QUALITY. PRODUCTIVITY. INNOVATION.

. Endava .Net Core Training

Processes, Threads and Asynchronous Programming

Agenda

Processes, Threads and Asynchronous Programming

- **Processes, Threads and Asynchronous Programming**
- **Background Worker Class**
- **Multi-threading Considerations. Synchronization**
- **Concurrent collections.**
- **Parallel Loops**
- **Task library**
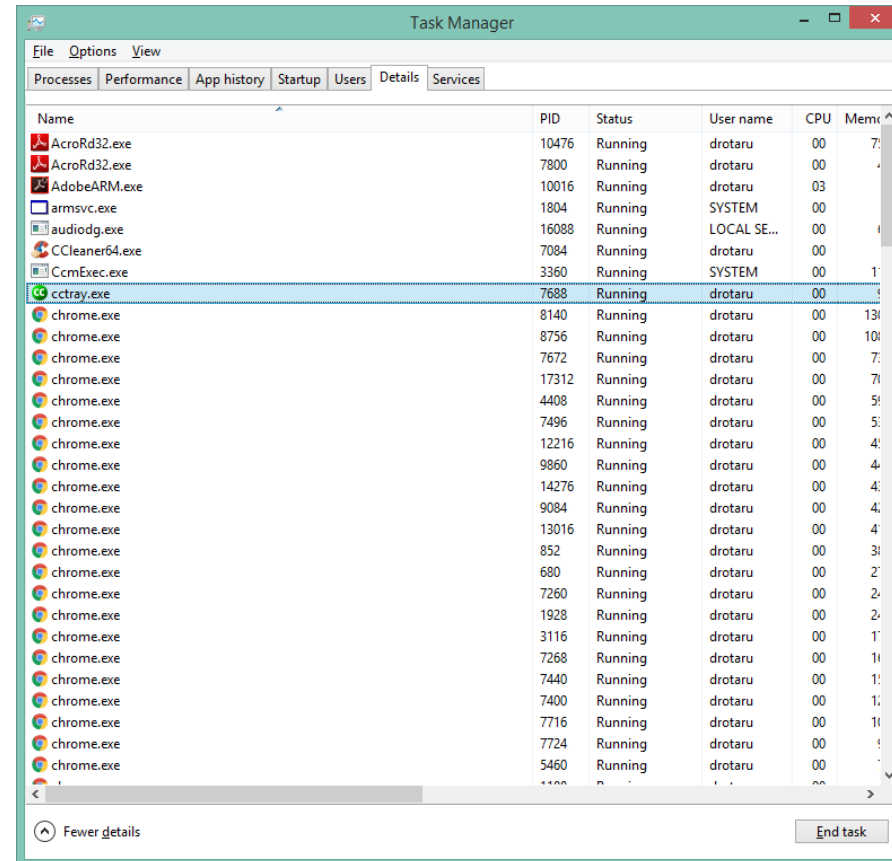- **Async/Await**
- **Timer**

Content

1. **Processes, Threads and Asynchronous Programming**
   1. What are Processes ?
   2. What is a thread?
   3. Thread class. Creating and executing a thread.
   4. BackgroudWorker Class
   5. Multi-threading and synchronization
   6. Locking. Thread safe.
   7. Concurrent collections.

QUALITY. PRODUCTIVITY. INNOVATION.

Processes

What is a process

- **A process is the set of resources that comprise a running program**

- **A process contains only a single thread, which executes from the beginning of the program to the end**
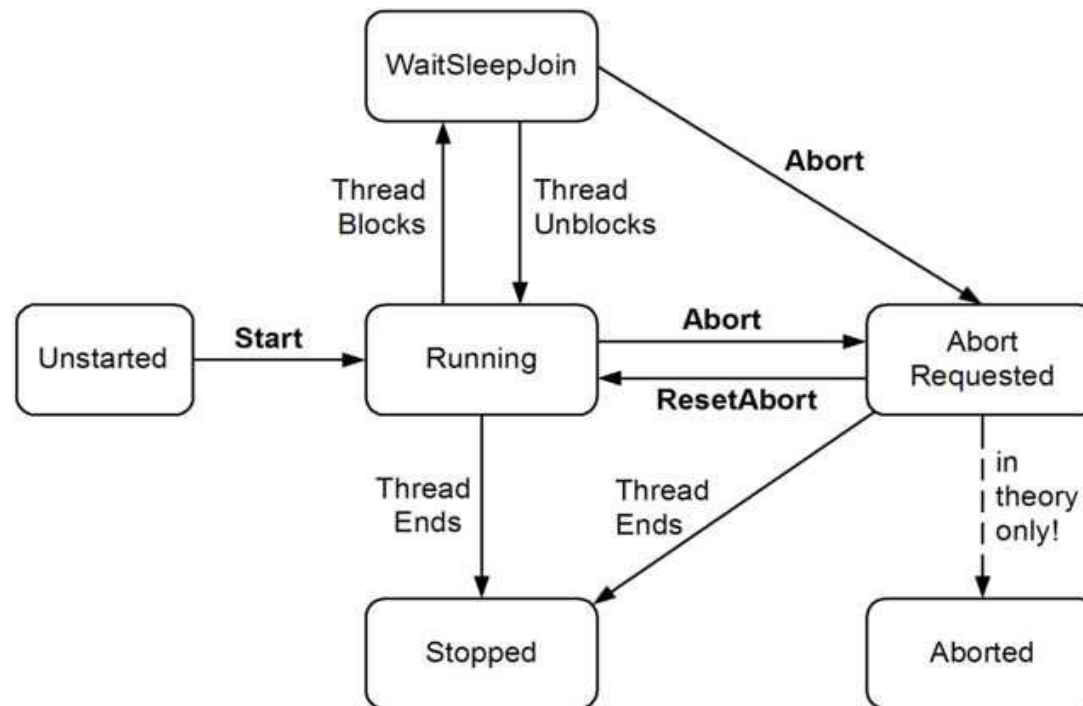
Threads

