: [
16                  { test: /\.vue$/, include: /ClientApp/, loader: 'vue-loader' },
17                  { test: /\.css$/, use: isDevBuild ? [ 'style-loader', 'css-loader' ] : Ext
18                  { test: /\.(png|jpg|jpeg|gif|svg)$/, use: 'url-loader?limit=25000' }
19              ]
20          },
21          output: {
22              path: path.join(__dirname, bundleOutputDir),
23              filename: '[name].js',
24              publicPath: 'dist/'
25          },
26          plugins: [
27              new webpack.DefinePlugin({
28                  'process.env': {
29                      NODE_ENV: JSON.stringify(isDevBuild ? 'development' : 'production')
30                  }
31              }),
32              new webpack.DllReferencePlugin({
33                  context: __dirname,
34                  manifest: require('./wwwroot/dist/vendor-manifest.json')
35              })
36          ].concat(isDevBuild ? [
37              // Plugins that apply in development builds only
38              new webpack.SourceMapDevToolPlugin({
39                  filename: '[file].map', // Remove this line if you prefer inline source map
40                  moduleFilenameTemplate: path.relative(bundleOutputDir, '[resourcePath]') /
41              })
```

```
        ] : [
            // Plugins that apply in productio
            new webpack.optimize.UglifyJsPlugi
            new ExtractTextPlugin('site.css')
        ])
    }];
};
```

webpack.config.js updated - vue and asp.net core integration hosted with ❤ by **GitHub** **view raw**

Now, go to components folder inside ClientApp. And then delete all the components and then delete the **css** folder. Now, add a new component with the name of **App.vue.** Now, add write some code as you do see below in the code file.

```
<template>
    <div>
        <h1> {{  message }} </h1>
    </div>
</template>

<script>
export default {
    name: 'app',
    data() {
        return {
            message: "Vue.js and asp.net core 3.1 integration setup"
        }
    }
}
</script>
```

App.vue - Vue and Asp.Net Core 3.1 integration hosted with ❤ by **GitHub** **view raw**

Now, change the **boot.ts** file name to **boot.js** and then change the file content as you do see below in the code file.

```
import Vue from 'vue';
import VueRouter from 'vue-router';
Vue.use(VueRouter);

const routes = [
];

new Vue({
    el: '#app-root',
    router: new VueRouter({ mode: 'history', routes: routes }),
    render: h => h(require('./components/App.vue'))
});
```

boot.js - vue and asp.net core 3.1 integration hosted with ❤ by **GitHub**

Now, go to **_Layout.cshtml** file inside **Views/Share** references.

READ **Define one to many relationship in Entity**

Now, go to **csproj** file and then modify it from **line # 10** to **22** as you do see below in the file.

Note: – You will see Entity Framework Packages in this file. We will use them later in this tutorial.

```
1  <Project Sdk="Microsoft.NET.Sdk.Web">
2
3    <PropertyGroup>
4      <TargetFramework>netcoreapp3.1</TargetFramework>
5      <TypeScriptCompileBlocked>true</TypeScriptCompileBlocked>
6      <TypeScriptToolsVersion>Latest</TypeScriptToolsVersion>
7      <IsPackable>false</IsPackable>
8    </PropertyGroup>
9
10   <ItemGroup>
11     <PackageReference Include="Microsoft.AspNetCore.App" Version="2.2.8" />
12     <PackageReference Include="Microsoft.AspNetCore.SpaServices" Version="3.1.1" />
13     <PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="3.1.1">
14       <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</Inc
15       <PrivateAssets>all</PrivateAssets>
16     </PackageReference>
17     <PackageReference Include="Microsoft.EntityFrameworkCore.Tools.DotNet" Version="2.0.3"
18   </ItemGroup>
19
20   <ItemGroup>
21     <DotNetCliToolReference Include="Microsoft.VisualStudio.Web.CodeGeneration.Tools" Versi
22   </ItemGroup>
23
24   <Target Name="DebugRunWebpack" BeforeTargets="Build" Condition=" '$(Configuration)' == 'I
25     <!-- Ensure Node.js is installed -->
26     <Exec Command="node --version" ContinueOnError="true">
27       <Output TaskParameter="ExitCode" PropertyName="ErrorCode" />
28     </Exec>
29     <Error Condition="'$(ErrorCode)' != '0'" Text="Node.js is required to build and run thi
30
31     <!-- In development, the dist files won't exist on the first run or when cloning to
32          a different machine, so rebuild them if not already present. -->
33     <Message Importance="high" Text="Performing first-run Webpack build..." />
34     <Exec Command="node node_modules/webpack/bin/webpack.js --config webpack.config.vendor
35     <Exec Command="node node_modules/webpack/bin/webpack.js" />
```

