Formación de Testing en .NET

Nafarroako Gobernua | Gobierno de Navarra

hiberus©

La compañía hiperespecializada en las TIC

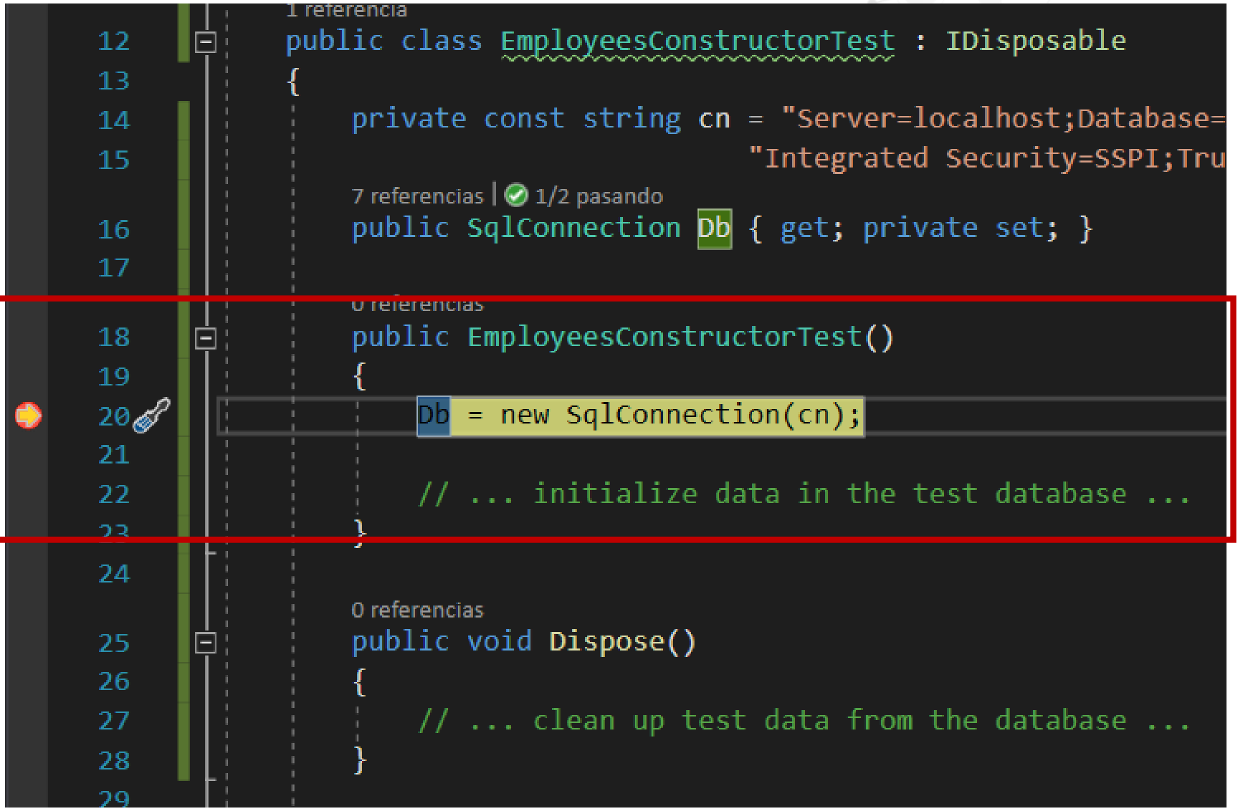
Tema 3: Moq

Nafarroako Gobernua Gobierno de Navarra

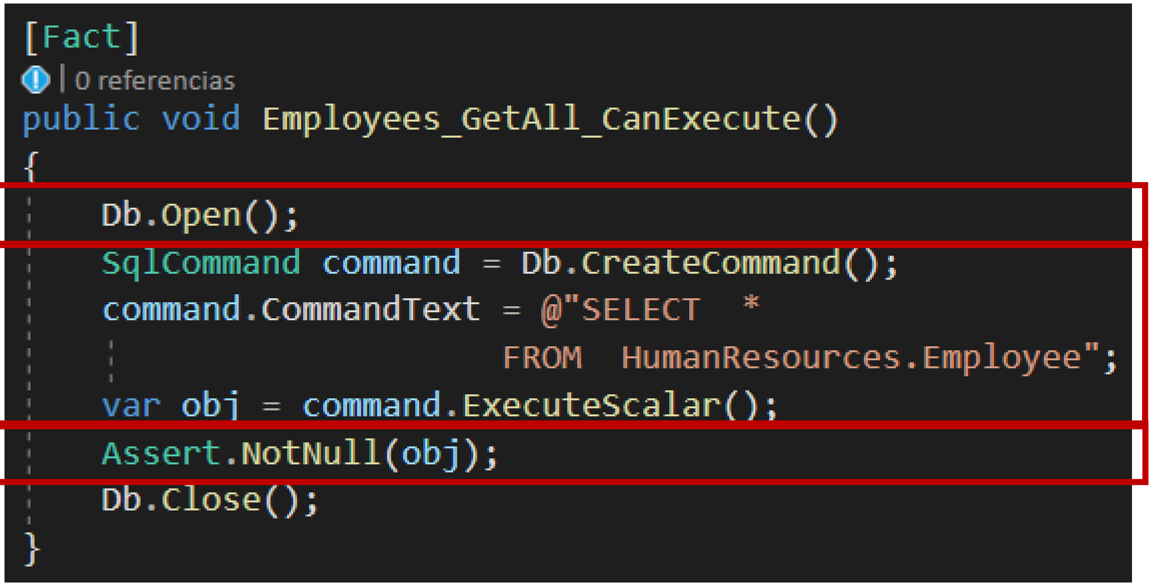
Why Moq?

Separating Responsabilities

#somoshiberus



```
1 referencia
public class EmployeesConstructorTest : IDisposable
{
    private const string cn = "Server=localhost;Database=
                             "Integrated Security=SSPI;Tru
    7 referencias | 1/2 pasando
    public SqlConnection Db { get; private set; }

    0 referencias
    public EmployeesConstructorTest()
    {
        Db = new SqlConnection(cn);

        // ... initialize data in the test database ...
    }

    0 referencias
    public void Dispose()
    {
        // ... clean up test data from the database ...
    }
```



```
[Fact]
| 0 referencias
public void Employees_GetAll_CanExecute()
{
    Db.Open();
    SqlCommand command = Db.CreateCommand();
    command.CommandText = @"SELECT  *
                          FROM  HumanResources.Employee";
    var obj = command.ExecuteScalar();
    Assert.NotNull(obj);
    Db.Close();
}
```

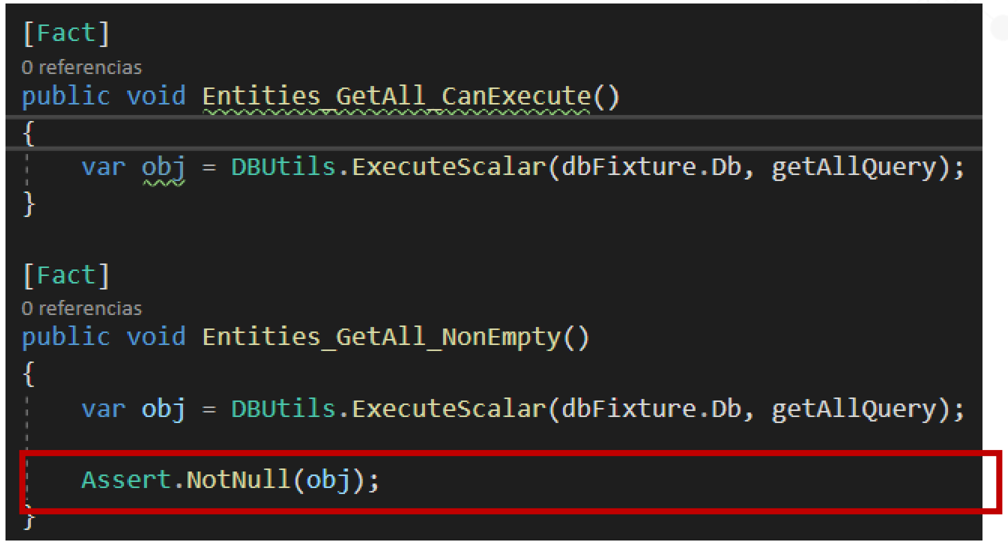
Nafarroako Gobernua | Gobierno de Navarra

www.hiberus.com

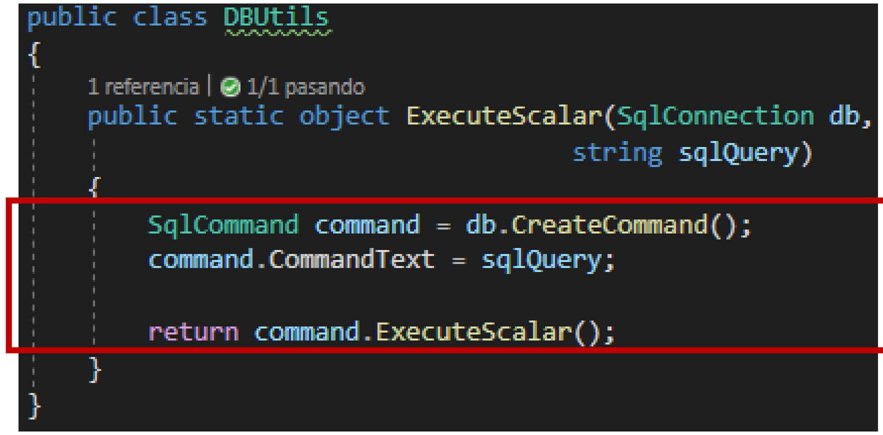
Why Moq?

Separating Responsabilities

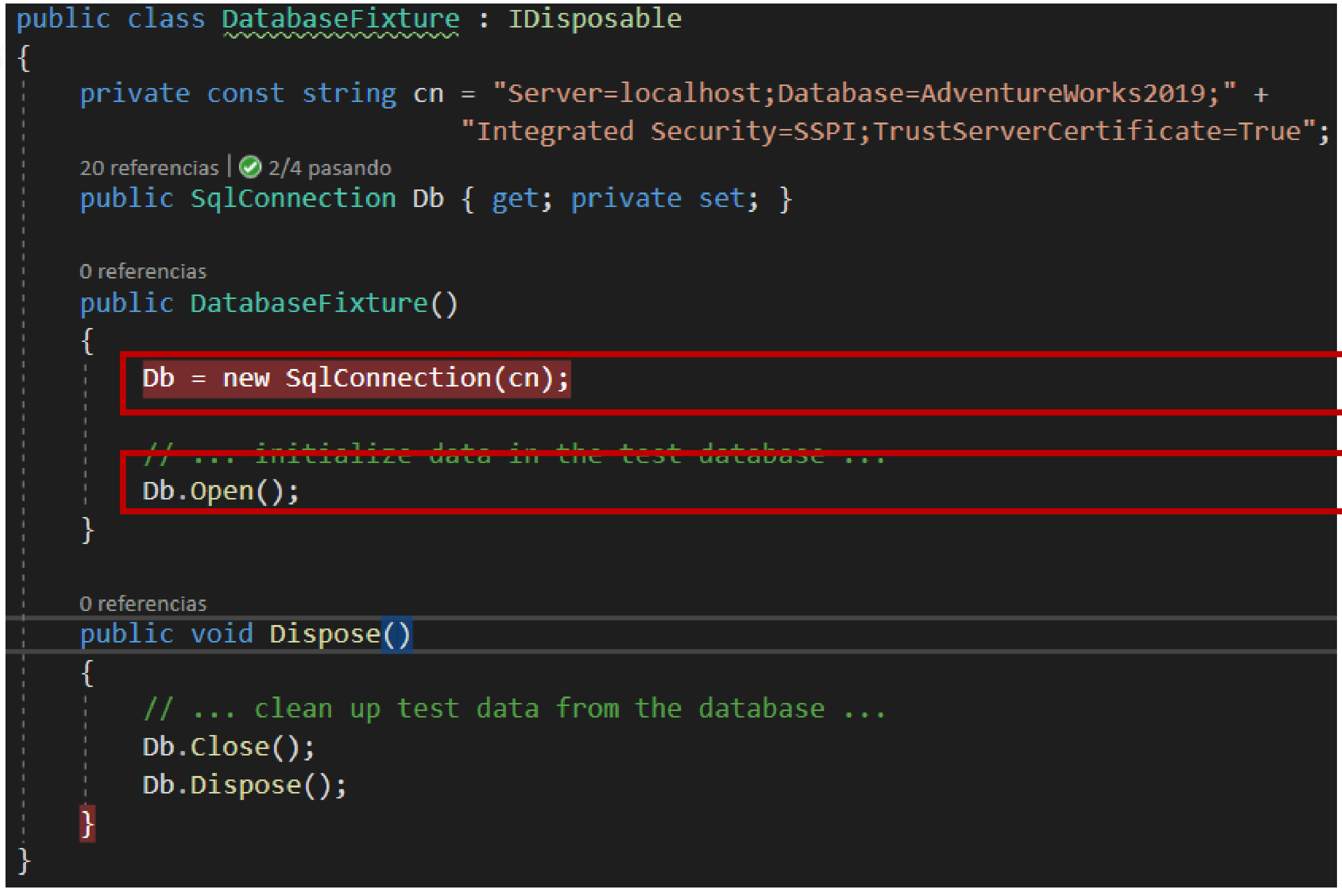
#somoshiberus



```
[Fact]
0 referencias
public void Entities_GetAll_CanExecute()
{
    var obj = DBUtils.ExecuteScalar(dbFixture.Db, getAllQuery);
}

[Fact]
0 referencias
public void Entities_GetAll_NonEmpty()
{
    var obj = DBUtils.ExecuteScalar(dbFixture.Db, getAllQuery);

    Assert.NotNull(obj);
}
```



```
public class DBUtils
{
    1 referencia | 1/1 pasando
    public static object ExecuteScalar(SqlConnection db,
                                       string sqlQuery)
    {
        SqlCommand command = db.CreateCommand();
        command.CommandText = sqlQuery;

        return command.ExecuteScalar();
    }
}
```

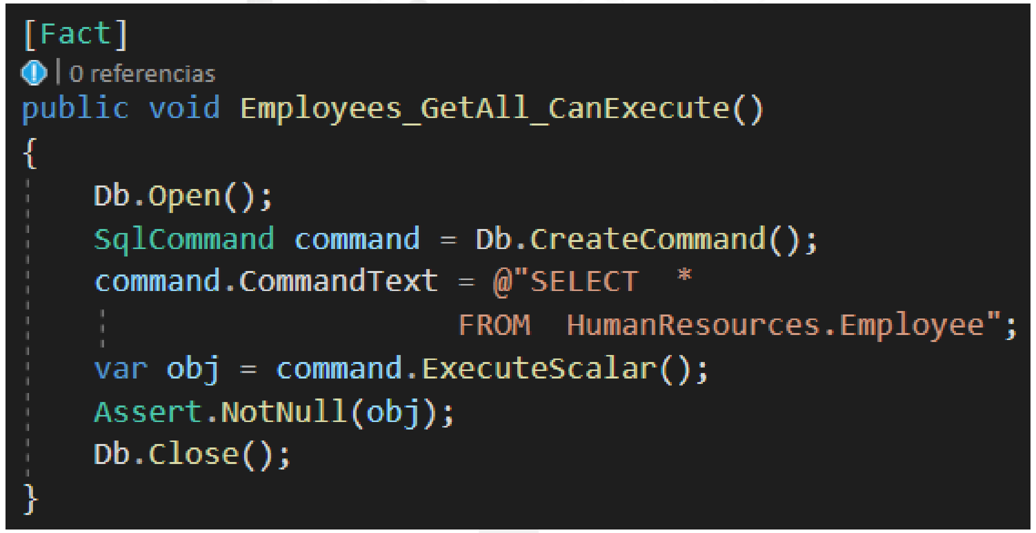


```
public class DatabaseFixture : IDisposable
{
    private const string cn = "Server=localhost;Database=AdventureWorks2019;" +
                              "Integrated Security=SSPI;TrustServerCertificate=True";
    20 referencias | 2/4 pasando
    public SqlConnection Db { get; private set; }

    0 referencias
    public DatabaseFixture()
    {
        Db = new SqlConnection(cn);

        // ... initialize data in the test database ...
        Db.Open();
    }

    0 referencias
    public void Dispose()
    {
        // ... clean up test data from the database ...
        Db.Close();
        Db.Dispose();
    }
}
```

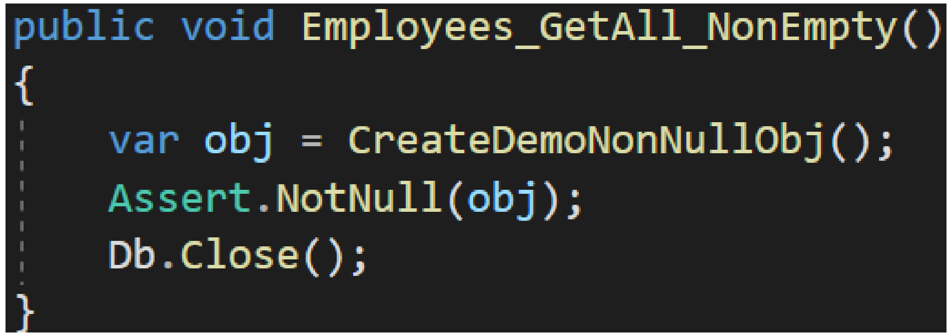
Why Moq?

Separating Responsabilities

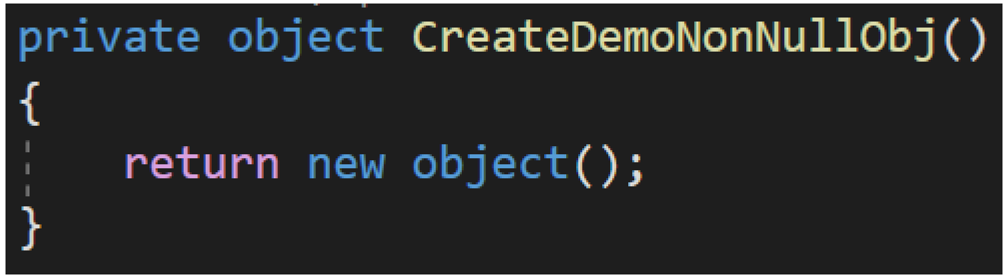
#somoshiberus



```
[Fact]
0 referencias
public void Employees_GetAll_CanExecute()
{
    Db.Open();
    SqlCommand command = Db.CreateCommand();
    command.CommandText = @"SELECT  *
                            FROM  HumanResources.Employee";
    var obj = command.ExecuteScalar();
    Assert.NotNull(obj);
    Db.Close();
}
```



```
public void Employees_GetAll_NonEmpty()
{
    var obj = CreateDemoNonNullObj();
    Assert.NotNull(obj);
    Db.Close();
}
```

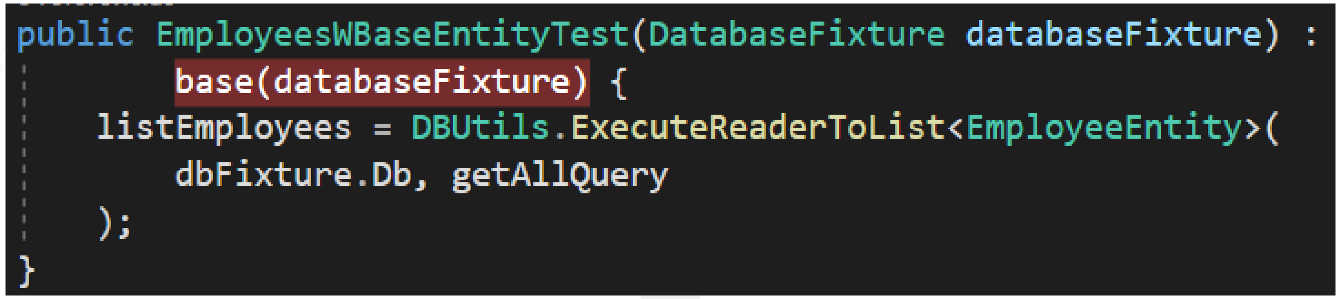


```
private object CreateDemoNonNullObj()
{
    return new object();
}
```

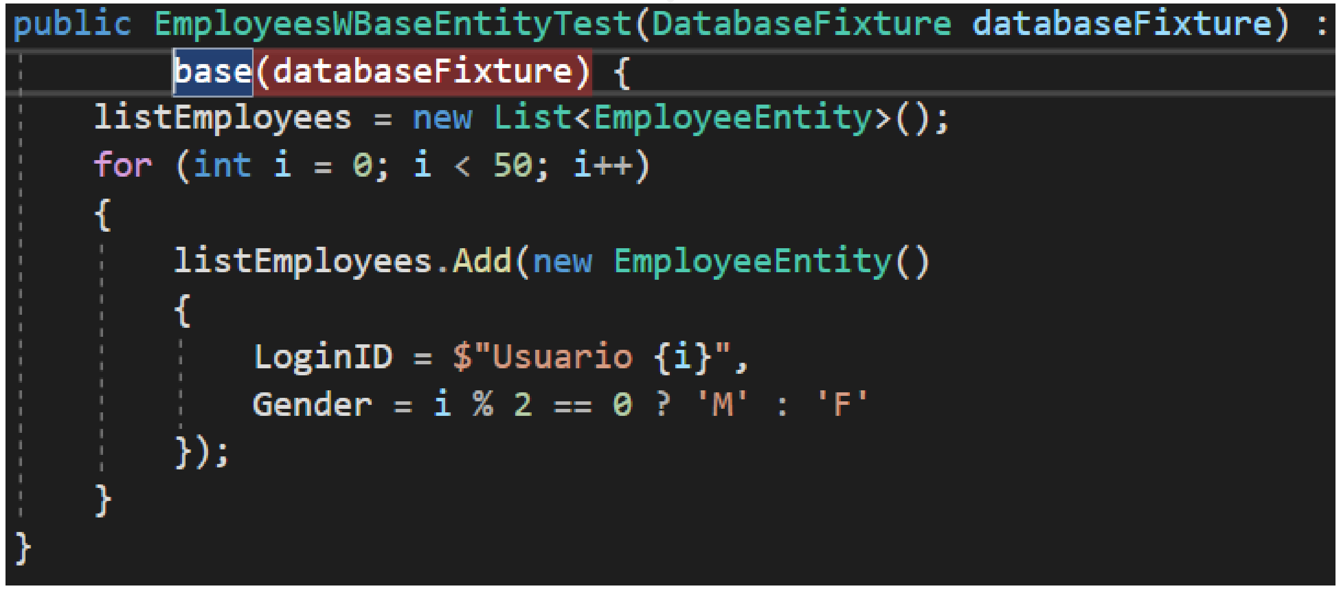
How Moq?

