[Creating task continuations](javascript:void(0)" \o ")

The [Task.ContinueWith](https://msdn.microsoft.com/en-us/library/system.threading.tasks.task.continuewith(v=vs.110).aspx) and [Task<TResult>.ContinueWith](https://msdn.microsoft.com/en-us/library/dd321274(v=vs.110).aspx) methods let you specify a task to start when the *antecedent task* finishes. The delegate of the continuation task is passed a reference to the antecedent task so that it can examine the antecedent task's status and, by retrieving the value of the [Task<TResult>.Result](https://msdn.microsoft.com/en-us/library/dd321468(v=vs.110).aspx) property, can use the output of the antecedent as input for the continuation.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace \_01\_console

{

class Program

{

static void Main(string[] args)

{

var getData = Task.Factory.StartNew(() =>

{

int i = 0;

int[] values = new int[10];

for (int ctr = 1; ctr <= 10; ctr++)

{

if (ctr % 2 != 0)

values[i++] = ctr;

}

return values;

});

var processData = getData.ContinueWith((x) =>

{

int sum = 0;

int n = x.Result.Length;

for (int i = 2; i <= x.Result.GetUpperBound(0); i++)

{

sum = sum + x.Result[i];

}

return Tuple.Create(n, sum);

});

var displayData = processData.ContinueWith ((x) => {

return String.Format("N={0:N0}, Total = {1:N0}", x.Result.Item1, x.Result.Item2); });

Console.WriteLine(displayData.Result);

Console.ReadLine();

}

}

}

Because [Task.ContinueWith](https://msdn.microsoft.com/en-us/library/dd270696(v=vs.110).aspx) is an instance method, you can chain method calls together instead of instantiating a [Task<TResult>](https://msdn.microsoft.com/en-us/library/dd321424(v=vs.110).aspx) object for each antecedent task.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace \_01\_console

{

class Program

{

static void Main(string[] args)

{

var displayData = Task.Factory.StartNew(() =>

{

int i = 0;

int[] values = new int[10];

for (int ctr = 1; ctr <= 10; ctr++)

{

if (ctr % 2 != 0)

values[i++] = ctr;

}

return values;

}).ContinueWith((x) =>

{

int sum = 0;

int n = x.Result.Length;

for (int i = 2; i <= x.Result.GetUpperBound(0); i++)

{

sum = sum + x.Result[i];

}

return Tuple.Create(n, sum);

}).ContinueWith ((x) => {

return String.Format("N={0:N0}, Total = {1:N0}", x.Result.Item1, x.Result.Item2); });

Console.WriteLine(displayData.Result);

Console.ReadLine();

}

}

}