bib International College

Threads

Teil 4

BackgroundWorker

bib International College

- Verwendung, wenn genau ein Thread für eine spezielle Aufgabe benötigt wird.
- Benutzeroberfläche kann während der Ausführung zeitaufwändiger Vorgänge (Downloads, Datenbank-transaktionen) weiter reagieren kann.
- BackgroundWorker zum Ausführen eines zeitaufwändigen Vorgangs im Hintergrund anlegen
- Ereignishandler für das DoWork-Ereignis hinzufügen und zeitaufwändigen Vorgang im Ereignishandler aufrufen
- RunWorkerAsync aufrufen, um den Vorgang zu starten.
- RunWorkerCompleted-Ereignis behandeln, um eine Benachrichtigung über den Abschluss des Vorgangs zu erhalten

```
public class Rechnung {
        private BackgroundWorker worker;
        // Akzeptiert einen BackgroundWorker, der die  Berechnung durchführt und den
        //Fortschritt meldet.
        public Rechnung(BackgroundWorker worker)   {
            this.worker = worker;
        }
        // Öffentliche Funktion, die die Berechnung enthält.
        public int Calculate(int digit)    {
            int summe = 0;
            if (digit > 0)  {
                for (int i = 1; i <= digit; i++)  {
                    summe = summe + i;
                    Thread.Sleep(100);
                    // Fortschrittsanzeige für den BackgroundWorker
                    worker.ReportProgress((i*100)/digit);
                }
            }
            return summe;
        }
}
```

