Biblioteka TPL jest biblioteką do programowania równoległego w języku C#

Biblioteka ta wprowadza pojęcie zadania (Task). Równoległość zadań (task parallelism) jest procesem równoległego uruchamiania tych zadań. Zadanie jest niezależną jednostką, która jest uruchamiana w ramach konkretnego programu. Korzyści z takiego podejścia są następujące:

-bardziej wydajne użycie zasobów systemowych;

-większa kontrola nad kodem niż ma to miejsce w przypadku użycia wątków.

Biblioteka TPL wykorzystuje wątki w tle, aby wykonywać zadania równolegle. Decyzja o liczbie używanych wątków jest dynamicznie obliczana w środowisku uruchomieniowym.

Tworzenie wątku wiąże się z ogromnym kosztem. Tworzenie dużej ilości wątków w aplikacji ma wpływ na przeciążenie przełączania pomiędzy kontekstami (Context Switching). W środowisku jednordzeniowym może to również prowadzić do złej wydajności, ponieważ mamy jeden rdzeń, który będzie obsługiwał wiele wątków.

Task, tj. nasze zadanie, dynamicznie oblicza czy potrzebuje utworzyć wiele różnych wątków aby to zadanie zrealizować. W tle używany jest ThreadPool w celu zarządzania pracą bez konieczności tworzenia lub przełączania się pomiędzy kolejnymi wątkami, jeżeli nie jest to wymagane.

Poniżej przykład pokazujący tworzenie zadań równoległych (parallel tasks) używając wątków (Threads) oraz zadań (Task):

```
// tworzenie nowego wątku
var thread = new Thread(start =>
    {
        // zadanie do wykonania
    });
thread.Start();
// tworzenie nowego zadania
Task.Factory.StartNew(()=>
{
    // zadanie do wykonania
});
```

Tworzenie nowych zadań:

```
var task = new Task(doSomework());
task.Start();
```

Oczekiwanie na wykonanie zadań:

```
Task task1 = Task.Factory.StartNew(doSomework());
Task task2 = Task.Factory.StartNew(doSomework());
Task task3 = Task.Factory.StartNew(doSomework());
Task.WaitAll(task1, task2, task3);
```

Wykonanie kolejnego zadania asynchronicznie, kiedy obecne zadanie zostanie ukończone:

```
1  Task.Factory.StartNew(doSomework()).ContinueWith(anotherAsyncWork());
```

W prawdziwym świecie często zachodzi potrzeba wykonania asynchronicznie wielu operacji. Poniższy fragment pokazuje jak możemy tego dokonać:

```
1  Task.Factory.StartNew(GetImagesFromTumblr())
2      .ContinueWith((Func<Task, List<string>>)GetImagesFromMyBirthday())
3      .ContinueWith(ShareThisOnFacebook())
4      .ContinueWith(SendNotificationToMyFriends())
5      .Continue(WaitForConfirmation());
```

Podsumowanie:

Biblioteka ta wraz z dostępnym rozszerzeniem pozwala na pełne wykorzystanie potencjału sprzętowego na którym uruchamiany jest program. Ten sam kod automatycznie dostosowuje się do platformy na której jest uruchamiany dzięki czemu możemy osiągnąć zauważalne korzyści na lepszych maszynach. Biblioteka ta poprawia również czytelność kodu oraz zmniejsza prawdopodobieństwo wystąpienia błędów.