Code style

Last updated by | Michiel Wouters | 12 feb 2024 at 16:07 CET

Code style

General

A code style is a set of guidelines that help developers write code in a consistent and readable way. A code style can also help prevent common errors, such as missing semicolons, incorrect indentation, or mismatched braces. A code style can be enforced by using tools such as code analysers, code formatters, or code editors that provide suggestions and warnings.

Some of the benefits of using a code style are:

- It makes the code easier to read and understand, which can improve productivity and collaboration.
- It reduces the chances of introducing bugs or errors due to inconsistent or unclear code.
- It helps maintain a high quality and professional appearance of the codebase.
- It can help adhere to the best practices and conventions of the programming language and the framework.

✋ Do use access modifiers

> Why? Access modifiers in C# are used to control the accessibility of a class, method, or variable. They help to define the scope of the class, method, or variable and determine which parts of the code can access it. Using access modifiers can help to improve the maintainability of your code by making it easier to understand and modify.

The following access modifiers are available in C#: (Access Modifiers (C# Programming Guide))

- public: The type or member can be accessed by any other code in the same assembly or another assembly that references it. The accessibility level of public members of a type is controlled by the accessibility level of the type itself.
- private: The type or member can be accessed only by code in the same class or struct.
- protected: The type or member can be accessed only by code in the same class, or in a class that is derived from that class.
- internal: The type or member can be accessed by any code in the same assembly, but not from another assembly. In other words, internal types or members can be accessed from code that is part of the same compilation.
- protected internal: The type or member can be accessed by any code in the assembly in which it's declared, or from within a derived class in another assembly.
- private protected: The type or member can be accessed by types derived from the class that are declared within its containing assembly.

💡 Tip

> You should have a compelling reason for every item you make public. If you don't, hide it.

✋ Do use the CodeMaid extension and/or an in-house edit config

> Why? Non-formatted code, double whitespaces, unused usings and instances, uneven indentations etc. can make code look 'sloppy' and it's wasted effort in trying to remember all guidelines on whitespaces, braces, etc. per language.

A tool like CodeMaid can automate this process and will almost always make code look better. Remember, a good developer is a lazy developer.

https://marketplace.visualstudio.com/items?itemName=SteveCadwallader.CodeMaid

✋ Do try to apply the SOLID guidelines as much as you can

> Why? As a DotNet Labs developer, you should have at least a passing knowledge of the SOLID principles. These will help to make your code more readable, testable, expandable and more resilient to change.

DotNetAcademy SOLID course
Tim Corey course on SOLID

Having troubles applying the SOLID guidelines? That's understandable since the concepts may be challenging to newcomers. Also, an existing codebase with differing coding styles might make it hard to implement these abstract concepts.

At the very least, try to implement the Single Responsibility and Dependency Inversion principles by writing short and focused methods that do one thing and do that one thing well. Try splitting up larger methods into smaller, readable, reusable and testable methods. To implement DI, try using Dependency Injection and avoiding the new keyword as much as possible when consuming Services.

New is glue