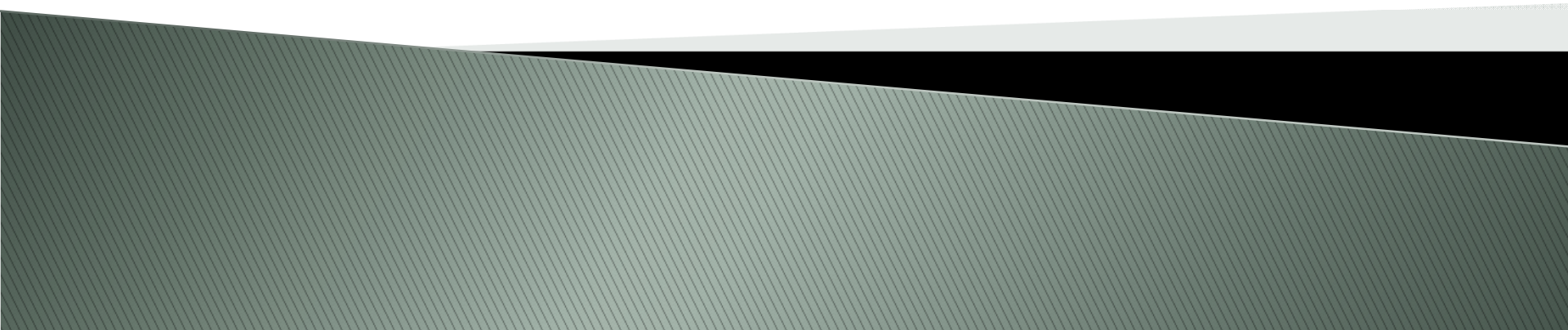


# Paralelné Programovanie

PV178 jaro 2011



# Motivace

- ▶ Prečo používať thready?
  - Izolovať kód od ostatného kódu
  - Zjednodušiť si písanie kódu
  - Dosiahnuť súbežné vykonávanie
- ▶ Ako často používať thready?
  - Lacnejšie ako process ale, nie sú zadarmo
  - Každý thread dobre zvážiť
  - Explicitne thready nepoužívať !!

# ThreadPool

- ▶ Optimalizované pridelovanie
- ▶ Recyklácia
- ▶ Heuristika
  - `ThreadPool.QueueUserWorkItem();`
  - Alebo!
  - Task Parallel Library

# Paralelné programovanie

- ▶ Výzva, vyžaduje prečítať viac ako jednu knihu o programovaní
- ▶ Synchronizace, false-sharing..
- ▶ Dekompozice:
  - Dátová (Data decomposition)
  - Úlohová (Task decomposition)
- ▶ Nástroj: Task Parallel Library

# Data parallelism

- ▶ AK nemáme závislosť medzi taskmi ale máme ich viacero rovnakých:

```
Parallel.ForEach(customers,  
    customer => ComputeStatistics(customer)..);
```

- ▶ Parallel.For, Parallel.Invoke(array of Actions)
- ▶ Ľahko sa používa ale:
  - Rozdelíme úlohy tak aby každý CPU pracoval, ale nie viac ako tak aby každý CPU pracoval
  - Rozdelíme úlohy tak aby v každej úlohe bolo čo najviac práce

# Tasky (TPL) v .NET 4.0

- ▶ Využívají ThreadPool, ale ponúkajú navyše:
  - **Stav operace**: task state– completed, faulted, waitingForChildren, cancelled
  - **Výsledok**: t.Result
  - **Štruktúru**: parent čaká na child tasky, alebo task.Wait()
    - `new Task<Int32>(Action<object>).Start();`
- ▶ WaitAll(), WaitAny(),

# Aggregate Exception

- ▶ Kolekce výnimiek z kolekce Taskov
- ▶ Umožňuje vrátiť Thread do ThreadPool
- ▶ InnerExceptions (nie InnerException)
  
- ▶ Vyhodená len pri Wait a Result, alebo pri Parallel metódach, inak ako Unobserved terminuje process

# Štruktúrovanie Taskov

- ▶ Začatie tasku okamžite po dokončení iného:

```
Task t = new Task<Int32>(n=>Sum(n));
```

```
t.ContinueWith(task =>  
    Console.WriteLine(task.Result),  
    TaskContinuationOptions);
```

- ▶ Child Tasky:
  - Parent task nie je Completed kým nie sú všetky child Completed (WaitingForChildrenToComplete)