```
  </Target>

  <Target Name="PublishRunWebpack" AfterTarget
    <!-- As part of publishing, ensure the JS
    <Exec Command="npm install" />
    <Exec Command="node node_modules/webpack/b
    <Exec Command="node node_modules/webpack/b

    <!-- Include the newly-built files in the publish output -->
    <ItemGroup>
      <DistFiles Include="wwwroot\dist\**" />
      <ResolvedFileToPublish Include="@(DistFiles->'%(FullPath)')" Exclude="@(ResolvedFile
        <RelativePath>%(DistFiles.Identity)</RelativePath>
        <CopyToPublishDirectory>PreserveNewest</CopyToPublishDirectory>
      </ResolvedFileToPublish>
    </ItemGroup>
  </Target>

</Project>
```

BookStore.csproj - Vue and Asp.Net Core 3.1 Integration hosted with ❤ by **GitHub** **view raw**

Now, go to the terminal and then run the below command to restore all the packages.

=> `dotnet restore`

Now, run the below command to install npm packages.

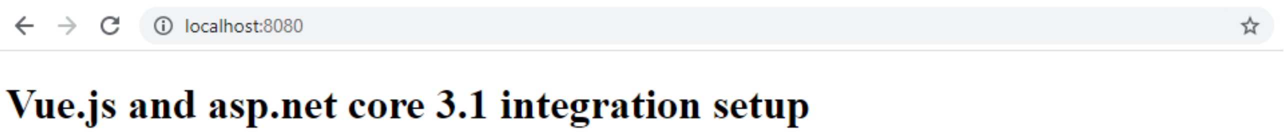
=> `npm install`

Now, run the below command to run the server side project.

=> `dotnet run`

Then open a second terminal of client app and then run the below command and then navigate to this url (http://localhost:8080/) and then you will see the output as you do see below in the screenshot.

=> `vue serve`



localhost:8080

Vue.js and asp.net core 3.1 integration setup

4- How to setup a database in Vu application?

Now, in this step, we will see how to setup a databa

We will use Entity Framework Core Identity to creat

Before going to the next, we need to add these b then run the below commands one by one to install the packages related to PostgreSQL and Entity Framework Core.

=> dotnet add package Microsoft.AspNetCore.Identity.EntityFrameworkCore --version 3.1.1

=> dotnet add package Npgsql.EntityFrameworkCore.PostgreSQL --version 3.1.0

=> dotnet add package Npgsql.EntityFrameworkCore.PostgreSQL.Design --version 2.0.0-preview1

So, go to project folder structure and then create a new folder with the name of "**Data**". We will add all the database related files within the "**Data**" folder. Now, right click on the **Data** folder and then add a new folder with the name of "**Entities**". Now, we will add all the classes inside the Entities folder. So, right click on the **Entities** folder and then add a new class with the name of **AppUser.cs** as you do see below in the code file.

```
using System.ComponentModel.DataAnnotations.Schema;
using Microsoft.AspNetCore.Identity;

namespace BookStore.Data.Entities
{
    public class AppUser : IdentityUser<int>
    {
        public string FirstName { get; set; }
        public string LastName { get; set; }

        [NotMapped]
        public string FullName
        {
            get {return $"{FirstName} {LastName}";}
        }
    }
}
```

AppUser.cs - Vue and Asp.Net Core 3.1 integration hosted with ❤ by **GitHub** **view raw**

Now, add a new class with the name of "**AppRole.cs**" as you do see below in the code file

```
using System.ComponentModel.DataAnnotations.Schema;
using Microsoft.AspNetCore.Identity;

namespace BookStore.Data.Entities
{
    public class AppRole : IdentityRole<int>
    {
        public AppRole() {}
        public AppRole(string name)
        {
            Name = name;
        }
    }
}
```

AppRole.cs - Vue and Asp.Net Core 3.1 Integration hosted with ❤ by **GitHub** **view raw**

Now, right on the **Data** folder and then add a context file with the name of "**BookStoreContext.cs**" and then write the code as you do see below in the file.