Separating Responsabilities

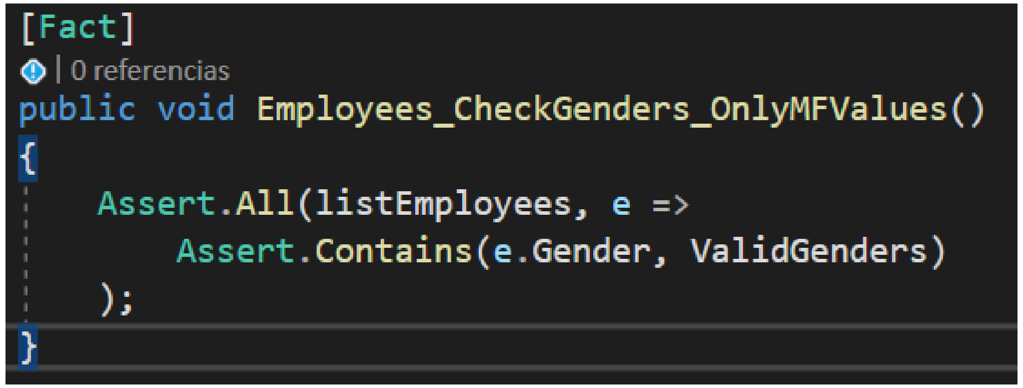
#somoshiberus



```
public EmployeesWBaseEntityTest(DatabaseFixture databaseFixture) :
        base(databaseFixture) {
    listEmployees = DBUtils.ExecuteReaderToList<EmployeeEntity>(
        dbFixture.Db, getAllQuery
    );
}
```



```
public EmployeesWBaseEntityTest(DatabaseFixture databaseFixture) :
        base(databaseFixture) {
    listEmployees = new List<EmployeeEntity>();
    for (int i = 0; i < 50; i++)
    {
        listEmployees.Add(new EmployeeEntity()
        {
            LoginID = $"Usuario {i}",
            Gender = i % 2 == 0 ? 'M' : 'F'
        });
    }
}
```



```
[Fact]
| 0 referencias
public void Employees_CheckGenders_OnlyMFValues()
{
    Assert.All(listEmployees, e =>
        Assert.Contains(e.Gender, ValidGenders)
    );
}
```

Moq Types

Test Doubles

#somoshiberus

- **Dummy**
  Passed around, but not actually used (e.g. need to fill parameter list)
- **Fake**
  Have working implementation, but have shortcuts (not to use in prod.)
- **Stubs**
  Provide canned answers to calls, not responding if other inputs
- **Spies**
  Stubs with memory; e.g. mail service with count of sent messages
- **Mocks**
  Pre-programmed objects with expected behaviour

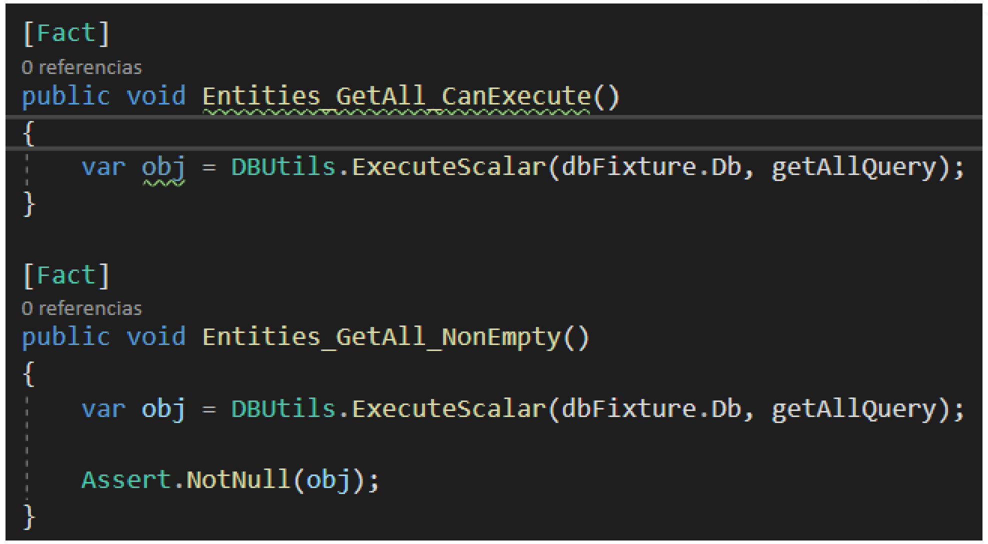
https://blog.cleancoder.com/uncle-bob/2014/05/14/TheLittleMocker.html

https://martinfowler.com/articles/mocksArentStubs.html

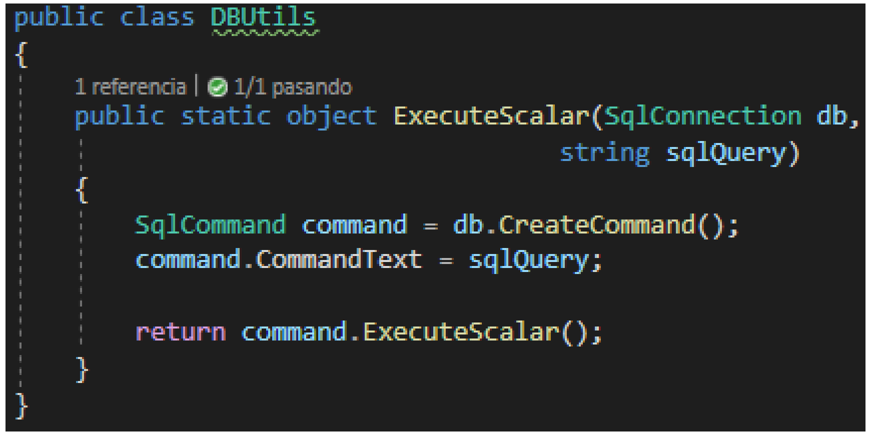
Moq Types

Test Doubles

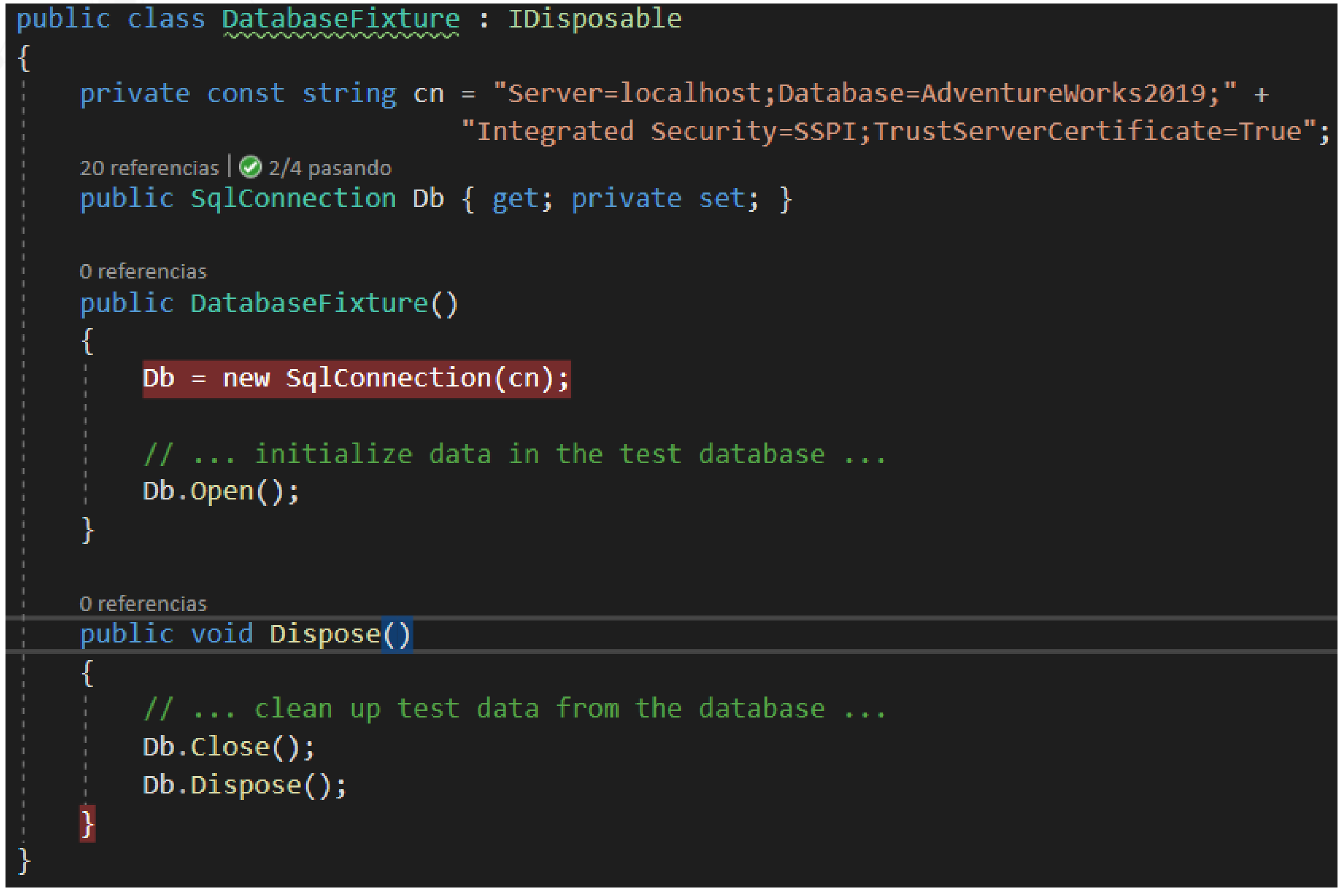
#somoshiberus



```
[Fact]
0 referencias
public void Entities_GetAll_CanExecute()
{
    var obj = DBUtils.ExecuteScalar(dbFixture.Db, getAllQuery);
}

[Fact]
0 referencias
public void Entities_GetAll_NonEmpty()
{
    var obj = DBUtils.ExecuteScalar(dbFixture.Db, getAllQuery);

    Assert.NotNull(obj);
}
```



```
public class DBUtils
{
    1 referencia | 1/1 pasando
    public static object ExecuteScalar(SqlConnection db,
                                       string sqlQuery)
    {
        SqlCommand command = db.CreateCommand();
        command.CommandText = sqlQuery;

        return command.ExecuteScalar();
    }
}
```



```
public class DatabaseFixture : IDisposable
{
    private const string cn = "Server=localhost;Database=AdventureWorks2019;" +
                              "Integrated Security=SSPI;TrustServerCertificate=True";
    20 referencias | 2/4 pasando
    public SqlConnection Db { get; private set; }

    0 referencias
    public DatabaseFixture()
    {
        Db = new SqlConnection(cn);

        // ... initialize data in the test database ...
        Db.Open();
    }

    0 referencias
    public void Dispose()
    {
        // ... clean up test data from the database ...
        Db.Close();
        Db.Dispose();
    }
}
```

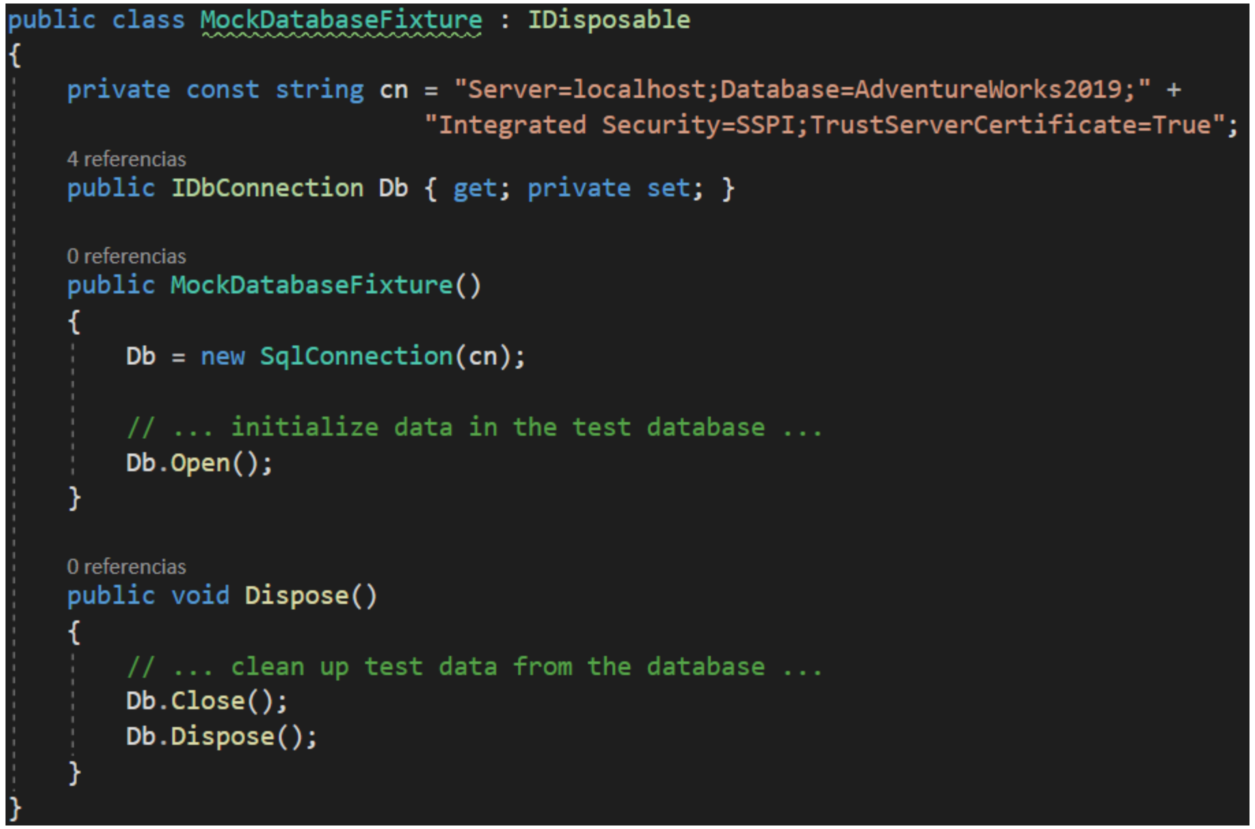
Nafarroako Gobernua Gobierno de Navarra

www.hiberus.com

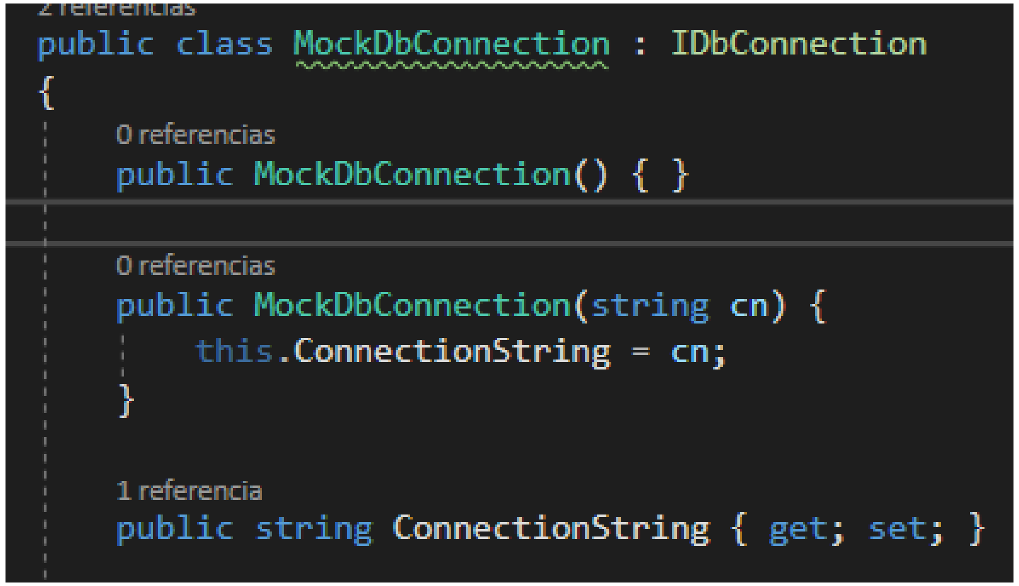
#somoshiberus

Moq Types

Dummy objects



```
public class MockDatabaseFixture : IDisposable
{
    private const string cn = "Server=localhost;Database=AdventureWorks2019;" +
                              "Integrated Security=SSPI;TrustServerCertificate=True";
    4 referencias
    public IDbConnection Db { get; private set; }

    0 referencias
    public MockDatabaseFixture()
    {
        Db = new SqlConnection(cn);

        // ... initialize data in the test database ...
        Db.Open();
    }

    0 referencias
    public void Dispose()
    {
        // ... clean up test data from the database ...
        Db.Close();
        Db.Dispose();
    }
}
```



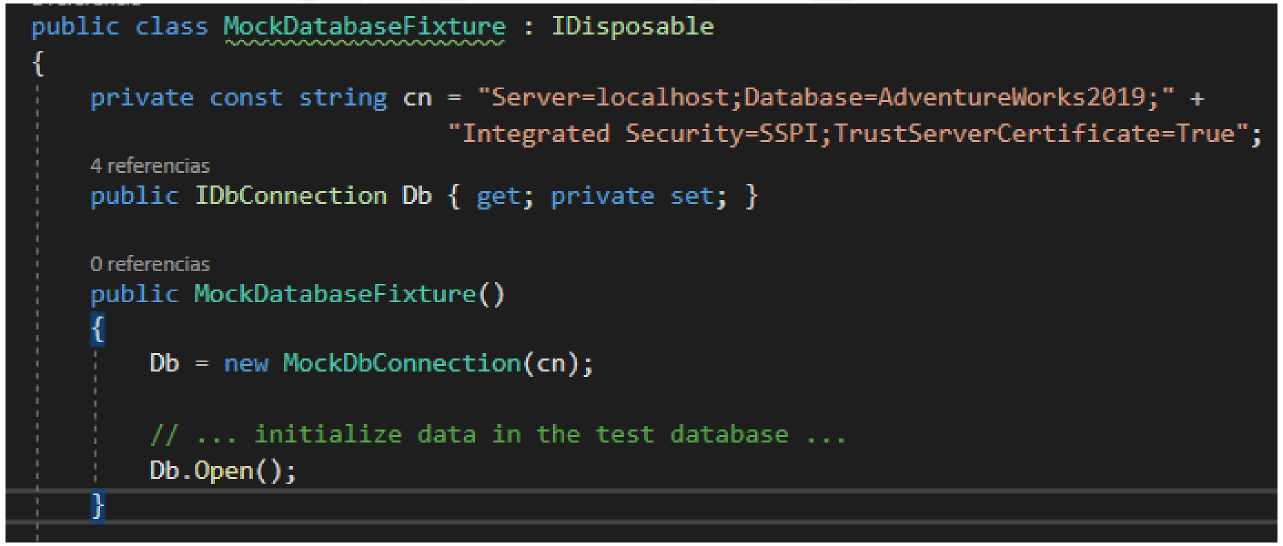
```
2 referencias
public class MockDbConnection : IDbConnection
{
    0 referencias
    public MockDbConnection() { }

    0 referencias
    public MockDbConnection(string cn) {
        this.ConnectionString = cn;
    }

    1 referencia
    public string ConnectionString { get; set; }
```

Nafarroako Gobernua Gobierno de Navarra

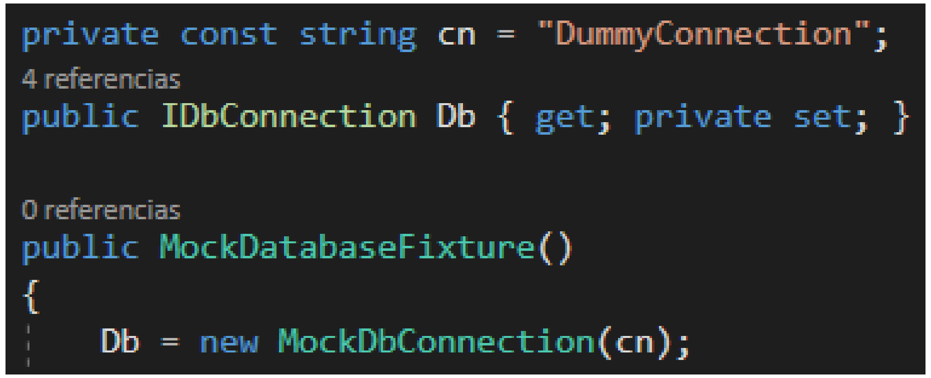
www.hiberus.com

Moq Types

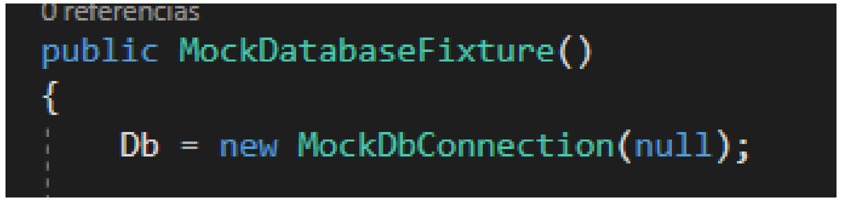
Dummy objects



```
public class MockDatabaseFixture : IDisposable
{
    private const string cn = "Server=localhost;Database=AdventureWorks2019;" +
                              "Integrated Security=SSPI;TrustServerCertificate=True";
    4 referencias
    public IDbConnection Db { get; private set; }

    0 referencias
    public MockDatabaseFixture()
    {
        Db = new MockDbConnection(cn);

        // ... initialize data in the test database ...
        Db.Open();
    }
```



```
private const string cn = "DummyConnection";
4 referencias
public IDbConnection Db { get; private set; }

0 referencias
public MockDatabaseFixture()
{
    Db = new MockDbConnection(cn);
```

