Single project that contain React plus ASP.NET Core MVC Application

[](https://i0.wp.com/www.srinadimpalli.com/wp-content/uploads/2019/12/blogpost_mainidea_image.jpg?ssl=1)

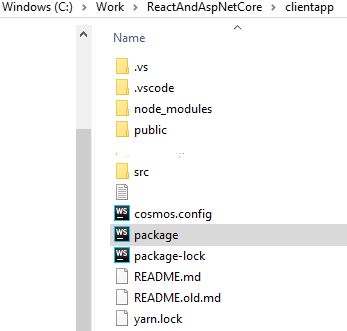
In this blog post, I will show how to create a project that contains ASP.NET Core MVC and React application, which means both parts of the projects can be developed using Visual Studio Code or Visual Studio 2019.

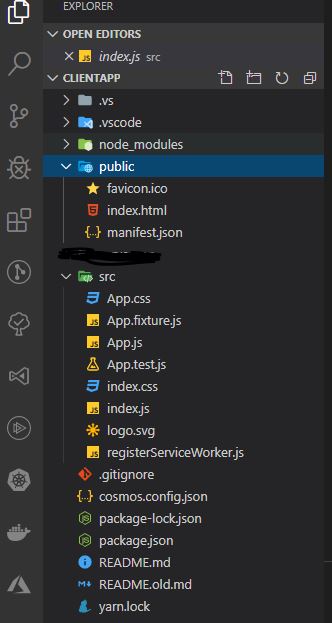
What is the context of combined projects?  
1. A combined project includes React and ASP.NET Core MVC in a single folder.  
2. A combined project makes it easy to develop both parts of an application using a single IDE.  
3. React application created first, followed by ASP.NET Core MVC. Additional NuGet packages are used when required as  
you progress.  
4. A combined project makes managing the development process easier. Still need good understanding of both React and  
ASP.NET Core MVC.

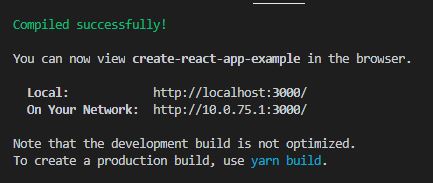
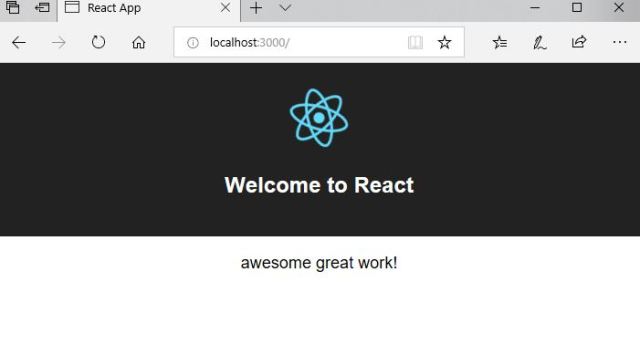
Preparing to Create a Project:  
Create a folder in your computer, you name it as you like, for example: ReactAspNetCore  
Open you command prompt and navigate to the folder ReactAspNetCore  
[https://i0.wp.com/www.srinadimpalli.com/wp-content/uploads/2019/12/rootfoler_command_path.jpg?resize=221%2C32&ssl=1](https://i0.wp.com/www.srinadimpalli.com/wp-content/uploads/2019/12/rootfoler_command_path.jpg?ssl=1)

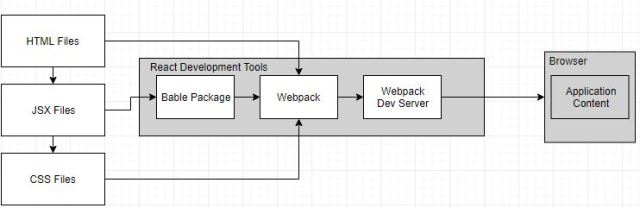
Before creating React project, you need to prepare for the React development environment and tools required to create the project.Check Node, if already node installed on your computer, you will see node version as below.  
Node -v  
v10.16.3 ( your computer might have different version)  
if not installed, go to [https://nodejs.org](https://nodejs.org/) and download the installer for Windows or macOS.  
npm -v  
6.9.0  
[Node installer includes the Node Package Manager (NPM)]  
There are two options to create React app, using create-react-app package, in this blog post I am using GitHub repository, the reason being is, its all configured and ready to add code. Click below link to download the create-react-app repository.

