An Introduction to Multi-Threaded Programming under .NET

BY

PRASEED PAI K.T.

KMUG meetup @ UST Global, Trivandrum

13th December 2008

http://k-mug.org/content/UGMeet131208.aspx

What is a Thread ?

- A Thread is a path of execution within a Process
- A Process can have one or more threads
- All the threads in a process executes within the context of the Process
- Foreground Threads vs Background Threads

Why Should I use Threads ?

- Resource Utilization - While Disk drives are accessed CPU might be idle. Multiple Threads will keep CPU busy.
- Responsiveness from the UI.
- Multi-Core and Multi-Processor machines
- ASP.net engine , SQL server engine , IE browser are some programs which exploits Multi Threading.

How do I create a Thread in .net ?

- Write a Method which matches any of the following delegates
- Void ThreadRoutine()
- Void ThreadRoutine(Object param)
- Instantiate Thread Object with a method matching above signature and call start method

Memory Management

- Threads have private stack
- Heap is shared
- At a granular level , local variable semantics is similar to sequential programs
- Static variables and Global variables are shared

Thread Local Storage

- TLS overview
- How do i create a TLS variable and use them?
- Possible Usage Scenarios

Synchronization

- Critical regions of the code needs to be serially executed to avoid memory updation anomalies.
- Synchronization Primitives
- Locks , Mutex , Semaphore , Monitors
- Synchronized Attribute

Safe Termination of Threads

- Thread.Interrupt
- Thread.Abort
- Safe cancellation

Performance considerations

- CPU bound programs
- I/O bound programs
- CPU bound vs I/O bound
- Locks should be granular
- Thread creation and pooling

Race , DeadLocks and other issues

- What is a race condition ?
- Dead Lock anti-pattern
- Multi-Core issues
- Incorrect coding can lead to crashes

Some real life programs

- Web Server
- Parellel For Loop emulation

Q&A

- Any Questions ?
- END