Table of Contents

Upgrade Guide

1.0 to 1.1	Error: Argument #1' cannot convert System.Collections.Generic.List<Xefier.Threading.Tasks.Task>' expression to type `System.Collections.Generic.IEnumerable<Xefier.Threading.Tasks.ITask>' Fix by replacing: ~~IEnumerable<Task>~~ IEnumerable<ITask>

Links

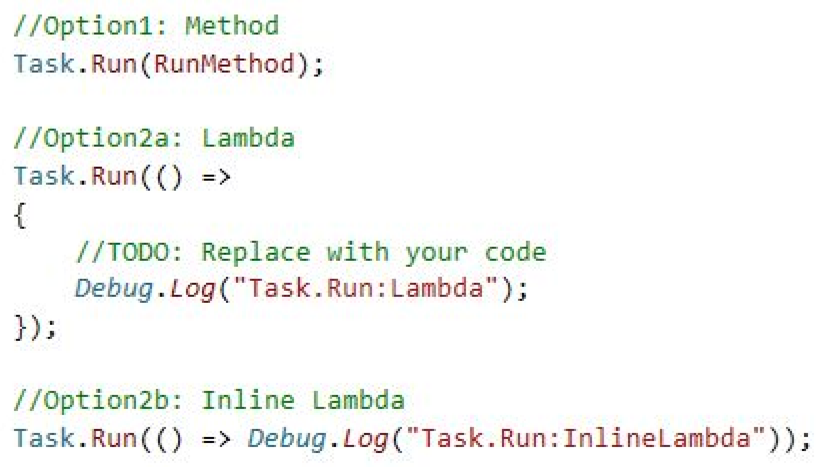
[Xefier.Threading.Tasks (Full Documentation)]()
[Forum]()

Examples

Assets\Xefier\Threading\Tasks\Examples\

Task.Run

Runs work on separate thread (pool)



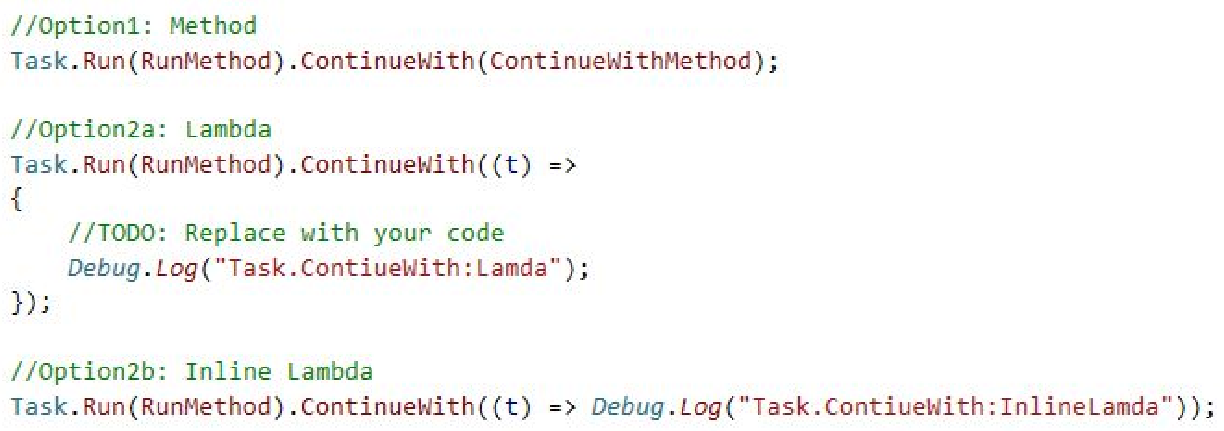
```
//Option1: Method
Task.Run(RunMethod);

//Option2a: Lambda
Task.Run(() =>
{
    //TODO: Replace with your code
    Debug.Log("Task.Run:Lambda");
});

//Option2b: Inline Lambda
Task.Run(() => Debug.Log("Task.Run:InlineLambda"));
```