<https://github.com/rajunadimpalli/create-react-app-example>

once downloaded, unzip and copy into the “C:\Work\ReactAspNetCore” or whatever the path you have locally set.  
Open command prompt and run “npm install” to download and install the react dependencies locally on your computer.  
[](https://i0.wp.com/www.srinadimpalli.com/wp-content/uploads/2019/12/clientapp_explorer.jpg?ssl=1)  
Install an Editor: React development ca be done with any programmer’s editor. Visual Studio Code or Visual Studio 2019.  
[https://code.visualstudio.com](https://code.visualstudio.com/)  
or  
<https://visualstudio.microsoft.com/vs/> In this post, I am using Visual Studio Code.

Understanding the Project Structure:  
Open the clientapp folder using your preferred editor, my preferred editor – Visual Studio Code, you will see the project structure shown below.  
[](https://i0.wp.com/www.srinadimpalli.com/wp-content/uploads/2019/12/clientapp_projectstructure.jpg?ssl=1)  
Public/index.html – This is the HTML file that is loaded by the browser.  
Src/index.js – This is the JavaScript file that is responsible for configuring and starting the React Application.  
Src/App.js – This is the React Component, which contains the HTML content that will be displayed to the user.  
Use the command prompt to run the commands, navigate to the clientapp folder and start the React Development tools.  
[https://i0.wp.com/www.srinadimpalli.com/wp-content/uploads/2019/12/client_npmstart.jpg?resize=369%2C25&ssl=1](https://i0.wp.com/www.srinadimpalli.com/wp-content/uploads/2019/12/client_npmstart.jpg?ssl=1)

It can take a moment for the development tools to start and compile the project for its first use. Once the “compiled Successfully” message is shown. Open a new browser window and navigate to [http://localhost:3000](http://localhost:3000/)  
[](https://i0.wp.com/www.srinadimpalli.com/wp-content/uploads/2019/12/client_compileScussfull.jpg?ssl=1)  
Running React application  
[](https://i0.wp.com/www.srinadimpalli.com/wp-content/uploads/2019/12/client_runcompilesuccess.jpg?ssl=1)

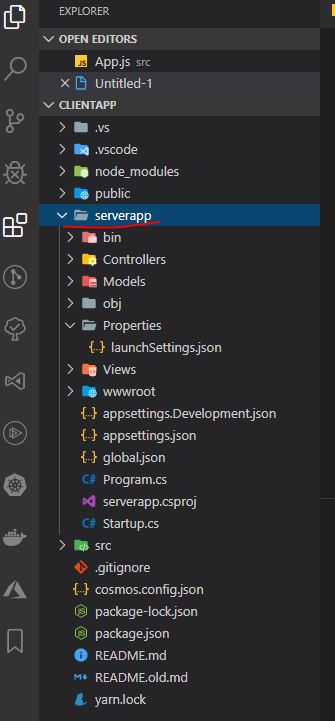
Here is the React Development tool chain in high level overview.  
[](https://i0.wp.com/www.srinadimpalli.com/wp-content/uploads/2019/12/react_development_toolprocess.jpg?ssl=1)

Creating the ASP.NET Core MVC part of the project:  
Once the React project has been set up, next step is to create an ASP.NET Core project. Use command prompt to run the commands as shown.

C:\Work\ReactAndAspNetCore\clientapp> mkdir serverapp  
C:\Work\ReactAndAspNetCore\clientapp> cd serverapp  
C:\Work\ReactAndAspNetCore\clientapp\serverapp> dotnet new globaljson –sdk-version 3.1

