Microsoft

Correcting Common Async/Await Mistakes in .NET

Brandon Minnick
Developer Advocate

@TheCodeTraveler

https://www.codetraveler.io/NDCSydney-2018-AsyncAwait

Multi Threading

ReadDataFromUrl

```
async Task ReadDataFromUrl(string url)
{
    WebClient wc = new WebClient();
    byte[] result = await wc.DownloadDataTaskAsync(url);
    string data = Encoding.ASCII.GetString(result);
    LoadData(data);
}
```

ReadDataFromUrl

```
async Task ReadDataFromUrl(string url)
{
    WebClient wc = new WebClient();
    byte[] result = await wc.DownloadDataTaskAsync(url);
    string data = Encoding.ASCII.GetString(result);
    LoadData(data);
}
```

Thread 1

ReadDataFromUrl

```
async Task ReadDataFromUrl(string url)
{
    WebClient wc = new WebClient();
    byte[] result = await wc.DownloadDataTaskAsync(url);
    string data = Encoding.ASCII.GetString(result);
    LoadData(data);
}
```

Thread 2

ReadDataFromUrl

```
async Task ReadDataFromUrl(string url)
{
    WebClient wc = new WebClient();
    byte[] result = await wc.DownloadDataTaskAsync(url);
    string data = Encoding.ASCII.GetString(result);
    LoadData(data);
}
```

Thread 1

Intermediate Language

ReadDataFromUr

```
async Task ReadDataF
{
    WebClient wc = ne
    byte[] result = a
    string data = Enc
    LoadData(data);
}
```

```
[CompilerGenerated]
private sealed class <ReadDataFromUrl>d__1 : IAsyncStateMachine
{
    // Fields
    public int <>1__state;
    private byte[] <>s__4;
    public AsyncVoidMethodBuilder <>t__builder;
    private TaskAwaiter<byte[]> <>u__1;
    private string <data>5__3;
    private byte[] <result>5__2;
    private WebClient <wc>5__1;
    public string url;

    // Methods
    public <ReadDataFromUrl>d__1();
    private void MoveNext();
    [DebuggerHidden]
    private void SetStateMachine(IAsyncStateMachine stateMachine);
}
```

ReadDataFromUrl

```
private sealed class <ReadDataFromUrl>d_1 : IAsyncStateMachine
```

```
async Task ReadDataF
{
    WebClient wc = ne
    byte[] result = a
    string data = Enc
    LoadData(data);
}
```

```
{
    [CompilerGenerated]
    // Fields
    public int <>1__state;
    private byte[] <>s__4;
    public AsyncVoidMethodBuilder <>t__builder;
    private TaskAwaiter<byte[]> <>u__1;
    private string <data>5__3;
    private byte[] <result>5__2;
    private WebClient <wc>5__1;
    public string url;

    // Methods
    public <ReadDataFromUrl>d__1();
    private void MoveNext();
    [DebuggerHidden]
    private void SetStateMachine(IAsyncStateMachine stateMachine);
}
```

ReadDataFromUr

```
async Task ReadDataF
{
    WebClient wc = ne
    byte[] result = a
    string data = Enc
    LoadData(data);
}
```

```
[CompilerGenerated]
private sealed class <ReadDataFromUrl>d__1 : IAsyncStateMachine
{
    // Fields
    public int <>1__state;
    private byte[] <>s__4;
    public AsyncVoidMethodBuilder <>t__builder;
    privat
    privat
    privat
    privat
    publi

    private string <data>5_3;
    private byte[] <result>5_2;
    private WebClient <wc>5_1;
    public string url;

    // Methods
    public <ReadDataFromUrl>d__1();
    private void MoveNext();
    [DebuggerHidden]
    private void SetStateMachine(IAsyncStateMachine stateMachine);
}
```

