Cancel async tasks after a period of time

Article • 09/08/2023

You can cancel an asynchronous operation after a period of time by using the CancellationTokenSource.CancelAfter method if you don't want to wait for the operation to finish. This method schedules the cancellation of any associated tasks that aren't complete within the period of time that's designated by the `CancelAfter` expression.

This example adds to the code that's developed in Cancel a list of tasks (C#) to download a list of websites and to display the length of the contents of each one.

This tutorial covers:

✓ Updating an existing .NET console application
✓ Scheduling a cancellation

Prerequisites

This tutorial requires the following:

- You're expected to have created an application in the Cancel a list of tasks (C#) tutorial
- .NET 5 or later SDK
- Integrated development environment (IDE)
  - We recommend Visual Studio or Visual Studio Code

Update application entry point

Replace the existing `Main` method with the following:

```csharp
static async Task Main()
{
    Console.WriteLine("Application started.");

    try
    {
        s_cts.CancelAfter(3500);

        await SumPageSizesAsync();
    }
    catch (OperationCanceledException)
    {
        Console.WriteLine("\nTasks cancelled: timed out.\n");
    }
    finally
    {
        s_cts.Dispose();
    }

    Console.WriteLine("Application ending.");
}
```

The updated `Main` method writes a few instructional messages to the console. Within the try-catch, a call to CancellationTokenSource.CancelAfter(Int32) schedules a cancellation. This will signal cancellation after a period of time.

Next, the `SumPageSizesAsync` method is awaited. If processing all of the URLs occurs faster than the scheduled cancellation, the application ends. However, if the scheduled cancellation is triggered before all of the URLs are processed, a OperationCanceledException is thrown.

Example application output

Console

```
Application started.

https://learn.microsoft.com                                       37,357
https://learn.microsoft.com/aspnet/core                           85,589
https://learn.microsoft.com/azure                                398,939
https://learn.microsoft.com/azure/devops                          73,663

Tasks cancelled: timed out.

Application ending.
```

Complete example

The following code is the complete text of the *Program.cs* file for the example.

C#

```
using System.Diagnostics;

class Program
{
    static readonly CancellationTokenSource s_cts = new
CancellationTokenSource();
```

```
    static readonly HttpClient s_client = new HttpClient
    {
        MaxResponseContentBufferSize = 1_000_000
    };

    static readonly IEnumerable<string> s_urlList = new string[]
    {
            "https://learn.microsoft.com",
            "https://learn.microsoft.com/aspnet/core",
            "https://learn.microsoft.com/azure",
            "https://learn.microsoft.com/azure/devops",
            "https://learn.microsoft.com/dotnet",
            "https://learn.microsoft.com/dynamics365",
            "https://learn.microsoft.com/education",
            "https://learn.microsoft.com/enterprise-mobility-security",
            "https://learn.microsoft.com/gaming",
            "https://learn.microsoft.com/graph",
            "https://learn.microsoft.com/microsoft-365",
            "https://learn.microsoft.com/office",
            "https://learn.microsoft.com/powershell",
            "https://learn.microsoft.com/sql",
            "https://learn.microsoft.com/surface",
            "https://learn.microsoft.com/system-center",
            "https://learn.microsoft.com/visualstudio",
            "https://learn.microsoft.com/windows",
            "https://learn.microsoft.com/maui"
    };

    static async Task Main()
    {
        Console.WriteLine("Application started.");

        try
        {
            s_cts.CancelAfter(3500);

            await SumPageSizesAsync();
        }
        catch (OperationCanceledException)
        {
            Console.WriteLine("\nTasks cancelled: timed out.\n");
        }
        finally
        {
            s_cts.Dispose();
        }

        Console.WriteLine("Application ending.");
    }

    static async Task SumPageSizesAsync()
    {
        var stopwatch = Stopwatch.StartNew();

        int total = 0;
        foreach (string url in s_urlList)
        {
            int contentLength = await ProcessUrlAsync(url, s_client,
s_cts.Token);
            total += contentLength;
        }

        stopwatch.Stop();

        Console.WriteLine($"\nTotal bytes returned:  {total:#,#}");
        Console.WriteLine($"Elapsed time:          {stopwatch.Elapsed}\n");
    }

    static async Task<int> ProcessUrlAsync(string url, HttpClient client,
CancellationToken token)
    {
        HttpResponseMessage response = await client.GetAsync(url, token);
        byte[] content = await response.Content.ReadAsByteArrayAsync(token);
        Console.WriteLine($"{url,-60} {content.Length,10:#,#}");

        return content.Length;
    }
}
```

See also

- CancellationToken
- CancellationTokenSource
- Asynchronous programming with async and await (C#)
- Cancel a list of tasks (C#)