Developing ASP.NET Core apps using dotnet watch

2017-3-9 2 min to read Contributors all

In this article

By Rick Anderson and Victor Hurdugaci

`dotnet watch` is a tool that runs a `dotnet` command when source files change. For example, a file change can trigger compilation, tests, or deployment.

In this tutorial we use an existing Web API app with two endpoints: one that returns a sum and one that returns a product. The product method contains a bug that we'll fix as part of this tutorial.

Download the sample app. It contains two projects, `WebApp` (a web app) and `WebAppTests` (unit tests for the web app).

In a console, navigate to the WebApp folder and run the following commands:

- `dotnet restore`
- `dotnet run`

The console output will show messages similar to the following (indicating that the app is running and waiting for requests):

console Copy

```console
$ dotnet run
Hosting environment: Production
Content root path: C:/Docs/aspnetcore/tutorials/dotnet-watch/sample/WebApp
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
```

In a web browser, navigate to `http://localhost:5000/api/math/sum?a=4&b=5`, you should see the result `9`.

Navigate to the product API (`http://localhost:5000/api/math/product?a=4&b=5`), it returns `9`, not `20` as you'd expect. We'll fix that later in the tutorial.

Add `dotnet watch` to a project

- Add `Microsoft.DotNet.Watcher.Tools` to the *.csproj* file:

XML

Copy

```
<ItemGroup>
 <DotNetCliToolReference Include="Microsoft.DotNet.Watcher.Tools" Version="1.0.0" />
</ItemGroup>
```

- Run `dotnet restore`.

Running `dotnet` commands using `dotnet watch`

Any `dotnet` command can be run with `dotnet watch`, for example:

Command	Command with watch
dotnet run	dotnet watch run
dotnet run -f net451	dotnet watch run -f net451
dotnet run -f net451 -- --arg1	dotnet watch run -f net451 -- --arg1
dotnet test	dotnet watch test

Run `dotnet watch run` in the `WebApp` folder. The console output will indicate `watch` has started.

Making changes with `dotnet watch`

Make sure `dotnet watch` is running.

Fix the bug in the `Product` method of the `MathController` so it returns the product and not the sum.

C#

Copy

```
public static int Product(int a, int b)
{
  return a * b;
}
```

Save the file. The console output will show messages indicating that `dotnet watch` detected a file change and restarted the app.

Verify `http://localhost:5000/api/math/product?a=4&b=5` returns the correct result.

Running tests using `dotnet watch`

- Change the `Product` method of the `MathController` back to returning the sum and save the file.
- In a command window, naviagate to the `WebAppTests` folder.
- Run `dotnet restore`
- Run `dotnet watch test`. You see output indicating that a test failed and that watcher is waiting for file changes:

  console

  Copy

  ```console
  Total tests: 2. Passed: 1. Failed: 1. Skipped: 0.
  Test Run Failed.
  ```

- Fix the `Product` method code so it returns the product. Save the file.

`dotnet watch` detects the file change and reruns the tests. The console output will show the tests passed.

dotnet-watch in GitHub

dotnet-watch is part of the GitHub DotNetTools repository.

The MSBuild section of the dotnet-watch ReadMe explains how dotnet-watch can be configured from the MSBuild project file being watched. The dotnet-watch ReadMe contains information on dotnet-watch not covered in this tutorial.