What is a thread

- **A thread is defined as the execution path of a program**

- **More than one thread can execute code inside the same process.**

- **More threads do not mean more performance (more resource consumed, context switching, complexity)**

Threads

With threads you can:

- **Maintain responsiveness**

- **Perform operations that consume a large amount of time without stopping the rest of the application**

- **Distinguish tasks of varying priority**

- **Parallel programming**

- **Allowing requests to be processed simultaneously (ASP.NET, WCF, Web Services)**

Thread pool

**Demo creating and starting threads.**

Threads

Thread class implements various methods & properties that allows to manipulate concurrently running threads

- **CurrentThread**

- **IsAlive**

- **IsBackground**

- **Name**

- **Priority**

- **ThreadState**

- **Suspend**

- **Resume**

- **Abort**

QUALITY. PRODUCTIVITY. INNOVATION.

BackgroundWorker class

- **BackgroundWorker class makes it simple to perform a task in the background on a separate thread**
- **Designed primarily for GUI programming (Windows Forms and WPF)**

QUALITY. PRODUCTIVITY. INNOVATION.
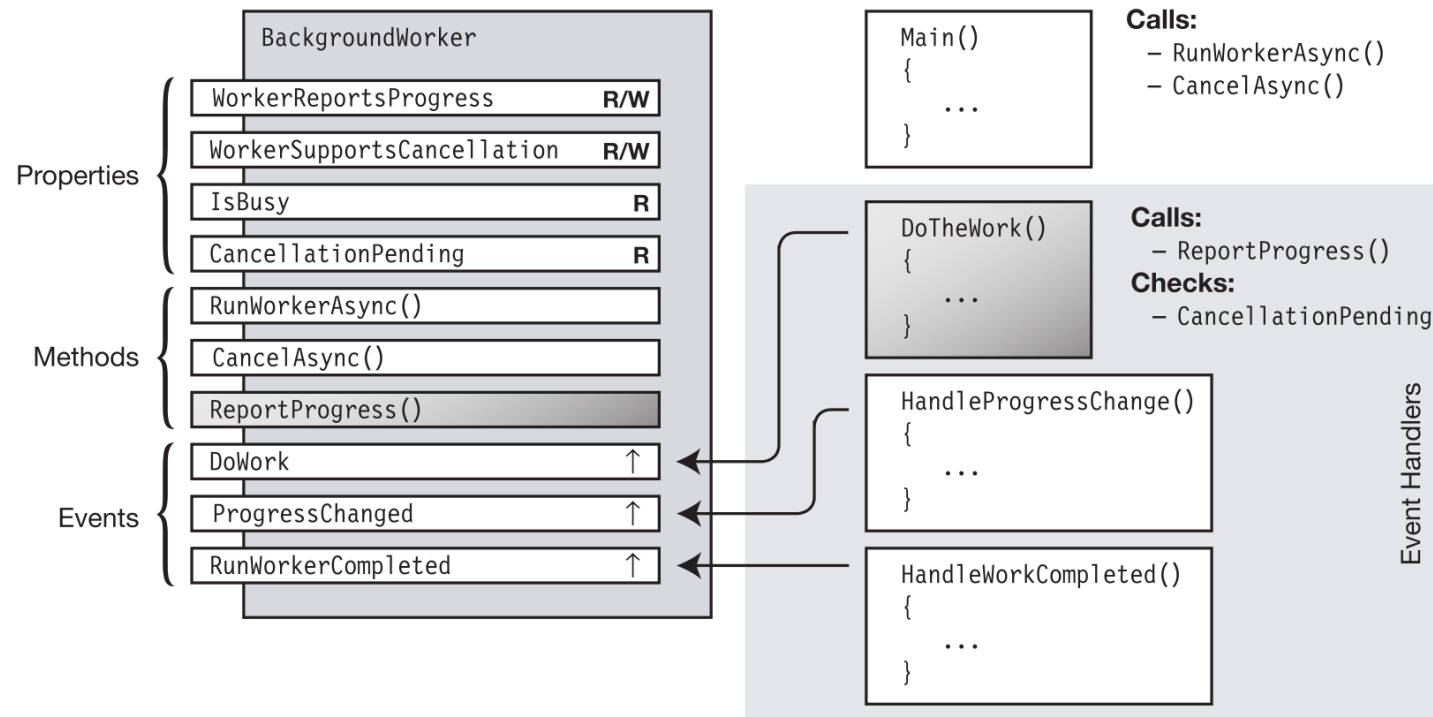
Multithreading. Synchronization.

Multithreading considerations

- **Multithreading can increase complexity - interaction between threads**

- **Threading also incurs a resource and CPU cost in scheduling and switching threads**

- **It can even slow down performance if used excessively or inappropriately**

Demo

Multithreading. Synchronization.

Synchronization

- **Lock / Monitor**

- **System.Threading.Interlocked**

- **Mutex – cross process**

- **Semaphore**

Demo

Concurrent collections

- **ConcurrentStack<T>**

- **ConcurrentQueue<T>**

- **ConcurrentBag<T>**

- **BlockingCollection<T>**

- **ConcurrentDictionary<TKey,TValue>**

Insert headline here
Use picture full slide from gallery on **public server**

Dan Rotaru

.Net Developer

✉ Dan.rotaru@endava.com

📞 +xx xxx xxx xxx

Ⓢ en_danrotaru

QUALITY. PRODUCTIVITY. INNOVATION.