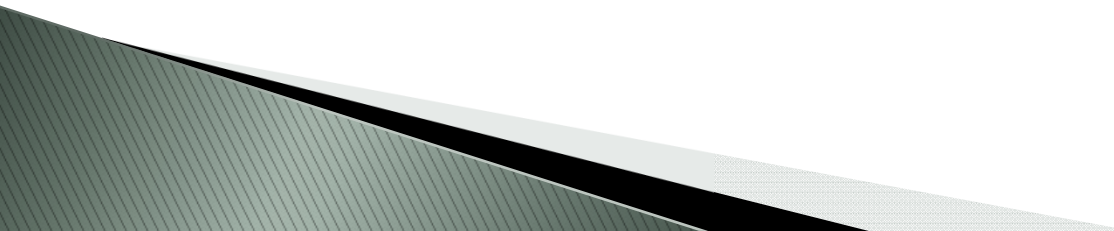
# TaskCreationOptions

- ▶ Parameter konstrukturu Tasku
- ▶ Výčet:
  - PreferFairness
  - LongRunning
  - AttachedToParent
- ▶ Tiež TaskContinuationOptions
  - ExecuteSynchronously
  - NotOnFaulted
  - NotOnCancelled
  - NotOnRanToCompletion

# Task Factories

- ▶ Niekedy chceme vytvárať viac Taskov s podobnými parametrami
  - Napríklad aby všetky boli child a continuation
- ▶ Použijeme TaskFactory
  - Všetky tasky z jednej faktory zdieľajú rovnaké nastavenie

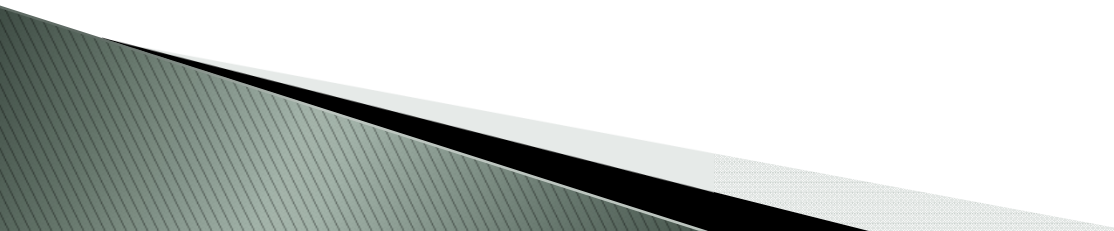
# Task Status

- ▶ Created
  - ▶ WaitingForActivation
  - ▶ WaitingToRun
  - ▶ Running
  - ▶ WaitingForChildrenToComplete
  - ▶ RanToCompletion
  - ▶ Canceled
  - ▶ Faulted
- 

# Cooperative Cancellation

- ▶ Kooperatívny vzor pre zrušenie prebiehajúcich taskov
- ▶ CancellationTokenSource a Property Token
- ▶ Token signalizuje po volaní Cancel na CTS
  
- ▶ Periodicky testujeme Token na IsCancellationRequested

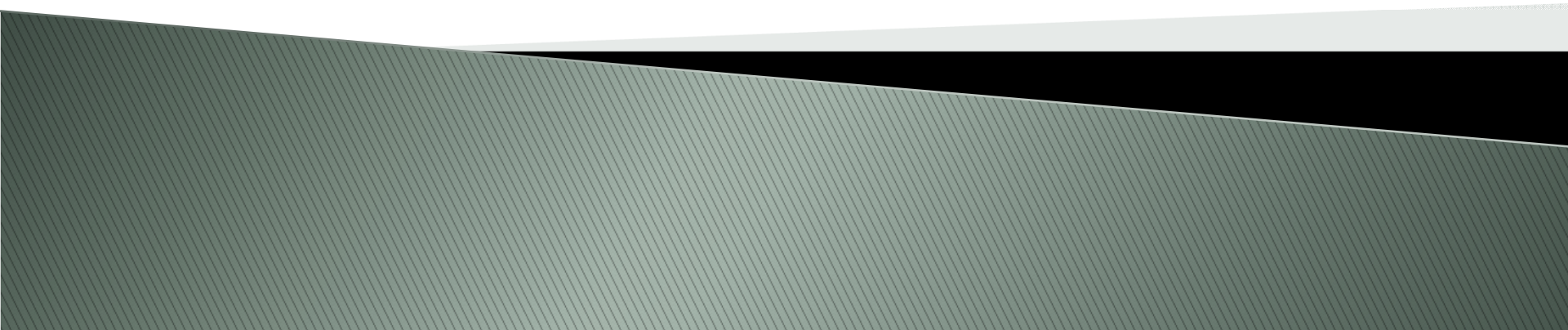
# PLINQ

- ▶ Parallel LINQ to Objects
  - ▶ Deklaratívne použitie Parallel metód
  - ▶ Pomocou extension metódy AsParallel konvertujeme z IEnumerable na ParallelQuery
  - ▶ Výsledok môžeme paralelne spracovať pomocou extension metódy ForAll
- 

