Projektový seminář 2: Vlákna

Petr Krajča

Katedra informatiky
Univerzita Palackého v Olomouci

Motivace

Problém č. 1

- Jak zařídit, aby uživatelské rozhraní reagovalo i v průběhu časově naročného výpočtu?
- $\Longrightarrow$ Použitím vláken.

Problém č. 2

- S uživatelským rozhraním lze manipulovat pouze z jednoho vlákna.
- $\Longrightarrow$ Existují komponenty řešící tento problém.

Problém č. 3

- Řešení existuje celá řada.
- $\Longrightarrow$ demonstrační příklady pro Javu (Swing) a .NET (Windows Forms).

Obecné řešení

- vytvoří se objekt ,,Worker"
- mající dvě metody
  1. jedna provádí výpočet (nesmí manipulovat s UI)
  2. je zavolána po skončení výpočtu a získá výsledek (může manipulovat s UI)
- s tímto objektem se manipuluje pomocí dvou metod
  1. zahájení výpočtu
  2. přerušení výpočtu

Spolupráce s UI

- uživatel vyvolá náročný výpočet
- UI se upraví adekvátním způsobem (signalizuje, že probíhá výpočet; zpřístupní se tlačítko přerušení výpočtu)
- worker zahají výpočet
- worker ukončí výpočet a výsledek promítne do UI a zbytku aplikace

Java: SwingWorker

- abstraktní třída `SwingWorker<T,V>` s metodami:
  - `protected abstract T doInBackground()`
  - `protected void done()`

- parametry třídy:
  - `T` – udává typ návratové hodnoty funkce, která se stará o výpočet
  - `V` – typ hodnot používaných k předávání informací o mezivýsledcích (pokud tuto funkcionalitu nepoužíváte, použijte `Void`)

- metody ovládající výpočet:
  - `public void execute()` – spustí výpočet
  - `public void cancel(boolean)` – přeruší výpočet
  - `public T get()` – slouží k získání výsledku výpočtu (např. v metodě `done()`)
  - `public boolean isCancelled()` – vrací `true`, pokud byl výpočet přerušen

```
public class Worker extends SwingWorker<Integer, Void> {
   private int arg;
   public Worker(int arg) {
      super();
      this.arg = arg;
   }

   @Override
   protected Integer doInBackground() throws Exception {
      return fib(this, arg);
   }

   @Override
   protected void done() {
      try {
         if (this.isCancelled()) displayAbort();
         else displayResult(this.get());
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
```

Java: Výpočet

```
public static int fib(SwingWorker<Integer, Void> sw, int n) {
   if (sw.isCancelled()) return -1;
   if (n == 0) return 0;
   if (n == 1) return 1;
   return fib(sw, n - 1) + fib(sw, n - 2);
}
```

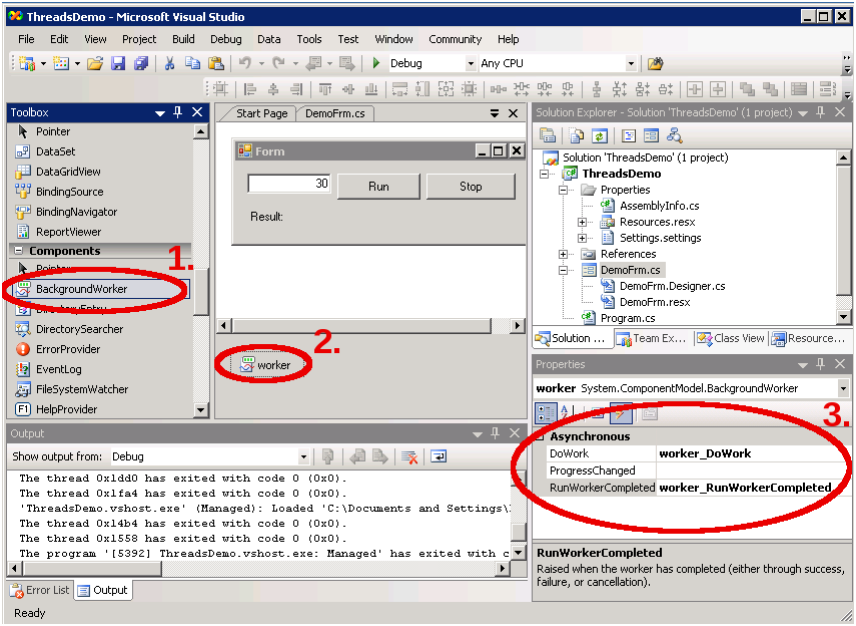
Java: Použití Workeru

```
btnStart.addActionListener(new ActionListener() {
   @Override
   public void actionPerformed(ActionEvent e) {
      final int arg = Integer.parseInt(txtInput.getText());
      adjustUI();
      sw = new Worker(arg);
      sw.execute();
   }
});

btnStop.addActionListener(new ActionListener() {
   @Override
   public void actionPerformed(ActionEvent e) {
      sw.cancel(true);
   }
});
```