Task.ContinueWith

Runs work asynchronously when Task completes



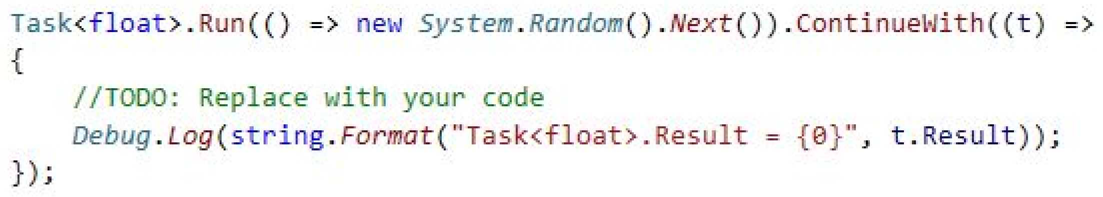
```
//Option1: Method
Task.Run(RunMethod).ContinueWith(ContinueWithMethod);

//Option2a: Lambda
Task.Run(RunMethod).ContinueWith((t) =>
{
    //TODO: Replace with your code
    Debug.Log("Task.ContiueWith:Lamda");
});

//Option2b: Inline Lambda
Task.Run(RunMethod).ContinueWith((t) => Debug.Log("Task.ContiueWith:InlineLamda"));
```

Task.Result

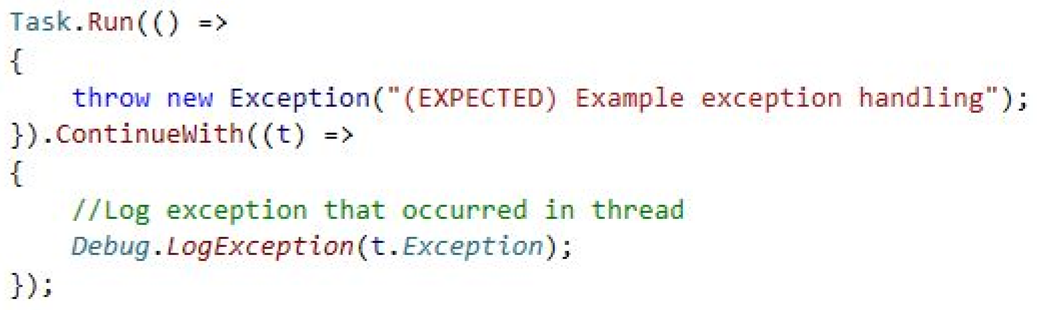
Task<T> performs same operations as Task except it has a Result where T = any type (int, float, etc)



```
Task<float>.Run(() => new System.Random().Next()).ContinueWith((t) =>
{
    //TODO: Replace with your code
    Debug.Log(string.Format("Task<float>.Result = {0}", t.Result));
});
```

Task.Exception

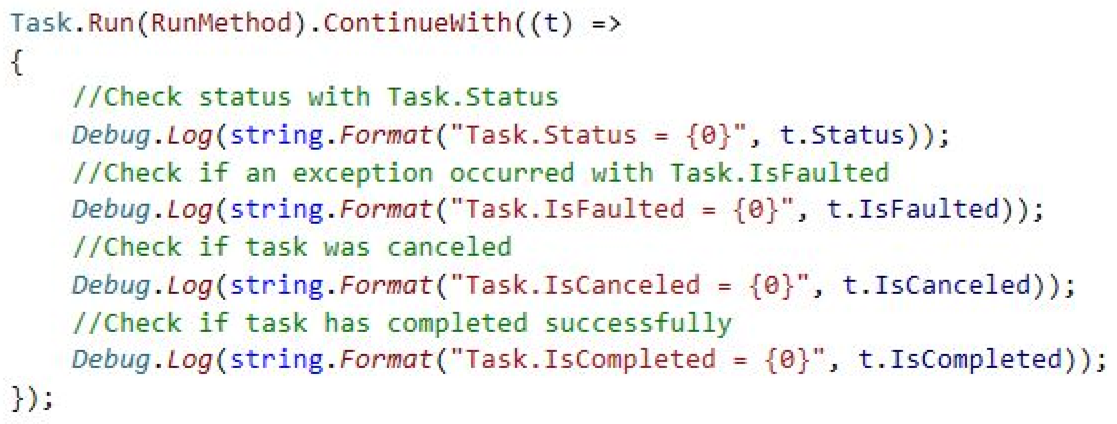
Gets exception from thread. Exceptions are usually tricky to handle in separate threads but Task.Exception provides an easy way to handle them!



```
Task.Run(() =>
{
    throw new Exception("(EXPECTED) Example exception handling");
}).ContinueWith((t) =>
{
    //Log exception that occurred in thread
    Debug.LogException(t.Exception);
});
```

