Module 7
WPF a vlákna

MGR. TOMÁŠ HAVETTA - MCT

Obsah

- Asynchronní zpracování
- Vytvoření neblokovaného UI ve WPF

Blok 1: Asynchronní zpracování

- Základy vláken
- Dispatcher
- ThreadPool a Thread
- Task
- Task Parallel Library (TPL)

Základy vláken

- Každá WPF aplikace používá několik vláken
- Dispatcher hlídá **Thread Affinity**
  - WPF objekt může být použitý jen ve vlákně, v kterém byl vytvořen => porušení = výjimka
- Možnosti práce s vlákny
  - Dispatcher – zajistí spuštění ve správném vlákně
  - ThreadPool - nepoužívat
  - Task – moderní zástupce vlákna (neplatí Task == Thread)
  - Task Parallel Library
  - BackgroundWorker – lze použít, ale je zastaralý

Dispatcher

- Umožňuje nastavit prioritu pro metodu v UI vlákně

```
myControl.Dispatcher.BeginInvoke(
  DispatcherPriority.Background,
  new Action(UserInterfaceUpdate));
```

- DispatcherTimer zajistí spuštění akce v požadovaném intervalu v UI vlákně (negarantováno)

```
myTimer = new DispatcherTimer(
  DispatcherPriority.Background,
  myControl.Dispatcher);
myTimer.Interval = TimeSpan.FromSeconds(2);
myTimer.Tick += new EventHandler(
  delegate(object s, EventArgs a)
  {
    // Update the user interface.
  });
myTimer.Start();
```

ThreadPool a Thread

- Každá .NET aplikace při spuštění vytvoří ThreadPool a v něm vlákna (počet je dán dle počtu CPU a dostupné RAM)
- Vlákna se přidělují pro vykonání výpočtu. Po ukončení výpočtu se nelikvidují, ale vracejí do ThreadPoolu
- NEOVLÁDAT PŘÍMO, ale pomocí Task

Task

- K dispozici od .NET Framework 4
- Dnes primární třída pro asynchronní operace
- Task a Task<T>
- C# 5 přidal pro pohodlnou práci async/await
- Systém se snaží minimalizovat přepínání vláken

Task Parallel Library (TPL)

Sequential **foreach** loop

```
foreach (var item in source)
{
  DoSomething(item);
}
```

Sequential tasks

```
DoSomething();
DoSomethingElse();
```

Sequential LINQ

```
var evenNums =
  from num in source
  where Compute(num) > 0
  select num;
```

Parallel **foreach** loop

```
Parallel.ForEach(
  source,
  item => DoSomething(item));
```

Parallel tasks

```
Parallel.Invoke(
  () => DoSomething(),
  () => DoSomethingElse());
```

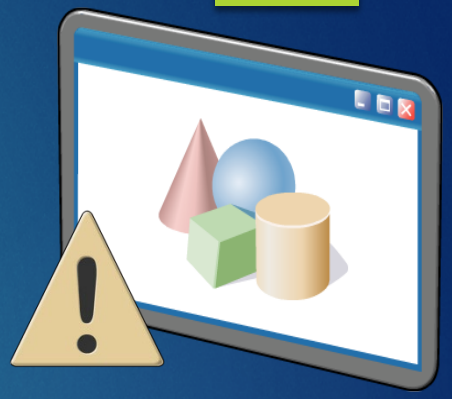
Parallel LINQ

```
var evenNums =
  from num in source.AsParallel()
  where Compute(num) > 0
  select num;
```

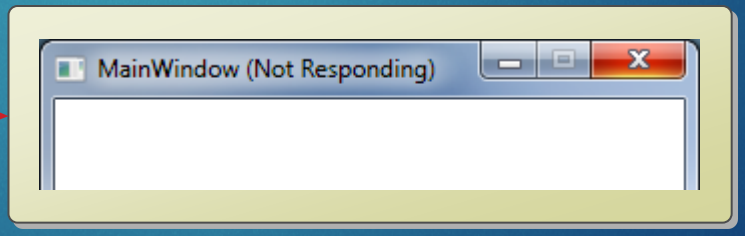
Blok 2: Vytvoření neblokovaného UI ve WPF

- Problémy Windows procesů
- Standardní postup zajištění funkčnosti oken

Problémy Windows procesů



- Pouze UI vlákno řeší události okna
- Jeho zablokování výpočtem či čekáním na HW nebo síť znamená problém
- Uživateli se aplikace jeví jako zablokovaná
- Aplikace může vypsat – Okno neodpovídá



Standardní postup zajištění funkčnosti oken

- Jakákoliv akce trvající déle než 0,1 s by se měla provádět v non-UI vlákně
- Ideálním je použít Task pro spuštění akce na pozadí
- Návrat zajistit pomocí async/await
- Využití více vláken NEMUSÍ zrychlit běh aplikace
- Pozor na Thread Safe kód

Lab: WPF vlákna

- Vytvořte aplikaci, která bude v pomocné třídě provádět dlouhý výpočet (alespoň 10 vteřin)
- Zajistěte pomocí Task a async/await plnou funkčnost okna i v době provádění výpočtu
- Informujte v ProgressBaru o průběhu výpočtu
- Implementujte možnost zrušení požadavku na výpočet