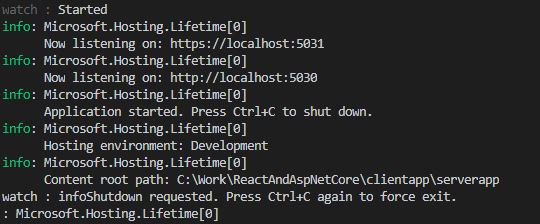
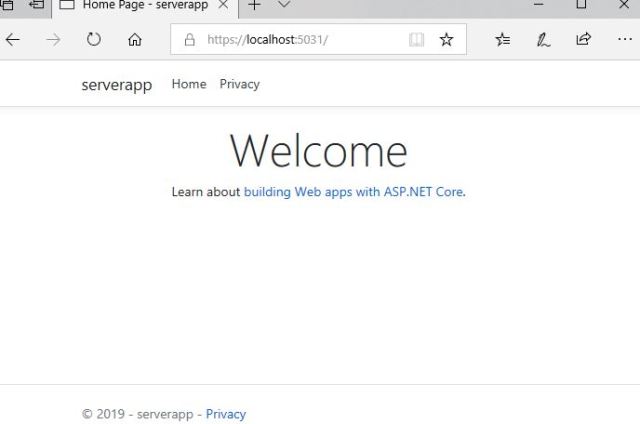
Above commands create clientapp/serverapp folder, which will contain the ASP.NET Core MVC project. The dotnet new globaljson command creates file named global.json that specifies the version of the .NET Core runtime that will be used, which will help ensure the projects work as expected.

Now run the command as shown below to create the ASP.NET Core MVC project.  
PS C:\Work\ReactAndAspNetCore\clientapp\serverapp> dotnet new mvc –language C# –auth None  
The dotnet new command adds all the files required for the basic ASP.NET Core MVC Project to the serverapp folder., alongside the React project created in the previous.

[](https://i0.wp.com/www.srinadimpalli.com/wp-content/uploads/2019/12/serverapp_filestructure.jpg?ssl=1)  
If visual studio prompted to add the C# extension, click yes and install the extension.

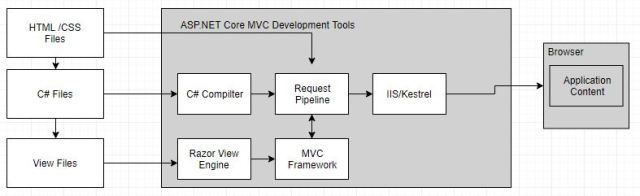
Building and Running the ASP.NET Core MVC Application:

The ASP.NET Core MVC part of the project can be compiled and executed from the command line or using the code editor. To build and run the project from the command line, run the command shown in  
in the serverapp folder.

PS C:\Work\ReactAndAspNetCore\clientapp\serverapp> dotnet watch run  
The ASP.NET Core MVC project will be compiled, and the runtime will start, producing the following output.The output reports the ports that are used to listen for HTTP and HTTPS requests. Open the browser and navigate to [https://localhost:5031](https://localhost:5031/), you will the output as below.  
[](https://i0.wp.com/www.srinadimpalli.com/wp-content/uploads/2019/12/serverapp_compilesuccess.jpg?ssl=1)  
[](https://i0.wp.com/www.srinadimpalli.com/wp-content/uploads/2019/12/serverapp_runtimeoutput.jpg?ssl=1)

Understanding the ASP.NET Core MVC Toolchain:

The development tools for ASP.NET Core MVC are entirely different from those used in React development. When the application is started, HTTP requests received by Internet Information Services (IIS) or the stand-alone ASP.NET Core Kestrel server and handled by the ASP.NET Core request pipeline, which is the backbone of ASP.NET Core Applications and allow MVC applications, REST API services, other frameworks to coexist in the same application.

[](https://i0.wp.com/www.srinadimpalli.com/wp-content/uploads/2019/12/aspnetcoremvc_toolchain.jpg?ssl=1)  
The request pipeline is the heart of the ASP.NET Core and is build around the series of middleware components that are able to inspect incoming requests and either modify them or produce results.

Connecting the React and ASP.NET Core Applications:

React and ASP.NET Core MVC applications share the same parent folder but are not connected in any way. It is possible to develop applications this way. A more useful approach is to connect the two toolchains so that HTTP requests are received by ASP.NET Core and passed on to either the MVC framework or the React development tools based on the URL.

In order to connect the two application tools chains, we need to add below package to ASP.NET Core  
Navigate to PS C:\Work\ReactAndAspNetCore\clientapp\serverapp> and run the command as below  
PS C:\Work\ReactAndAspNetCore\clientapp\serverapp> dotnet add package Microsoft.AspNetCore.SpaServices.Extensions –version 3.1.0. The Microsoft.AspNetCore.SpaServices.Extensions package is provided by Microsoft to allow single-page application (SPA) frameworks to be used with ASP.NET Core.