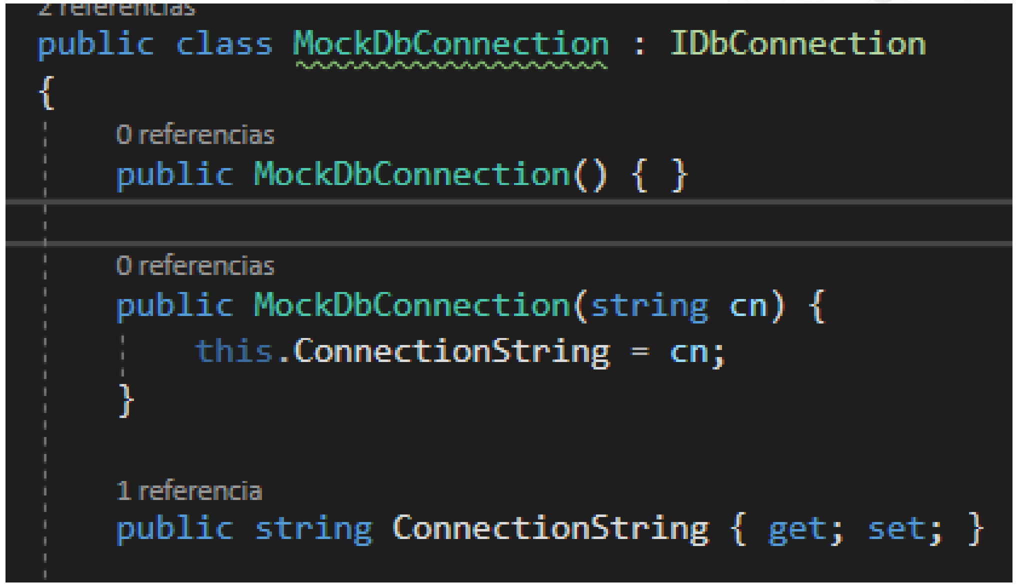


```
0 referencias
public MockDatabaseFixture()
{
    Db = new MockDbConnection(null);
```

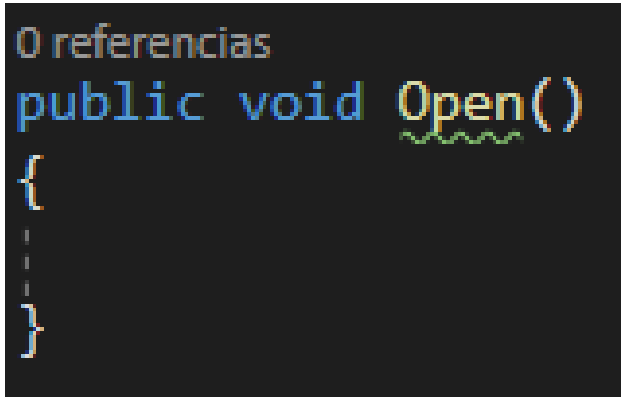
Moq Types

Fake objects

#somoshiberus



```
2 referencias
public class MockDbConnection : IDbConnection
{
    0 referencias
    public MockDbConnection() { }

    0 referencias
    public MockDbConnection(string cn) {
        this.ConnectionString = cn;
    }

    1 referencia
    public string ConnectionString { get; set; }
```



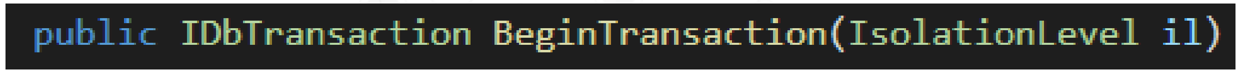
```
0 referencias
public void Open()
{

}
```

https://mikhail.io/2016/02/unit-testing-dapper-repositories/

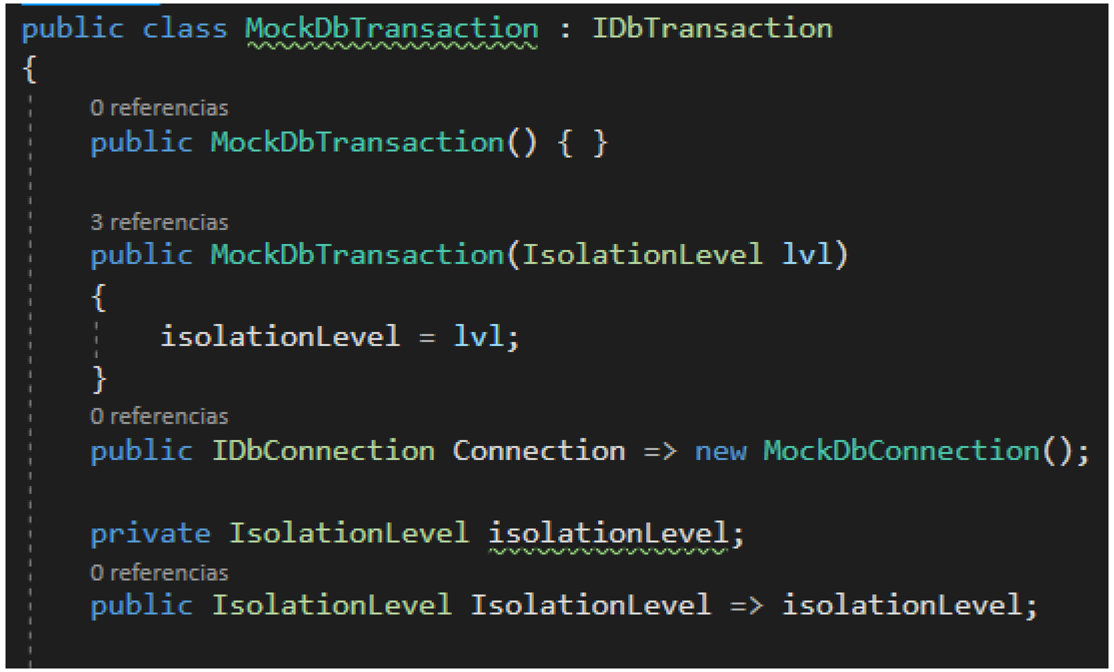
Moq Types

Stub objects

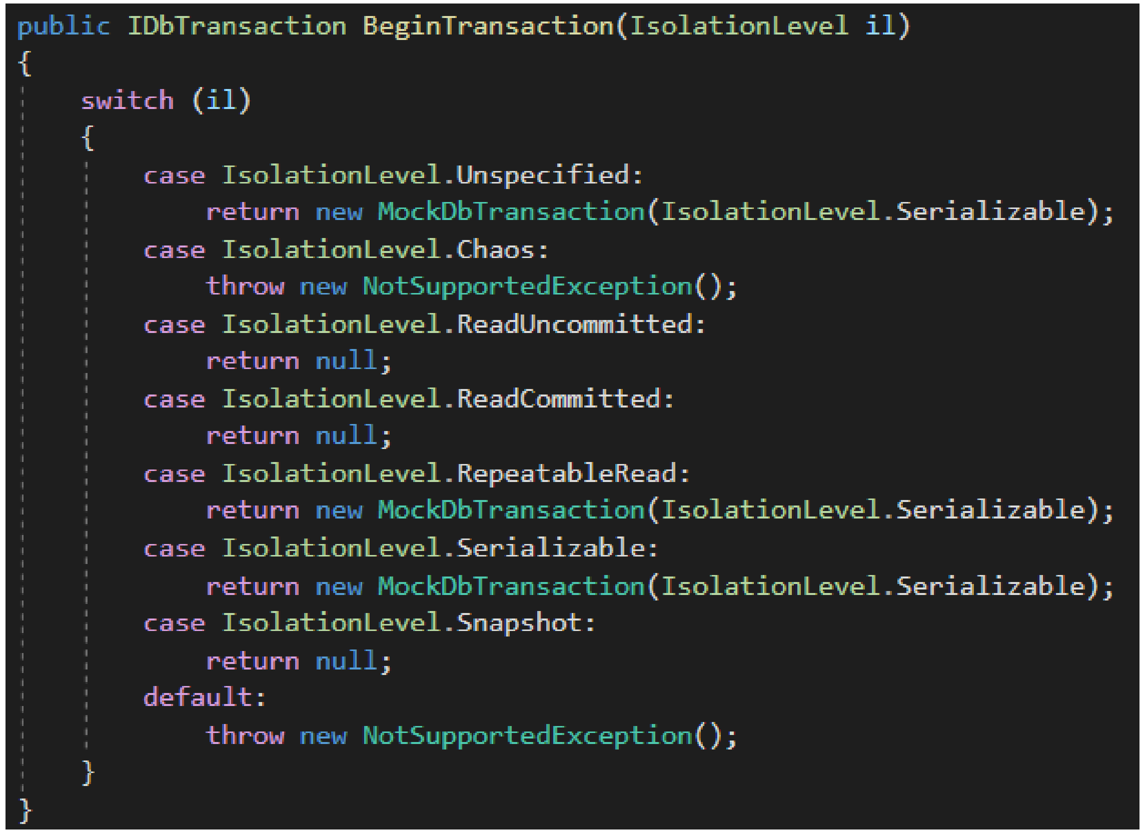
#somoshiberus



```
public IDbTransaction BeginTransaction(IsolationLevel il)
```



```
public class MockDbTransaction : IDbTransaction
{
    0 referencias
    public MockDbTransaction() { }

    3 referencias
    public MockDbTransaction(IsolationLevel lvl)
    {
        isolationLevel = lvl;
    }
    0 referencias
    public IDbConnection Connection => new MockDbConnection();

    private IsolationLevel isolationLevel;
    0 referencias
    public IsolationLevel IsolationLevel => isolationLevel;
```

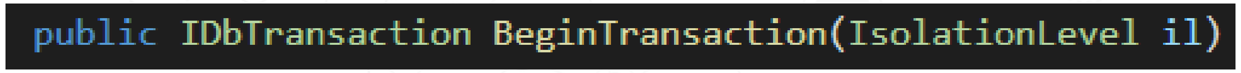


```
public IDbTransaction BeginTransaction(IsolationLevel il)
{
    switch (il)
    {
        case IsolationLevel.Unspecified:
            return new MockDbTransaction(IsolationLevel.Serializable);
        case IsolationLevel.Chaos:
            throw new NotSupportedException();
        case IsolationLevel.ReadUncommitted:
            return null;
        case IsolationLevel.ReadCommitted:
            return null;
        case IsolationLevel.RepeatableRead:
            return new MockDbTransaction(IsolationLevel.Serializable);
        case IsolationLevel.Serializable:
            return new MockDbTransaction(IsolationLevel.Serializable);
        case IsolationLevel.Snapshot:
            return null;
        default:
            throw new NotSupportedException();
    }
}
```

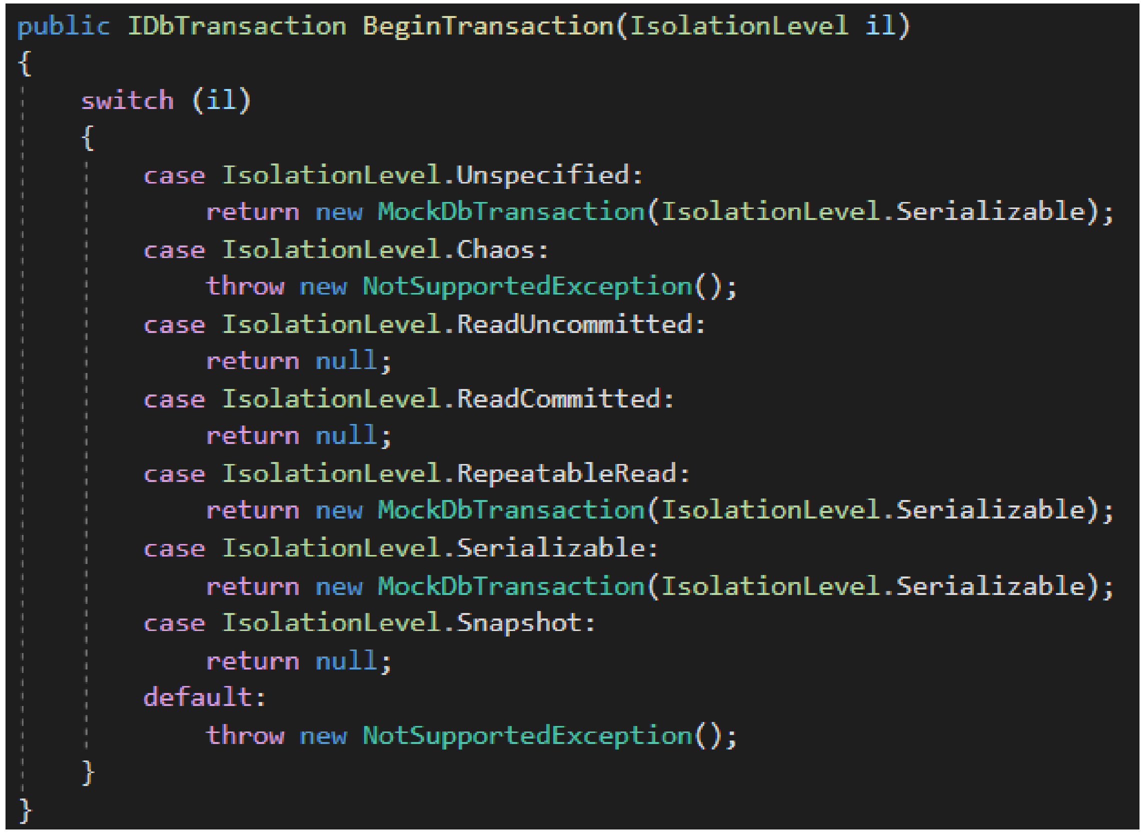
Moq Types

"Spies"

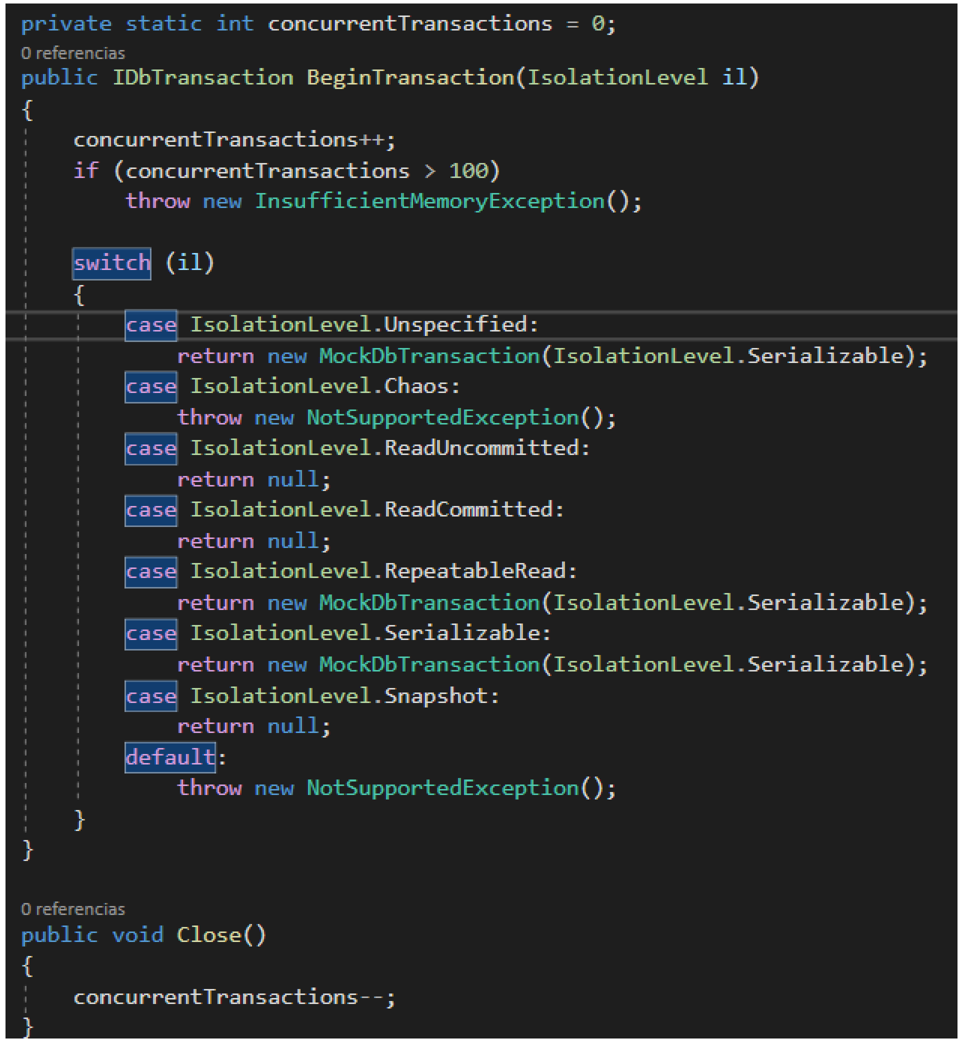
#somoshiberus



```
public IDbTransaction BeginTransaction(IsolationLevel il)
```



```
public IDbTransaction BeginTransaction(IsolationLevel il)
{
    switch (il)
    {
        case IsolationLevel.Unspecified:
            return new MockDbTransaction(IsolationLevel.Serializable);
        case IsolationLevel.Chaos:
            throw new NotSupportedException();
        case IsolationLevel.ReadUncommitted:
            return null;
        case IsolationLevel.ReadCommitted:
            return null;
        case IsolationLevel.RepeatableRead:
            return new MockDbTransaction(IsolationLevel.Serializable);
        case IsolationLevel.Serializable:
            return new MockDbTransaction(IsolationLevel.Serializable);
        case IsolationLevel.Snapshot:
            return null;
        default:
            throw new NotSupportedException();
    }
}
```

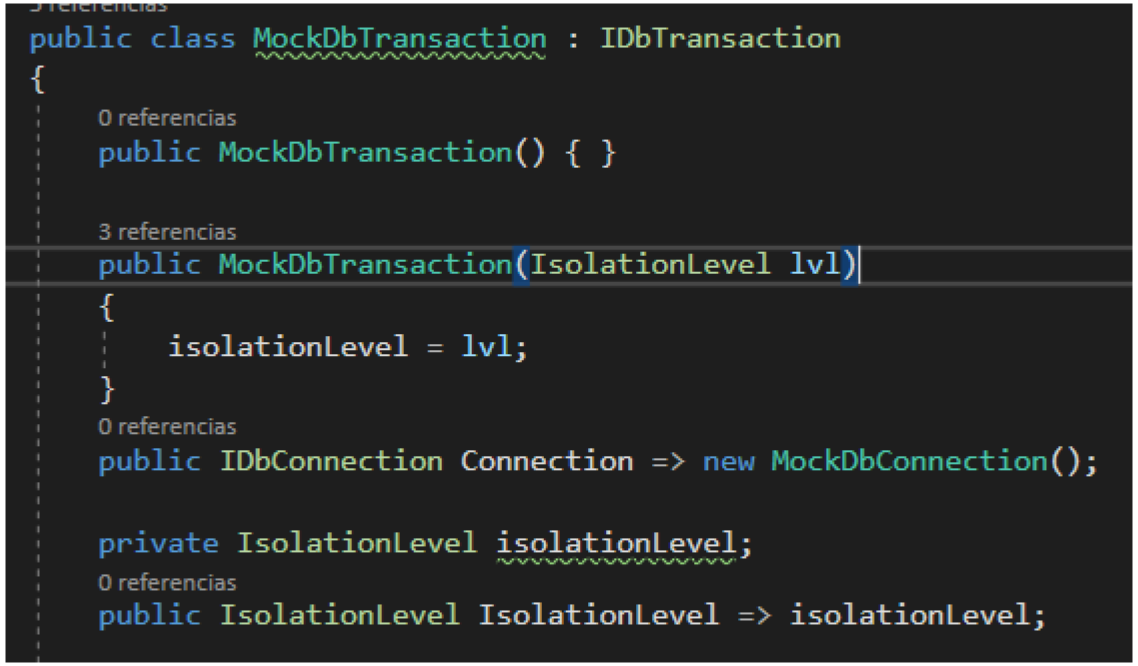


```
private static int concurrentTransactions = 0;
0 referencias
public IDbTransaction BeginTransaction(IsolationLevel il)
{
    concurrentTransactions++;
    if (concurrentTransactions > 100)
        throw new InsufficientMemoryException();

    switch (il)
    {
        case IsolationLevel.Unspecified:
            return new MockDbTransaction(IsolationLevel.Serializable);
        case IsolationLevel.Chaos:
            throw new NotSupportedException();
        case IsolationLevel.ReadUncommitted:
            return null;
        case IsolationLevel.ReadCommitted:
            return null;
        case IsolationLevel.RepeatableRead:
            return new MockDbTransaction(IsolationLevel.Serializable);
        case IsolationLevel.Serializable:
            return new MockDbTransaction(IsolationLevel.Serializable);
        case IsolationLevel.Snapshot:
            return null;
        default:
            throw new NotSupportedException();
    }
}

0 referencias
public void Close()
{
    concurrentTransactions--;
}
```

