Architecting an ASP.NET Core MVC Application for Unit Testability

WHAT SHOULD I TEST?

Benjamin Day
TRAINER | COACH | DEVELOPER

@benday www.benday.com

Overview

Sketch out the vision for the course

Why unit testing?

What should you test?

How do I think about testing?

Unit testing demos

A little bit of background

I’ve done 9 courses for Pluralsight…

…but not much content about unit testing.

Why?

I kept running out of time.

The Lost Modules

I am <u>passionate</u> about software testing.

I help teams and companies streamline their software development and delivery.

Agile, Scrum, DevOps, TFS, VSTS, etc.

Therapist for Teams

Successful, happy, productive teams almost always are good at automated testing.

High-performing Teams Write Unit Tests

If you know your stuff is working, it's a lot easier to deliver software.

If you know your stuff is NOT working, it's a lot easier to NOT deliver software.

Easy Decision vs. Tough Decision

If you know your stuff is working, it’s a lot easier to deliver software.

? ?

Is it working?

Should we ship it?

? ?

If you know your stuff is NOT working, it’s a lot easier to NOT deliver software.

"Blah blah blah. Whatever. That's why we have QA testers."

QA Testers

Human-based testing

Run test cases

Bang on the app and see if it breaks

Not very efficient

Humans are slow

I’m not saying fire your QA testers.

I ♥ QA testers.

I want you to think about quality from the beginning.

Don’t make “quality” wait until the end.

Bad QA vs. Good QA

Write code

It pretty much works…maybe

“Kick it over the wall to QA”

They send back bugs

(repeat)

Write your code with tests

99% sure it works

QA verifies it works

QA focuses on Exploratory Testing

Is it a good use of their time to run a bunch of boring tests that a computer could run instead?

No.

Wise, Strategic Use of QA Testers: Focus on Exploratory Testing

How do you let QA do more exploratory testing?

Write unit tests.

Unit Tests

Little chunks of test code that exercise and validate little chunks of application code

Let your unit tests tell your code is broken

Automated

- Run hundreds of tests in seconds
- Run these tests hundreds of times per day
- Great for DevOps, CI/CD

Give much higher quality stuff to QA testers

More time for exploratory testing

Next up:
What Should You Test?

What should you test?

Assumption:
You’re writing an app with ASP.NET Core

Demos will be in ASP.NET Core

This is a software architecture course.

The concepts apply to just about anything.

What should you test?

What can you test?

What can you test easily?

What will help you decide if the app is working?

A Lot of Things to Test

ASP.NET

- Controllers & Views
- Security Logic
- Routing
- Configuration

Adapter Logic / Boundary Logic

- ASP.NET ViewModels to/from Domain Models
- Entity Framework Entities to/from Domain Models

Validation & Calculation Logic

Service Layer / Use Case Logic

I want to test these pieces in isolation.

I want my tests to be as focused and small as possible.

Dependencies make things hard to test.

Unit Test vs. Integration Test

Test a chunk of code
in isolation →
Unit Test

No dependencies

Deploy the app to test a
chunk of code →
Integration Test

Dependencies

I want to write mostly or only unit tests.

Dependencies are the enemies of unit tests.

Architect my system that allows me to write unit tests rather than integration tests

I want to architect my system to manage dependencies so that I can focus on unit tests.

My Goal:
Design for Testability

Just Writing Code vs. Design for Testability

Just Writing Code	Design for Testability
Writing code without tests	Writing code with tests
Design decision	Design decision
Choose whatever is easiest to implement	Choose the option that is the most testable
Nothing wrong with this	Probably takes more effort
Design for speed of initial delivery	Higher quality application
	More maintainable code
12 months later you want to add tests → hard to add tests	Much easier to refactor

Does it take more time to write code with unit tests?

Yes, it probably does…

…but it depends on what you measure

- Time to write the code?
- Time to deliver the application?

Ensure quality early →

- Fewer bug fixes at the end
- Less maintenance headache

Slower to code

Must faster to validate, deliver, and maintain

Next up:
The Architecture of Design for Testability

The Architecture of Design for Testability: An Overview

Disclaimer:
I’m going to mention a bunch of stuff fast.

If you don't know what I'm talking about...

