Access Modifiers

.NET / MICROSOFT DYNAMICS 365

Access modifiers are keywords used to specify the declared ***accessibility*** *of a member or type.*

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/access-modifiers

Access Modifiers - Class Accessibility

https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/access-modifiers

- ***Classes*** and ***structs*** declared directly within a namespace (not nested within other classes or structs) can be either ***public*** or ***internal***.
- ***Internal*** is the default if no access modifier is specified.
- ***Derived*** classes can't have greater accessibility than their ***base*** types.

Access Modifiers - Class Member Accessibility

https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/classes-and-objects#accessibility

Access Modifiers control which regions of program text can access the member.

- public - Access isn't limited.
- private (default)- This class only.
- internal - current assembly (.exe, .dll).
- protected internal - Child classes, or classes within the same assembly.
- protected - Derived classes only.
- private protected - This class or derived classes only.

Access Modifiers – Public

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/public

The ***public*** keyword is an access modifier for types and type members.

There are no restrictions on accessing public members.

```
class PointTest
{
    public int x;
    public int y;
}

class MainClass4
{
    static void Main()
    {
        var p = new PointTest();
        // Direct access to public members.
        p.x = 10;
        p.y = 15;
        Console.WriteLine($"x = {p.x}, y = {p.y}");
    }
}
// Output: x = 10, y = 15
```

Access Modifiers – Private

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/private

- ***Private*** access is the least permissive access level. ***Private*** members are accessible only within the body of the ***class*** or the ***struct*** in which they are declared.
- Nested types in the same body can also access those ***private*** members.

```
class Employee
{
    private int i;
    double d;   // private access by default
}
```

Access Modifiers – Private Example

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/private

This Employee class contains two ***private*** data members. As ***private*** members, they can only be accessed by member methods.

Public methods, GetName() and Salary(), are added to allow controlled access to the ***private*** members.

```
class Employee2
{
    private string name = "FirstName, LastName";
    private double salary = 100.0;

    public string GetName()
    {
        return name;
    }

    public double Salary
    {
        get { return salary; }
    }
}

class PrivateTest
{
    static void Main()
    {
        var e = new Employee2();

        // The data members are inaccessible (private), so
        // they can't be accessed like this:
        //    string n = e.name;
        //    double s = e.salary;

        // 'name' is indirectly accessed via method:
        string n = e.GetName();

        // 'salary' is indirectly accessed via property
        double s = e.Salary;
    }
}
```

Access Modifiers – Internal

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/internal

Internal types or members are accessible only within files in the same assembly.

A common use of ***internal*** access is in component-based development because it enables a group of components to cooperate in a private manner without being exposed to the rest of the application code.

```
public class BaseClass
{
    // Only accessible within the same assembly.
    internal static int x = 0;
}
```

Access Modifiers – Protected Internal

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/protected-internal

- A ***protected internal*** member of a base class is accessible from any type within its containing assembly.
- It is also accessible in a derived class located in another assembly only if the access occurs through a variable of the derived class type.
- Struct members cannot be ***protected internal*** because the struct cannot be inherited.

Access Modifiers – Protected Internal

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/protected-internal

BaseClass and TestAccess are in the same assembly. TestAccess can access myValue. In the second file, an attempt to access myValue through an instance of BaseClass will produce an error, while an access to this member through DerivedClass succeeds.

```
// Assembly1.cs
// Compile with: /target:library
public class BaseClass
{
   protected internal int myValue = 0;
}

class TestAccess
{
    void Access()
    {
        var baseObject = new BaseClass();
        baseObject.myValue = 5;
    }
}
```

```
// Assembly2.cs
// Compile with: /reference:Assembly1.dll
class DerivedClass : BaseClass
{
    static void Main()
    {
        var baseObject = new BaseClass();
        var derivedObject = new DerivedClass();

        // Error CS1540, because myValue can only be accessed by
        // classes derived from BaseClass.
        // baseObject.myValue = 10;

        // OK, because this class derives from BaseClass.
        derivedObject.myValue = 10;
    }
}
```

Access Modifiers – Protected

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/protected

A ***protected*** member is accessible through ***derived*** class instances only.

```
class A
{
    protected int x = 123;
}

class B : A
{
    static void Main()
    {
        var a = new A();
        var b = new B();

        // Error CS1540, because x can only be accessed by
        // classes derived from A.
        // a.x = 10;

        // OK, because this class derives from A.
        b.x = 10;
    }
}
```

Access Modifiers – Private Protected

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/private-protected

- A ***private protected*** member is accessible by types derived from the containing class, but only within its containing assembly.
- Assembly1.cs contains ***public*** BaseClass, and ***derived***, DerivedClass1.
- BaseClass owns a ***private protected*** myValue, which DerivedClass1 tries to access in two ways.
- Accessing myValue through an instance of BaseClass will produce an error.
- Using it as an inherited member in DerivedClass1 succeeds.
- In Assembly2.cs, accessing myValue as an inherited member of DerivedClass2 produces an error, because it's in a different assembly.

```
// Assembly1.cs
// Compile with: /target:library
public class BaseClass
{
    private protected int myValue = 0;
}

public class DerivedClass1 : BaseClass
{
    void Access()
    {
        var baseObject = new BaseClass();

        // Error CS1540, because myValue can only be accessed by
        // classes derived from BaseClass.
        // baseObject.myValue = 5;

        // OK, accessed through the current derived class instance
        myValue = 5;
    }
}
```

C#

```
// Assembly2.cs
// Compile with: /reference:Assembly1.dll
class DerivedClass2 : BaseClass
{
    void Access()
    {
        // Error CS0122, because myValue can only be
        // accessed by types in Assembly1
        // myValue = 10;
    }
}
```