ReadDataFromUr

```
async Task ReadDataF
{
    WebClient wc = ne
    byte[] result = a
    string data = Enc
    LoadData(data);
}
```

```
[CompilerGenerated]
private sealed class <ReadDataFromUrl>d__1 : IAsyncStateMachine
{
    // Fields
    public int <>1__state;
    private byte[] <>s__4;
    public AsyncVoidMethodBuilder <>t__builder;
    private TaskAwaiter<byte[]> <>u__1;
    private string <data>5__3;
    private byte[] <result>5__2;
    private WebClient <wc>5__1;
    public string url;

    // Methods
    public
    private void MoveNext();
    [DebuggerHidden]
    private void SetStateMachine(IAsyncStateMachine stateMachine);
}
```

```
private void MoveNext();
```

ReadDataFromUrl_MoveNext

```
public void MoveNext()
{
   uint num = (uint)this.$PC;
   this.$PC = -1;
   try {
      switch (num) {
         case 0:
            this.<wc>__0 = new WebClient();
            this.$awaiter0 = this.<wc>__0.DownloadDataTaskAsync(this.url).GetAwaiter();
            this.$PC = 1;
            ...
            return;
            break;
         case 1:
            this.<result>__1 = this.$awaiter0.GetResult();
            this.<data>__2 = Encoding.ASCII.GetString(this.<result>__1);
            this.$this.LoadData(this.<data>__2);
            break;
         default:
            return;
      }
   }
   catch (Exception exception) { ... }
   this.$PC = -1;
   this.$builder.SetResult();
}
```

ReadDataFromUrl_MoveNext

```
public void MoveNext()
{
   uint num = (uint)this.$PC;
   this.$PC = -1;
   try {
      switch (num) {
         case 0:
            this.<wc>__0 = new WebClient();
            this.$awaiter0 = this.<wc>__0.DownloadDataTaskAsync(this.url).GetAwaiter();
            this.$PC = 1;
            ...
            return;
         default:
            return;
      }
   }
   catch (Exception exception) { ... }
   this.$PC = -1;
   this.$builder.SetResult();
}
```

ReadDataFromUrl_MoveNext

```
public void MoveNext()
{
   uint num = (uint)this.$PC;
   this.$PC = -1;
   try {
      switch (num) {
         case 0:
            this.<wc>__0 = new WebClient();
            this.$awaiter0 = this.<wc>__0.DownloadDataTaskAsync(this.url).GetAwaiter();
            if (!this.$awaiter0.IsCompleted) {
               this.$PC = 0;
               this.$builder.AwaitUnsafeOnCompleted(ref this.$awaiter0, ref this);
               return;
            }
            goto case 1;
         case 1:
            this.<result>__1 = this.$awaiter0.GetResult();
            this.<data>__2 = Encoding.ASCII.GetString(this.<result>__1);
            this.$this.LoadData(this.<data>__2);
            break;
         default:
            return;
      }
   }
   catch (Exception exception) { ... }
   this.$PC = -1;
   this.$builder.SetResult();
}
```


ReadDataFromUrl_MoveNext

```
public void MoveNext()
{
   uint num = (uint)this.$PC;
   this.$PC = -1;
   try {
      switch (num) {
         case 0:
            this.<wc>__0 = new WebClient();
            this.$awaiter0 = this.<wc>__0.DownloadDataTaskAsync(this.url).GetAwaiter();
            this.$PC = 1;
            ...
            return;
            break;
         case 1:
            this.<result>__1 = this.$awaiter0.GetResult();
            this.<data>__2 = Encoding.ASCII.GetString(this.<result>__1);
            this.$this.LoadData(this.<data>__2);
            break;
         default:
            return;
      }
   }
   catch (Exception exception) { ... }
   this.$PC = -1;
   this.$builder.SetResult();
}
```

try { . . . }

catch (Exception exception) { . . . }

Quick Review

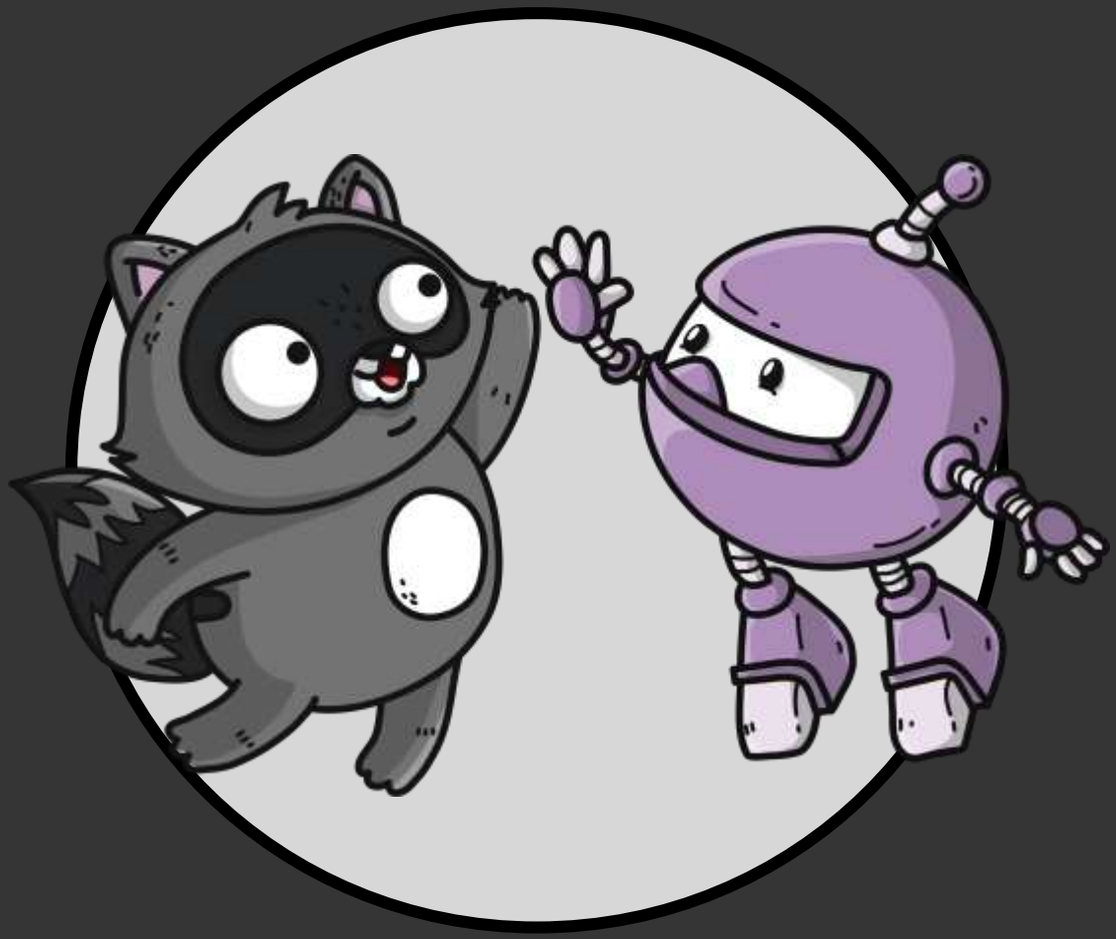
Async Keyword Adds 100 Bytes

Every Async Method Becomes a Class

Await Every Async Method

Non-awaited Async Methods Hide Exceptions

Let's Fix Some Code



Async/Await Best Practices

Async/Await Best Practices

Never Use `.Wait()` or `.Result`

- Always use `await`
- If synchronous, use `.GetAwaiter().GetResult()`

Async/Await Best Practices

Fire and Forget Tasks

- Use `ICommand`
- Use `async void`
  - Only from the Main Thread

Async/Await Best Practices

Avoid `return await`

- Remove `**async**` keyword
  - Except: In `**try/catch**` blocks
  - Except: In `**using( … )**` blocks

Async/Await Best Practices

Utilize `.ConfigureAwait(false)`

- Except: When needing to return to calling thread

Xamarin University

CSC350: Using Async Await

@TheCodeTraveler

https://www.codetraveler.io/NDCSydney-2018-AsyncAwait

Resources

https://www.codetraveler.io/NDCSydney-2018-AsyncAwait

Thank You

https://www.codetraveler.io/NDCSydney-2018-AsyncAwait