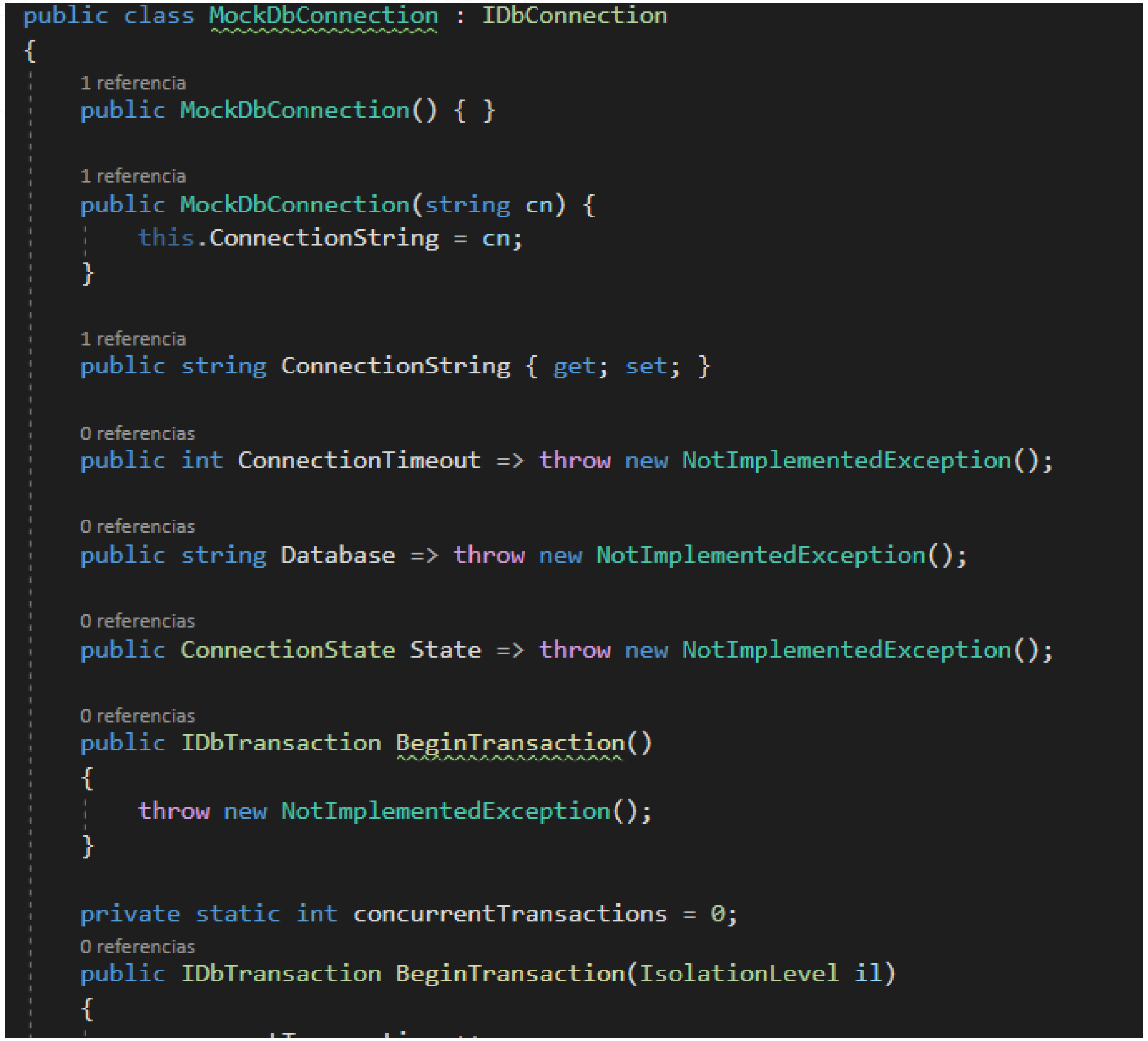
Moq Types

"Mocks"

#somoshiberus



```
public class MockDbTransaction : IDbTransaction
{
    0 referencias
    public MockDbTransaction() { }

    3 referencias
    public MockDbTransaction(IsolationLevel lvl)
    {
        isolationLevel = lvl;
    }
    0 referencias
    public IDbConnection Connection => new MockDbConnection();

    private IsolationLevel isolationLevel;
    0 referencias
    public IsolationLevel IsolationLevel => isolationLevel;
```



```
public class MockDbConnection : IDbConnection
{
    1 referencia
    public MockDbConnection() { }

    1 referencia
    public MockDbConnection(string cn) {
        this.ConnectionString = cn;
    }

    1 referencia
    public string ConnectionString { get; set; }

    0 referencias
    public int ConnectionTimeout => throw new NotImplementedException();

    0 referencias
    public string Database => throw new NotImplementedException();

    0 referencias
    public ConnectionState State => throw new NotImplementedException();

    0 referencias
    public IDbTransaction BeginTransaction()
    {
        throw new NotImplementedException();
    }

    private static int concurrentTransactions = 0;
    0 referencias
    public IDbTransaction BeginTransaction(IsolationLevel il)
    {
```

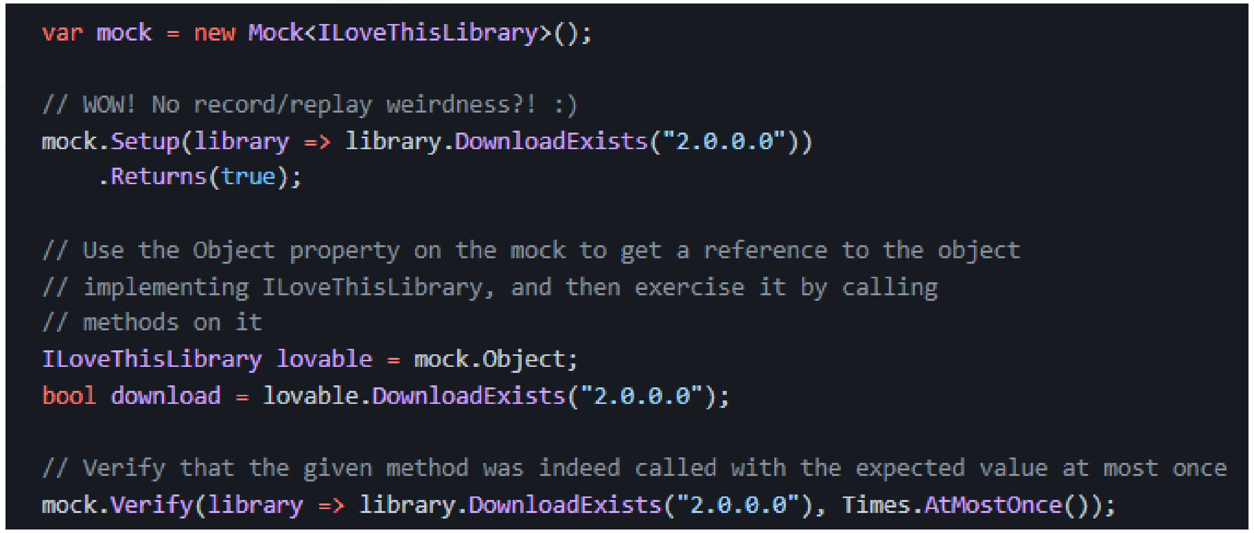
Nafarroako Gobernua Gobierno de Navarra

www.hiberus.com

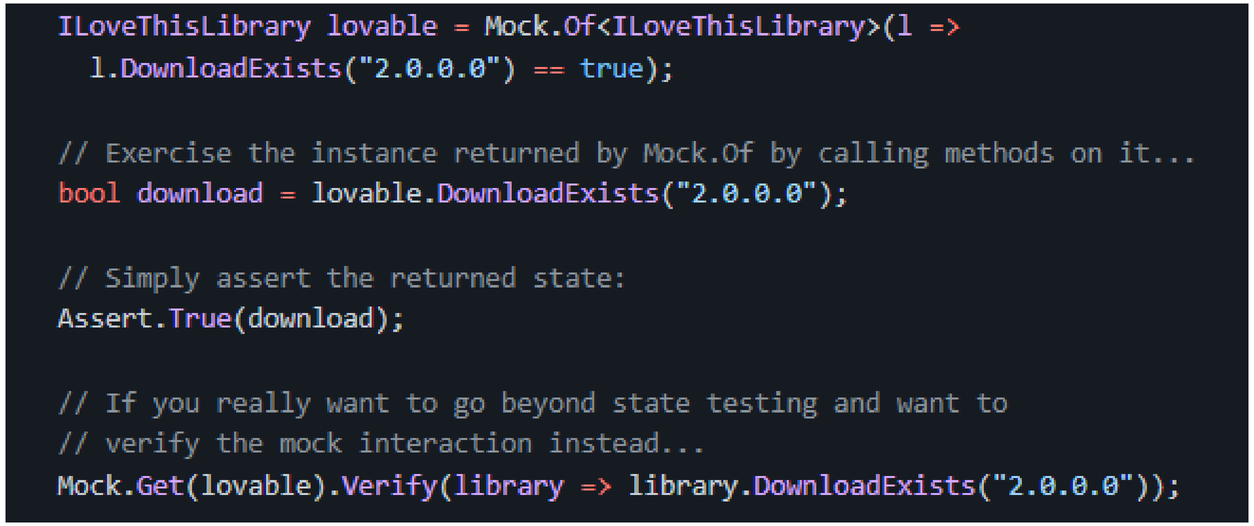
Moq

Let there be Mock!

#somoshiberus



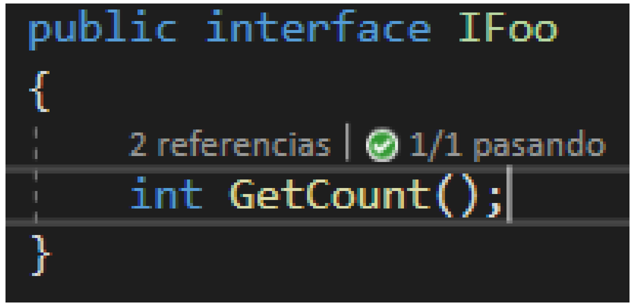
```
var mock = new Mock<ILoveThisLibrary>();

// WOW! No record/replay weirdness?! :)
mock.Setup(library => library.DownloadExists("2.0.0.0"))
    .Returns(true);

// Use the Object property on the mock to get a reference to the object
// implementing ILoveThisLibrary, and then exercise it by calling
// methods on it
ILoveThisLibrary lovable = mock.Object;
bool download = lovable.DownloadExists("2.0.0.0");

// Verify that the given method was indeed called with the expected value at most once
mock.Verify(library => library.DownloadExists("2.0.0.0"), Times.AtMostOnce());
```



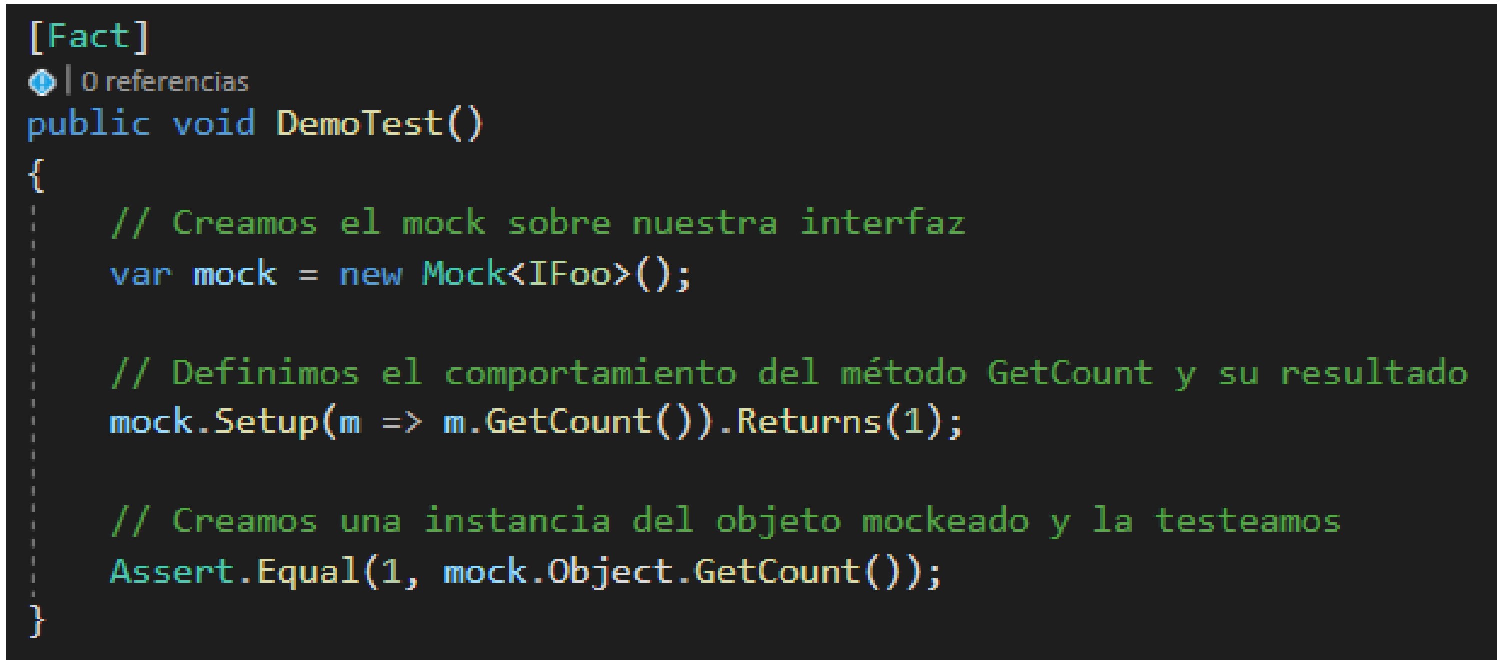
```
ILoveThisLibrary lovable = Mock.Of<ILoveThisLibrary>(l =>
  l.DownloadExists("2.0.0.0") == true);

// Exercise the instance returned by Mock.Of by calling methods on it...
bool download = lovable.DownloadExists("2.0.0.0");

// Simply assert the returned state:
Assert.True(download);

// If you really want to go beyond state testing and want to
// verify the mock interaction instead...
Mock.Get(lovable).Verify(library => library.DownloadExists("2.0.0.0"));
```

Moq

Let there be Mock!



```
public interface IFoo
{
        2 referencias | 1/1 pasando
        int GetCount();
}
```

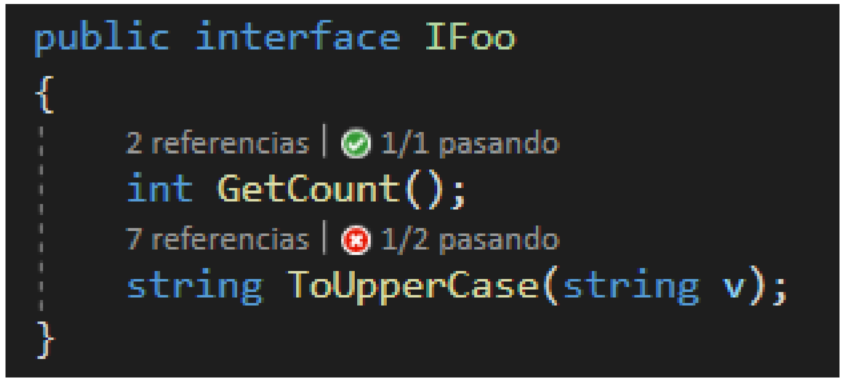


```
[Fact]
 | 0 referencias
public void DemoTest()
{
    // Creamos el mock sobre nuestra interfaz
    var mock = new Mock<IFoo>();

    // Definimos el comportamiento del método GetCount y su resultado
    mock.Setup(m => m.GetCount()).Returns(1);

    // Creamos una instancia del objeto mockeado y la testeamos
    Assert.Equal(1, mock.Object.GetCount());
}
```

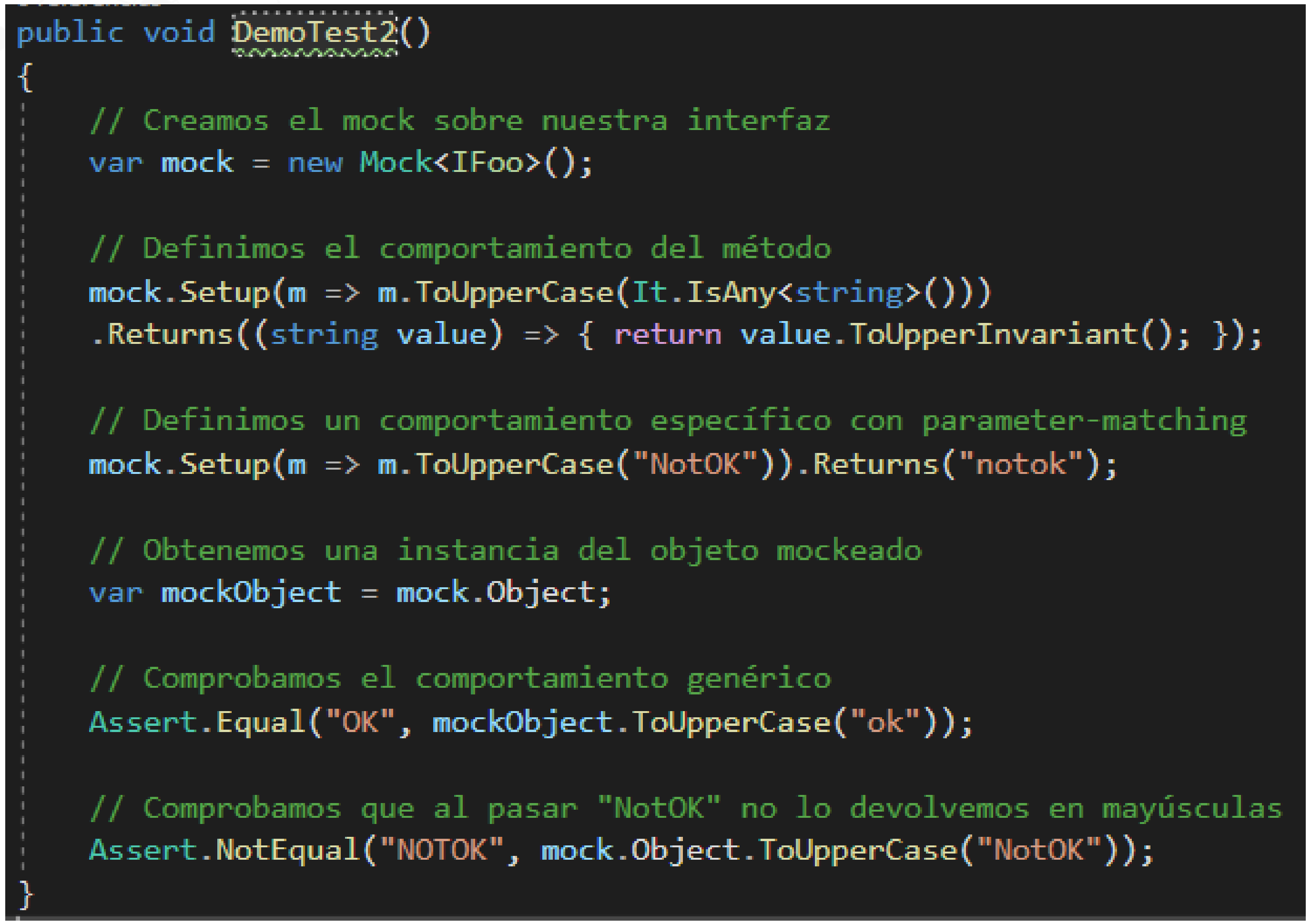
Moq

Let there be Mock!

#somoshiberus



```
public interface IFoo
{
    2 referencias | ✓ 1/1 pasando
    int GetCount();
    7 referencias | ✗ 1/2 pasando
    string ToUpperCase(string v);
}
```

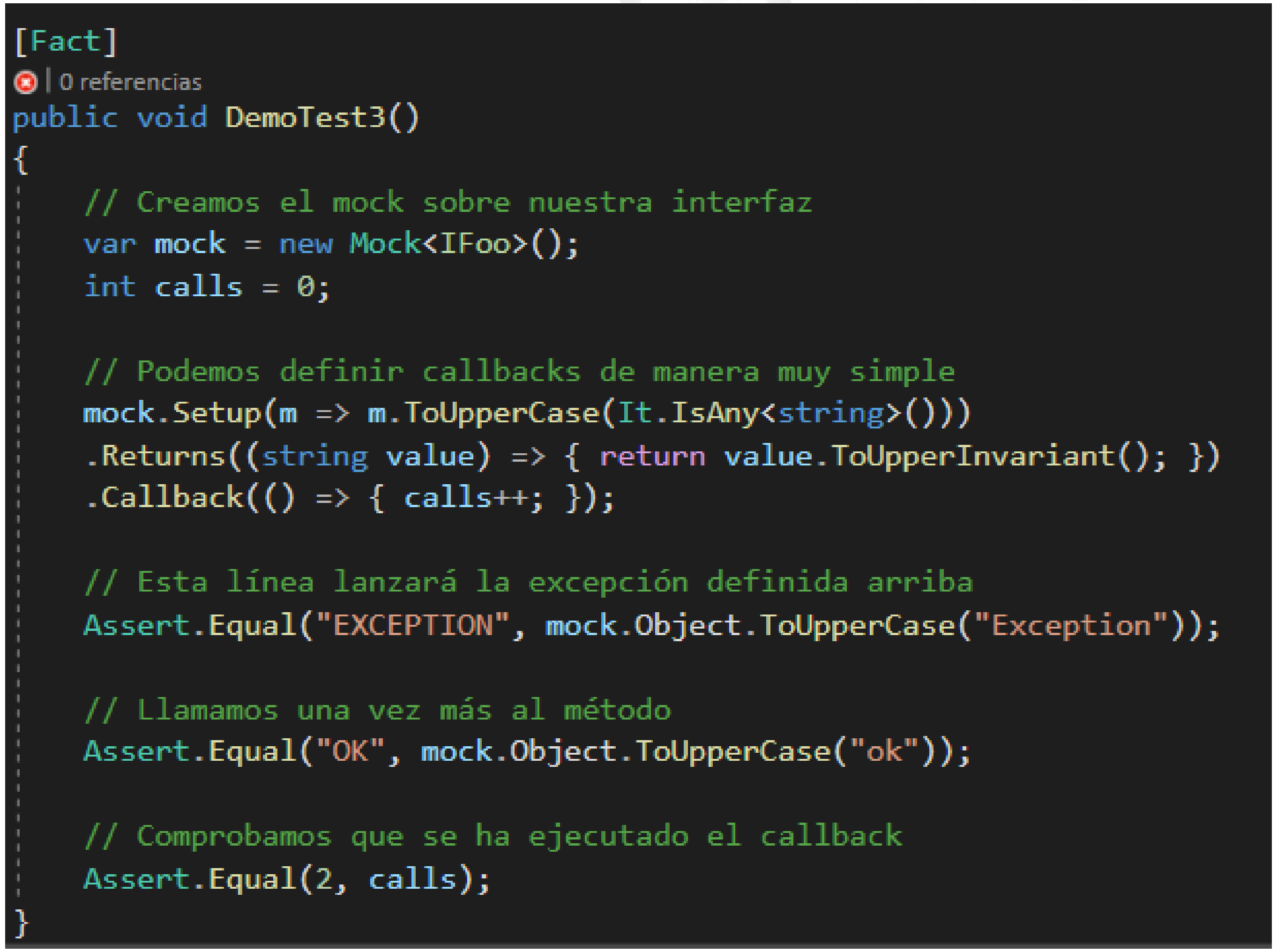


```
public void DemoTest2()
{
    // Creamos el mock sobre nuestra interfaz
    var mock = new Mock<IFoo>();

    // Definimos el comportamiento del método
    mock.Setup(m => m.ToUpperCase(It.IsAny<string>()))
    .Returns((string value) => { return value.ToUpperInvariant(); });

    // Definimos un comportamiento específico con parameter-matching
    mock.Setup(m => m.ToUpperCase("NotOK")).Returns("notok");

    // Obtenemos una instancia del objeto mockeado
    var mockObject = mock.Object;

    // Comprobamos el comportamiento genérico
    Assert.Equal("OK", mockObject.ToUpperCase("ok"));

    // Comprobamos que al pasar "NotOK" no lo devolvemos en mayúsculas
    Assert.NotEqual("NOTOK", mock.Object.ToUpperCase("NotOK"));
}
```

Moq

Let there be Mock!