# Timers

- ▶ Ak potrebujeme periodicky alebo v daných časových momentoch volať metódy
- ▶ Timer použije ThreadPool vlákno k vykonaniu metódy v danom čase
- ▶ Namespace `System.Threading.Timer`
  - `public Timer(callback, state, dueTime, period)`

# Windows Communication Foundation



# WCF

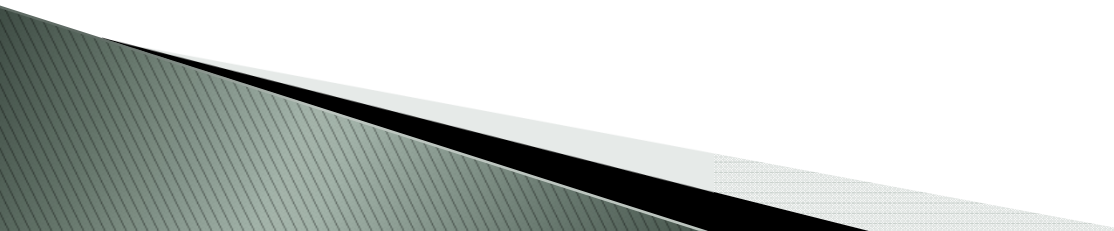
- ▶ Nástroj pre vývoj a nasadzovanie služieb vo Windows
- ▶ Nachádza sa v System.ServiceModel
- ▶ Framework pre budovanie aplikácií, ktoré komunikujú
- ▶ Používame od medziprocesovej po internetovú komunikáciu



# WCF Služba

- ▶ Služba je zverejnené jednotka funkcionality
- ▶ Služby používame pre vývoj distribuovaných aplikácií, princíp SOA
- ▶ Služby a klienti interagujú cez posialanie správ
- ▶ Správy sú zväčša SOAP
- ▶ Používajú mnoho transportov, nie len http
- ▶ Definícia služby a technológie/standardu pre prenos správ služby je oddelená – transparentná
- ▶ Poskytuje Metadata – môže komunikovať aj s non-WCF službami a klientmi

# WCF Klient

- ▶ Klient vždy používa proxy pre komunikovanie so službou
  - ▶ Proxy poskytuje rovnaké metódy ako služba – zaobaluje serializaci a réžiu komunikace
  - ▶ Svcutil.exe vygeneruje kód klientskej proxy zo skompilovaného kódu služby
- 

# ABC

- ▶ Vytvoriť WCF službu znamená definovať hlavne:
  - **A**dresu služby – URI
  - **B**inding – vybrať protokol (prípadne nastaviť)
  - **C**ontract – definícia zverejnenej funkcionality
- ▶ Týmto sme definovali **Endpoint**

# Adresa

- ▶ Určuje kde sa služba nachádza
- ▶ `<scheme>://<domain>[:port]/[path]`
  - Príklady:
    - `http://example.com/ServiceA`
    - `net.msmq://localhost`
- ▶ Musí byť unikátna pre každý endpoint

# Binding

- ▶ Určuje ako pristupovať k službe
- ▶ Transportný protokol:
  - TCP
  - HTTP
  - Name pipe
  - MSMQ
- ▶ Kódovanie a formát správy
- ▶ Bezpečnosť
- ▶ ....

# Service Contract

- ▶ Určuje čo služba poskytuje
- ▶ Definuje operace zverejnené službou
- ▶ Pomocou atribútov
  - Trieda ktorá predstavuje službu sa označí
    - [ServiceContract]
  - Metódy tejto triedy ktoré chceme zverejniť
    - [OperationContract]

# Data Contract

- ▶ Ak potrebujeme aby služba používala komplexný typ
- ▶ DataContractSerializer serializuje náš typ do XML pre prenos
- ▶ Definujeme pomocou atribútov podobne ako pri Service Contract
  - [DataContract]
  - [DataMember]

# Hosting

- ▶ IIS
- ▶ Narozdiel od webových služieb WCF služby môžeme hostovať aj vo vlastnom procese
  - Class ServiceHost
- ▶ WCF služba môže byť stavová
- ▶ Endpointy a ostatné nastavenia sa dajú zadať aj v konfigurácii
  - VS2010 obsahuje konfiguračné klikátko