MODULE 17

MULTITHREADING AND WPF

MODULE TOPICS

WPF Threading Model
Thread Safety
The Dispatcher Object
Using Async and Await

WPF THREADING MODEL

- Multithreading should be used to keep Windows applications responsive
- Do not block the UI thread with possibly long running synchronous operations

WPF THREADING MODEL

- Console application, Windows Services, and ASP.NET applications do not impose any kind of threading model
  - Any thread can do whatever it wants when it wants
- Windows Forms, WPF, and Silverlight applications impose a threading model where the thread that created the window is the only thread allowed to update the window
  - Thread pool threads must have the GUI thread update the UI

THE DISPATCHER

- A dispatcher manages the work that takes place in a WPF application
- Owns the application thread and manages a queue of work items
- You can retrieve the dispatcher for the current thread using the static Dispatcher.CurrentDispatcherproperty
- Once you have a dispatcher, you can call BeginInvoketo marshal code to the dispatcher thread

THE DISPATCHER

```
Dispatcher.BeginInvoke(
    (Action)(() =>
        {
            // Notify user
            StatusTextBlock.Text =
            string.Format("Loop # {0}", loopIndex++);
        }
    )
);
```

ASYNC AWAIT

```
public static async Task<user[]> GetUsersAsync(IEnumerable<int> userIds)
{
    var getUserTasks = userIds.Select(id => GetUserAsync(id));
    return await Task.WhenAll(getUserTasks);
}

</int></user[]>
```

ANY QUESTIONS?