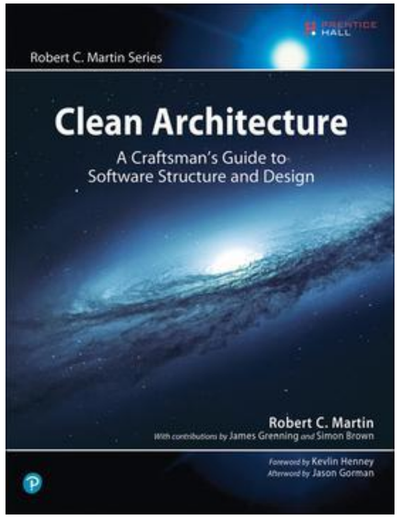
…keep watching the course.

My Big Software Architecture Influences

Robert C. Martin

Martin Fowler

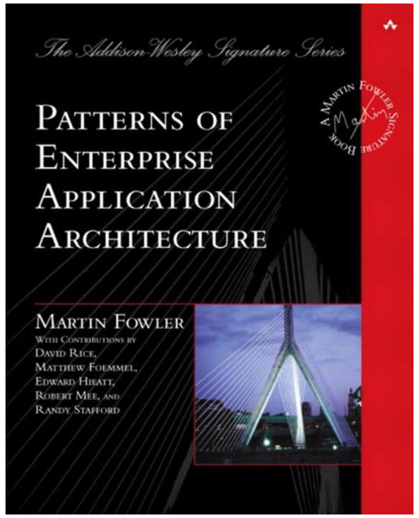
Two Great Books



“Clean Architecture”

by Robert C. Martin

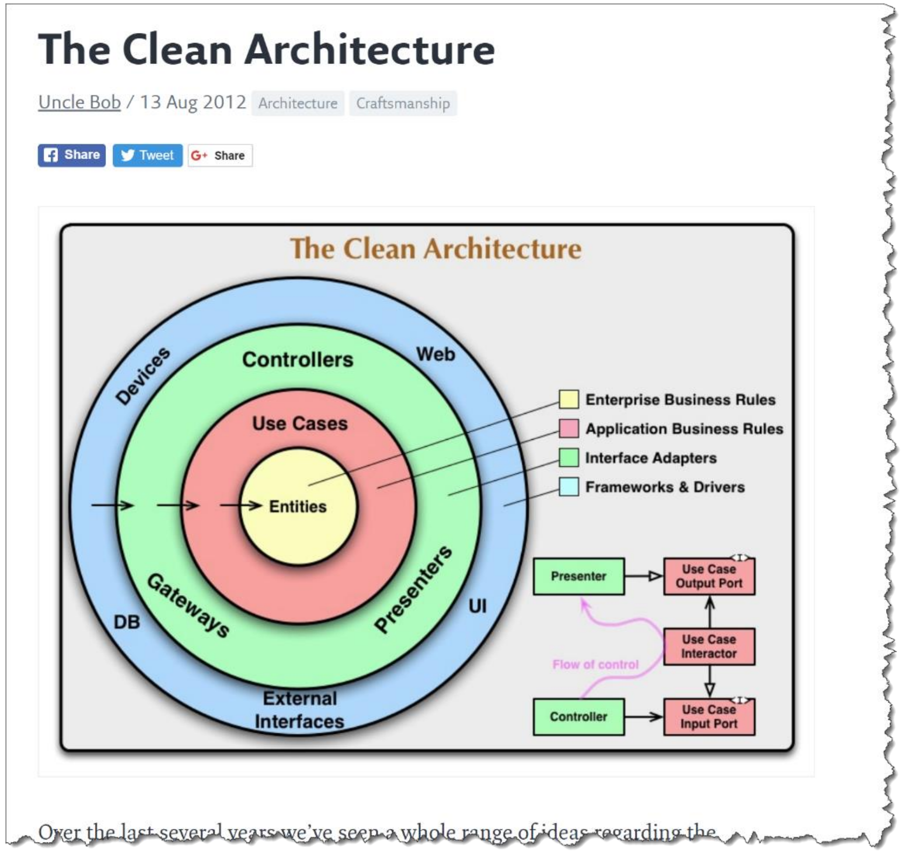
http://a.co/eVXxP9x



“Patterns of Enterprise Application Architecture”

by Martin Fowler et al.