#somoshiberus



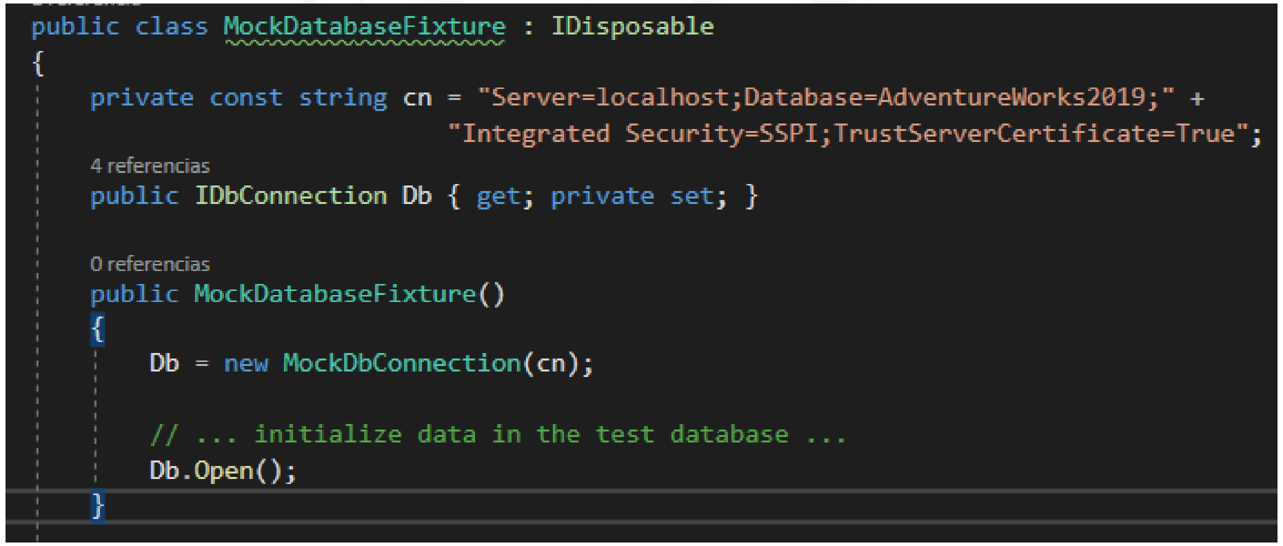
```
[Fact]
0 referencias
public void DemoTest3()
{
    // Creamos el mock sobre nuestra interfaz
    var mock = new Mock<IFoo>();
    int calls = 0;

    // Podemos definir callbacks de manera muy simple
    mock.Setup(m => m.ToUpperCase(It.IsAny<string>()))
    .Returns((string value) => { return value.ToUpperInvariant(); })
    .Callback(() => { calls++; });

    // Esta línea lanzará la excepción definida arriba
    Assert.Equal("EXCEPTION", mock.Object.ToUpperCase("Exception"));

    // Llamamos una vez más al método
    Assert.Equal("OK", mock.Object.ToUpperCase("ok"));

    // Comprobamos que se ha ejecutado el callback
    Assert.Equal(2, calls);
}
```

https://github.com/Moq/moq4/wiki/Quickstart

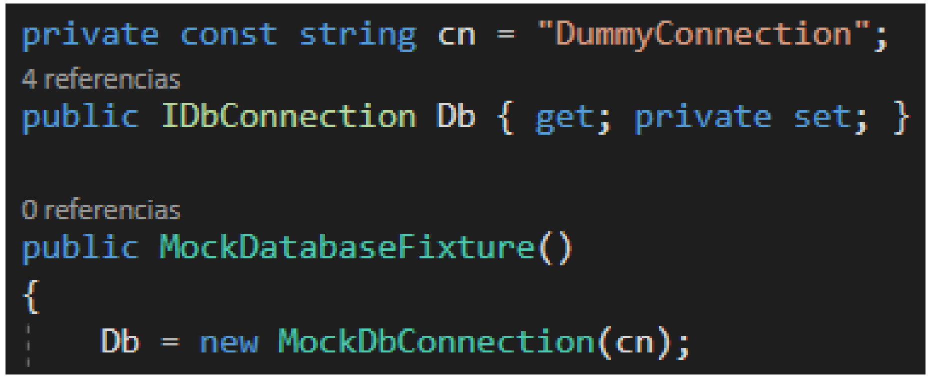
Moq

Dummy objects

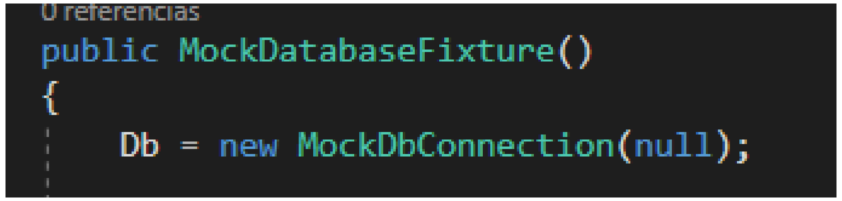
#somoshiberus



```
public class MockDatabaseFixture : IDisposable
{
    private const string cn = "Server=localhost;Database=AdventureWorks2019;" +
                              "Integrated Security=SSPI;TrustServerCertificate=True";
    4 referencias
    public IDbConnection Db { get; private set; }

    0 referencias
    public MockDatabaseFixture()
    {
        Db = new MockDbConnection(cn);

        // ... initialize data in the test database ...
        Db.Open();
    }
```



```
private const string cn = "DummyConnection";
4 referencias
public IDbConnection Db { get; private set; }

0 referencias
public MockDatabaseFixture()
{
    Db = new MockDbConnection(cn);
```



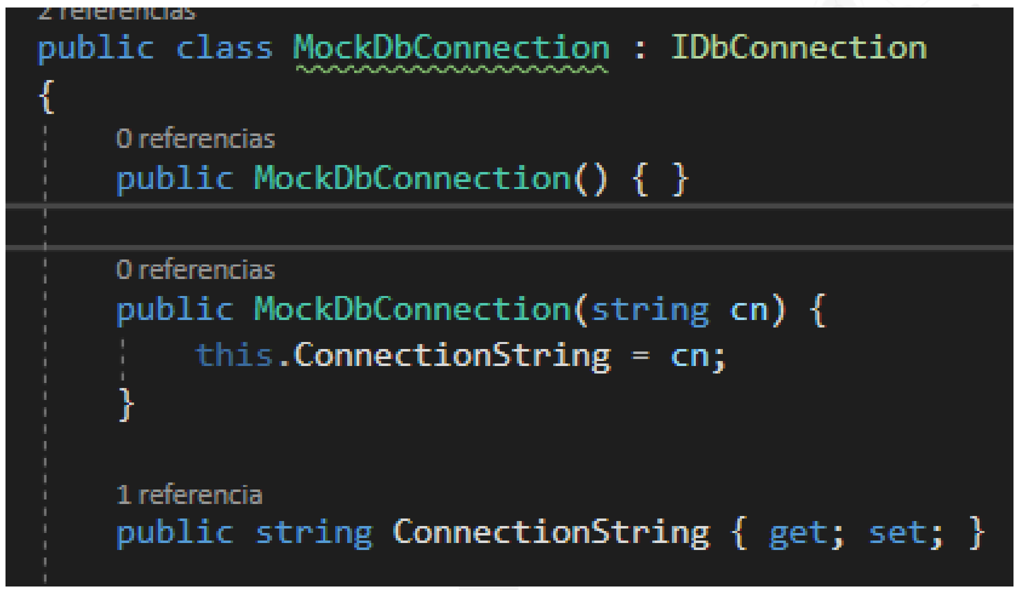
```
0 referencias
public MockDatabaseFixture()
{
    Db = new MockDbConnection(null);
```

Nafarroako Gobernua Gobierno de Navarra

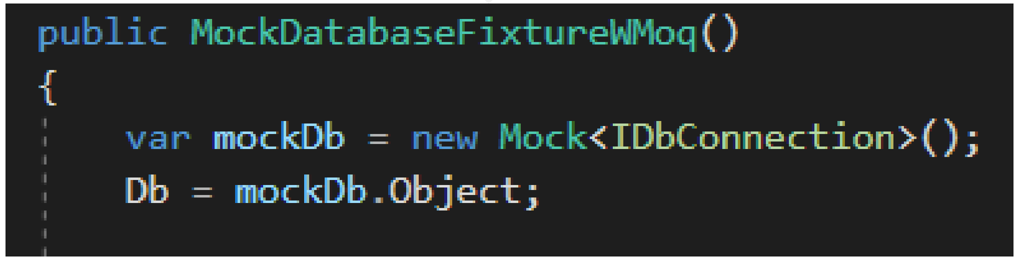
Moq

Fake objects

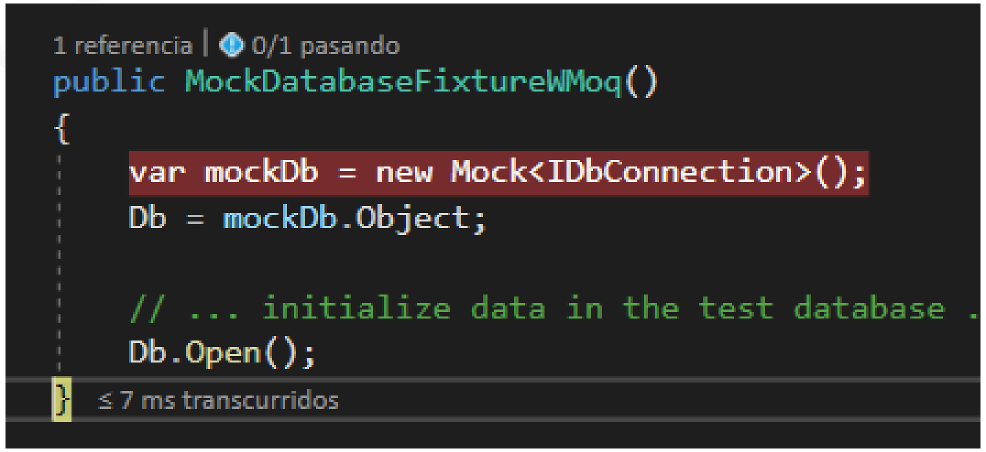
#somoshiberus



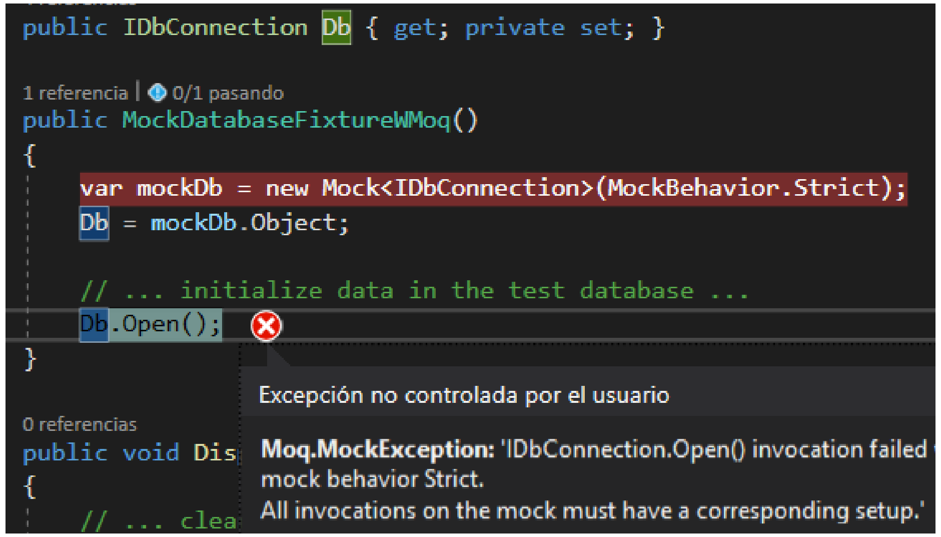
```
2 referencias
public class MockDbConnection : IDbConnection
{
    0 referencias
    public MockDbConnection() { }

    0 referencias
    public MockDbConnection(string cn) {
        this.ConnectionString = cn;
    }

    1 referencia
    public string ConnectionString { get; set; }
```



```
public MockDatabaseFixtureWMoq()
{
    var mockDb = new Mock<IDbConnection>();
    Db = mockDb.Object;
```



```
1 referencia | 0/1 pasando
public MockDatabaseFixtureWMoq()
{
    var mockDb = new Mock<IDbConnection>();
    Db = mockDb.Object;

    // ... initialize data in the test database .
    Db.Open();
} ≤ 7 ms transcurridos
```



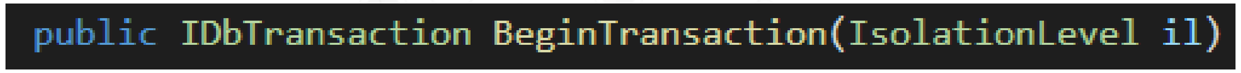
```
public IDbConnection Db { get; private set; }

1 referencia | 0/1 pasando
public MockDatabaseFixtureWMoq()
{
    var mockDb = new Mock<IDbConnection>(MockBehavior.Strict);
    Db = mockDb.Object;

    // ... initialize data in the test database ...
    Db.Open();
}

0 referencias
public void Dis
{
    // ... clea
```

Excepción no controlada por el usuario

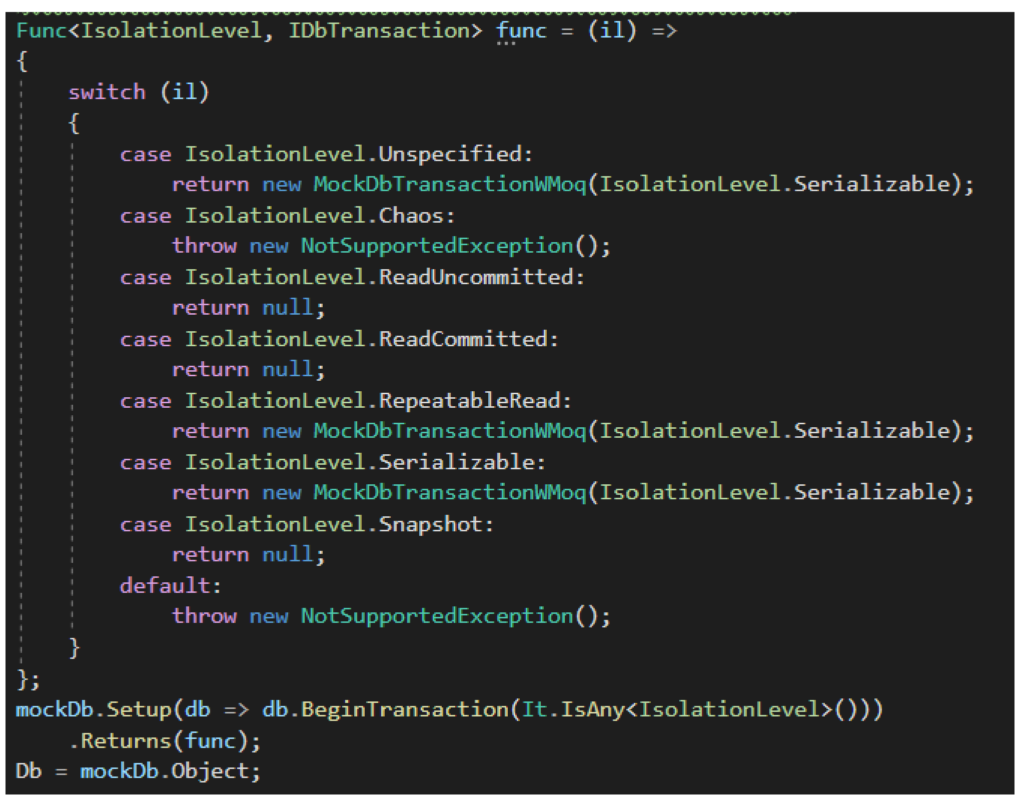
Moq.MockException: 'IDbConnection.Open() invocation failed
mock behavior Strict.
All invocations on the mock must have a corresponding setup.'

Moq Types

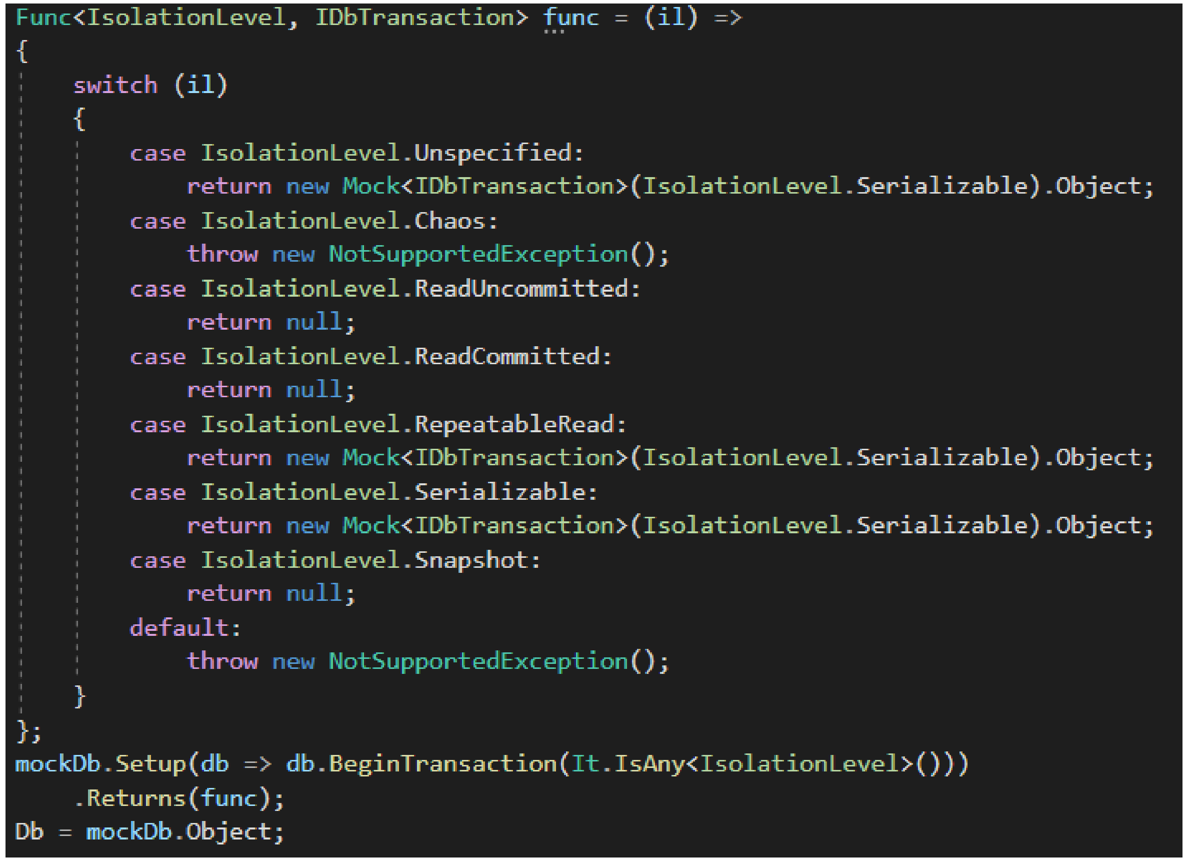
Stub objects



```
public IDbTransaction BeginTransaction(IsolationLevel il)
```



```
Func<IsolationLevel, IDbTransaction> func = (il) =>
{
    switch (il)
    {
        case IsolationLevel.Unspecified:
            return new MockDbTransactionWMoq(IsolationLevel.Serializable);
        case IsolationLevel.Chaos:
            throw new NotSupportedException();
        case IsolationLevel.ReadUncommitted:
            return null;
        case IsolationLevel.ReadCommitted:
            return null;
        case IsolationLevel.RepeatableRead:
            return new MockDbTransactionWMoq(IsolationLevel.Serializable);
        case IsolationLevel.Serializable:
            return new MockDbTransactionWMoq(IsolationLevel.Serializable);
        case IsolationLevel.Snapshot:
            return null;
        default:
            throw new NotSupportedException();
    }
};
mockDb.Setup(db => db.BeginTransaction(It.IsAny<IsolationLevel>()))
    .Returns(func);
Db = mockDb.Object;
```



