

Podcast #128: We chat with Kent C Dodds about why he loves React and discuss what life was like in the dark days before Git. [Listen now](#).

What is the default for AddJsonFile options in configuration under .NET Core 2.2?

Asked 10 months ago Active 10 months ago Viewed 633 times

▲ According to [some wise source](#), there's a method called *AddJsonFile* for reading in a config file. [One of its signatures](#) allows to specify *optional* and *reloadOnChange*.

1



```
IConfigurationRoot config = new ConfigurationBuilder()  
    .SetBasePath(Directory.GetCurrentDirectory())  
    .AddJsonFile("config.json", false, false)  
    .Build();
```

However, I can't find info on what's the default value for that in case I use a signature without those explicitly specified. I can assume that it's *false*, basing it on the fact that it's the default value of a boolean (i.e. *default(bool)* is *false*).

But I prefer to have stuff explicitly and linkably stated.

c#

configuration

asked Jan 20 at 15:47



[Konrad Viltersten](#)


22.2k 37 155 291

1 Answer



Refer the source code present at: <https://github.com/aspnet/Configuration/blob/master/src/Config.Json/JsonConfigurationExtensions.cs>

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).



```

    /// <summary>
    /// Adds the JSON configuration provider at <paramref name="path"/> to <paramref
name="builder"/>.
    /// </summary>
    /// <param name="builder">The <see cref="IConfigurationBuilder"/> to add to.
</param>
    /// <param name="path">Path relative to the base path stored in
    /// <see cref="IConfigurationBuilder.Properties"/> of <paramref
name="builder"/>.</param>
    /// <returns>The <see cref="IConfigurationBuilder"/>.</returns>
    public static IConfigurationBuilder AddJsonFile(this IConfigurationBuilder
builder, string path)
    {
        return AddJsonFile(builder, provider: null, path: path, optional: false,
reloadOnChange: false);
    }
}

```

As you mentioned earlier in the question, you can use another overload of this method to set reloadOnChange value. Below is sample code [from MSDN](#):

```

public class Program
{
    public static void Main(string[] args)
    {
        CreateWebHostBuilder(args).Build().Run();
    }

    public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
        WebHost.CreateDefaultBuilder(args)
            .ConfigureAppConfiguration((hostingContext, config) =>
            {
                config.AddJsonFile("appsettings.json", optional: false, reloadOnChange:
false);

                config.AddCommandLine(args);
            })
            .UseStartup<Startup>();
}

```

answered Jan 20 at 16:10

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

-
- 2 Yeah, that's what got me suspicious. Usually, I prefer not to set explicitly such that already is implicitly defaulted (for the sake of compactness and brevity). However, I've see numerous examples with those two parameters set to false and that got me thinking that, perhaps, those are set to true by default and, hence, require falsifying explicitly. I realize now that it's a case of misleading example copied over and over in numerous blogs. Same goes for the excessively specification of the named parameters. That's not needed and, in my view, confusing. – [Konrad Viltersten](#) Jan 20 at 17:51
-