file (C# Reference)

Article • 11/18/2022

Beginning with C# 11, the `file` contextual keyword is a type modifier.

The `file` modifier restricts a top-level type's scope and visibility to the file in which it's declared. The `file` modifier will generally be applied to types written by a source generator. File-local types provide source generators with a convenient way to avoid name collisions among generated types. The `file` modifier declares a file-local type, as in this example:

C#

```
file class HiddenWidget
{
    // implementation
}
```

Any types nested within a file-local type are also only visible within the file in which it's declared. Other types in an assembly may use the same name as a file-local type. Because the file-local type is visible only in the file where it's declared, these types don't create a naming collision.

A file-local type can't be the return type or parameter type of any member that is more visible than `file` scope. A file-local type can't be a field member of a type that has greater visibility than `file` scope. However, a more visible type may implicitly implement a file-local interface type. The type can also explicitly implement a file-local interface but explicit implementations can only be used within the `file` scope.

Example

The following example shows a public type that uses a file-local type to provide a worker method. In addition, the public type implements a file-local interface implicitly:

C#

```
// In File1.cs:
file interface IWidget
{
    int ProvideAnswer();
}

file class HiddenWidget
```

```
{
    public int Work() => 42;
}

public class Widget : IWidget
{
    public int ProvideAnswer()
    {
        var worker = new HiddenWidget();
        return worker.Work();
    }
}
```

In another source file, you can declare types that have the same names as the file-local types. The file-local types aren't visible:

C#

```
// In File2.cs:
// Doesn't conflict with HiddenWidget
// declared in File1.cs
public class HiddenWidget
{
    public void RunTask()
    {
        // omitted
    }
}
```

C# language specification

For more information, see Declared accessibility in the C# Language Specification, and the C# 11 - File local types feature specification.

See also

- C# Reference
- C# Programming Guide
- C# Keywords
- Access Modifiers
- Accessibility Levels
- Modifiers
- public
- protected
- internal