Innovating Learning Mentoring

Upcoming Presentations

Date	Time	Topic	Presenter
March 12	4-6 PM	Distributed Programming Using .NET Remoting	Lester Temple of Landl Softwares
April 4	4-6 PM	Reflection in .NET	Andrew Troelson of Intertech
May 2	4-6 PM	C# and Flash	Rick Waldvogel of Motivaction
June 6	4-6 PM	Mobil web development with ASP.NET	Umer Faruq

User Group's Web Site

- New web site for the user group.
  - http://www.ilmservice.com/twincitiesnet
  - Bunch of new features
    - List of latest news and events
    - Details of upcoming presentations
    - Code Zone – code download area
    - Presentations submitted by the user group's members
    - Threaded discussion lists. The more you use it, the better it gets ☺
  - Volunteers needed to provide content. Send email to farhan@ilmlearning.com

Attend Get.NET! For Free

- Drawing for a free Get.NET! seminar
  - Remember to turn-in your evaluation form
  - Drawing will be conducted after the presentation
  - One lucky person will walk away with free registration to the Get.NET! seminar on March 6th
  - Everyone else gets a 25% discount by using the coupon code “tcnug” during registration

Multi Threading

- What is multi threaded programming
  - Allows you to run several sections of your code simultaneously
  - On a single CPU machine
    - Operating system balances the use of processor among all threads
    - Operating system simulates parallel processing by constantly switching between threads

Multi Threading

- When to use multi threading ?
  - When you need to quickly respond to users' interactions
    - You can leverage .NET Remoting or ASP.NET web service to distribute processing load to multiple computers
  - Consider multi threading when you need to process several independent transactions

Multi Threading

- Meet Minne-500
  - Simple car racing game.
  - An old car “Oldie” races with a new car “newbie”
  - Shows multi-threaded programming
  - Change thread priorities and watch its effect

Multi Threading

- Considerations while using multiple threads
  - Creates memory overhead because the context for each thread needs to be saved separately
  - Creates more work for the processor because it needs to switch frequently between multiple threads of execution
  - Can slow down your application if operated in a single processor environment
  - Multi threading works best in parallel processor systems, or in a distributed computing environment

Multi Threading

- Creating threads
  - Using ThreadPool class
    - Contained inside System.Threading namespace
    - Only contains static methods
    - By default, contains a pool of 25 threads
    - Processing can not be aborted after its started
    - Can not set priority on the thread
    - Use “QueueUserWorkItem” method to start parallel processing
    - “QueueUserWorkItem” receives a “WaitCallBack” delegate.
    - “WaitCallBack” delegate receives an object as a parameter, which can be used to pass state info.

Multi Threading

- Creating threads
  - Creating custom threads
    - Threads are created by instantiating Thread class
    - Contained in System.Threading namespace
    - Complete control over prioritizing, aborting threads.
    - Thread class receives a “ThreadStart” delegate as a parameter to the constructor
    - “ThreadStart” delegate does not receive any parameter and does not return any value

Multi Threading

- Starting a thread
  - Use the Start method of the Thread class to start its execution
- Stopping a thread
  - Use the Abort method of the Thread class to stop its execution
- Handling thread abortion
  - When a thread is aborted, .NET runtime throws “ThreadAbortException”
  - This exception can be handled to perform any necessary cleanup

Multi Threading

- Suspending a thread
  - Use the “Suspend” method of the Thread class to halt its execution. Its like pushing the pause button on VCR.
- Resuming a thread
  - Use the “Resume method of the Thread to resume its execution.
- Sleeping a thread
  - Use the static method “Sleep” of the Thread class to cause the thread to become dormant for a specified period of time.

Multi Threading

- Waiting for another thread
  - Use the “Join” method of the Thread class to specify that you intend to wait for another thread to finish running.
- Synchronizing threads
  - Meet the “Thread Pull”, designed to simulate multiple threads working together
  - Use Interlocked class for simple increment and decrement operations
  - Use “lock” keyword to synchronize access to a code block.
  - This keyword will cause all threads accessing the code block to queue up and execute in a sequence

Multi Threading

- Synchronizing threads
  - Using Monitor
    - Use the “Wait” method of the Monitor class to suspend a thread until another event occurs in the system.
    - Use the “Pulse” or “PulseAll” method of the Monitor class to activate thread(s) sitting in wait mode.