

Forum:[Hack Forums](#)

Thread: [C# Multi-Threading With Bgworkers \[Intermediate Level Tutorial\]](#)

Pulser - October 20, 2010, 9:08 pm

C# Back Ground Workers

What Is It:

When you are trying to do multiple things in your program sometimes your program can get confused. It can crash, lag, or not work or execute code properly! This looks very unprofessional and it can enrage your users. By using a background worker you dedicate a separate thread to run a time consuming, or tricky bit of code!

Getting Started:

Begin by creating the following:

- A progress bar
- A button
- A Background Worker

Insert the following code into the background worker:

Code do

```
{  
    progressBar1.Value += 1;  
  
} while (progressBar1.Value < 1000);
```

This is obviously just a Do While Loop, if you do not know what this is, then please check out my other tutorials.

Double click the button and type:

Code `backgroundWorker1.RunWorkerAsync();` This simply begins doing whatever is in the `Do_Work` event of your background worker.

Now, when you attempt to access a control from a separate thread you will get an error, in order to prevent this, double click the form and insert this in `Form_Load`:

Code `CheckForIllegalCrossThreadCalls = false;` This is NOT a good way of doing this, it can lead to bad programming habits. However, for the sake of this tutorial we are going to do it like this.

This program is going to run the Do While loop on a separate thread, by doing this you prevent lag, you also have a completely separate thread dedicated to this loop!

Conclusion:

Multi-Threading is ALWAYS a good idea to use, every experienced programmer uses it, and the next time you create an application I suggest you try using it!

Code:

```
Code private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)  
{
```

```
do
{
    progressBar1.Value += 1;

} while (progressBar1.Value < 1000);
}

private void button1_Click(object sender, EventArgs e)
{
    backgroundWorker1.RunWorkerAsync();
}

private void Form1_Load(object sender, EventArgs e)
{
    CheckForIllegalCrossThreadCalls = false;
}
```