Async and Await in C#

Vivek Kumar Updated date Sep 24, 2020

Basics of C# async await. In this article, you'll learn what C# async and C# await keywords are, and how to use async and await in C# code.

Nowadays, Asynchronous programming is very popular with the help of the async and await keywords in C#. When we are dealing with UI and on button click, we use a long running method like reading a large file or something else which will take a long time, in that case, the entire application must wait to complete the whole task. In other words, if any process is blocked in a synchronous application, the entire application gets blocked and our application stops responding until the whole task completes.
Asynchronous programming is very helpful in this condition. By using Asynchronous programming, the Application can continue with the other work that does not depend on the completion of the whole task.

We will get all the benefits of traditional Asynchronous programming with much less effort by the help of async and await keywords.

methods are not dependent on each other and Method1 is taking a long time to complete its task. In Synchronous programming, it will execute the first Method1 and it will wait for completion of this method and then it will execute Method2. Thus, it will be a time intensive process even though both the methods are not depending on each other.

We can run all the methods parallelly by using the simple thread programming but it will block UI and wait to complete all the tasks. To come out of this problem, we have to write too many codes in traditional programming but if we will simply use the async and await keywords, then we will get the solutions in much less code.

Async and await in C# are the code markers, which marks code positions from where the control should resume after a task completes.

Let's start with practical examples for understanding the programming concept.

Code examples of C# async await

We are going to take a console application for our demonstration.

Example 1

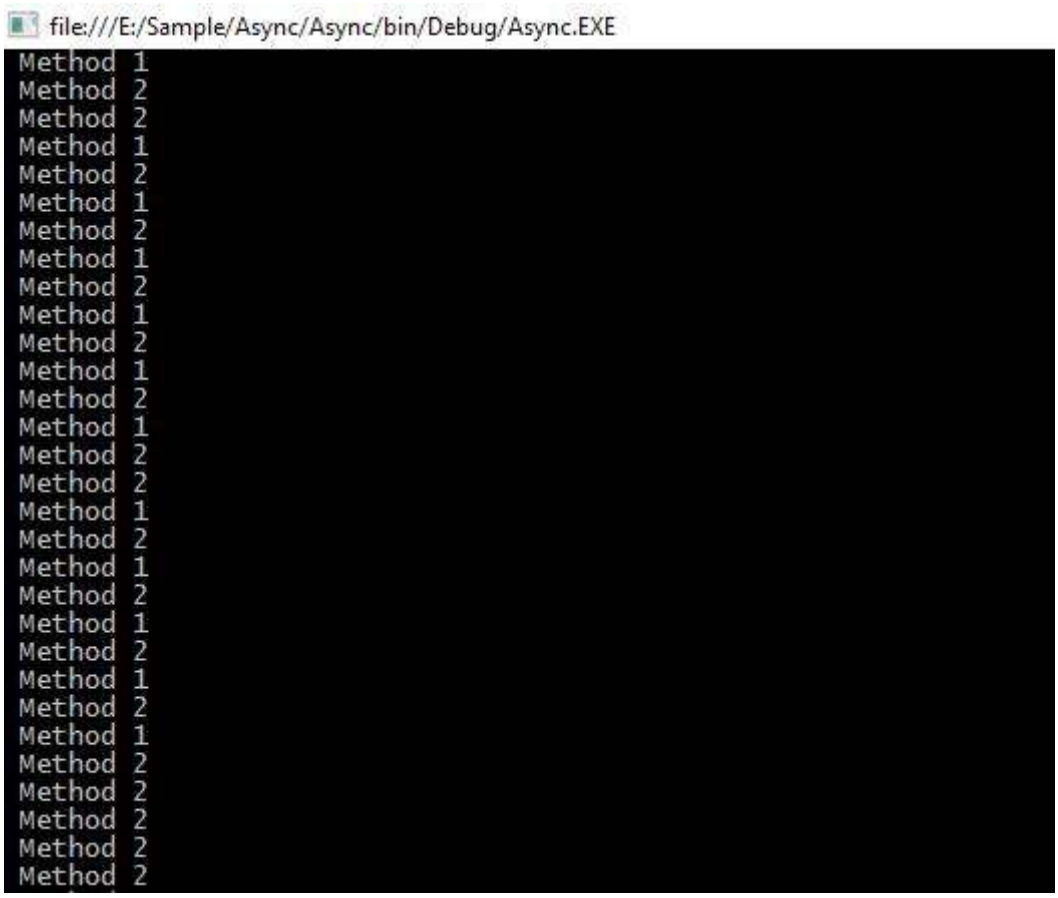
In this example, we are going to take two methods, which are not dependent on each other.

Code sample

```
01. class Program
02. {
03.     static void Main(string[] args)
04.     {
05.         Method1();
06.         Method2();
07.         Console.ReadKey();
08.     }
09.
10.     public static async Task Method1()
11.     {
12.         await Task.Run(() =>
13.         {
14.             for (int i = 0; i < 100; i++)
15.             {
16.                 Console.WriteLine(" Method 1");
19.     }
20.
21.
22.     public static void Method2()
23.     {
24.         for (int i = 0; i < 25; i++)
25.         {
26.             Console.WriteLine(" Method 2");
27.         }
28.     }
29. }
```

Here, we can clearly see Method1 and Method2 are not waiting for each other.

Output



Now, coming to the second example, suppose we have Method3, which is dependent on Method1

In this example, Method1 is returning total length as an integer value and we are passing a parameter as a length in a Method3, which is coming from Method1.

Here, we have to use await keyword before passing a parameter in Method3 and for it, we have to use the async keyword from the calling method.

We can not use await keyword without async and we cannot use async keyword in the Main method for the console Application because it will give the error given below.

