A Cancellable ProgressBar While Processing on a Background Thread

Start

```
//File:Window.xaml.cs
using System.ComponentModel;
using System.Threading;
using System.Windows;
using System.Windows.Input;

namespace WpfApplication1
{
public partial class Window1 : Window
{
private BackgroundWorker worker = new BackgroundWorker();

public Window1()
{
InitializeComponent();
worker.WorkerReportsProgress = true;
worker.WorkerSupportsCancellation = true;
worker.DoWork += new DoWorkEventHandler(worker_DoWork);
worker.RunWorkerCompleted += new
RunWorkerCompletedEventHandler(worker_RunWorkerCompleted);
worker.ProgressChanged += worker_ProgressChanged;
```

```
}

private void button_Click(object sender, RoutedEventArgs e){
if(!worker.IsBusy)
{
this.Cursor = Cursors.Wait;
worker.RunWorkerAsync();
button.Content = “Cancel”;
}else{
worker.CancelAsync();
}
}

private void worker_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e)
{
this.Cursor = Cursors.Arrow;
if(e.Cancelled)
{
MessageBox.Show(“Operation was canceled”);
}
else if(e.Error != null)
{
MessageBox.Show(e.Error.Message);
}

button.Content = “Start”;
}

private void worker_DoWork(object sender, DoWorkEventArgs e)
{
for(int i = 1; i <= 100; i++) { if(worker.CancellationPending)
{ e.Cancel = true; return; } Thread.Sleep(100);
worker.ReportProgress(i); } } private void
worker_ProgressChanged(object sender, ProgressChangedEventArgs
e) { progressBar.Value = e.ProgressPercentage; } } } [/csharp]
```