Working Effectively with Legacy Code
Legacy Code

OCHUTNÁVKA
Legacy Code?

“Code without tests.” [Michael Feathers]

“Source code inherited from someone else.”

“Source code inherited from an older version of the software.”

„Veškerý kód, s kterým aktuálně nejste spokojeni (nebo byste alespoň neměli být).“

[Robert Haken]
Legacy Code

Non-uniform coding style
Nesrozumitelný
Málo/bez testů
Bad Design
Code Smell
...

Refactoring Mindset

LEGACY:
„Do not touch working code, unless needed.“

NOW:
„Leave the code in better condition than you found it.“ [The Boy Scout Rule]

Planned refactoring

Refactoring as you go.
Předpoklady

Sdílené vlastnictví kódu
Source Code Management
Continuous Integration builds + runs Tests
Refactoring Justification

Quality
Clean Code
Professionalism
Right Thing

Economics
Roslyn CodeAnalysis + baseline

DEMO
Code Analyzers

Microsoft.AnalyzerPowerPack (Roslyn Team)
System.Runtime.[CSharp.]Analyzers
StyleCop Analyzers
SonarLint [SonarQube]

VS Perf-tip:
Tools / Options / Text Editor / C# / Advanced / Enable solution wide analysis = OFF
Refactoring

Continuous, As you go
Small steps
IDE/Tooling supported safe steps
Comprehension Refactoring (Rename, Extract, ...)

Podpořeno testy

- Pomáhají porozumět kódu
- Guard Conditions, Contract.Requires
- Debug.Assert, InvalidOperationException, ...
- Unit-Tests
- Integration Tests
"Good" Unit Test

- automated + repeatable
- fully isolated
- consistent in its results
- runs quickly
- full control of the unit under test (all dependencies)
- relevant tomorrow
- easy to implement
- able to run it at the push of a button
- if fails => easy to detect what was expected
Poor Man’s Testability
Extract dependant call to virtual method

DEMO
"Good" Unit Test

- automated + repeatable
- fully isolated
- consistent in its results
- runs quickly
- full control of the unit under test (all dependencies)
- relevant tomorrow
- easy to implement
- able to run it at the push of a button
- if fails => easy to detect what was expected
Mocking

DEMO
"Good" Unit Test

- automated + repeatable
- fully isolated
- consistent in its results
- runs quickly
- full control of the unit under test (all dependencies)
- relevant tomorrow
- easy to implement
- able to run it at the push of a button
- if fails => easy to detect what was expected
Dependency Injection

DEMO
Tips & Tricks

[assembly::InternalsVisibleTo(MyTestAssembly)]
[Obsolete]
Treat Warnings as Errors
Ambient Context (ale ne ServiceLocator nebo public container!)
Service Factories
Analyze / Analyze Solution for Code Clones
Test / Analyze Code Coverage
Q & A