.NET: BackgroundWorker

- `BackgroundWorker` – komponenta .NET frameworku podobná SwingWorkeru
- události
  - `DoWork` – je provedena, pokud je spuštěn výpočet
  - `RunWorkerCompleted` – je provedena, pokud je výpočet ukončen
- metody a vlastnosti
  - metoda `RunWorkerAsync(object)` – spustí výpočet a argument předá události `DoWork`
  - metoda `CancelAsync()` – přeruší výpočet
  - vlastnost `WorkerSupportsCancellation` – udává, jestli worker podporuje přerušení výpočtu (je potřeba nastavit na `True`)
  - vlastnost `CancellationPending` – signalizuje, že byla zavolána metoda `CancelAsync`
- Událost `DoWork`:
  - události `DoWork` je předán argument typu `DoWorkEventArgs` mající tyto vlastnosti:
  - `Argument` – obsahuje hodnotu se kterou byl výpočet zahájen
  - `Result` – ukládá se do ní výsledek výpočtu
  - `Cancel` – slouží k oznámení, že výpočet byl přerušen


ThreadsDemo - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Test Window Community Help
Debug
Any CPU
Toolbox
Pointer
DataSet
DataGridView
BindingSource
BindingNavigator
ReportViewer
Components
BackgroundWorker
DirectorySearcher
ErrorProvider
EventLog
FileSystemWatcher
HelpProvider
1.
Start Page
DemoFrm.cs
Form
30
Run
Stop
Result:
worker
2.
Solution Explorer - Solution 'ThreadsDemo' (1 project)
Solution 'ThreadsDemo' (1 project)
ThreadsDemo
Properties
AssemblyInfo.cs
Resources.resx
Settings.settings
References
DemoFrm.cs
DemoFrm.Designer.cs
DemoFrm.resx
Program.cs
Solution ...
Team Ex...
Class View
Resource...
Properties
worker System.ComponentModel.BackgroundWorker
3.
Asynchronous
DoWork worker_DoWork
ProgressChanged
RunWorkerCompleted worker_RunWorkerCompleted
RunWorkerCompleted
Raised when the worker has completed (either through success, failure, or cancellation).
Output
Show output from: Debug
The thread 0x1dd0 has exited with code 0 (0x0).
The thread 0x1fa4 has exited with code 0 (0x0).
'ThreadsDemo.vshost.exe' (Managed): Loaded 'C:\Documents and Settings\
The thread 0x14b4 has exited with code 0 (0x0).
The thread 0x1558 has exited with code 0 (0x0).
The program '[5392] ThreadsDemo.vshost.exe: Managed' has exited with c
Error List
Output
Ready

Ukázky C# (1/2)

```
private void worker_DoWork(object sender, DoWorkEventArgs e)
{
    BackgroundWorker bw = sender as BackgroundWorker;
    int o = (int)e.Argument;
    int r = fib(o, bw, e);
    e.Result = r;
}

private void worker_RunWorkerCompleted(object sender,
                                       RunWorkerCompletedEventArgs e)
{
    if (e.Cancelled) lbOutput.Text = "Result: N/A";
    else lbOutput.Text = "Result: " + e.Result;

    btnRun.Enabled = true;
    btnStop.Enabled = false;
}
```

Ukázky C# (2/3)

```
private static int fib(int n, BackgroundWorker bw, DoWorkEventArgs ev)
{
    if (bw.CancellationPending)
    {
        ev.Cancel = true;
        return -1;
    }
    if (n == 0) return 0;
    if (n == 1) return 1;
    return fib(n - 1, bw, ev) + fib(n - 2, bw, ev);
}
```

