**Thread in c#**

**Introduction**

C# supports parallel execution of code through multithreading. A thread is an independent execution path, able to run simultaneously with other threads. A C# program starts in a single thread created automatically by the CLR and operating system (the "main" thread), and is made multi-threaded by creating additional threads.

The classes and interfaces in the *System.Threading* namespace provide the multithreading support in the .NET platform. *System.Threading.Thread* is the main class for creating threads and controlling them.

**ThreadStart Delegate**

* To create a thread, we need to instantiate the *Thread* class, passing a *ThreadStart* delegate (S*ystem.Threading.ThreadStart)* in its constructor. This delegate contains the method where the thread will begin execution, when started. The Start() method of the Thread class then starts the execution of a new thread.

Example 1:

using System;

using System.Threading ;

class Thread\_App

{

public static void First\_Thread()

{

Console.WriteLine("First thread created");

}

public static void Main()

{

// Create Delegate & Pass Function Name

ThreadStart funcstart= new ThreadStart(First\_Thread);

Console.WriteLine("Creating the first thread ");

// Use Delegate Name while creating new Thread

Thread fThread = new Thread(funcstart);

// Start new Thread

fThread.Start();

}

}

**Note:**

* we are creating a new thread called fThread, which when started executes the function called First\_Thread(). Delegate ThreadStart(funcstart) contains address of the function(First\_Thread) that needs to be executed when the fThread’s Start() is called.

**Thread Methods**

* Start(): starts the execution of the thread.
* Suspend(): suspends the thread, if the thread is already suspended, nothing happens.
* Resume(): resumes a thread that has been suspended.
* Join(): blocks a calling thread until the thread terminates.
* Sleep(int x): suspends the thread for specified amount of time (in milliseconds).
* Interrupt(): interrupts a thread that is in the wait, sleep or join stage.
* Abort(): Begins the process of terminating the thread. Once the thread terminates, it cannot be restarted by calling the function Start() again.

**Thread Priority**

* *System.Threading.Thread.Priority* enumeration defines the priority states of a thread that in turn determines how much processor time a thread gets to execute.
* Threads are executed based on their priority levels.
* You can then change the priority value to any of the values shown below
  + Highest
  + AboveNormal
  + Normal
  + BelowNormal
  + Lowest

**Thread State**

* *System.Threading.Thread.ThreadState* property defines the state in which a thread is, during the execution of the thread.
* Once created, a thread is always in at least one of the states till it is terminated.
* When a thread is created it is in *Unstarted* state.
* Start() method of the *Thread* class changes this state to a *Running* state for the thread.
* The thread continues to be in this state till it goes into a sleep or is suspended or is aborted or the thread terminates naturally.
* When the thread is suspended is goes into the *Suspended* state till another thread resumes it, when it is back again to the *Running* state.
* On being aborted or terminated, the thread is stopped and the Thread State is *Stopped.*
* Once stopped, a thread can never leave the *Stopped* state just as once started a thread can never return to the *Unstarted* state.