The most commonly used connection technique is to configure the ASP.NET Core runtime so that it starts the React development server when the first HTTP request is received. Add the code below into Startup.cs class to configure ASP.NET Core so that it is responsible for managing the React development server.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  using Microsoft.AspNetCore.Builder;  using Microsoft.AspNetCore.Hosting;  using Microsoft.AspNetCore.HttpsPolicy;  using Microsoft.Extensions.Configuration;  using Microsoft.Extensions.DependencyInjection;  using Microsoft.Extensions.Hosting;  &amp;amp;amp;lt;strong&amp;amp;amp;gt;using Microsoft.AspNetCore.SpaServices.ReactDevelopmentServer;&amp;amp;amp;lt;/strong&amp;amp;amp;gt;    namespace serverapp  {  public class Startup  {  public Startup(IConfiguration configuration)  {  Configuration = configuration;  }  public IConfiguration Configuration { get; }  // This method gets called by the runtime. Use this method to add services to the container.  public void ConfigureServices(IServiceCollection services)  {  services.AddControllersWithViews();  }  // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.  public void Configure(IApplicationBuilder app, IWebHostEnvironment env)  {  if (env.IsDevelopment())  {  app.UseDeveloperExceptionPage();  }  else  {  app.UseExceptionHandler("/Home/Error");  app.UseHsts();  }  app.UseHttpsRedirection();  app.UseStaticFiles();    app.UseRouting();    app.UseAuthorization();    app.UseEndpoints(endpoints =&amp;amp;amp;gt;  {  endpoints.MapControllerRoute(  name: "default",  pattern: "{controller=Home}/{action=Index}/{id?}");  });  app.UseSpa(spa =&amp;amp;amp;gt; {  spa.Options.SourcePath="../src";  spa.UseReactDevelopmentServer("start");  });  }  }  } |

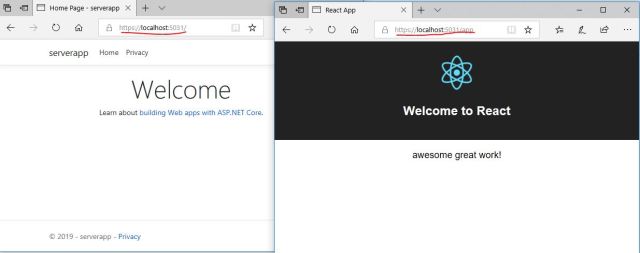
After adding above code , you need to build and run the server app, so navigate to command prompt and run below command.

C:\Work\ReactAndAspNetCore\clientapp\serverapp> dotnet watch run

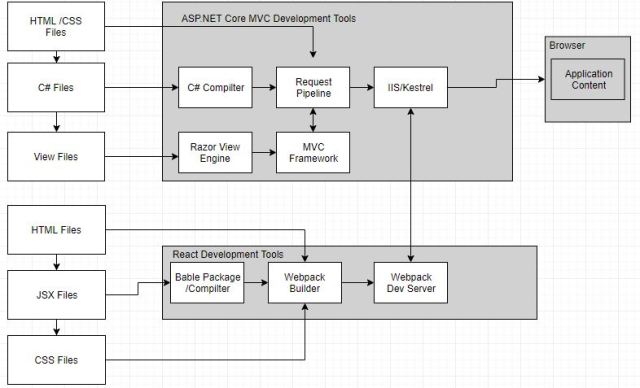
Open a web browser and navigate to [https://localhost:5031](https://localhost:5031/); you will see the content generated by the MVC Framework, as shown below.Next navigate to [https:/localhost:5031/app](https://www.srinadimpalli.com/localhost:5031/app); you will see the output from the React application, as shown below.  
The first URL is handled by the MVC Framework because the default URL is translated into a request for the Index action on the Home controller by this statement.

|  |
| --- |
| app.UseEndpoints(endpoints =&amp;amp;amp;gt;  {  endpoints.MapControllerRoute(  name: "default",  pattern: "{controller=Home}/{action=Index}/{id?}");  }); |

The app segment of the second URL (https:localhost:5031/app) doesn’t correspond to the name of an MVC controller, so the MVC framework can’t handle the request, which is then forwarded to the React Development Server. See below screenshot.

[](https://i0.wp.com/www.srinadimpalli.com/wp-content/uploads/2019/12/final_bothserverandclientbrowser.jpg?ssl=1)

Behind the scenes, The ASP.NET Core middleware start the React Development Server and forwards HTTP Requests that are not handled by the middleware components, such as MVC framework as below screenshot.