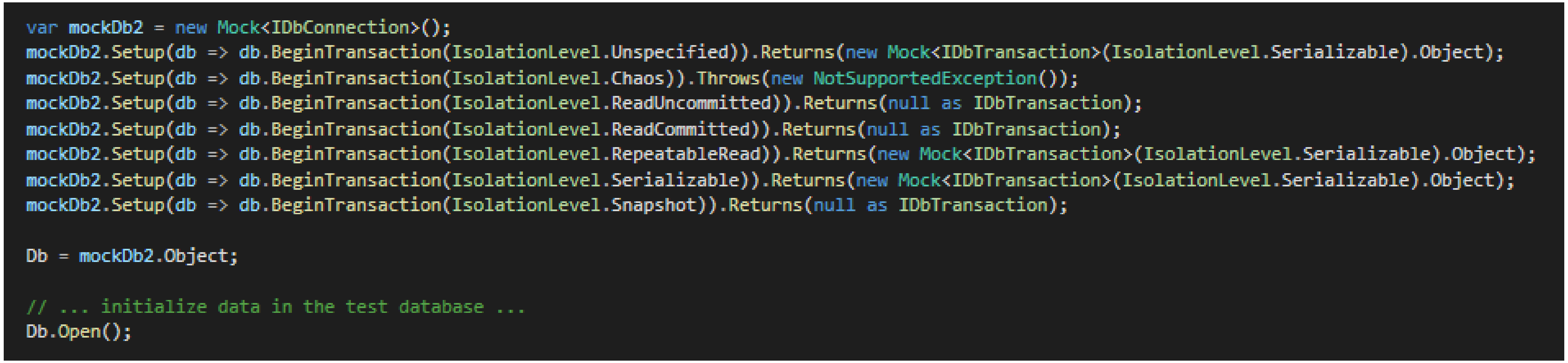
```
Func<IsolationLevel, IDbTransaction> func = (il) =>
{
    switch (il)
    {
        case IsolationLevel.Unspecified:
            return new Mock<IDbTransaction>(IsolationLevel.Serializable).Object;
        case IsolationLevel.Chaos:
            throw new NotSupportedException();
        case IsolationLevel.ReadUncommitted:
            return null;
        case IsolationLevel.ReadCommitted:
            return null;
        case IsolationLevel.RepeatableRead:
            return new Mock<IDbTransaction>(IsolationLevel.Serializable).Object;
        case IsolationLevel.Serializable:
            return new Mock<IDbTransaction>(IsolationLevel.Serializable).Object;
        case IsolationLevel.Snapshot:
            return null;
        default:
            throw new NotSupportedException();
    }
};
mockDb.Setup(db => db.BeginTransaction(It.IsAny<IsolationLevel>()))
    .Returns(func);
Db = mockDb.Object;
```

Moq Types

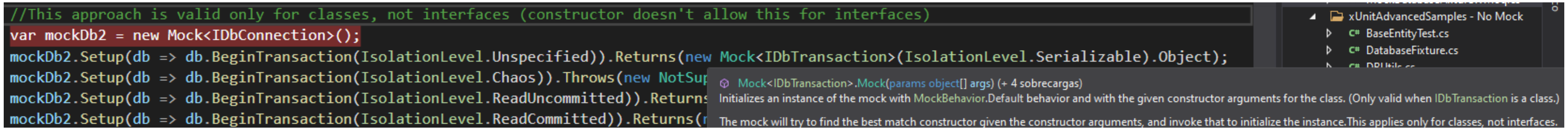
Stub objects

#somoshiberus

```
public IDbTransaction BeginTransaction(IsolationLevel il)
```



```
var mockDb2 = new Mock<IDbConnection>();
mockDb2.Setup(db => db.BeginTransaction(IsolationLevel.Unspecified)).Returns(new Mock<IDbTransaction>(IsolationLevel.Serializable).Object);
mockDb2.Setup(db => db.BeginTransaction(IsolationLevel.Chaos)).Throws(new NotSupportedException());
mockDb2.Setup(db => db.BeginTransaction(IsolationLevel.ReadUncommitted)).Returns(null as IDbTransaction);
mockDb2.Setup(db => db.BeginTransaction(IsolationLevel.ReadCommitted)).Returns(null as IDbTransaction);
mockDb2.Setup(db => db.BeginTransaction(IsolationLevel.RepeatableRead)).Returns(new Mock<IDbTransaction>(IsolationLevel.Serializable).Object);
mockDb2.Setup(db => db.BeginTransaction(IsolationLevel.Serializable)).Returns(new Mock<IDbTransaction>(IsolationLevel.Serializable).Object);
mockDb2.Setup(db => db.BeginTransaction(IsolationLevel.Snapshot)).Returns(null as IDbTransaction);

Db = mockDb2.Object;

// ... initialize data in the test database ...
Db.Open();
```



```
//This approach is valid only for classes, not interfaces (constructor doesn't allow this for interfaces)
var mockDb2 = new Mock<IDbConnection>();
mockDb2.Setup(db => db.BeginTransaction(IsolationLevel.Unspecified)).Returns(new Mock<IDbTransaction>(IsolationLevel.Serializable).Object);
mockDb2.Setup(db => db.BeginTransaction(IsolationLevel.Chaos)).Throws(new NotSup
mockDb2.Setup(db => db.BeginTransaction(IsolationLevel.ReadUncommitted)).Returns
mockDb2.Setup(db => db.BeginTransaction(IsolationLevel.ReadCommitted)).Returns(r
```

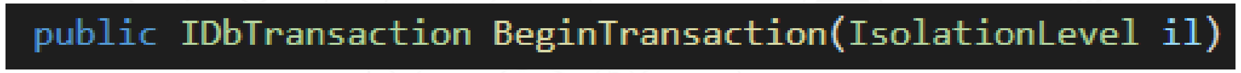
Mock<IDbTransaction>.Mock(params object[] args) (+ 4 sobrecargas)
Initializes an instance of the mock with MockBehavior.Default behavior and with the given constructor arguments for the class. (Only valid when IDbTransaction is a class.)

The mock will try to find the best match constructor given the constructor arguments, and invoke that to initialize the instance.This applies only for classes, not interfaces.

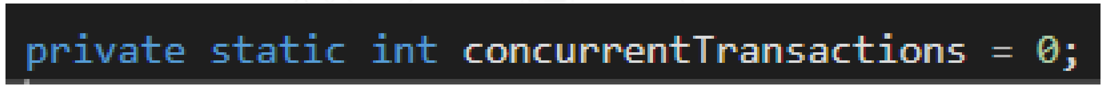
Moq Types

"Spies"

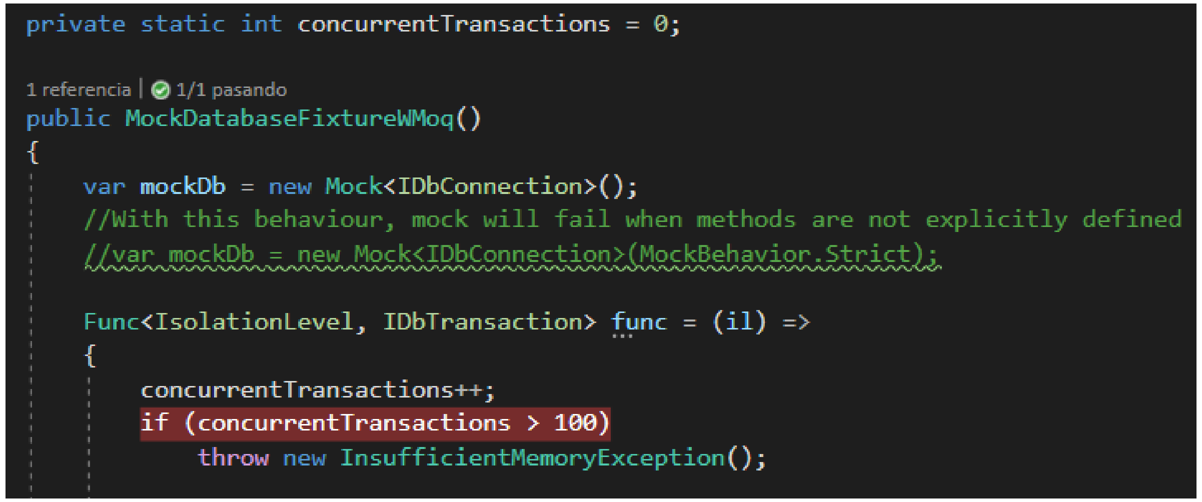
#somoshiberus



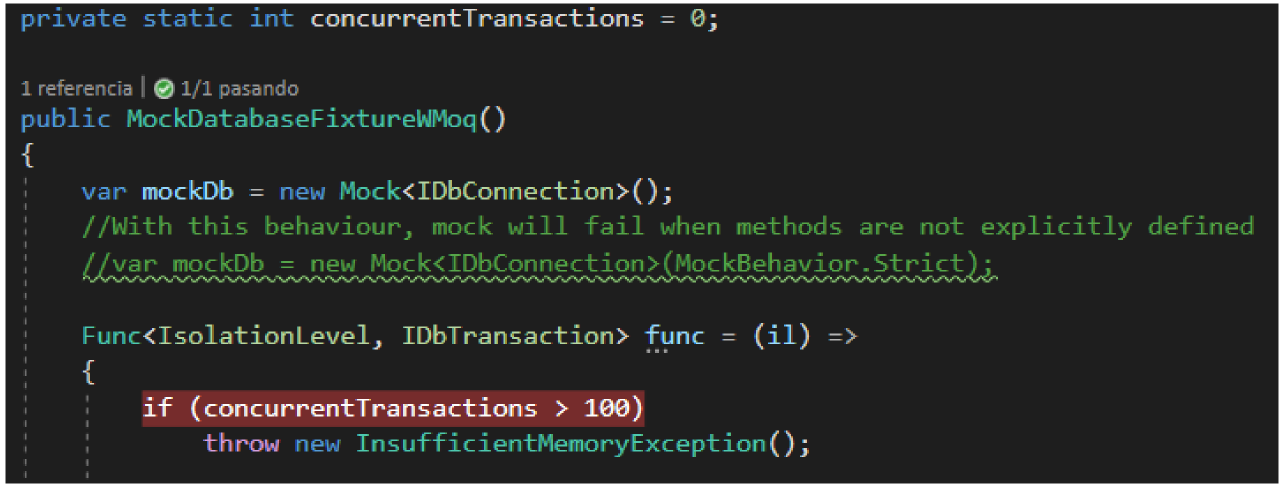
```
public IDbTransaction BeginTransaction(IsolationLevel il)
```



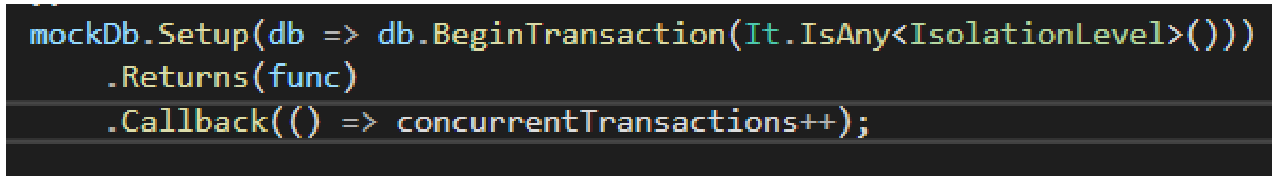
```
private static int concurrentTransactions = 0;
```



```
private static int concurrentTransactions = 0;

1 referencia | 1/1 pasando
public MockDatabaseFixtureWMoq()
{
    var mockDb = new Mock<IDbConnection>();
    //With this behaviour, mock will fail when methods are not explicitly defined
    //var mockDb = new Mock<IDbConnection>(MockBehavior.Strict);

    Func<IsolationLevel, IDbTransaction> func = (il) =>
    {
        concurrentTransactions++;
        if (concurrentTransactions > 100)
            throw new InsufficientMemoryException();
```



```
private static int concurrentTransactions = 0;

1 referencia | 1/1 pasando
public MockDatabaseFixtureWMoq()
{
    var mockDb = new Mock<IDbConnection>();
    //With this behaviour, mock will fail when methods are not explicitly defined
    //var mockDb = new Mock<IDbConnection>(MockBehavior.Strict);

    Func<IsolationLevel, IDbTransaction> func = (il) =>
    {
        if (concurrentTransactions > 100)
            throw new InsufficientMemoryException();
```

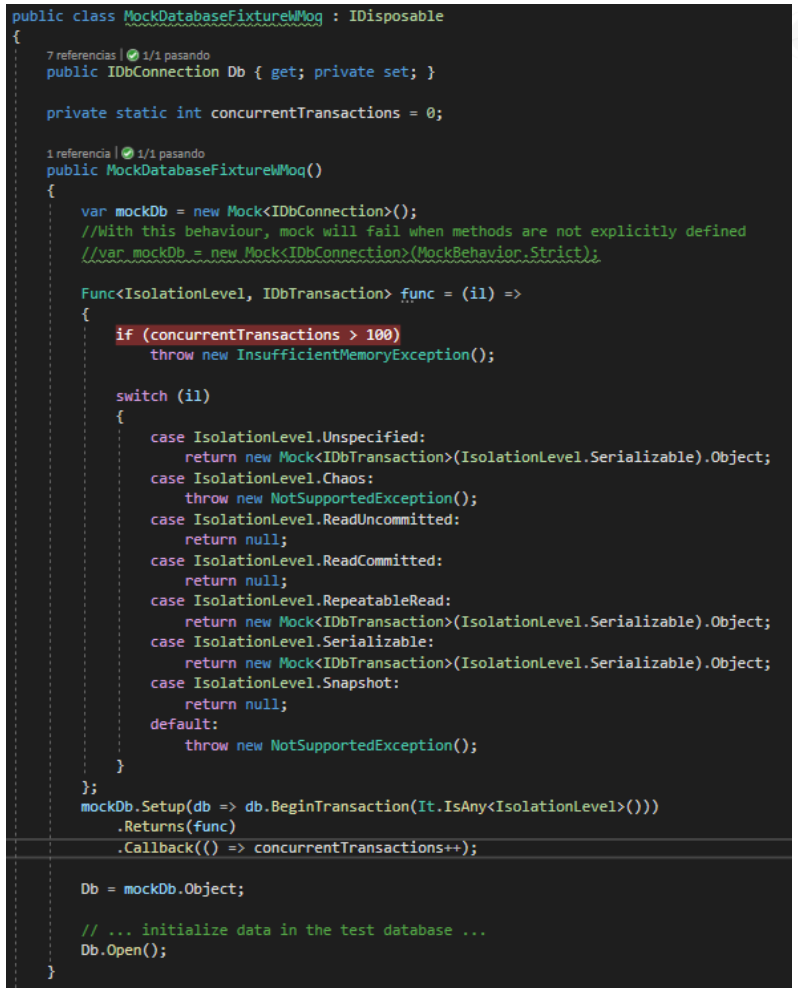


```
mockDb.Setup(db => db.BeginTransaction(It.IsAny<IsolationLevel>()))
    .Returns(func)
    .Callback(() => concurrentTransactions++);
```

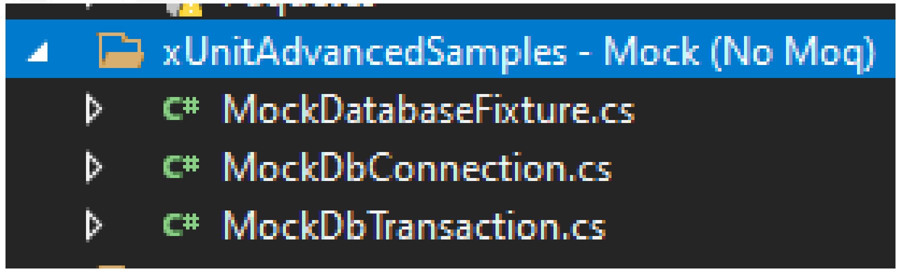
Moq Types

“Mocks"

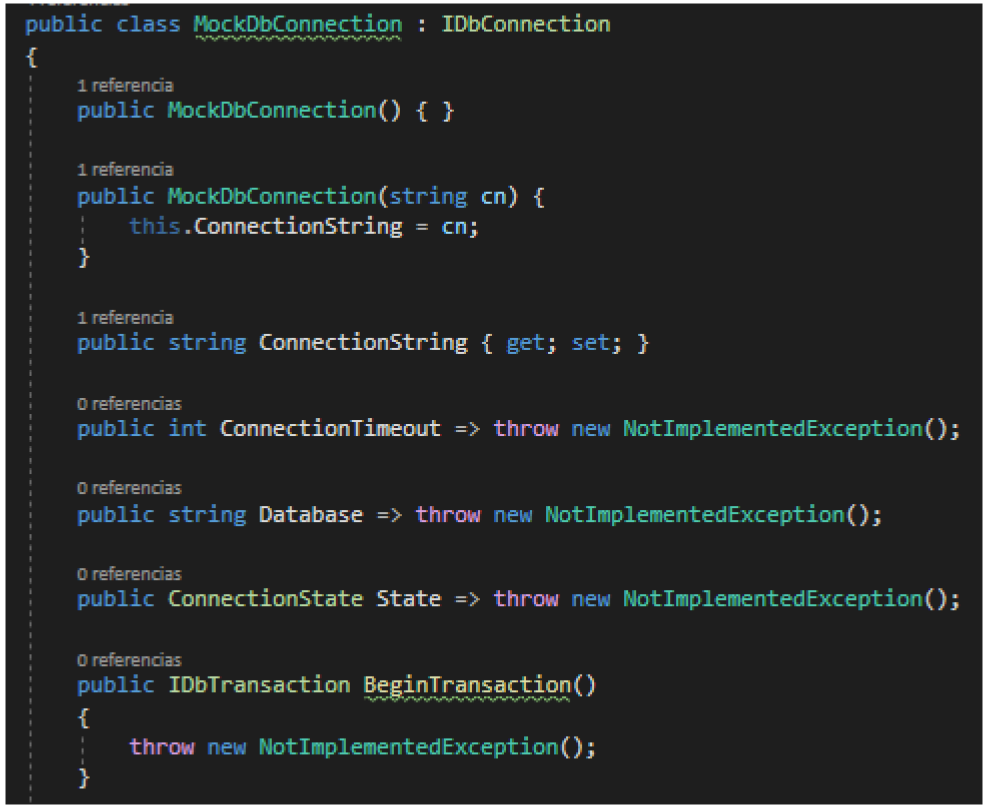
#somoshiberus



```
public class MockDatabaseFixtureWMoq : IDisposable
{
    7 referencias | 1/1 pasando
    public IDbConnection Db { get; private set; }

    private static int concurrentTransactions = 0;

    1 referencia | 1/1 pasando
    public MockDatabaseFixtureWMoq()
    {
        var mockDb = new Mock<IDbConnection>();
        //With this behaviour, mock will fail when methods are not explicitly defined
        //var mockDb = new Mock<IDbConnection>(MockBehavior.Strict);

        Func<IsolationLevel, IDbTransaction> func = (il) =>
        {
            if (concurrentTransactions > 100)
                throw new InsufficientMemoryException();

            switch (il)
            {
                case IsolationLevel.Unspecified:
                    return new Mock<IDbTransaction>(IsolationLevel.Serializable).Object;
                case IsolationLevel.Chaos:
                    throw new NotSupportedException();
                case IsolationLevel.ReadUncommitted:
                    return null;
                case IsolationLevel.ReadCommitted:
                    return null;
                case IsolationLevel.RepeatableRead:
                    return new Mock<IDbTransaction>(IsolationLevel.Serializable).Object;
                case IsolationLevel.Serializable:
                    return new Mock<IDbTransaction>(IsolationLevel.Serializable).Object;
                case IsolationLevel.Snapshot:
                    return null;
                default:
                    throw new NotSupportedException();
            }
        };
        mockDb.Setup(db => db.BeginTransaction(It.IsAny<IsolationLevel>()))
            .Returns(func)
            .Callback(() => concurrentTransactions++);

        Db = mockDb.Object;

        // ... initialize data in the test database ...
        Db.Open();
    }
```



```
xUnitAdvancedSamples - Mock (No Moq)
    C# MockDatabaseFixture.cs
    C# MockDbConnection.cs
    C# MockDbTransaction.cs
```



```
public class MockDbConnection : IDbConnection
{
    1 referencia
    public MockDbConnection() { }

    1 referencia
    public MockDbConnection(string cn) {
        this.ConnectionString = cn;
    }

    1 referencia
    public string ConnectionString { get; set; }

    0 referencias
    public int ConnectionTimeout => throw new NotImplementedException();

    0 referencias
    public string Database => throw new NotImplementedException();

    0 referencias
    public ConnectionState State => throw new NotImplementedException();

    0 referencias
    public IDbTransaction BeginTransaction()
    {
        throw new NotImplementedException();
    }
```

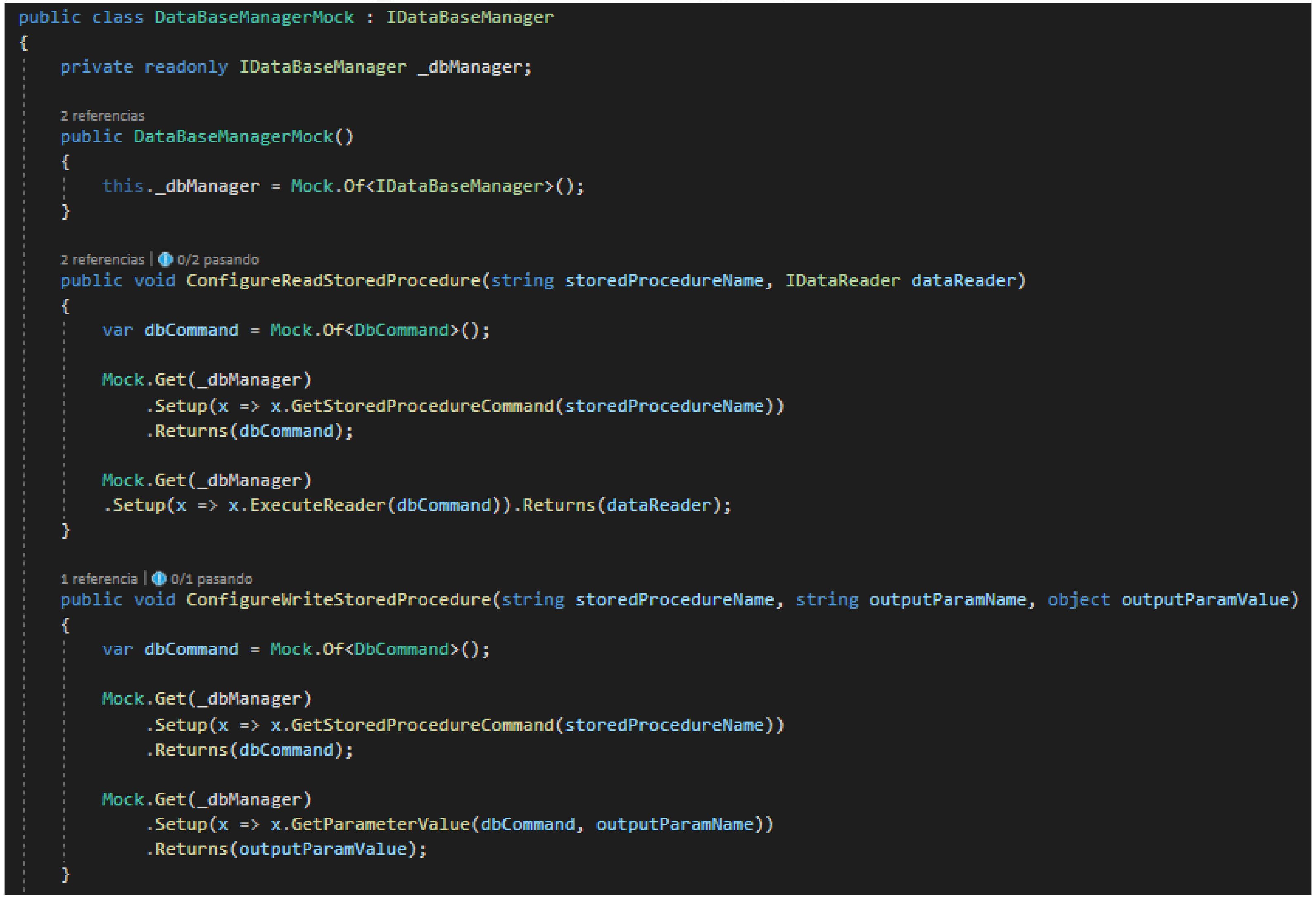
Nafarroako Gobernua | Gobierno de Navarra

www.hiberus.com

Moq Types

“Mocks"