Task.Status

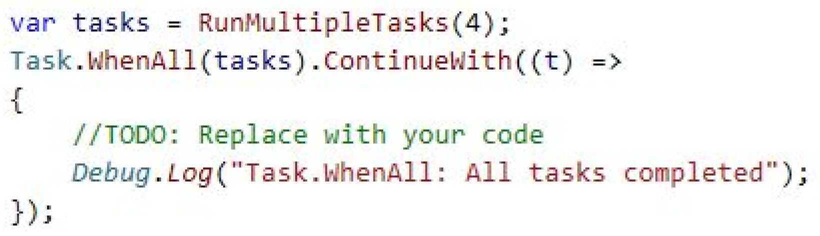
Check task's status



```
Task.Run(RunMethod).ContinueWith((t) =>
{
    //Check status with Task.Status
    Debug.Log(string.Format("Task.Status = {0}", t.Status));
    //Check if an exception occurred with Task.IsFaulted
    Debug.Log(string.Format("Task.IsFaulted = {0}", t.IsFaulted));
    //Check if task was canceled
    Debug.Log(string.Format("Task.IsCanceled = {0}", t.IsCanceled));
    //Check if task has completed successfully
    Debug.Log(string.Format("Task.IsCompleted = {0}", t.IsCompleted));
});
```

Task.WhenAll

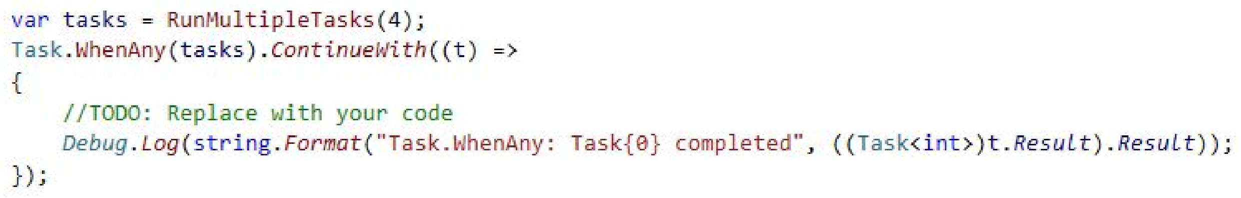
Creates a task that completes when all specified tasks are complete



```
var tasks = RunMultipleTasks(4);
Task.WhenAll(tasks).ContinueWith((t) =>
{
    //TODO: Replace with your code
    Debug.Log("Task.WhenAll: All tasks completed");
});
```

Task.WhenAny

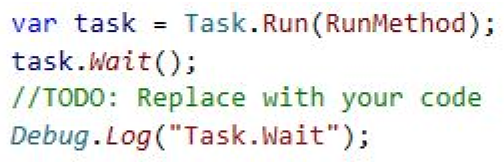
Creates a task that completes when any specified tasks are complete



```
var tasks = RunMultipleTasks(4);
Task.WhenAny(tasks).ContinueWith((t) =>
{
    //TODO: Replace with your code
    Debug.Log(string.Format("Task.WhenAny: Task{0} completed", ((Task<int>)t.Result).Result));
});
```

Task.Wait

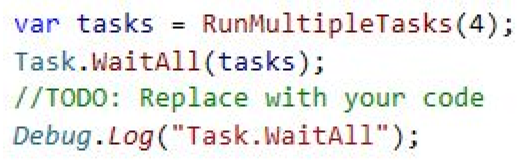
Waits for Task to complete



```
var task = Task.Run(RunMethod);
task.Wait();
//TODO: Replace with your code
Debug.Log("Task.Wait");
```

Task.WaitAll

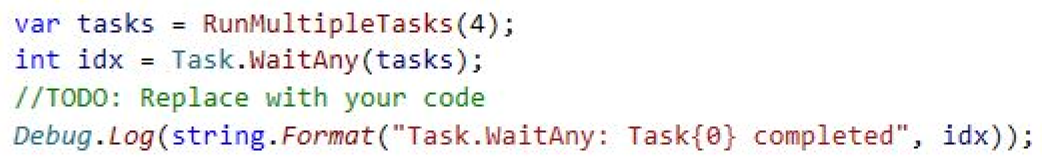
Waits for all specified tasks to complete



```
var tasks = RunMultipleTasks(4);
Task.WaitAll(tasks);
//TODO: Replace with your code
Debug.Log("Task.WaitAll");
```

Task.WaitAny

Waits for any specified tasks to complete



```
var tasks = RunMultipleTasks(4);
int idx = Task.WaitAny(tasks);
//TODO: Replace with your code
Debug.Log(string.Format("Task.WaitAny: Task{0} completed", idx));
```

Want to spawn objects on a thread?

Async Objects: https://www.assetstore.unity3d.com/#!/content/81192

Want to run for loops in Parallel?

Parallel for Unity: https://www.assetstore.unity3d.com/en/#!/content/81738