Design Pattern & Clean Code

Testable Code

Thomas Ley | @CleanCodeCoach

Goals

- Pattern for "testable code"
- Separate "untestable code".

Unit tests

- Cost time and money
- Changes when code changed
- No value for the customer
- Learn an additional concept.

Unit tests

- Not only check input --> output
- They define what happens in error-execution path
  - Logging
  - Return values
  - Rethrow exception?
- The document the usage of a class
  - Parameter usage
  - Return values
  - Expected behaviour

Unit tests

- Double check code
- Detect unwanted changes
- Keeps your code clean.

About testing

- Myth: One assert per test
- Truth: One path per test.

About testing

- Myth: Each method has a test method
- Truth: Each method will have multiple test methods
  - One "happy path"
  - Multiple "error path".

Testing

- Brings all pieces together
- Clean code to test [SRP] classes
- [DIP] to mock dependencies.

[Static Class Wrapper]

- Create Wrapper
- Extract interface

☝ Used for Static classes
☝ Used for classes without interface

Mock HttpClient

- Create HttpClientWrapper
- Extract interface IHttpClient
- Create HttpClientMock
- Use interface and inject implementation

Demo

- Project

Random(), DateTime()

- Same pattern applies to "changing" data
- Predict random numbers
- Change date time during test (e.g. test cache expiration)

"Mocking" Frameworks

- NSubstitute
- System.Io.Abstractions
- ContextFor<T>

Demo

- Project

Architectural Tests

- Unit tests to verify architectural decisions
- E.g. Each class must have a test
- Operator methods must be virtual

Have a break...

... have a workout

7 Minute Workout or Bring Sally Up