#somoshiberus



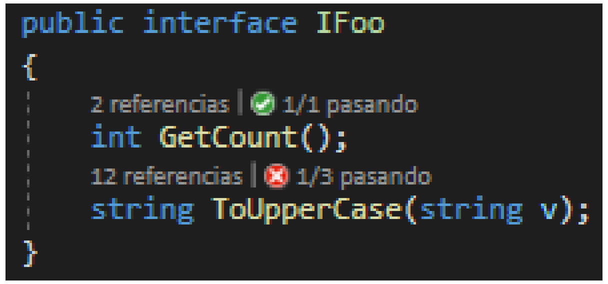
```
public class DataBaseManagerMock : IDataBaseManager
{
    private readonly IDataBaseManager _dbManager;

    2 referencias
    public DataBaseManagerMock()
    {
        this._dbManager = Mock.Of<IDataBaseManager>();
    }

    2 referencias | 0/2 pasando
    public void ConfigureReadStoredProcedure(string storedProcedureName, IDataReader dataReader)
    {
        var dbCommand = Mock.Of<DbCommand>();

        Mock.Get(_dbManager)
            .Setup(x => x.GetStoredProcedureCommand(storedProcedureName))
            .Returns(dbCommand);

        Mock.Get(_dbManager)
        .Setup(x => x.ExecuteReader(dbCommand)).Returns(dataReader);
    }

    1 referencia | 0/1 pasando
    public void ConfigureWriteStoredProcedure(string storedProcedureName, string outputParamName, object outputParamValue)
    {
        var dbCommand = Mock.Of<DbCommand>();

        Mock.Get(_dbManager)
            .Setup(x => x.GetStoredProcedureCommand(storedProcedureName))
            .Returns(dbCommand);

        Mock.Get(_dbManager)
            .Setup(x => x.GetParameterValue(dbCommand, outputParamName))
            .Returns(outputParamValue);
    }
```

Nafarroako Gobernua Gobierno de Navarra

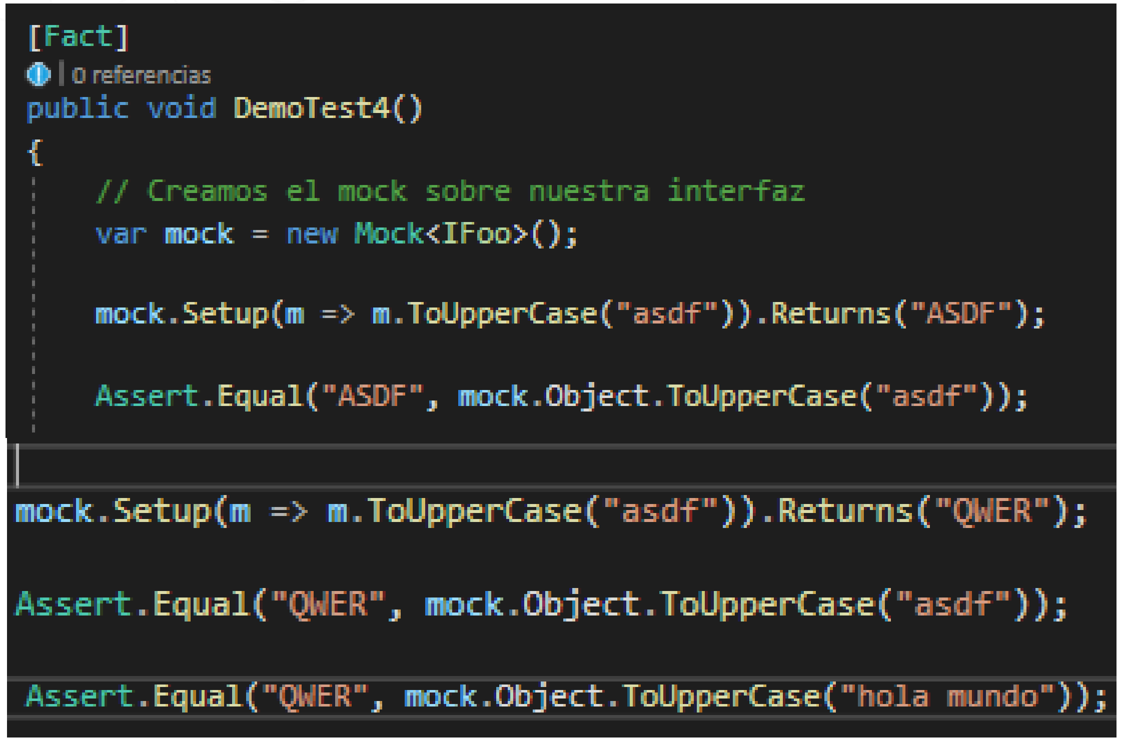
Moq

Overriding behaviours

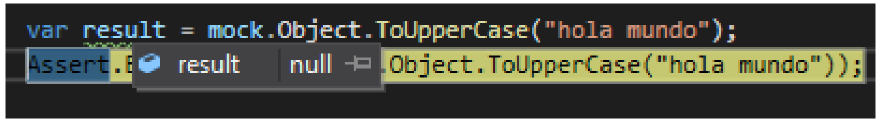
#somoshiberus



```
public interface IFoo
{
    2 referencias | 1/1 pasando
    int GetCount();
    12 referencias | 1/3 pasando
    string ToUpperCase(string v);
}
```



```
[Fact]
0 referencias
public void DemoTest4()
{
    // Creamos el mock sobre nuestra interfaz
    var mock = new Mock<IFoo>();

    mock.Setup(m => m.ToUpperCase("asdf")).Returns("ASDF");

    Assert.Equal("ASDF", mock.Object.ToUpperCase("asdf"));

mock.Setup(m => m.ToUpperCase("asdf")).Returns("QWER");

Assert.Equal("QWER", mock.Object.ToUpperCase("asdf"));

Assert.Equal("QWER", mock.Object.ToUpperCase("hola mundo"));
```



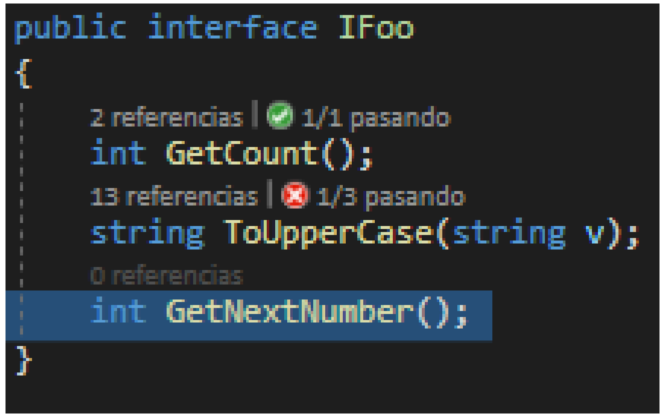
```
var result = mock.Object.ToUpperCase("hola mundo");
Assert.E  result   null  .Object.ToUpperCase("hola mundo"));
```

Nafarroako Gobernua Gobierno de Navarra

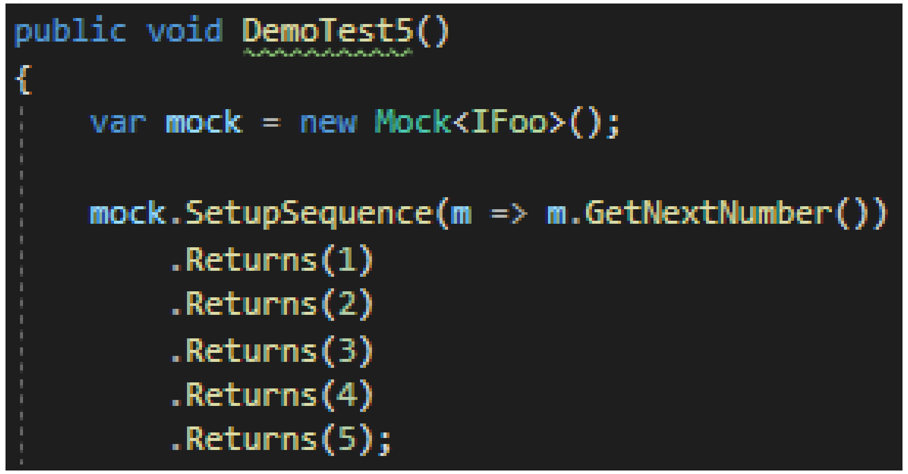
Moq

Sequences

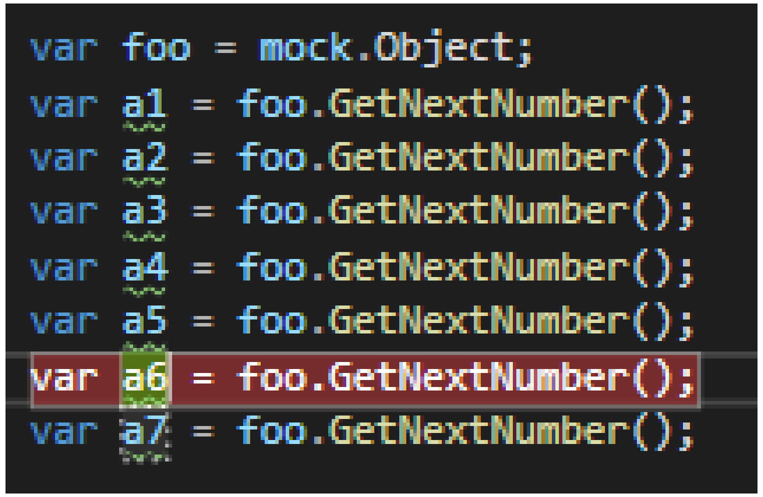
#somoshiberus



```
public interface IFoo
{
    2 referencias | 1/1 pasando
    int GetCount();
    13 referencias | 1/3 pasando
    string ToUpperCase(string v);
    0 referencias
    int GetNextNumber();
}
```



```
public void DemoTest5()
{
    var mock = new Mock<IFoo>();

    mock.SetupSequence(m => m.GetNextNumber())
        .Returns(1)
        .Returns(2)
        .Returns(3)
        .Returns(4)
        .Returns(5);
```

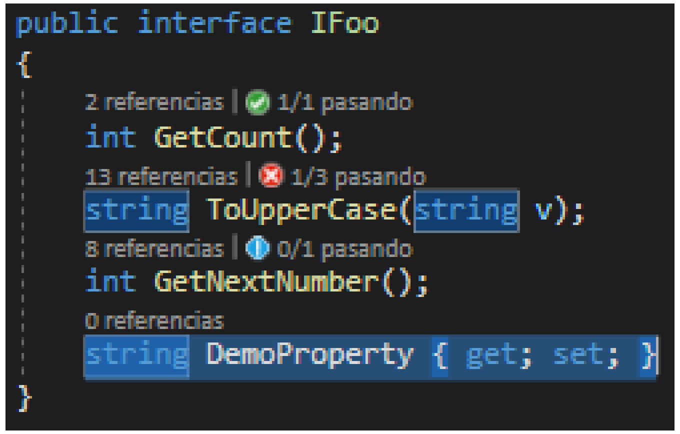


```
var foo = mock.Object;
var a1 = foo.GetNextNumber();
var a2 = foo.GetNextNumber();
var a3 = foo.GetNextNumber();
var a4 = foo.GetNextNumber();
var a5 = foo.GetNextNumber();
var a6 = foo.GetNextNumber();
var a7 = foo.GetNextNumber();
```

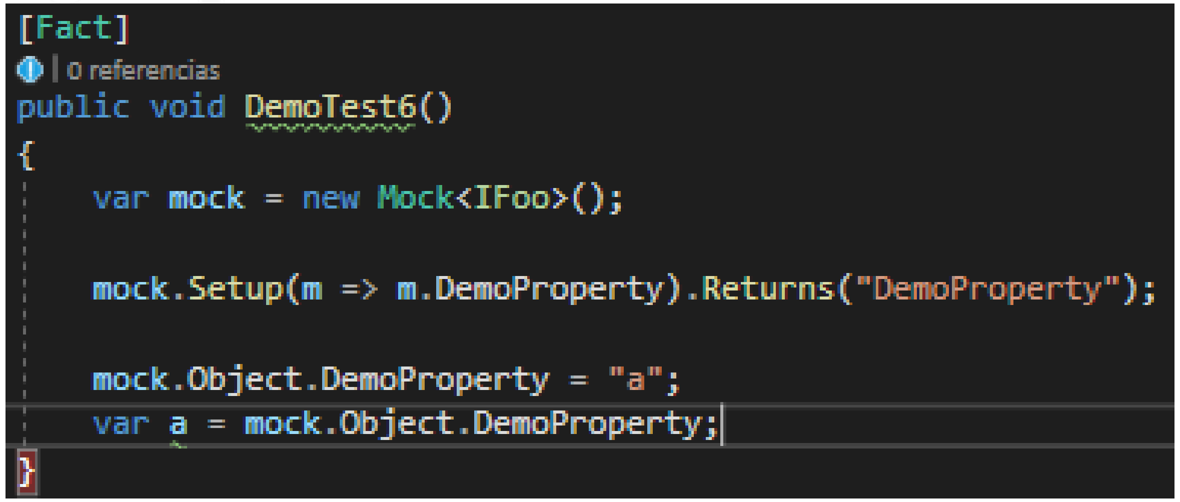
Moq

Properties

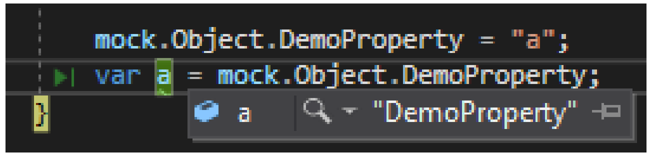
#somoshiberus



```
public interface IFoo
{
    2 referencias | 1/1 pasando
    int GetCount();
    13 referencias | 1/3 pasando
    string ToUpperCase(string v);
    8 referencias | 0/1 pasando
    int GetNextNumber();
    0 referencias
    string DemoProperty { get; set; }
}
```



```
[Fact]
0 referencias
public void DemoTest6()
{
    var mock = new Mock<IFoo>();

    mock.Setup(m => m.DemoProperty).Returns("DemoProperty");

    mock.Object.DemoProperty = "a";
    var a = mock.Object.DemoProperty;
}
```



```
    mock.Object.DemoProperty = "a";
    var a = mock.Object.DemoProperty;
}       a    "DemoProperty"
```

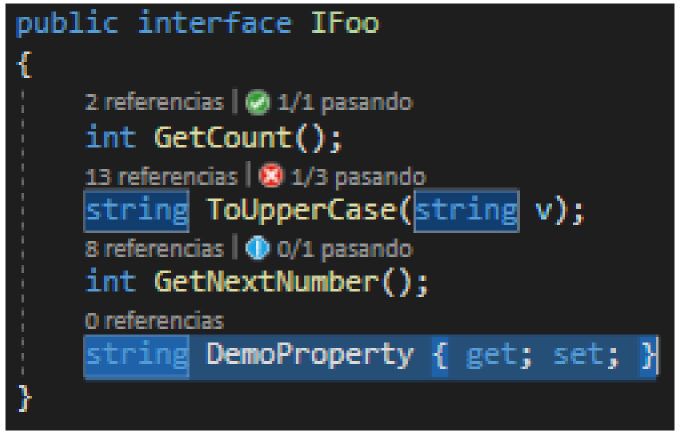
Nafarroako Gobernua Gobierno de Navarra

www.hiberus.com

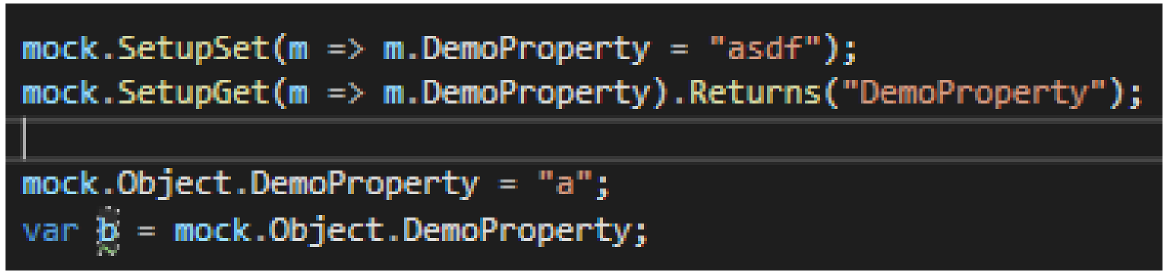
Moq

Properties

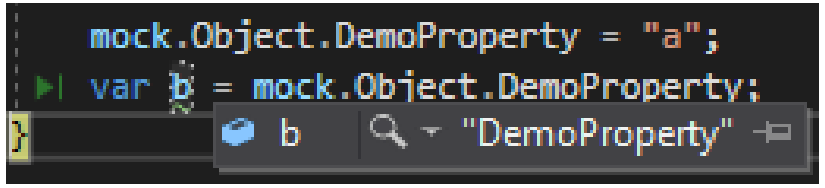
#somoshiberus



```
public interface IFoo
{
    2 referencias | ✓ 1/1 pasando
    int GetCount();
    13 referencias | ✗ 1/3 pasando
    string ToUpperCase(string v);
    8 referencias | 0/1 pasando
    int GetNextNumber();
    0 referencias
    string DemoProperty { get; set; }
}
```



```
mock.SetupSet(m => m.DemoProperty = "asdf");
mock.SetupGet(m => m.DemoProperty).Returns("DemoProperty");

mock.Object.DemoProperty = "a";
var b = mock.Object.DemoProperty;
```

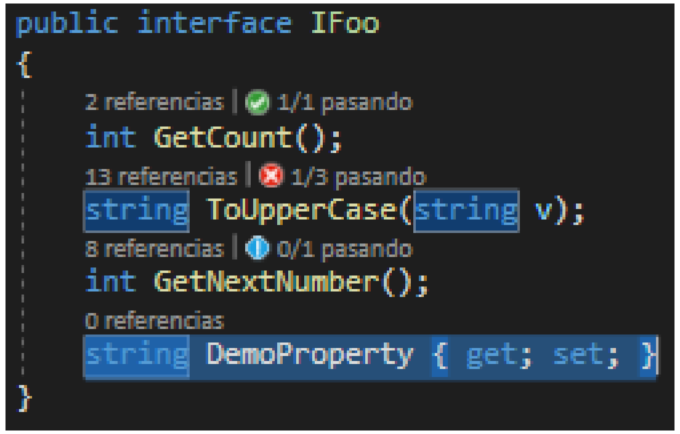


```
    mock.Object.DemoProperty = "a";
    var b = mock.Object.DemoProperty;
}       b    "DemoProperty"
```

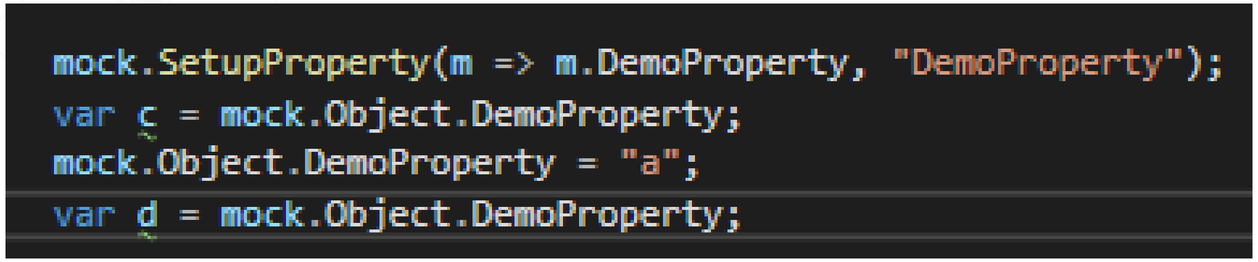
Moq

Properties with state

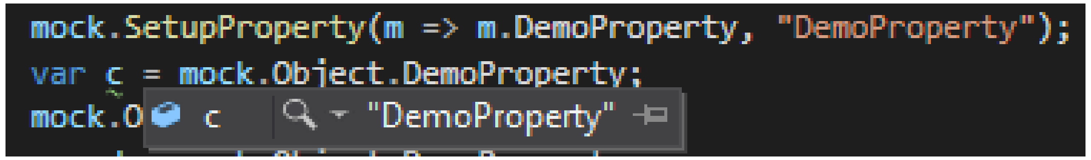
#somoshiberus



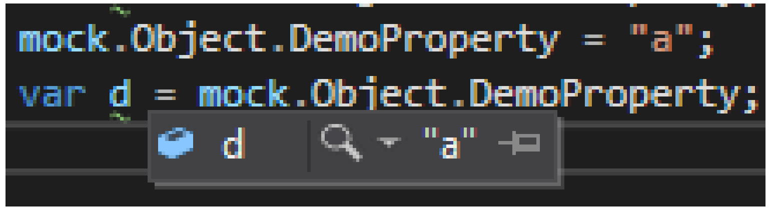
```
public interface IFoo
{
    2 referencias | 1/1 pasando
    int GetCount();
    13 referencias | 1/3 pasando
    string ToUpperCase(string v);
    8 referencias | 0/1 pasando
    int GetNextNumber();
    0 referencias
    string DemoProperty { get; set; }
}
```



