

Service Base

Contributed by Phillip Brown
Saturday, 23 September 2006
Last Updated Monday, 06 November 2006

```
public abstract class Base : ServiceBase {
    protected Thread m_ServiceThread;
    protected ManualResetEvent m_ShutdownEvent;
    protected TimeSpan m_ServiceDelay;

    public Base() {
        // Set our service loop time to a default of 1 second
        // this can be changed by the property below
        // this may later be defaulted to a larger value
        m_ServiceDelay = new TimeSpan(0, 0, 1);
    }

    public TimeSpan Delay {
        set { m_ServiceDelay = value;}
    }

    protected override void OnStart(string[] args) // We are overriding the OnStart method in our ServiceBase base.
    {
        // create our threadstart object to wrap our delegate method
        ThreadStart _ThreadStart = new ThreadStart(this.ServiceMain);

        // create the manual reset event and
        // set it to an initial state of unsignaled
        m_ShutdownEvent = new ManualResetEvent(false);

        // create the worker thread
        m_ServiceThread = new Thread(_ThreadStart);

        // go ahead and start the worker thread
        m_ServiceThread.Start();

        // call the base class so it has a chance
        // to perform any work it needs to
        base.OnStart(args);
    }

    protected override void OnStop() {
        // signal the event to shutdown
        m_ShutdownEvent.Set();

        // wait for the thread to stop
        m_ServiceThread.Join((int)m_ServiceDelay.TotalSeconds);

        // call the base class
        base.OnStop();
    }

    protected virtual void ServiceMain() {
        bool bSignaled = false;

        Initialize();

        while (true) {
            // wait for the event to be signaled
            // or for the configured delay
            bSignaled = m_ShutdownEvent.WaitOne(m_ServiceDelay, true);

            // if we were signaled to shutdown, exit the loop
            if (bSignaled == true)
                break;
        }
    }
}
```

```
        // let's do some work
        ServiceThread();
    }

    Dispose();
}

protected virtual void ServiceThread() {
    return;
}

protected virtual void Initialize() {
    return;
}

protected virtual new void Dispose() {
    return;
}
}
```