```
using BookStore.Data.Entities;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace BookStore.Data
{
    public class BookStoreContext : IdentityDbContext<AppUser, AppRole, int>
    {
        public BookStoreContext(DbContextOptions<BookStoreContext> options) : base(options
        {}
    }
}
```

BookStoreContext - Vue and Asp.Net Core 3.1 Integration hosted with ❤ by **GitHub** **view raw**

Now, we will add dependency injection for database context. So, go to project folder structure and then open the **startup.cs** file and then go to **ConfigureServices** method and then write the code as you do see below in the code file.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using BookStore.Data;
using BookStore.Data.Entities;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.SpaServices.Webpack;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection

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

        readonly string MyAllowSpecificOrigins = "_myAllowSpecificOrigins";
        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddCors(options =>
        {
            options.AddPolicy(MyAllowSpecificOrigins,
            builder =>
            {
                builder.WithOrigins("http://localhost:5000/",
                                    "http://localhost:8080/");
            });
        });
            services.AddDbContext<BookStoreContext>(options => options.UseNpgsql(Configura

            services.AddIdentity<AppUser, AppRole>()
            .AddEntityFrameworkStores<BookStoreContext>()
            .AddDefaultTokenProviders();
            services.AddMvc(options => options.EnableEndpointRouting = false);
        }

        // This method gets called by the runtime. Use this method to configure the HTTP r
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
                app.UseWebpackDevMiddleware(new WebpackDevMiddlewareOptions
                {
                    HotModuleReplacement = true
                });
            }
            else
            {
```

```
60                 app.UseExceptionHandler("/Home/Error");
61             }
62 
63             app.UseStaticFiles();
64             app.UseCors(MyAllowSpecificOrigins
65             app.UseMvc(routes =>
66             {
67                 routes.MapRoute(
68                     name: "default",
69                     template: "{controller=Home}/{action=Index}/{id?}");
70 
71                 routes.MapSpaFallbackRoute(
72                     name: "spa-fallback",
73                     defaults: new { controller = "Home", action = "Index" });
74             });
75         }
76     }
77 }
```

startup.cs - vue and asp.net core 3.1 integration hosted with ❤ by **GitHub** **view raw**

Let's understand the above code:

READ **Entity Framework Core 3.0 Cache Busting | Dotnet Detail**

Line # 39: here in this line, we are registering the database context with dependency injection container.

Line # 41: here in this line, we are registering the Asp.Net Core Identity service.

Now, we will add connection string for the database. So, go to project folder structure and then open the **appsettings.json** file and then add the connection string as you do see below in the file.

Note: – Make sure you have added the correct username and password for your database server.

```
{
  "ConnectionStrings": {
    "DefaultConnection": "host=localhost; database=BookStore;username=postgres;password= a
  },
  "Logging": {
    "LogLevel": {
      "Default": "Warning"
    }
    }
}
```

appsettings.json - vue and asp.net core 3.1 integration hosted wit

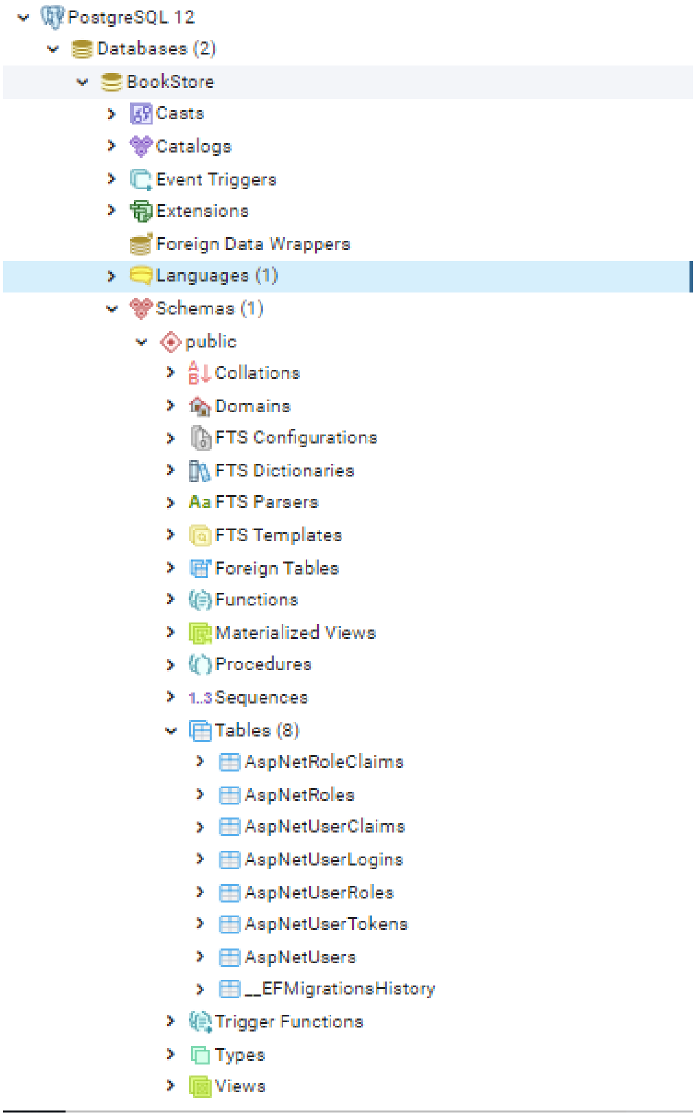
Now, add migration by running this below comm name of **Migrations** within the project folder structu

```
=> dotnet ef migrations add initial
```