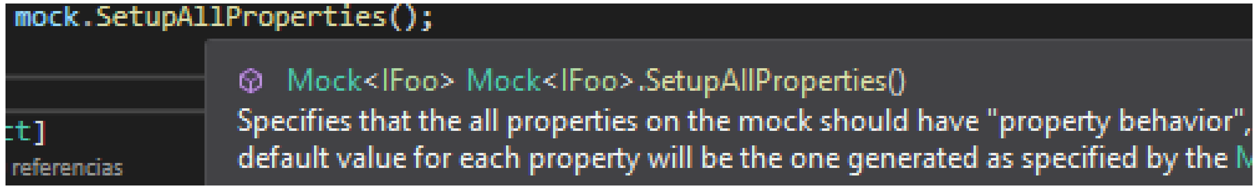
```
mock.SetupProperty(m => m.DemoProperty, "DemoProperty");
var c = mock.Object.DemoProperty;
mock.Object.DemoProperty = "a";
var d = mock.Object.DemoProperty;
```



```
mock.SetupProperty(m => m.DemoProperty, "DemoProperty");
var c = mock.Object.DemoProperty;
mock.O  c  "DemoProperty"
```



```
mock.Object.DemoProperty = "a";
var d = mock.Object.DemoProperty;
  d  "a"
```



```
mock.SetupAllProperties();
```

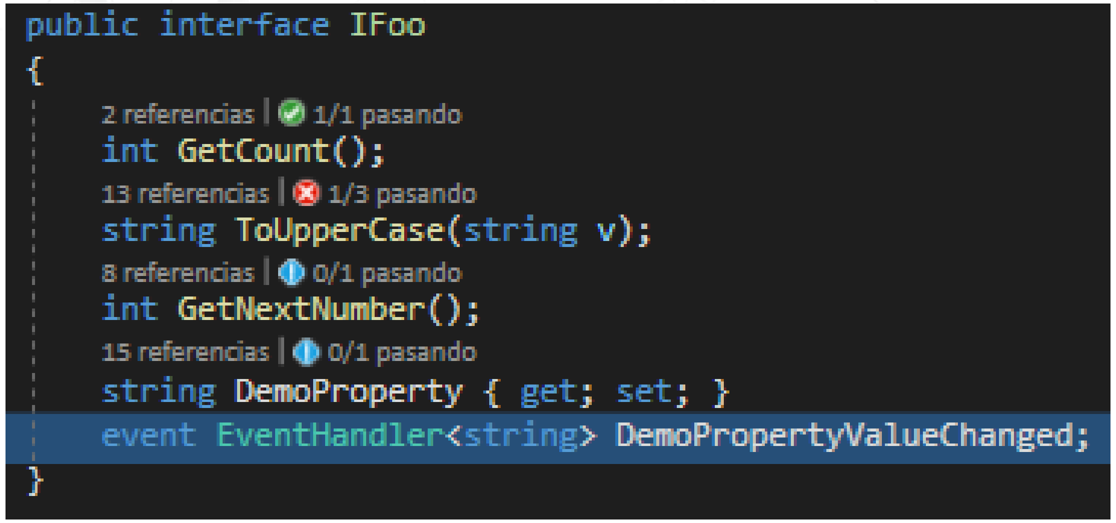
Mock<IFoo> Mock<IFoo>.SetupAllProperties()
Specifies that the all properties on the mock should have "property behavior",
default value for each property will be the one generated as specified by the

Nafarroako Gobernua Gobierno de Navarra

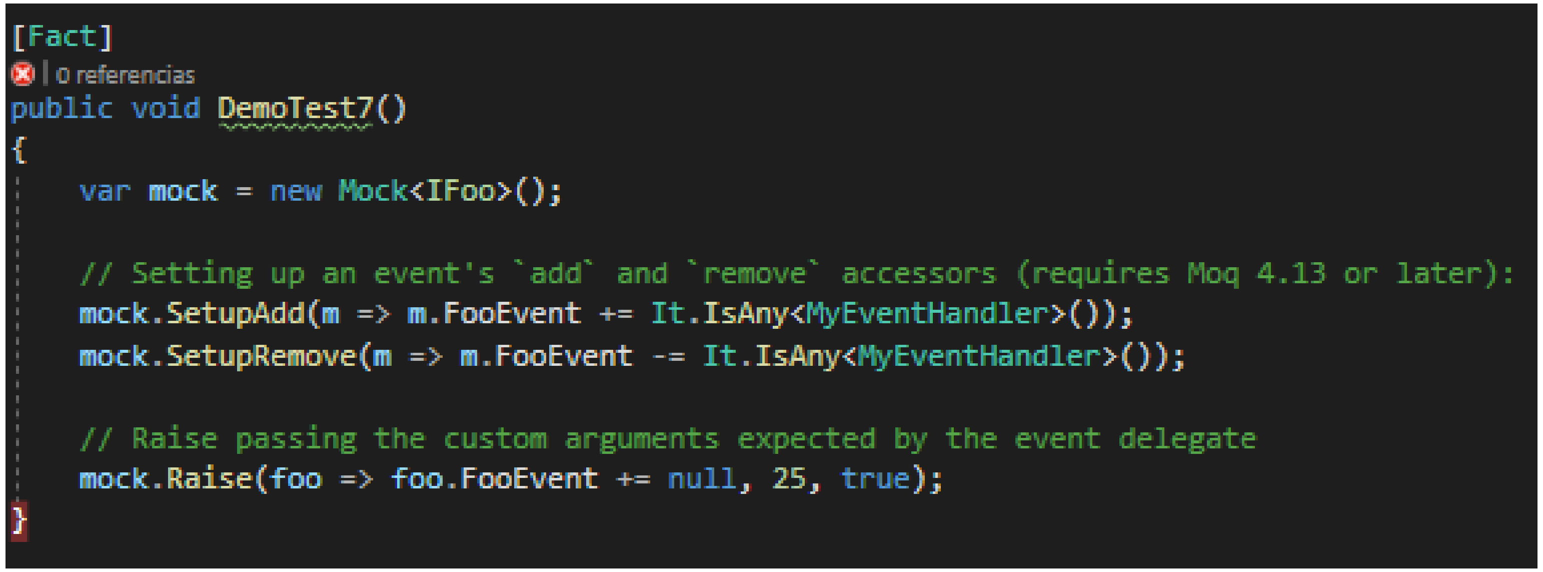
Moq

Events

#somoshiberus



```
public interface IFoo
{
    2 referencias | 1/1 pasando
    int GetCount();
    13 referencias | 1/3 pasando
    string ToUpperCase(string v);
    8 referencias | 0/1 pasando
    int GetNextNumber();
    15 referencias | 0/1 pasando
    string DemoProperty { get; set; }
    event EventHandler<string> DemoPropertyValueChanged;
}
```

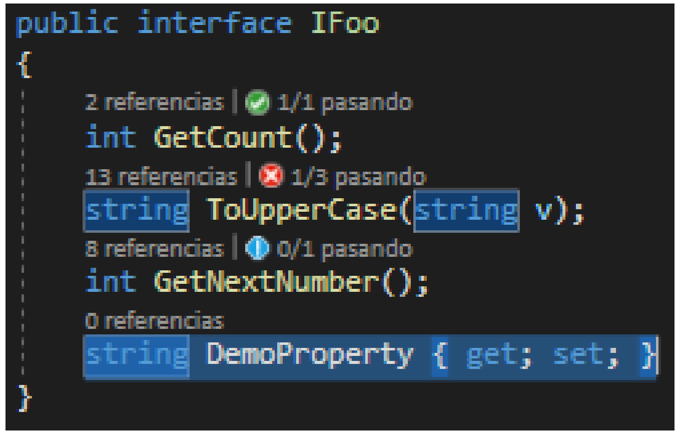


```
[Fact]
| 0 referencias
public void DemoTest7()
{
    var mock = new Mock<IFoo>();

    // Setting up an event's `add` and `remove` accessors (requires Moq 4.13 or later):
    mock.SetupAdd(m => m.FooEvent += It.IsAny<MyEventHandler>());
    mock.SetupRemove(m => m.FooEvent -= It.IsAny<MyEventHandler>());

    // Raise passing the custom arguments expected by the event delegate
    mock.Raise(foo => foo.FooEvent += null, 25, true);
}
```

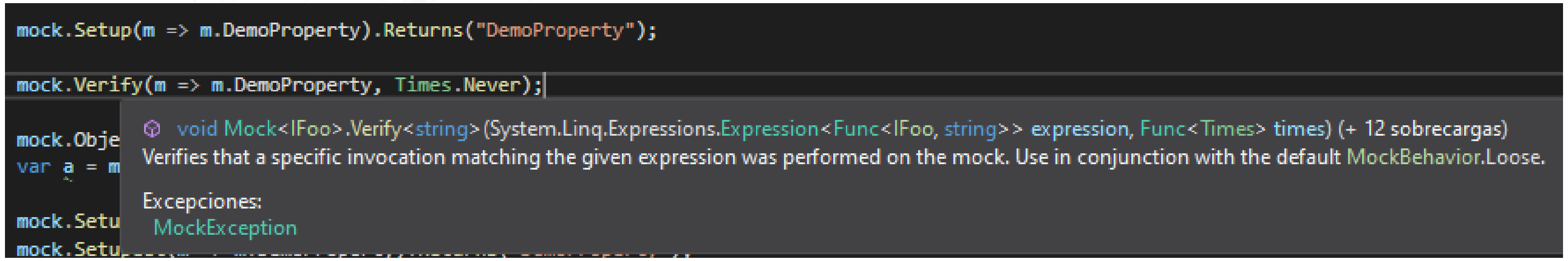
Moq

Verify

#somoshiberus



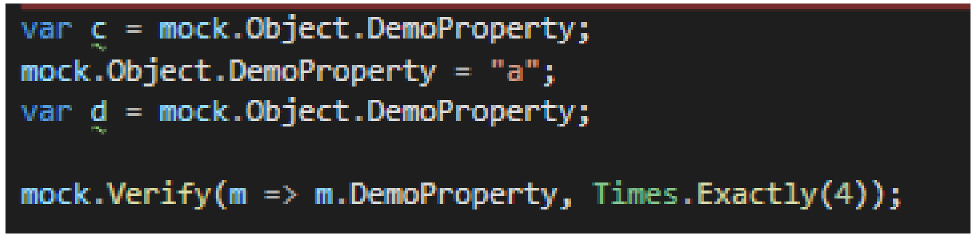
```
public interface IFoo
{
    2 referencias | 1/1 pasando
    int GetCount();
    13 referencias | 1/3 pasando
    string ToUpperCase(string v);
    8 referencias | 0/1 pasando
    int GetNextNumber();
    0 referencias
    string DemoProperty { get; set; }
}
```



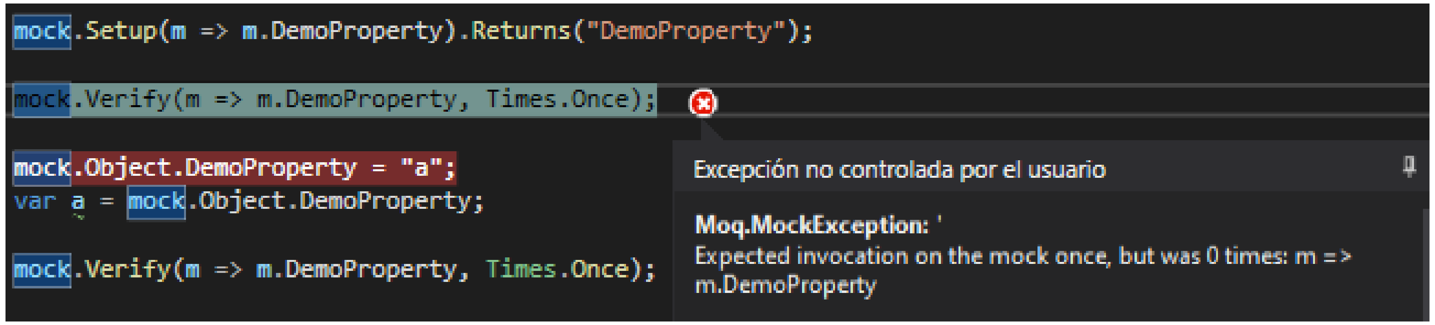
```
mock.Setup(m => m.DemoProperty).Returns("DemoProperty");

mock.Verify(m => m.DemoProperty, Times.Never);

mock.Obje
var a = m

mock.Setu
mock.Setu
```

void Mock<IFoo>.Verify<string>(System.Linq.Expressions.Expression<Func<IFoo, string>> expression, Func<Times> times) (+ 12 sobrecargas)
Verifies that a specific invocation matching the given expression was performed on the mock. Use in conjunction with the default MockBehavior.Loose.

Excepciones:
MockException



```
var c = mock.Object.DemoProperty;
mock.Object.DemoProperty = "a";
var d = mock.Object.DemoProperty;

mock.Verify(m => m.DemoProperty, Times.Exactly(4));
```



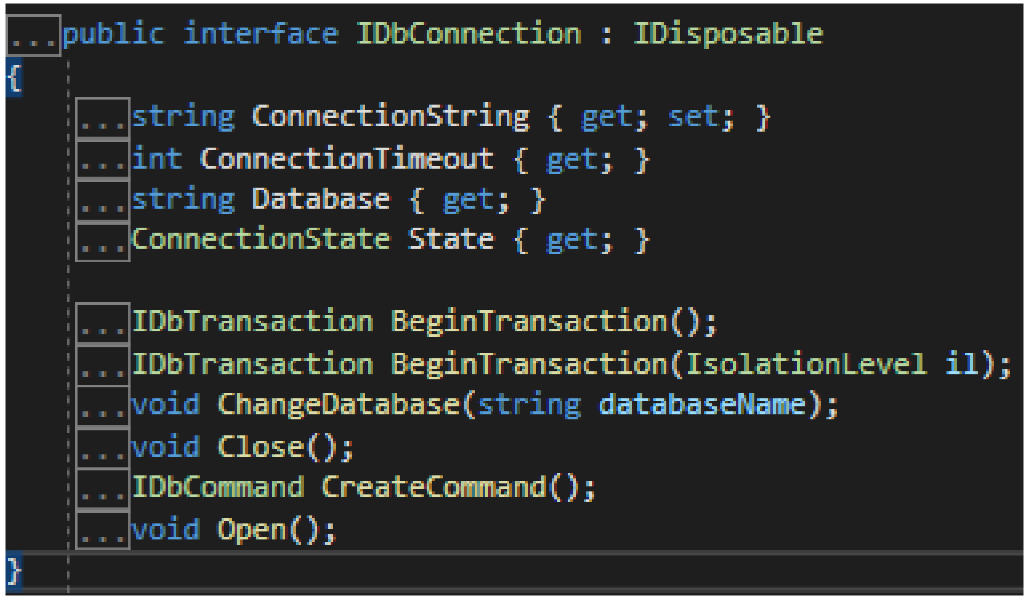
```
mock.Setup(m => m.DemoProperty).Returns("DemoProperty");

mock.Verify(m => m.DemoProperty, Times.Once);

mock.Object.DemoProperty = "a";
var a = mock.Object.DemoProperty;

mock.Verify(m => m.DemoProperty, Times.Once);
```

Excepción no controlada por el usuario

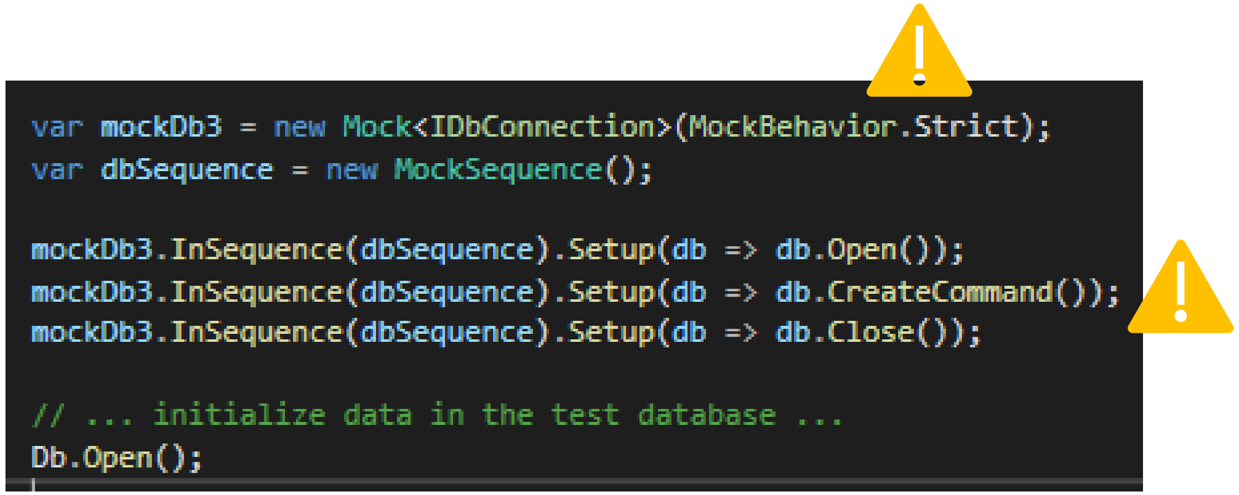
Moq.MockException: '
Expected invocation on the mock once, but was 0 times: m => m.DemoProperty

Moq

MockSequence



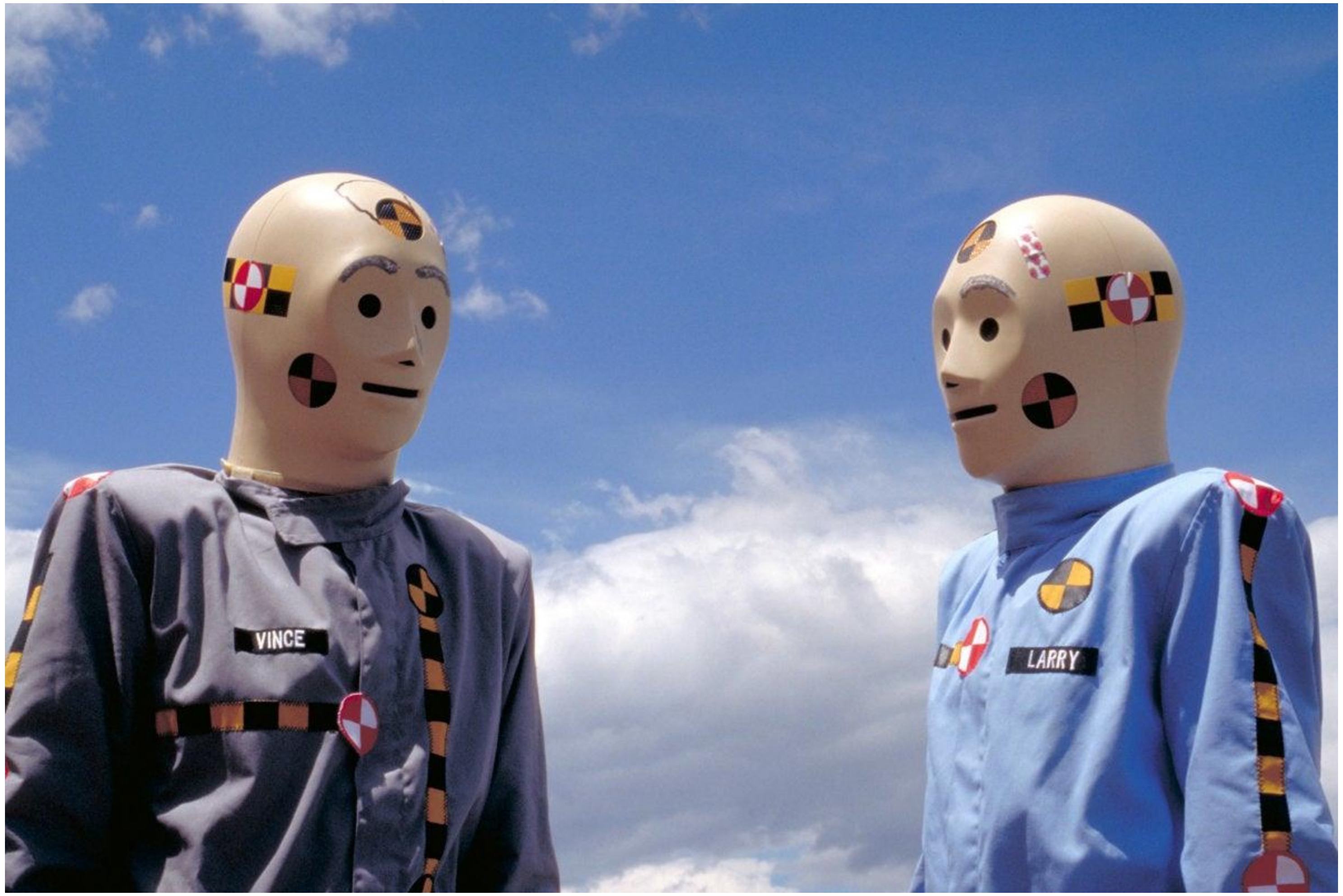
```
public interface IDbConnection : IDisposable
{
    string ConnectionString { get; set; }
    int ConnectionTimeout { get; }
    string Database { get; }
    ConnectionState State { get; }

    IDbTransaction BeginTransaction();
    IDbTransaction BeginTransaction(IsolationLevel il);
    void ChangeDatabase(string databaseName);
    void Close();
    IDbCommand CreateCommand();
    void Open();
}
```



```
var mockDb3 = new Mock<IDbConnection>(MockBehavior.Strict);
var dbSequence = new MockSequence();

mockDb3.InSequence(dbSequence).Setup(db => db.Open());
mockDb3.InSequence(dbSequence).Setup(db => db.CreateCommand());
mockDb3.InSequence(dbSequence).Setup(db => db.Close());

// ... initialize data in the test database ...
Db.Open();
```

Moq

Only Moq is not OK



hiberus© TECNOLOGIA

La compañía hiperespecializada en las TIC

www.hiberus.com