Code sample

```
01. class Program
02. {
03.     static void Main(string[] args)
04.     {
05.         callMethod();
06.         Console.ReadKey();
07.     }
08.
09.     public static async void callMethod()
10.     {
11.         Task<int> task = Method1();
12.         Method2();
13.         int count = await task;
14.         Method3(count);
15.     }
16.
17.     public static async Task<int> Method1()
18.     {
19.         int count = 0;
20.         await Task.Run(() =>
21.         {
22.             for (int i = 0; i < 100; i++)
23.             {
24.                 Console.WriteLine(" Method 1");
25.                 count += 1;
26.             }
27.         });
28.         return count;
29.     }
30.
31.     public static void Method2()
32.     {


35.             Console.WriteLine(" Method 2");
36.         }
37.     }
38.
39.     public static void Method3(int count)
40.     {
41.         Console.WriteLine("Total count is " + count);
42.     }
43. }
```

In the code given above, Method3 requires one parameter, which is the return type of Method1. Here, await keyword is playing a vital role for waiting of Method1 task completion

Real time example

There are some supporting API's from the .NET Framework 4.5 and the Windows runtime contains methods that support async programming.

We can use all of these in the real time project with the help of async and await keyword for the faster execution of the task.

Some APIs that contain async methods are HttpClient, SyndicationClient, StorageFile, StreamWriter, StreamReader, XmlReader, MediaCapture, BitmapEncoder, BitmapDecoder etc.

In this example, we are going to read all the characters from a large text file asynchronously and get the total length of all the characters.

Sample code

```
01. class Program
02. {
03.     static void Main()
04.     {
05.         Task task = new Task(CallMethod);
06.         task.Start();
07.         task.Wait();
08.         Console.ReadLine();
09.     }
10.
11.     static async void CallMethod()
12.     {
13.         string filePath = "E:\\sampleFile.txt";
14.         Task<int> task = ReadFile(filePath);
15.
16.
17.         Console.WriteLine(" Other Work 2");
18.         Console.WriteLine(" Other Work 3");
19.
20.         int length = await task;
21.         Console.WriteLine(" Total length: " + length);
22.
23.         Console.WriteLine(" After work 1");
24.         Console.WriteLine(" After work 2");
25.     }
26.
27.     static async Task<int> ReadFile(string file)
28.     {
29.         int length = 0;
30.
```

```
        {
            // Reads all characters from the  he end o
            // and returns them as one string.
            string s = await reader.ReadToEndAsync();

            length = s.Length;
        }
        Console.WriteLine(" File reading is completed");
        return length;
    }
}
```

In the code given above, we are calling a ReadFile method to read the contents of a text file and get the length of the total characters present in the text file.

In our sampleText.txt, file contains too many characters, so It will take a long time to read all the characters.

Here, we are using async programming to read all the contents from the file, so it will not wait to get a return value from this method and execute the other lines of code but it has to wait for the line of code given below because we are using await keyword and we are going to use the return value for the line of code given below..

```
int length = await task;
Console.WriteLine(" Total length: " + length);
```

Subsequently, other lines of code will be executed sequentially.

```
Console.WriteLine(" After work 1");
Console.WriteLine(" After work 2");
```

Output

Here, we have to understand very important points that if we are not using await keyword, then the method works as a synchronous method. The compiler will show the warning to us but it will not show any error.

In this easy way, we can use async and await keywords in C# to implement async programming.

If you're new to Async programming, here is a detailed tutorial - **Tutorial on Asynchronous programming in C#**.

Next Recommended Article

C# Asynchronous Programming - Async and Await

Vivek Kumar TOP 500

Microsoft Certified Professional | C# Corner MVP | Dzone MVB | Author | Speaker

https://vivekkumar.com

180 **2.9m** **2**

View Previous Comments

75 **37**

Type your comment here and press Enter Key (Minimum 10 characters)

Is this article updated ? In the 'Example 2' , it says ,"We cannot use async on Main method" . But that is before c#7.1. After c# 7.1 we can use. Is this article written before 2018 ?

Siva Sankaran S

1973 **7** **0**

Sep 09, 2020

0 0 Reply

Hey, Vivek. Thank you so much. your explanation is simple and clear.

hiuwa li

1969 **11** **0**

Aug 10, 2020

0 0

Ardmore Ardmore
1976 4 0
Feb 11, 2020
0 Reply

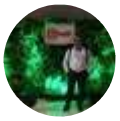
Great explanation
Mallikarjun Dasari
1938 42 0
Feb 11, 2020
0 0 Reply

If we call method4 just after method 3,so it holds the compiler for method 4 too.so,where is async programming..?
Summit Rajput
1939 41 11.5k
Jan 29, 2020
0 1 Reply

If the method4 is independent then we can keep that method above the await, dependent methods should go to end
Mallikarjun Dasari
1938 42 0
Feb 11, 2020
0

Very confusing article
Summit Rajput
1939 41 11.5k
Jan 29, 2020
1 1 Reply

Read it carefully, need to practice each and every example mentioned in this article.
Khaja Moizuddin
193 11.1k 988.1k
Feb 01, 2020
1



Nice explanation.
Imran Shaikh
1397 591 30.6k
Nov 21, 2019
0 0 Reply

Its doubt: the same thing threading was done then why we need asyn methods,if any thing exists tell me the different
sivaprakash sekar
Oct 15, 2019

Atta Kumah
1832 148 4.3k
Sep 28, 2019
0 0 Reply

Good article. We can easily understand... Thank you
Bavani Mayan
1967 13 0
Sep 28, 2019
0 0 Reply

FEATURED ARTICLES

Getting Started Wth MongoDb Atlas

What Is a Full Stack Developer