Now, run the below command to create a database.

```
=> dotnet ef database update
```

After running successfully these above commands, now go to server and then you will see the output as you do see below in the screenshot.



```
PostgreSQL 12
  Databases (2)
    BookStore
      Casts
      Catalogs
      Event Triggers
      Extensions
      Foreign Data Wrappers
      Languages (1)
      Schemas (1)
        public
          Collations
          Domains
          FTS Configurations
          FTS Dictionaries
          FTS Parsers
          FTS Templates
          Foreign Tables
          Functions
          Materialized Views
          Procedures
          Sequences
          Tables (8)
            AspNetRoleClaims
            AspNetRoles
            AspNetUserClaims
            AspNetUserLogins
            AspNetUserRoles
            AspNetUserTokens
            AspNetUsers
            __EFMigrationsHistory
          Trigger Functions
          Types
          Views
```

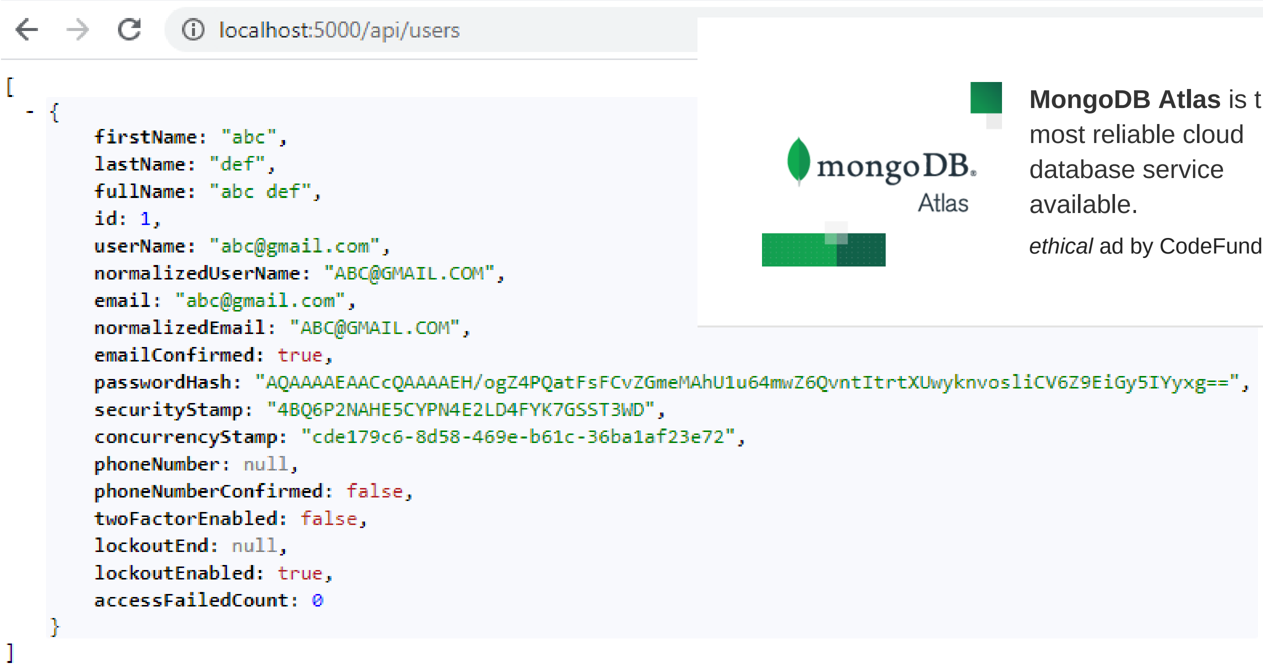
Now, go to project folder structure and then create a new file with the name of **UserController.cs** inside the **Controllers** folder. Now, write the code as you do see below in the file.

```
using System.Threading.Tasks;
using BookStore.Data;
using BookStore.Data.Entities;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

[Route("api/users")]
    public class UsersController : ControllerBase
    {
        private readonly BookStoreContext _db;
        private readonly UserManager<AppUser> _userManager;
        public UsersController(BookStoreContext db, UserManager<AppUser> userManager)
        {
            _db = db;
            _userManager = userManager;
        }
        [HttpGet]
        public async Task<IActionResult> GetUser()
        {
            if(_userManager.FindByEmailAsync("abc@gmail.com").GetAwaiter().GetResult() ==
            {
                var user = new AppUser
                {
                    FirstName = "abc",
                    LastName = "def",
                    UserName = "abc@gmail.com",
                    Email = "abc@gmail.com",
                    EmailConfirmed = true,
                    LockoutEnabled = false
                };
                _userManager.CreateAsync(user,"Abc@123").GetAwaiter().GetResult();
            }
            return Ok(await _db.Users.ToListAsync());
        }
    }
```

