Early and late binding in C#

When an object is assigned to an object variable of the specific type, then the C# compiler performs the binding with the help of .NET Framework. C# performs two different types of bindings which are:

- **Early Binding or Static Binding**
- **Late Binding or Dynamic Binding**

Early Binding

It recognizes and checks the methods, or properties during compile time. In this binding, the compiler already knows about what kind of object it is and what are the methods or properties it holds, here the objects are static objects. The performance of early binding is fast and it is easy to code. It decreases the number of run-time errors.

Example:

```
// C# program to illustrate the
// concept of early binding
using System;

class Geeks {

    // data members
    public string name;
    public string subject;

    // public method
    public void details(string name, string subject)
    {
        this.name = name;
        this.subject = subject;
        Console.WriteLine("Myself: " + name);
        Console.WriteLine("My Favorite Subject is: " + subject);
    }
}

// Driver class
class GFG {

    // Main Method
    static void Main(string[] args)
    {

        // creating object of Geeks class
        Geeks g = new Geeks();

        // Calling the method of Geeks class
        g.details("Ankita", "C#");

        // Calling "mymethod()" gives error
        // because this method does not
        // belong to class Geeks or compiler
        // does not know mymethod() at compile time
        g.mymethod();
    }
}
```

Compile-Time error:

> *prog.cs(34, 5): error CS1061: Type `Geeks' does not contain a definition for `mymethod' and no extension method `mymethod' of type `Geeks' could be found. Are you missing an assembly reference?*
> *prog.cs(5, 7): (Location of the symbol related to previous error)*

Explanation: In the above example, we have a class named as *Geeks*. This class contains *details()* method. Here, the compiler already knows about the properties and methods present in *Geeks*. But when we try to call *mymethod()* then it will throw an error because this method is not known by the compiler.

Late Binding

In late binding, the compiler does not know about what kind of object it is and what are the methods or properties it holds, here the objects are dynamic objects. The type of the object is decided on the bases of the data it holds on the right-hand side during run-time. Basically, late binding is achieved by using *virtual* methods. The performance of late binding is slower than early binding because it requires lookups at run-time.

Example: In the below, program the *obj* holds integer type data and *obj1* holds double type data. But the compiler doesn't resolve these at compile-time. At the runtime, these dynamic objects get detected and converted into `System.Int32` and `System.Double` respectively. Thats why the run-time resolving process is termed as late binding.

```
// C# program to illustrate the
// concept of late binding
using System;

class GFG {
    static void Main()
    {
        // Dynamic objects
        dynamic obj = 4;
        dynamic obj1 = 5.678;

        // Display the type of objects
        Console.WriteLine("The type of the objects are :");

        // GetType() method is
        // used to get the type
        Console.WriteLine(obj.GetType());
        Console.WriteLine(obj1.GetType());
    }
}
```

Output :

```
The type of the objects are :
System.Int32
System.Double
```

Recommended Posts:

Late Binding using Reflection in C#

Difference between C and C#

Writing to Excel Sheet Using EPPlus in C#

Lambda Expressions in C#

How to Get Started with Game Development?

Basics of File Handling in C#

C# | ToolTip Class

C# | RichTextBox Class

C# | MaskedTextBox Class

C# | NumericUpDown Class

C# | DateTimePicker Class

C# | GroupBox Class

C# | ListBox Class

Different ways to convert String to Integer in C#

ankita_saini
Check out this Author's contributed articles.

Article Tags : C# CSharp-OOP

1

0

☐ To-do ☐ Done

No votes yet.

Feedback/ Suggest Improvement Add Notes Improve Article

▲