# ASP.NET на Вашем Mac

В предыдущих разделах мы обсуждали JavaScript и TypeScript, но сейчас я хотел бы вернуться к C#, который также очень популярен среди web разработчиков. Конечно, когда мы говорим о C#, то речь идет о технологии ASP.NET. Традиционно ASP.NET является составной частью .NET Framework и работает на Windows. Но сегодня мы видим много изменений в этом направлении и сейчас мы поговорим о том, как приступить к разработке ASP.NET приложений прямо на Вашем Mac.

Первым делом нам необходимо установить .NET Version Manager, который позволит управлять .NET окружением на нашем Mac. Чтобы установить .NET Version Manager (DNVM) Вам необходимо выполнить следующую команду:

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

Если при выполнении этой команды Вы получили сообщение о том, что не найден .bash профиль, то последний можно создать с помощью команд ниже, после чего запустить установку заново:

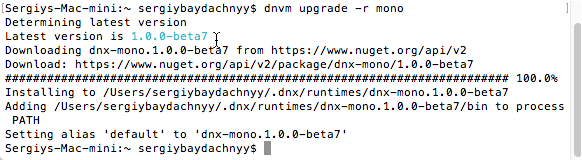
**cd ~/**

**touch .bash\_profile**

Как только менеджер версий установлен, Вы можете использовать dnvm для установки .NET окружения (.NET Execution Environment - DNX).

Обратите внимание, что DNX базируется на открытой реализации .NET Framework и тут есть возможность выбрать между coreCLR и Mono. Поскольку CoreCLR находится все ее в разработке, да и VS Code не поддерживает его должным образом, мы будем использовать Mono. Следовательно, если у Вас все еще не установлен Mono, то выполните установку с сайта <http://www.mono-project.com> и приступайте к установке DNX:

**dnvm upgrade -r mono**



Как только установка завершена, мы можем приступить к созданию нашего первого проекта.

Чтобы не тратить много времени, я предлагаю начать с руководства, которое находится по адресу <http://docs.asp.net/en/latest/dnx/console.html>. Создадим пустой файл на C# и скопируем туда следующий код:

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");

}

}

}

Как Вы видите Visual Studio Code не поддерживает IntelliSense систему должным образом, так как мы не предоставили информацию о нашем проекте. В случае с ASP.NET мы можем воспользоваться project.json файлом, который в общем случае может выглядеть следующим образом:

{

"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"

}

}

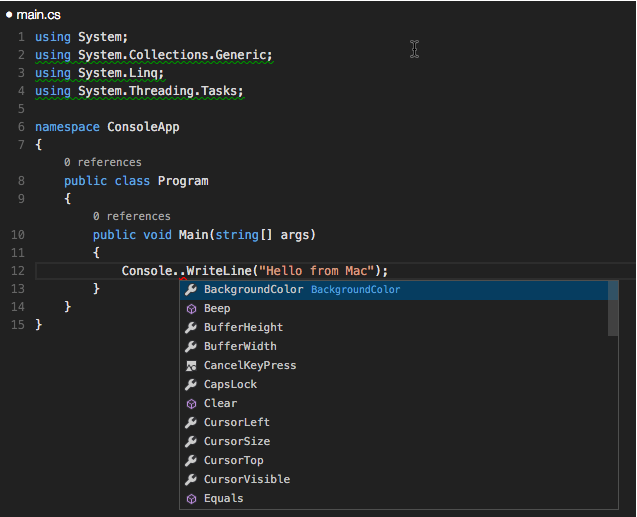
}

}

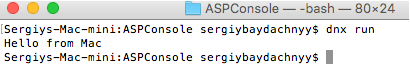
Как только мы создали файл проекта, Visual Studio Code отобразит следующее сообщение:



Принимая предложение, Вы позволите Code установить все зависимости. Как только это сделано, Вы получите привычное дружественное окружение:



Откройте терминал и выполните команду **dnx run**, чтобы запустить проект:



Поздравляем, наше первое приложение на ASP.NET готово. Но хотелось бы попробовать создать и небольшое web приложение. Для этого создайте Startup.cs файл и разместите там следующий код:

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!");

});

}

}

}

Этот класс является точкой входа для нашего web приложения. Обычно тут Вы можете найти много кода, который конфигурирует приложение, но в нашем случае мы хотим реализовать что-то простое, поэтому просто будем запускать приложение, которое возвращает в контекст текстовое сообщение.

Модифицируем project.json следующим образом:

{

"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"

}

}

Тут мы добавили основные библиотеки, включая зависимость на локальный веб сервер Kestrel. Как только VS Code загрузит все зависимые пакеты, Вы можете запустить приложение с посмощью следующей команды:

dnx kestrel



Как только приложение запущено. Попробуйте открыть браузер и перейти по ссылке <http://localhost:5000>. Если все прошло нормально, то на экране появится следующее сообщение:

