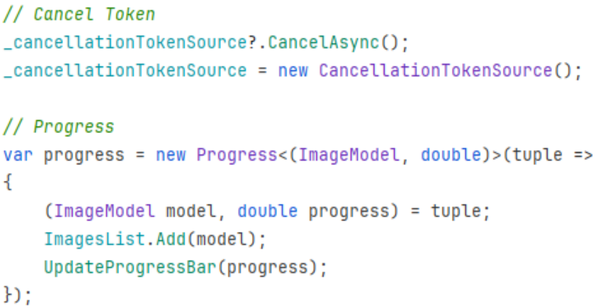
Async Image Processing Tool in C#

Sebastian Hollersbacher

Demo

Loading Images

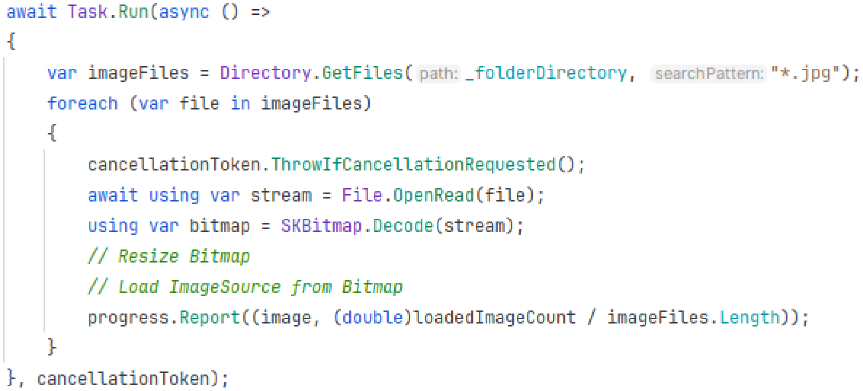


```
// Cancel Token
_cancellationTokenSource?.CancelAsync();
_cancellationTokenSource = new CancellationTokenSource();

// Progress
var progress = new Progress<(ImageModel, double)>(tuple =>
{
    (ImageModel model, double progress) = tuple;
    ImagesList.Add(model);
    UpdateProgressBar(progress);
});
```

Progress created in UI-Thread → runs in UI-Thread

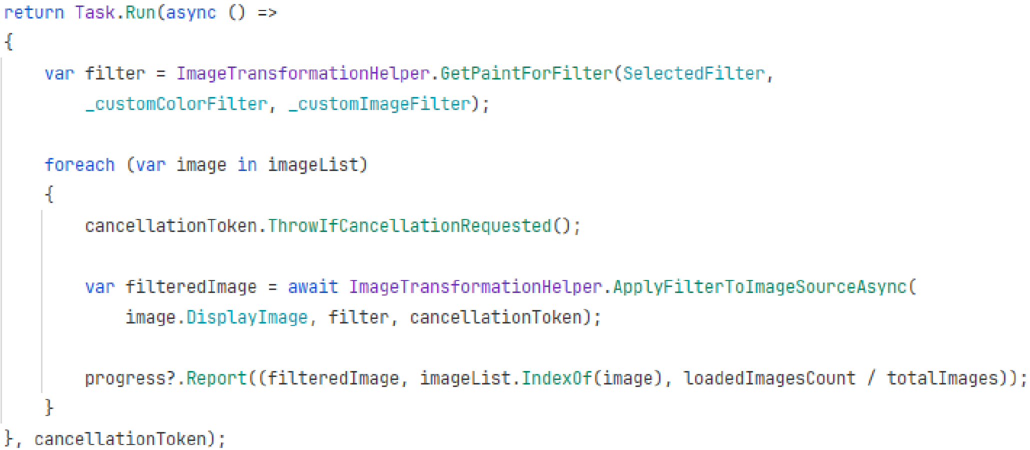
Loading Images



```
await Task.Run(async () =>
{
    var imageFiles = Directory.GetFiles( path: _folderDirectory,  searchPattern: "*.jpg");
    foreach (var file in imageFiles)
    {
        cancellationToken.ThrowIfCancellationRequested();
        await using var stream = File.OpenRead(file);
        using var bitmap = SKBitmap.Decode(stream);
        // Resize Bitmap
        // Load ImageSource from Bitmap
        progress.Report((image, (double)loadedImageCount / imageFiles.Length));
    }
}, cancellationToken);
```

Only 1 Background-Task to not overload UI

Filters



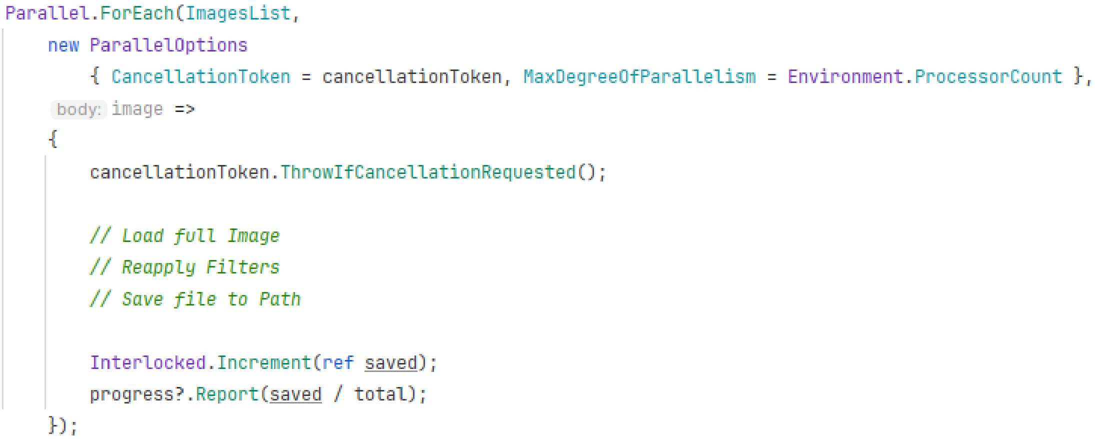
```
return Task.Run(async () =>
{
    var filter = ImageTransformationHelper.GetPaintForFilter(SelectedFilter,
        _customColorFilter, _customImageFilter);

    foreach (var image in imageList)
    {
        cancellationToken.ThrowIfCancellationRequested();

        var filteredImage = await ImageTransformationHelper.ApplyFilterToImageSourceAsync(
            image.DisplayImage, filter, cancellationToken);

        progress?.Report((filteredImage, imageList.IndexOf(image), loadedImagesCount / totalImages));
    }
}, cancellationToken);
```

Saving



```
Parallel.ForEach(ImagesList,
    new ParallelOptions
        { CancellationToken = cancellationToken, MaxDegreeOfParallelism = Environment.ProcessorCount },
    body: image =>
    {
        cancellationToken.ThrowIfCancellationRequested();

        // Load full Image
        // Reapply Filters
        // Save file to Path

        Interlocked.Increment(ref saved);
        progress?.Report(saved / total);
    });
```

Reload Image because only a resized image was used until now