# Bring ASP.NET to your Mac

We spent some time for JavaScript and TypeScript, and I want to devote some time for C# which is very popular among web developers as well. Of course, since we are talking about web development and C#, ASP.NET is our friend there. Traditionally, ASP.NET is a part of .NET Framework and woks on Windows platform only. But everything is changing quickly and today we will discuss how to bring ASP.NET to your Mac and use Visual Studio Code to create the first web sites on this runtime.

First of all, we need to install .NET Version Manager that will help us to manage .NET Environments later. To install .NET Version Manager or DNVM you need to execute the following command:

**curl -ssl https://raw.githubusercontent.com/aspnet/Home/dev/dnvminstall.sh | DNX\_BRANCH=dev sh && source ~/.dnx/dnvm/dnvm.sh**

if you have a message that .bash profile is not found, you need to create one and restart the command above. To create a new bash profile you can execute the following commands:

**cd ~/**

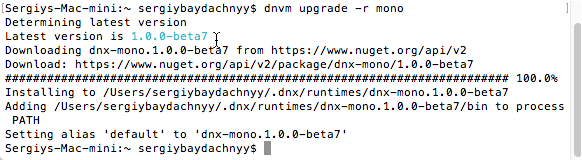
**touch .bash\_profile**

Once the Version Manager is installed you can use **dnvm** tool to install .NET Execution Environment (DNX) for running asp.net 5.0 applications.

Pay special attention that DNX is based on open source implementation of .NET Framework and there are two of them: mono and CoreCLR. And DNX support both of them, so you can install DNX based on Mono or DNC based on CoreCLR. Because CoreCLR is in preview only and is not recommended for production we will use Mono. Additionally, you can use Mono for many different things and we already discussed how to use it to debug Unity projects.

Therefore, if you still don’t have Mono, you can install it from <http://www.mono-project.com> and right after that you can install the execution environment:

**dnvm upgrade -r mono**



Everything is done and we can start working with ASP.NET projects.

So, I propose to start with a simple console application. You can copy some code from official tutorial using the following link <http://docs.asp.net/en/latest/dnx/console.html>. Let’s open a new empty folder, create a C# file there and add the following code:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace ConsoleApp

{

public class Program

{

public void Main(string[] args)

{

Console.WriteLine("Hello from Mac");

}

}

}

You can see that Visual Studio Code doesn’t support full IntelliSense because we didn’t provide any information about the project. In case of ASP.NET we need to work with project.json file. I think that you can try to create sln and csproj files but I don’t think that I can do it by hand. But in case of project.json you can use the following code, that is not very complex:

{

"version": "1.0.0-\*",

"description": "Console Application",

"commands": {

"ConsoleApp": "ConsoleApp"

},

"frameworks": {

"dnx451": { },

"dnxcore50": {

"dependencies": {

"Microsoft.CSharp": "4.0.1-beta-23225",

"System.Collections": "4.0.11-beta-23225",

"System.Console": "4.0.0-beta-23225",

"System.Linq": "4.0.1-beta-23225",

"System.Threading": "4.0.11-beta-23225"

}

}

}

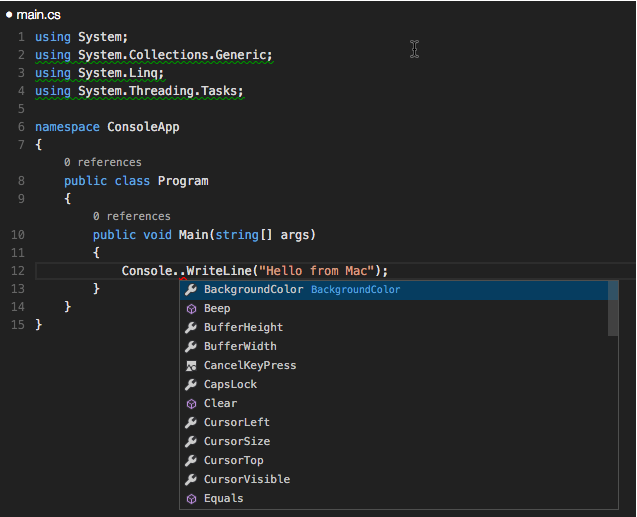
}

Once you create this file, Visual Studio Code will start reading information about the project and will try to restore dependencies that you provided in the file. To notify you about it Visual Studio Code will display the following message:

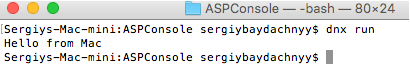


Just click Restore and it’s done.

Right now you can see that IntelliSense system works fine and you can work in friendly environment:



So, open the Terminal window in the context of your folder and use dnx run to start the project:



Congratulations, you build the first .NET console application on your Mac.

Let’s try to create the first web application. In order to do it you can start with Startup.cs file that you can place to the root of your project. Let’s add the following code there:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNet.Builder;

using Microsoft.AspNet.Http;

using Microsoft.Framework.DependencyInjection;

namespace ConsoleApp

{

public class Startup

{

public void Configure(IApplicationBuilder app)

{

app.Run(async (context) =>

{

await context.Response.WriteAsync("Hello World!");

});

}

}

}

This class provides the entry point for our web application. Usually we need to configure the application there but we are going to develop a very simple application, so we will simply run the application and return Hello World to the given context.

Our project.json will look a little bit different:

{

"webroot": "wwwroot",

"version": "1.0.0-\*",

"dependencies": {

"Microsoft.AspNet.Server.IIS": "1.0.0-beta7",

"Microsoft.AspNet.Server.WebListener": "1.0.0-beta7",

"Microsoft.AspNet.Server.Kestrel": "1.0.0-beta7"

},

"commands": {

"kestrel": "Microsoft.AspNet.Hosting --server Microsoft.AspNet.Server.Kestrel --server.urls http://localhost:5000",

"web": "Microsoft.AspNet.Hosting --server Microsoft.AspNet.Server.WebListener --server.urls http://localhost:5001"

}

}

We need to include basic libraries there including local hosting service Kestrel. Once we do it, VS Code will restore all needed references and you can run the web application. In order to do it type the following command:

dnx kestrel



And that’s all. Right now you can open your browser and navigate to <http://localhost:5000>. You should see Hello World on the screen:

