Just the Best Practices



Elton StonemanConsultant & Trainer

@EltonStoneman blog.sixeyed.com

Must Haves



Dispose of IDisposable objects as soon as you can



Disposing Disposable Objects

Typical.cs

```
using (var obj = new Custom())
{
    // work with obj
}
// obj.Dispose() is called here
```

Alternative.cs

```
var obj = new Custom();
try
    // work with obj
finally
    obj.Dispose();
```

Enable static analysis with rule CA2000



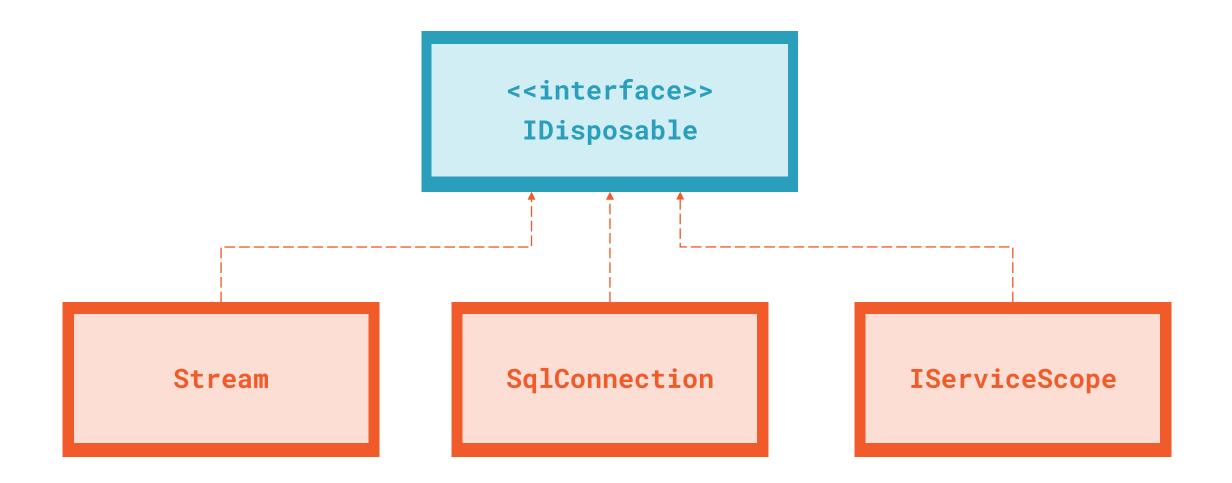
CA2000

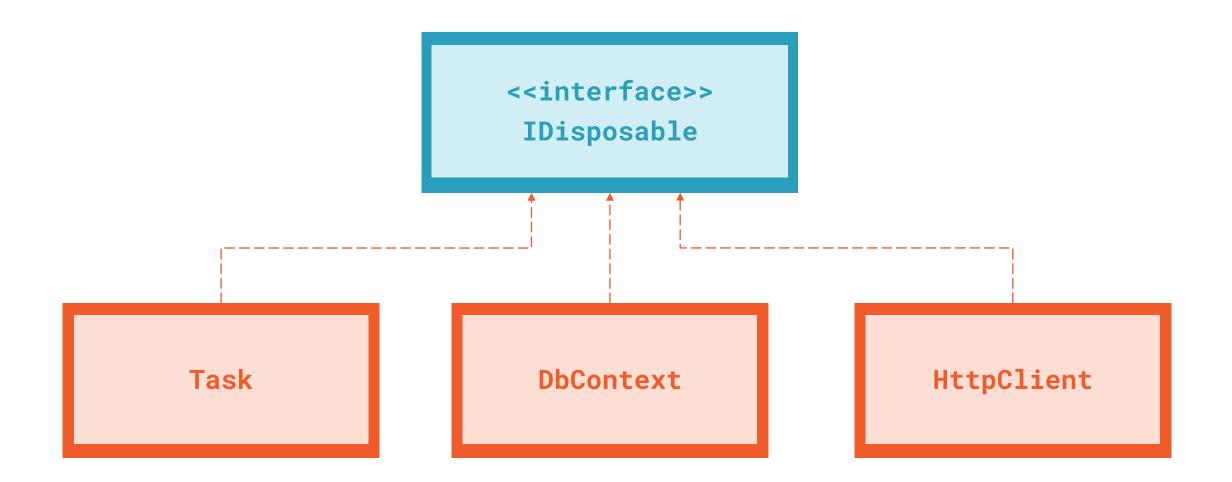
Dispose objects before losing scope



Know your domain:)







Nice to Haves



If you use IDisposable objects as instance fields, implement IDisposable



Implementing IDisposable

DatabaseState.cs

```
public class DatabaseState : IDisposable
{
   public void Dispose()
   {
      Dispose(true);
      GC.SuppressFinalize(this);
   }
```

Program.cs

```
using (var s = new DatabaseState())
{
   Console.WriteLine(s.GetDate());
}
```

Allow Dispose() to be called multiple times and don't throw exceptions



Dispose Safely

Example.cs

```
protected SqlConnection _connection;
protected void Dispose(bool disposing)
   if (_disposed)
        return;
   if (disposing)
           (_connection != null)
            _connection.Dispose();
            _connection = null;
        _disposed = true;
```

Implement IDisposable to support disposing resources in a class hierarchy



Dispose Pattern

BaseClass.cs

```
public void Dispose()
{
    Dispose(true);
    GC.SuppressFinalize(this);
}
protected virtual void Dispose(bool disposing)
{
    // dispose only *this* class's resources
}
```

DerivedClass.cs

```
protected override void Dispose(bool disposing)
{
   // dispose only *this* class's resources
}
```

Edge Cases



If you use unmanaged resources, declare a finalizer which cleans them up



Finalizers

ClassWithFinalizer.cs

```
~UnmanagedDatabaseState()
    Dispose(false);
protected override void Dispose(bool disposing)
    if (disposing)
       // clean up managed resources
    // clean up unmanaged resources
    base.Dispose(disposing);
```

Implement IAsyncDisposable if your class uses an async disposable field



IAsyncDisposable

WithAsyncCleanup.cs

```
public class WithAsyncCleanup : IAsyncDisposable
   public async ValueTask DisposeAsync()
        await DisposeAsyncCore();
       GC.SuppressFinalize(this);
   protected virtual async ValueTask DisposeAsyncCore()
       // clean up managed resources
```

We're Done!



So...

- Please leave a rating
- Follow @EltonStoneman on Twitter
- And watch my other courses ©

