**Part 88 - ThreadStart delegate**

In this video we will discuss the purpose of **ThreadStart delegate**. This is continuation to [Part 87](http://csharp-video-tutorials.blogspot.com/2014/03/part-87-advantages-and-disadvantages-of_8.html). Please watch [Part 87](http://csharp-video-tutorials.blogspot.com/2014/03/part-87-advantages-and-disadvantages-of_8.html) before proceeding.

**Let us understand the purpose of ThreadStart delegate with an example.**

usingSystem**;**

usingSystem.Threading**;**

namespaceThreadStartDelegateExample

**{**

classProgram

**{**

publicstaticvoidMain**()**

**{**

ThreadT1=newThread**(**Number.PrintNumbers**);**

T1.Start**();**

**}**

**}**

classNumber

**{**

publicstaticvoidPrintNumbers**()**

**{**

for**(**inti=1**;**i<=10**;**i++**)**

**{**

Console.WriteLine**(**i**);**

**}**

**}**

**}**

**}**

**In the example above to create a THREAD,** we created an instance of Thread class and to it's constructor we have passed the name of the function that we want the thread to execute.  
ThreadT1=newThread**(**Number.PrintNumbers**);**  
  
**We can rewrite the above line using ThreadStart delegate as shown below.**  
ThreadT1=newThread**(**newThreadStart**(**Number.PrintNumbers**));**  
  
**Why a delegate need to be passed as a parameter to the Thread class constructor?**  
The purpose of creating a Thread is to execute a function. A delegate is a type safe function pointer, meaning it points to a function that the thread has to execute. In short, all threads require an entry point to start execution. Any thread you create will need an explicitly defined entry point i.e a pointer to the function where they should begin execution. So threads always require a delegate.  
  
**In the code below, we are not explicitly creating the ThreadStart delegate, then how is it working here?**  
ThreadT1=newThread**(**Number.PrintNumbers**);**  
  
It's working in spite of not creating the ThreadStart delegate explicitly because the framework is doing it automatically for us.  
  
**We can also rewrite the same line using delagate() keyword as shown below.**

ThreadT1=newThread**(**delegate**() {**Number.PrintNumbers**(); });**

**We can also rewrite the same line using lambda expression as shown below.**

ThreadT1=newThread**(()**=>Number.PrintNumbers**());**

**Thread function need not be a static function always.** It can also be a non-static function. In case of non-static function we have to create an instance of the class. An example is shown below.

classProgram

**{**

publicstaticvoidMain**()**

**{**

Numbernumber=newNumber**();**

ThreadT1=newThread**(**number.PrintNumbers**);**

T1.Start**();**

**}**

**}**

classNumber

**{**

publicvoidPrintNumbers**()**

**{**

for**(**inti=1**;**i<=10**;**i++**)**

**{**

Console.WriteLine**(**i**);**

**}**

**}**

**}**