

# Module 07:

## "Architecture Tests"



# Agenda

- ▶ Introduction
- ▶ **Architecture Tests**
- ▶ ArchUnitNET
- ▶ Summary



# Why Test Architecture?

- ▶ Architectures rules and best practices are enforced by conventions and code reviews
- ▶ Tend to "fade" over time with
  - in/out-flux of developers
  - Project complexity
- ▶ Expresses and maintains the design decisions in code
- ▶ Self-testing Architecture
  - Can be checked automatically in pipelines etc.



# Architecture Tests

- ▶ Checks could include:
  - Classes and interfaces are in correct namespace and projects given their type
  - Dependency rules are adhered to
  - Naming conventions are obeyed
  - Visibility and access modifier constraints are satisfied
  - No cross-version references for endpoints
  
- ▶ Note: This is not the same as linting..! 😊



# Agenda

- ▶ Introduction
- ▶ Architecture Tests
- ▶ **ArchUnitNET**
- ▶ Summary



# TngTech.ArchUnitNET

- ▶ Freely available nuget package
  - <https://archunitnet.readthedocs.io/en/latest/>
- ▶ **ArchUnitNET** is a .NET library for building assertions of types and structure using a Fluent API syntax
  - Works with any unit testing framework through additional packages
- ▶ Often combined with **FluentAssertions**
  - But not directly related
- ▶ Alternatives do exist, e.g.
  - **NetArchTest.Rules** available at <https://github.com/BenMorris/NetArchTest>



# ArchUnitNet Basics

- ▶ Set up rules in a number of steps
  1. Define the **Architecture**
    - Assemblies to be tested etc.
  2. Define collections of layers and types
  3. Select types and filter with predicates and combine with **And()** or **Or()**
  4. Apply conditions using **Should()** and **NotXXX()**
  5. Finally, assert with **Check()**



# Summary

- ▶ Introduction
- ▶ Architecture Tests
- ▶ ArchUnitNET