UserController - vue and asp.net core 3.1 integration hosted with ❤ by **GitHub** **view raw**

Here in the above **GetUser** function, we are creating a new user if it is not exists and then return the users list.

Now, run the project by running this command (**dotnet run**) and then navigate to the url (http://localhost:5000/api/users) and then you will see the output as you do see below in the screenshot.



```
[
  - {
        firstName: "abc",
        lastName: "def",
        fullName: "abc def",
        id: 1,
        userName: "abc@gmail.com",
        normalizedUserName: "ABC@GMAIL.COM",
        email: "abc@gmail.com",
        normalizedEmail: "ABC@GMAIL.COM",
        emailConfirmed: true,
        passwordHash: "AQAAAAEAACcQAAAAEH/ogZ4PQatFsFCvZGmeMAhU1u64mwZ6QvntItrtXUwyknvosliCV6Z9EiGy5IYyxg==",
        securityStamp: "4BQ6P2NAHE5CYPN4E2LD4FYK7GSST3WD",
        concurrencyStamp: "cde179c6-8d58-469e-b61c-36ba1af23e72",
        phoneNumber: null,
        phoneNumberConfirmed: false,
        twoFactorEnabled: false,
        lockoutEnd: null,
        lockoutEnabled: true,
        accessFailedCount: 0
    }
]
```

Congratulations! We have successfully integrated Vue and Asp.Net Core 3.1 application using visual studio code.

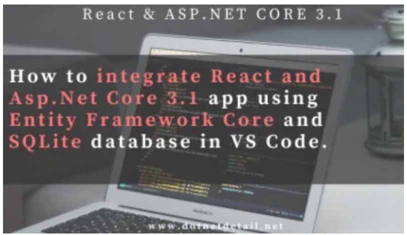
Note:- If you found this blog helpful then: Buy me a coffee

Thank you for reading. Please keep visiting and sharing it within your community.

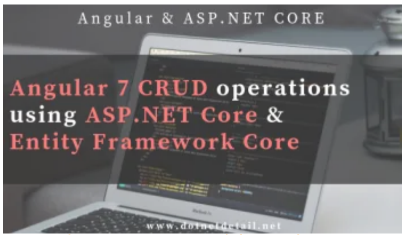
« **Document Viewer in Asp.Net Core 3.0 Application**

How to integrate React and Asp.Net Core 3.1 app using VS Code »

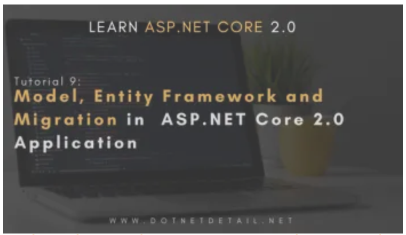
Related


React & ASP.NET CORE 3.1
How to integrate React and Asp.Net Core 3.1 app using Entity Framework Core and SQLite database in VS Code.
www.dotnetdetail.net

