Using VS Code for Building .NET Core Applications: A Step By Step Guide

Prerequisites

- **Install .NET Core SDK**: Download and install the .NET Core SDK from the [.NET website](https://dotnet.microsoft.com/download).
- **Install Visual Studio Code**: Download and install VS Code from the [VS Code website](https://code.visualstudio.com/).
- **Install C# Extension for VS Code**: Open VS Code, go to the Extensions view by clicking the square icon on the sidebar or pressing `Ctrl+Shift+X`, and search for 'C#'. Install the extension provided by Microsoft.

Step-by-Step Guide

Step 1: Create a New .NET Core Project

Open a terminal in VS Code by clicking on `Terminal > New Terminal` or pressing ``Ctrl+` ``.

Navigate to the directory where you want to create your project.

Run the following command to create a new .NET Core console application:

```sh
 dotnet new console -n MyFirstApp
```

This will create a new directory called `MyFirstApp` with the necessary files.

Step 2: Open the Project in VS Code

Open the project folder in VS Code by clicking `File > Open Folder` and selecting the `MyFirstApp` directory.

VS Code should automatically detect the project and prompt you to add required assets for building and debugging. Click 'Yes' to add these.

Step 3: Explore the Project Structure

- **Program.cs**: This file contains the entry point of your application.
- **MyFirstApp.csproj**: This is the project file that contains information about the project and its dependencies.

Step 4: Run the Application

Open `Program.cs` and you should see the default 'Hello World' code:

```csharp
 using System;
```

```
namespace MyFirstApp
{
class Program
{
static void Main(string[] args)
{
Console.WriteLine("Hello World!");
}
}
}
```

Run the application by opening the terminal and executing:

```sh
dotnet run
```

You should see the output 'Hello World!' in the terminal.

Step 5: Debug the Application

Set a breakpoint by clicking in the left margin next to the `Console.WriteLine("Hello World!");` line in `Program.cs`.

Start debugging by pressing `F5` or going to `Run > Start Debugging`.

The debugger will stop at the breakpoint, allowing you to inspect variables and step through the code.

Step 6: Add a NuGet Package

Open a terminal and navigate to your project directory.

Run the following command to add a NuGet package, for example, `Newtonsoft.Json`:

```sh
dotnet add package Newtonsoft.Json
```

Restore the dependencies by running:

```sh
dotnet restore
```

Step 7: Build the Application

You can build the application without running it by executing:

```sh
 dotnet build
```

This will compile the application and generate the necessary binaries in the `bin` directory.

Step 8: Create a Class Library

To add a class library to your solution, run the following command:

```sh
 dotnet new classlib -n MyLibrary
```

Add the class library as a dependency to your console application by editing the `MyFirstApp.csproj` file and adding:

```xml
 <ProjectReference Include="..\MyLibrary\MyLibrary.csproj" />
```

Restore the dependencies again:

```sh
 dotnet restore
```

Step 9: Add Unit Tests

Create a new xUnit test project by running:

```sh
 dotnet new xunit -n MyFirstApp.Tests
```

Add the test project as a dependency to your solution by editing the `MyFirstApp.Tests.csproj` file:

```xml
 <ProjectReference Include="..\MyFirstApp\MyFirstApp.csproj" />
```

Write a simple test in `UnitTest1.cs`:

```csharp
 using System;
 using Xunit;

 namespace MyFirstApp.Tests
```

```
{
public class UnitTest1
{
[Fact]
public void Test1()
{
Assert.Equal(4, 2 + 2);
}
}
}
```

Run the tests using:

```sh
dotnet test
```

Step 10: Publish the Application

Publish your application to prepare it for deployment:

```sh
dotnet publish -c Release -o ./publish
```

This will compile the application and place the output in the `publish` directory.

Tips

- **IntelliSense**: Use IntelliSense to get code suggestions and documentation as you type.
- **Extensions**: Explore other useful extensions like `Debugger for Chrome` for web development, `Azure Tools` for cloud integration, etc.
- **Shortcuts**: Learn VS Code shortcuts to improve your productivity.