

# Threading Tips

## 10 Best Practices and Tips

# Tip 10 – Control.Invoke

- Update a Control only from the thread it was created on.
  - Detect if invoke is required.
  - Use delegate to forward to Control's thread.
  - InvalidOperationException thrown if updating from wrong thread.

.NET 2.0

# Tip 9 – Control.BeginInvoke

- Delegate's BeginInvoke/EndInvoke
  - Required.
  - Not calling EndInvoke will leak resources.
- Control's BeginInvoke/EndInvoke
  - Only required to obtain return value.
  - Won't leak resources if not called.

# Tip 8 – Background Thread

- Understand the semantics of thread types.
  - Foreground
    - CLR cannot end process until all foreground threads terminate.
  - Background
    - Stopped – without notification - once all foreground threads have terminated.
    - Set `Thread.IsBackground` to true.

# Tip 7 – Thread.Interrupt

- ThreadInterruptedException may not get thrown.
- Exception is throw if the thread
  - In a wait state.
  - Or sometime later if enters a wait state.
  - Otherwise, no exception is thrown.

# Tip 6 – Thread Yield

- Thread work should be divided into small chunks.
- Foreach chunk thread notifications should be checked.
  - Time to quit?
- Foreach chunk yield your thread.
  - Enter a wait state for at least 1ms.

# Tip 5 – lock semantics

- C# lock statement is compiler sugar for `Monitor.Enter` and `Monitor.Exit`
  - Requires a reference to use as lock.
  - Compiler checks that specified lock is not a value type.
    - Compiler does not check `Monitor.Enter/Exit`
  - object lock is released regardless if exception is thrown.

# Tip 4 – Monitor.TryEnter

- Consider Monitor.TryEnter instead of Monitor.Enter.
  - If lock cannot be obtained do not enter critical code section.
  - Reduces deadlock potential.
  - Adds coding complexity.
    - Fallback if lock cannot be obtained.

# Tip 3 – locks are public

- lock can be associated with any reference type.

- Don't lock on Type

```
lock(typeof(myType)) { /* .. */ }
```

- Don't lock on "this"

```
lock(this) { /* .. */ }
```

- Don't lock on string literals

```
lock("myString") { /* .. */ }
```

## Tip 2 – AvoidMethodImpl for sync

- Attribute found in namespace
  - System.Runtime.CompilerServices
- Can be applied to methods and properties.
  - Either instance or static
- Instance locks obtain with  
`lock(this)`
- Static locks obtained with  
`lock(typeof(yourMethodsClass))`

# Tip -1 – Double Check Lock

- Used when lazy creating a Singleton.
  - Increases performance since less locking.
  - Cool factor is high.
- There is an easier way
  - Just as fast as double check technique.
  - Simpler to code.