How to integrate React and Asp.Net Core 3.1 app using VS Code


Angular & ASP.NET CORE
Angular 7 CRUD operations using ASP.NET Core & Entity Framework Core
www.dotnetdetail.net

Angular 7 CRUD with web API using Asp.Net Core


LEARN ASP.NET CORE 2.0
Tutorial 9:
Model, Entity Framework and Migration in ASP.NET Core 2.0 Application
WWW.DOTNETDETAIL.NET

What is Entity Framework Core in ASP.NET Core 2.0

FILED UNDER: ASP.NET CORE, ENTITY FRAMEWORK, VUE
TAGGED WITH: .NET CORE 3.1, ASP.NET CORE 3.1, ENTITY FRAMEWORK CORE, POSTGRESQL, VUE

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

□ Notify me of follow-up comments by email.

□ Notify me of new posts by email.

Get Notifications ✖

POST COMMENT

Buy me a coffee

JOIN US

Dotnet Detail
1,337 likes

Like Page

Contact Us

Be the first of your friends to like this

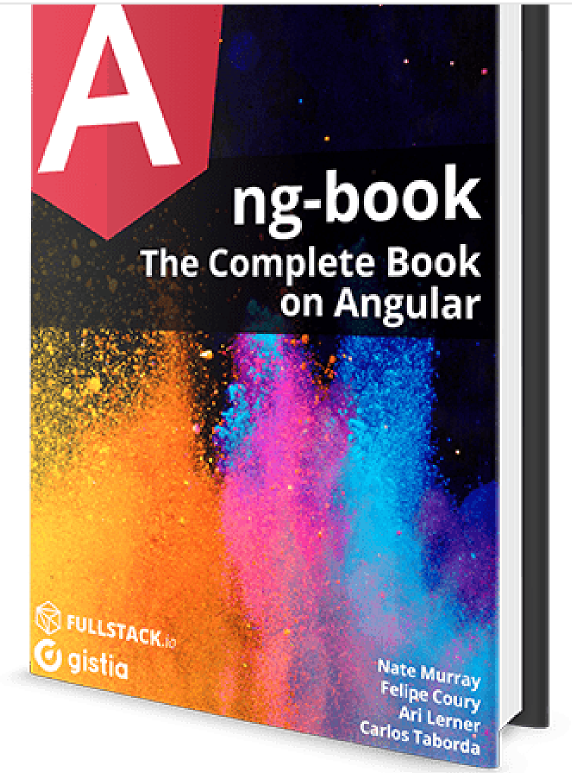
SUBSCRIBE TO BLOG VIA EMAIL

Enter your email address to subscribe.

Email Address

SUBSCRIBE

Get Notifications ✖



RECENT POSTS

- How to integrate React and Asp.Net Core 3.1 app using VS Code
- How to integrate Vue and Asp.Net Core 3.1 app using VS Code
- Document Viewer in Asp.Net Core 3.0 Application
- Role Based Authorization in Asp.Net Core 3.0
- How to use AutoMapper in Asp Net Core 3.0 App

Get Notifications

Asp.Net Core 3.0 Web API Versioning

.NET CORE .NET Core 2.0 .NET Core 3.0 .NET Core 3.1 .NET CORE MIDDLEWARE 2FA ANGULAR ANGULAR 5 ANGULAR 5 TUTORIAL Angular 6 Angular 6 Tutorial Angular 7 Angular 7 Tutorials ANGULAR AND .NET CORE ANGULAR TUTORIAL ASP.NET ASP.NET CORE ASP.NET CORE 2.0 ASP.NET Core 2.1 Asp.Net Core 3.0 Asp.Net Core 3.1 ASP.NET CORE MIDDLEWARE ASP.NET Core MVC ASP.NET Core Routing ASP.NET CORE TUTORIAL ASP.NET MVC6 ASP NET MVC Authentication Authentication and Authorization Azure Core Tutorial CRUD Deploy Asp.Net Core App to azure Deployment EF Core ENTITY FRAMEWORK Entity Framework Core Interview Questions JWT MVC 6 MVC COREL MVC CORE 2.0 Reactjs Two Factor Authentication Visual Studio 2019 WEB API

Get Notifications ✖