Ukázky C# (3/3)

```
private void btnRun_Click(object sender, EventArgs e)
{
    btnRun.Enabled = false;
    btnStop.Enabled = true;

    // spusti vypocet se zadanym argumentem
    worker.RunWorkerAsync(Int32.Parse(txtInput.Text));
}

private void btnStop_Click(object sender, EventArgs e)
{
    worker.CancelAsync();
}
```

Možnost informovat o průběhu výpočtu

SwingWorker

- metoda `protected void publish(V... chunks)` – umožnuje oznámit stav výpočtu z metody `doInBackground()`
- metoda `protected void process(List<V> chunks)` – umožňuje prezentovat data poslané metodou `publish()` v UI

BackgroundWorker

- událost `ProgressChanged`
- metoda `ReportProgress`

Alternativní řešení

Manuální vytvoření vlákna

- možnost vytvořit vlastní vlákno explicitně (instance třídy `Thread`)

Synchronizace

- vyvolání události v uživatelském rozhraní z druhého vlákna
- $\Longrightarrow$ předání informací vláknu starajícímu se o uživatelské rozhraní
- pokud vlákno chce zasahovat do UI je to potřeba v Javě provést přes volání `SwingUtilities.invokeLater(Runnable)`, resp. `Control.Invoke()` v .NETu, která zajistí provedení v kontextu vlákna starající se o UI

Přerušení výpočtu

- přerušení výpočtu je potřeba řešit pomocí sdílené proměnné (`volatile`), v případě složitější komunikace mezi vlákny je potřeba použít kritické sekce
- explicitní přerušení výpočtu voláním metody `interrupt()` nebo `abort()` třídy `Thread` se nedoporučuje
- možnost pozastavit běh vlákna

Řešení: Java (1/3)

```
public class FibThread extends Thread {
   private int arg;
   public volatile boolean stopRequest;

   public FibThread(int arg) {
      this.arg = arg;
   }

   public void run() {
      stopRequest = false;
      final int r = fib(this, arg);
      SwingUtilities.invokeLater(new Runnable() {

         public void run() {
            if (stopRequest) displayAbort();
            else displayResult(r);
         }
      });
   }
```

Řešení: Java (2/3)

```
btnStart.addActionListener(new ActionListener() {
   @Override
   public void actionPerformed(ActionEvent e) {
      int arg = Integer.parseInt(txtInput.getText()); // nacte vstup
      adjustGUI();
      fibThread = new FibThread(arg);
      fibThread.start();
   }
});

btnStop.addActionListener(new ActionListener() {
   @Override
   public void actionPerformed(ActionEvent e) {
      fibThread.stopRequest = true;
   }
});
```

Řešení: Java (3/3)

```
public int fib(FibThread thr, int n) {
   if (thr.stopRequest) return -1;
   if (n == 0) return 0;
   if (n == 1) return 1;
   return fib(thr, n - 1) + fib(thr, n - 2);
}
```

Řešení: .NET/C# (1/3)

```
private class FibCalc {
    public MethodInvoker done;
    public volatile bool StopRequest;
    private Control control;
    private int arg;
    public int result;

    public FibCalc(Control control, int arg)
    {
        this.arg = arg;
        this.control = control;
        StopRequest = false;
    }
    public void run()
    {
        int n = fib(this, arg);
        result = n;
        control.Invoke(done);
    }
}
```

Řešení: .NET/C# (2/3)

```
private void btnStart_Click(object sender, EventArgs e)
{
    btnStart.Enabled = false;
    btnStop.Enabled = true;

    int arg = Int32.Parse(txtInput.Text);

    fibCalc = new FibCalc(this, arg);
    fibCalc.done = computationDone;

    Thread thr = new Thread(fibCalc.run);
    thr.Start();
}
private void computationDone()
{
    if (fibCalc.StopRequest) lblOutput.Text = "Vysledek: N/A";
    else lblOutput.Text = "Vysledek: " + fibCalc.result;
    btnStop.Enabled = false;
    btnStart.Enabled = true;
}
```

Řešení: .NET/C# (3/3)

```
private static int fib(FibCalc fibCalc, int n)
{
    if (fibCalc.StopRequest) return -1;
    if (n == 0) return 0;
    if (n == 1) return 1;
    return fib(fibCalc, n - 1) + fib(fibCalc, n - 2);
}

private void btnStop_Click(object sender, EventArgs e)
{
    fibCalc.StopRequest = true;
    btnStart.Enabled = true;
    btnStop.Enabled = false;
}
```