[](https://i0.wp.com/www.srinadimpalli.com/wp-content/uploads/2019/12/usingseprateserver_development.jpg?ssl=1)

Each time the ASP.NET Core runtime restarts, The React development server will also be restarted and won’t be able to respond to requests until after the initial compilation phase has completed.  
User Control+C to stop the ASP.NET Core runtime.

Using the ASP.NET Core MVC Proxy Feature:

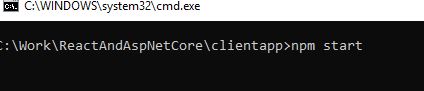
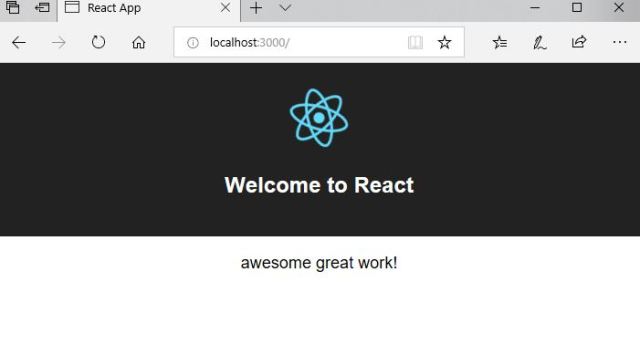
The alternative connection technique is to start the React Development Tools separately from the ASP.NET Core runtime. Requests are still forwarded to the React server, but the ASP.NET Core Runtime isn’t responsible for starting the React tools. In this approach the restarts are faster, which can create a smoother flow during iterative development. Each part of the project responds independently, so that a change in C# class, doesn’t affect the React Development Server. Only drawback is, you need to run two commands and monitor two streams of output.  
Before modifying the Startup.cs file, you need to add a new configuration section under appsettings.Development.json file, add below statements.

“DevTools”: {  
“ConnectionStrategy”: “proxy”  
}

|  |
| --- |
| To change alternative connections, you need to change the Startup.cs file, add below statements.    {  endpoints.MapControllerRoute(  name: "default",  pattern: "{controller=Home}/{action=Index}/{id?}");  });  app.UseSpa(spa =&amp;amp;amp;gt;  {  string strategy = Configuration.GetValue&amp;amp;amp;lt;string&amp;amp;amp;gt;("DevTools:ConnectionStrategy");  if (strategy == "proxy")  {  spa.UseProxyToSpaDevelopmentServer("<http://localhost:3000/>");  }  else if (strategy == "managed")  {  spa.Options.SourcePath = "../src";  spa.UseReactDevelopmentServer("start");  }  }); |

The configuration object is used to read the value of ConnectionStrategy in the DevTools section. If the setting is managed, then the technique from the previous section, if the setting is “proxy”, then the useSpa method is used to specify the URL of the React Development server as an argument to the  
spa.UseProxyToSpaDevelopmentServer(“http://localhost:3000/“);

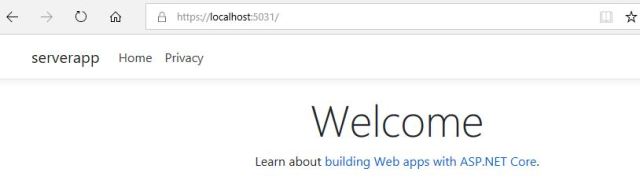
To start the React development server, open the new command prompt, navigate to clientapp folder and run the command shown below to start the React development server.

[](https://i0.wp.com/www.srinadimpalli.com/wp-content/uploads/2019/12/seprate_coonection_react_commandprompt.jpg?ssl=1)  
[](https://i0.wp.com/www.srinadimpalli.com/wp-content/uploads/2019/12/client_runcompilesuccess-1.jpg?ssl=1)

Starting the ASP.NET Core server

dotnet watch run

To start the ASP.NET Core server, open a second command prompt , navigate to clientapp/serverapp folder  
PS C:\Work\ReactAndAspNetCore\clientapp\serverapp> dotnet watch run

Now open browser and navigate to “https:localhost:5021/” URL.  
[](https://i0.wp.com/www.srinadimpalli.com/wp-content/uploads/2019/12/seprate_connection_aspnetcore_runbrowser.jpg?ssl=1)

Once both servers are running, navigate to https://localhost:5031/app URL, you will see the same results as in the previous section because requests are being forwarded in the same way. Difference is that React development server started separately.  
That’s all for now.