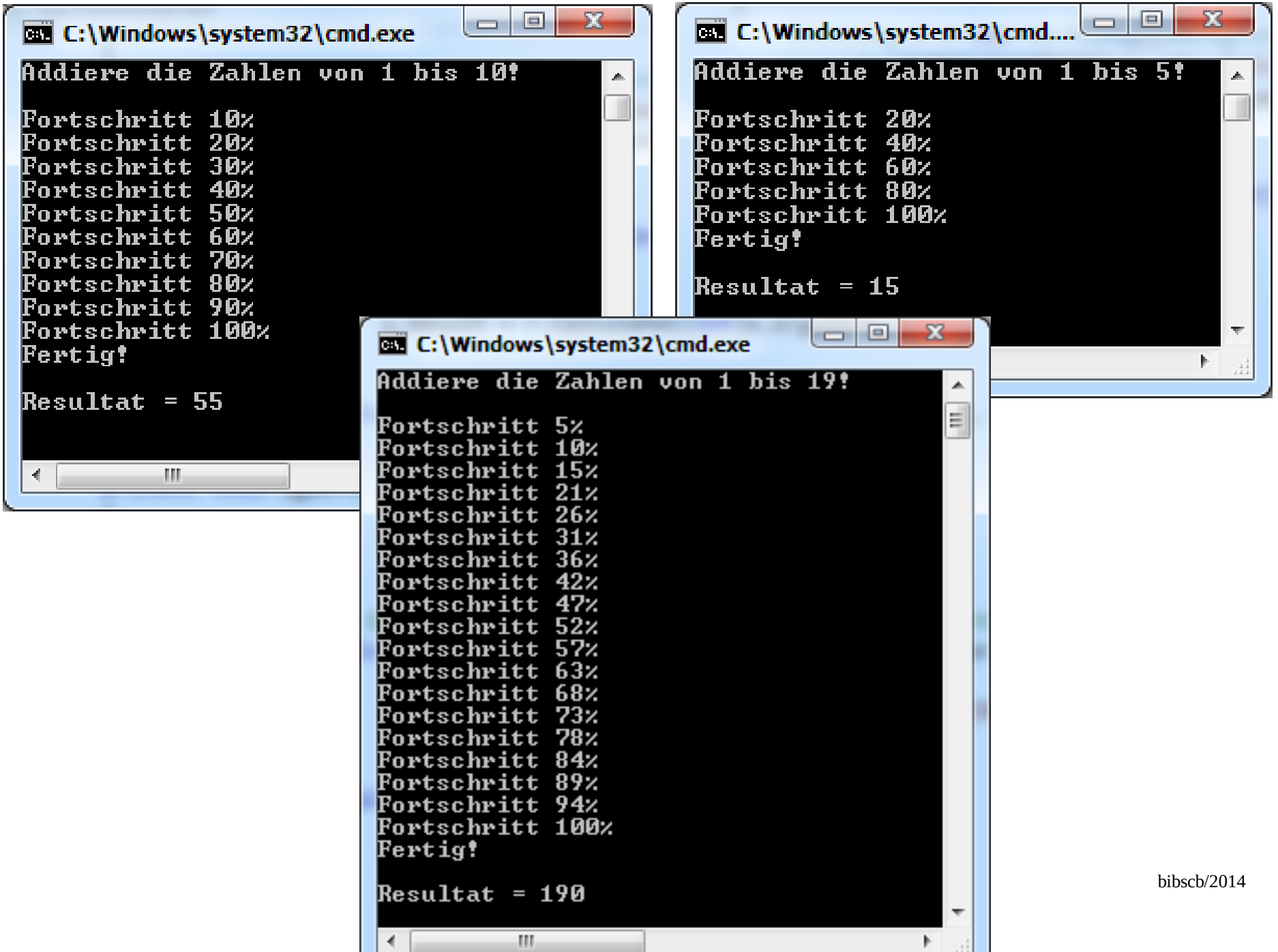
```
public class RechnungTest {
    private bool workCompleted = false;
    private int result;
    public RechnungTest(int digits){
        BackgroundWorker bgWorker = new BackgroundWorker();
        bgWorker.DoWork += new DoWorkEventHandler(DoWork);
        bgWorker.RunWorkerCompleted +=
            new RunWorkerCompletedEventHandler(WorkCompleted);
        bgWorker.ProgressChanged +=
            new ProgressChangedEventHandler(ProgressChanged);
        bgWorker.RunWorkerAsync(digits);
        bgWorker.WorkerReportsProgress = true;
        while (!workCompleted) {
            Thread.Sleep(500);
        }
        Console.WriteLine("Resultat = " + result);
        Console.Read();
    }
```

```
    private void  DoWork(object sender, DoWorkEventArgs e)        {
        Rechnung r = new Rechnung((BackgroundWorker)sender);
        Console.WriteLine("Addiere die Zahlen von 1 bis {0}!\n", e.Argument);
        e.Result = r.Calculate((int)e.Argument);
    }

    private void WorkCompleted(object sender, RunWorkerCompletedEventArgs e){
        Console.WriteLine("Fertig!\n");
        result = (int)e.Result;
        workCompleted = true;
    }

    // Der Ereignishandler aktualisiert den Fortschritt
    private void  ProgressChanged(object sender, ProgressChangedEventArgs e) {
        Console.WriteLine(String.Format("Fortschritt {0}% ", e.ProgressPercentage));
    }

    public static void Main(string[] args){
        new RechnungTest(new Random().Next(20));
    }
}
```


C:\Windows\system32\cmd.exe
Addiere die Zahlen von 1 bis 10!
Fortschritt 10%
Fortschritt 20%
Fortschritt 30%
Fortschritt 40%
Fortschritt 50%
Fortschritt 60%
Fortschritt 70%
Fortschritt 80%
Fortschritt 90%
Fortschritt 100%
Fertig!
Resultat = 55
C:\Windows\system32\cmd....
Addiere die Zahlen von 1 bis 5!
Fortschritt 20%
Fortschritt 40%
Fortschritt 60%
Fortschritt 80%
Fortschritt 100%
Fertig!
Resultat = 15
C:\Windows\system32\cmd.exe
Addiere die Zahlen von 1 bis 19!
Fortschritt 5%
Fortschritt 10%
Fortschritt 15%
Fortschritt 21%
Fortschritt 26%
Fortschritt 31%
Fortschritt 36%
Fortschritt 42%
Fortschritt 47%
Fortschritt 52%
Fortschritt 57%
Fortschritt 63%
Fortschritt 68%
Fortschritt 73%
Fortschritt 78%
Fortschritt 84%
Fortschritt 89%
Fortschritt 94%
Fortschritt 100%
Fertig!
Resultat = 190