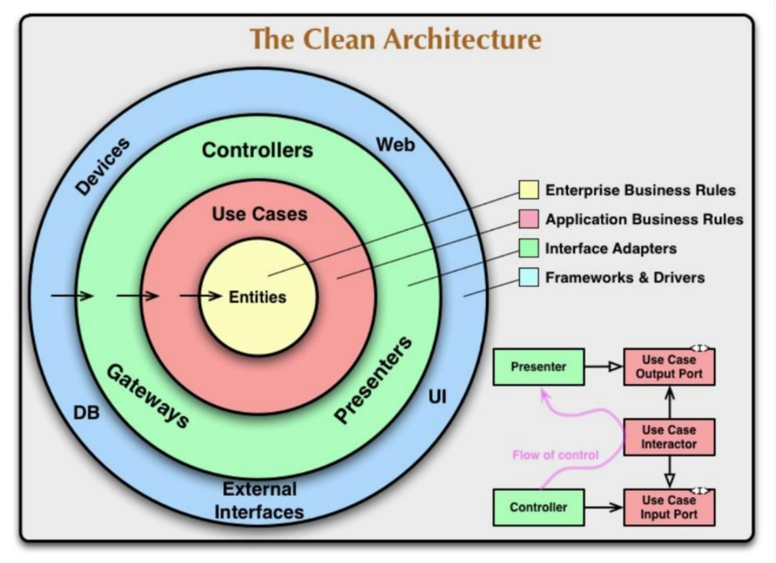
http://a.co/3MzPQ7O



The Clean Architecture

Uncle Bob / 13 Aug 2012 Architecture Craftsmanship

Share Tweet Share



Over the last several years we've seen a whole range of ideas regarding the

Clean Architecture

Robert C. Martin

https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html

It's in the book

Way of layering and organizing code

Use abstractions to manage dependencies

SOLID Principles

Robert C. Martin

https://en.wikipedia.org/wiki/SOLID

Single Responsibility Principle (SRP)

Open / Closed Principle (OCP)

Liskov Substitution Principle (LSP)

Interface Segregation Principle (ISP)

Dependency Inversion Principle (DIP)

Single Responsibility &
Dependency Inversion
are essential for testability.

Single Responsibility Principle

An class should have one and only one reason to change

- A class should only do one thing

Keep things clean and organized

This keeps your unit tests targeted and clean and organized

Dependency Inversion Principle

Dependencies between chunks of code should always be modeled as abstractions

C# interfaces rather than concrete types

If an class depends on an interface…

- Always construct instances of the concrete type outside of that class
- Pass the dependency in on the constructor

I’m also going to talk about design patterns.

Design Patterns: Don't Re-invent the Wheel and Have It Come out Square

Design Patterns

Model-View-Controller (MVC)

- Abstraction of a user interface

Repository

- Abstraction of persistence logic
- Data Access

Adapter

- Turn one kind of object into another
- Helps you move between layers

Strategy

- Abstraction of algorithms
- Validations and calculations

Next up:
Demos

Unit Test Demos

Handful of Stuff

“System Under Test”

- Application code that I’m testing
- “sut”

Guideline: At least one unit test per public method in the SUT

Microsoft Test Framework

- MSTest V2
- https://github.com/Microsoft/testfx

Please don’t freak out about unit test framework choices.

Demo

Part 1 of 2

Calculator

Add, Subtract, Multiply, Divide

Code Coverage

Code Coverage

How good are your tests?

What have you tested?

What have you missed?

Code Coverage with MSTest V2 and .NET Core

Works in Visual Studio 2017

- (Doesn't seem to work using dotnet command line)

You need to edit your project file

- *.csproj

Change the type of *.pdb file that's generated

Supposedly this gets fixed in Q3 2018

- https://github.com/Microsoft/vstest-docs/blob/master/RFCs/0021-CodeCoverageForNetCore.md

```
<PropertyGroup>
    <TargetFramework>
        netcoreapp2.1
    </TargetFramework>

    <IsPackable>
        false
    </IsPackable>

    <DebugType>
        Full
    </DebugType>
</PropertyGroup>
```

- Edit your *.csproj file
- Add DebugType property
- Set it to Full

Demo

Part 2 of 2

Calculator

Edit *.csproj file

Code Coverage

Summary

Sketched out the vision for the course

Why unit testing?

What should I test?

How do I think about testing?

Unit testing demos

Next up:
Testing ASP.NET