**Creating a console application in .NET**

**Step 1: Install .NET SDK**

Firstly, ensure you have the .NET SDK installed on your machine. You can download it from Dot Net.

**Step 2: Open a Terminal or Command Prompt**

Open your terminal or command prompt or **VSCode** where you want to create your project.

**Step 3: Create a New Console Application**

To create a new console application, use the following command:

**dotnet new console -o MyConsoleAppName**

**cd MyConsoleApp**

This command does the following:

* **dotnet new console**: Creates a new console application template.
* **-o** **MyConsoleAppName**: Specifies the output directory with the name MyConsoleApp (you can replace **MyConsoleAppName** with your desired project name).
* **Cd -** Navigate into your newly created project directory:

**OR**

**dotnet new console**

Creates a new console application template.

**Step 4:**

This will take the folder name as the Application Name

using System;

class Program

{

    static void Main(string[] args)

    {

        Console.WriteLine("Hello, World!");

        // Your code here

    }

}

**Step 5: Build and Run Your Application**

To build and run your console application, use the following commands:

**dotnet build**

**dotnet run**

* **dotnet build**: Compiles the application.
* **dotnet run**: Runs the application.

These are some of the essential dotnet CLI commands used in .NET development for managing projects, dependencies, testing, and deployment. Each command has additional options and parameters that can be explored further using **dotnet --help or dotnet <command> --help** for detailed usage instructions.

Execute a .NET application.

runtime-options:

  --additionalprobingpath <path>   Path containing probing policy and assemblies to probe for.

  --additional-deps <path>         Path to additional deps.json file.

  --depsfile                       Path to <application>.deps.json file.

  --fx-version <version>           Version of the installed Shared Framework to use to run the application.

  --roll-forward <setting>         Roll forward to framework version  (LatestPatch, Minor, LatestMinor, Major, LatestMajor, Disable).

  --runtimeconfig                  Path to <application>.runtimeconfig.json file.

path-to-application:

  The path to an application .dll file to execute.

Usage: dotnet [sdk-options] [command] [command-options] [arguments]

Execute a .NET SDK command.

sdk-options:

  -d|--diagnostics  Enable diagnostic output.

  -h|--help         Show command line help.

  --info            Display .NET information.

  --list-runtimes   Display the installed runtimes.

  --list-sdks       Display the installed SDKs.

  --version         Display .NET SDK version in use.

SDK commands:

  add               Add a package or reference to a .NET project.

  build             Build a .NET project.

  build-server      Interact with servers started by a build.

  clean             Clean build outputs of a .NET project.

  format            Apply style preferences to a project or solution.

  help              Show command line help.

  list              List project references of a .NET project.

  msbuild           Run Microsoft Build Engine (MSBuild) commands.

  new               Create a new .NET project or file.

  nuget             Provides additional NuGet commands.

  pack              Create a NuGet package.

  publish           Publish a .NET project for deployment.

  remove            Remove a package or reference from a .NET project.

  restore           Restore dependencies specified in a .NET project.

  run               Build and run a .NET project output.

  sdk               Manage .NET SDK installation.

  sln               Modify Visual Studio solution files.

  store             Store the specified assemblies in the runtime package store.

  test              Run unit tests using the test runner specified in a .NET project.

  tool              Install or manage tools that extend the .NET experience.

  vstest            Run Microsoft Test Engine (VSTest) commands.

  workload          Manage optional workloads.

Additional commands from bundled tools:

  dev-certs         Create and manage development certificates.

  fsi               Start F# Interactive / execute F# scripts.

  user-jwts         Manage JSON Web Tokens in development.

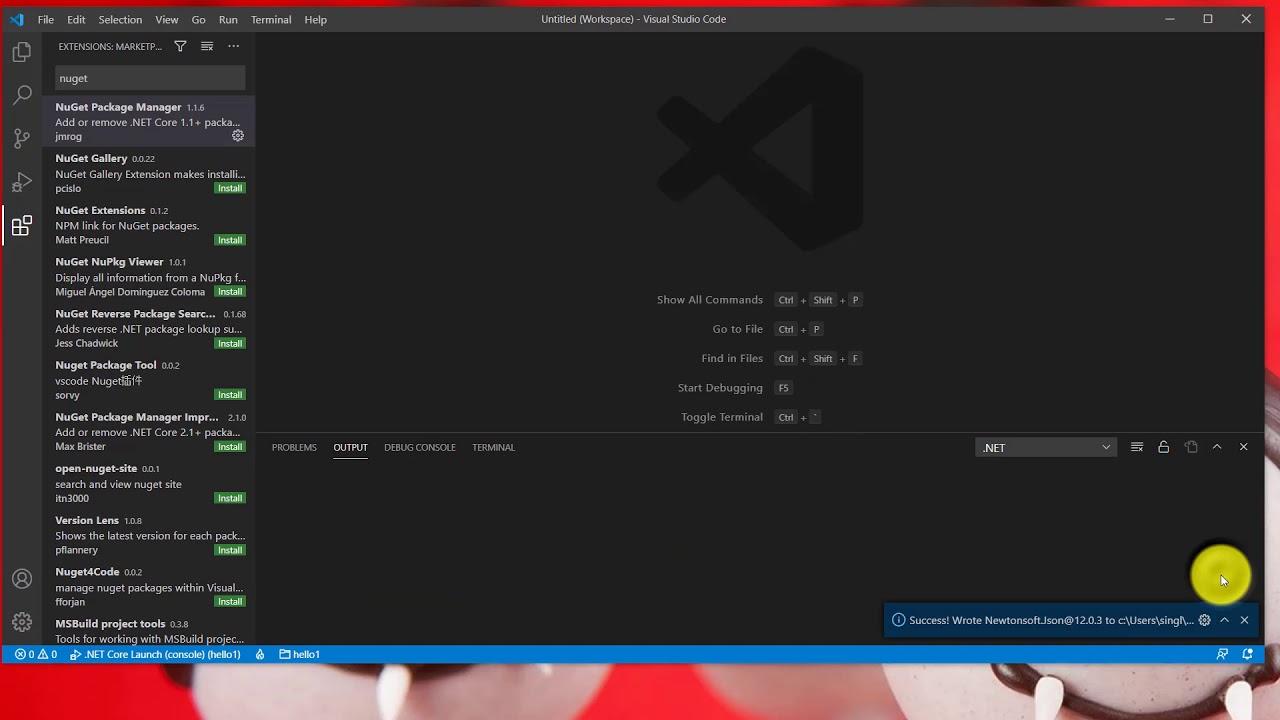
  user-secrets      Manage development user secrets.

  watch             Start a file watcher that runs a command when files change.

Run 'dotnet [command] --help' for more information on a command.

**Extension VSCode**

* NuGet package manager  ( <https://youtu.be/e1GhVs85Rss?si=jV3UEZb7GunjT3Xl> )
* C# Dev Kit
* C#
* .net install tool
* C# Extentions

YouTube

.Net core - Easy to use Nuget manager with VS Code

allmnet의 Python 개발 채널

16.1K views · 4 years ago

OpenShare in meetingBy using YouTube, you agree to the privacy policy, terms of use, and permissions.

[Pawaskar, Rashmi Darshan (Unverified): path-to-application:    The path to an app...](https://teams.microsoft.com/l/message/19:meeting_ZDU3MDllMmQtM2M0OC00YzM2LTg1NjgtNWZkNGVkYTVkMDBk@thread.v2/1737525743280?context=%7B%22contextType%22%3A%22chat%22%7D)

sent on January 22, 2025 11:32 AM