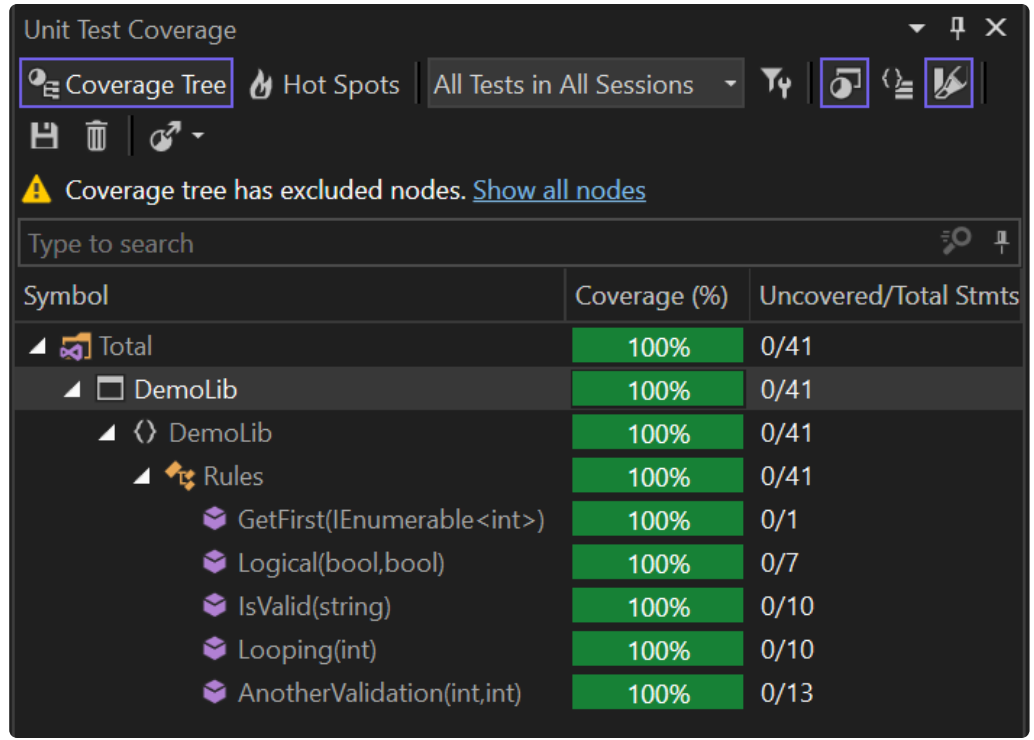
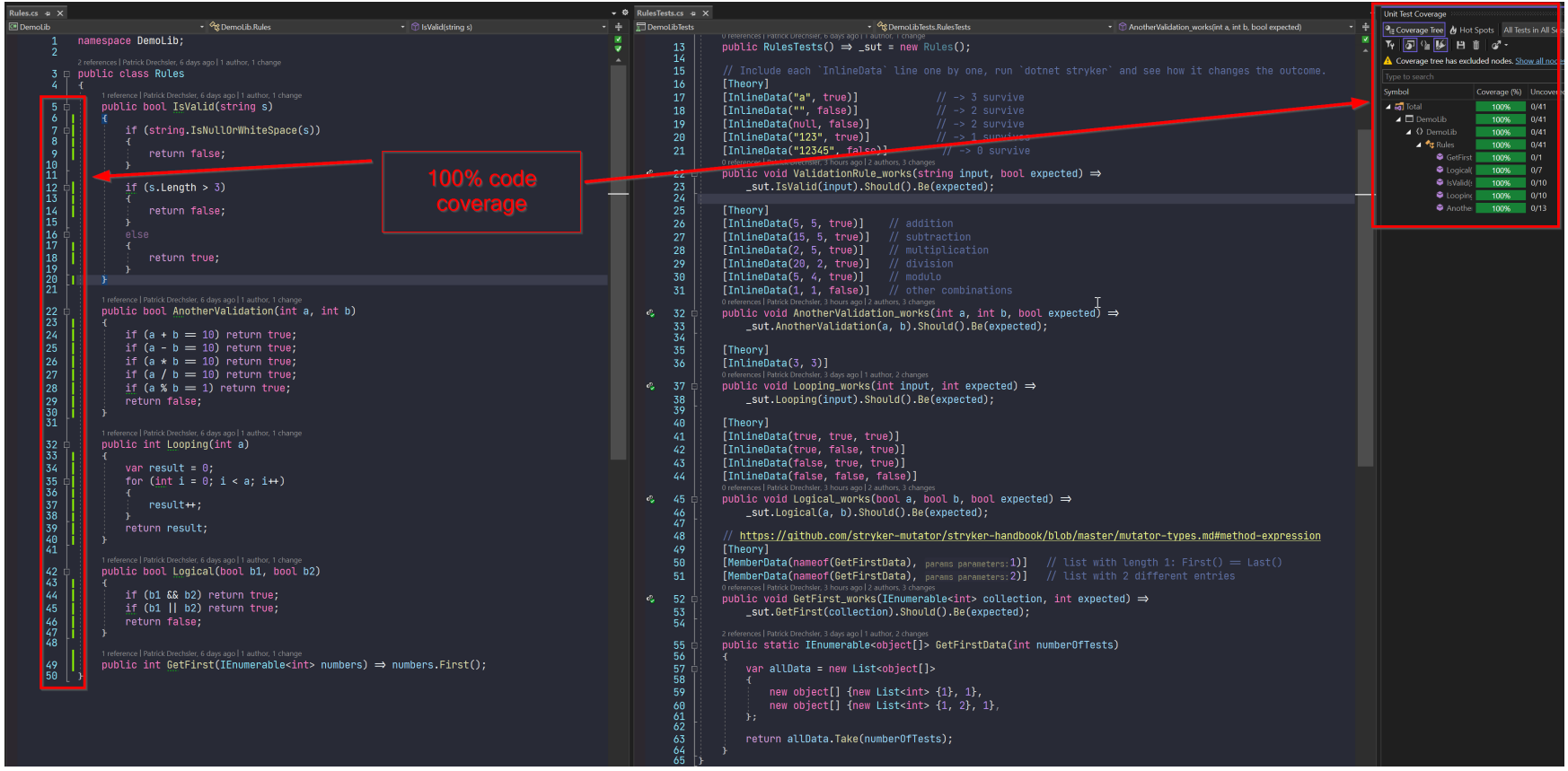
Mutation Testing

PATRICK DRECHSLER

Code Coverage 100%

what more do you want?




Rules.cs
DemoLib
DemoLib.Rules
IsValid(string s)
namespace DemoLib;
public class Rules
{
public bool IsValid(string s)
{
if (string.IsNullOrWhiteSpace(s))
{
return false;
}
if (s.Length > 3)
{
return false;
}
else
{
return true;
}
}
public bool AnotherValidation(int a, int b)
{
if (a + b == 10) return true;
if (a - b == 10) return true;
if (a * b == 10) return true;
if (a / b == 10) return true;
if (a % b == 1) return true;
return false;
}
public int Looping(int a)
{
var result = 0;
for (int i = 0; i < a; i++)
{
result++;
}
return result;
}
public bool Logical(bool b1, bool b2)
{
if (b1 && b2) return true;
if (b1 || b2) return true;
return false;
}
public int GetFirst(IEnumerable<int> numbers) => numbers.First();
100% code coverage
RulesTests.cs
DemoLibTests
DemoLibTests.RulesTests
AnotherValidation_works(int a, int b, bool expected)
public RulesTests() => _sut = new Rules();
// Include each `InlineData` line one by one, run `dotnet stryker` and see how it changes the outcome.
[Theory]
[InlineData("a", true)] // -> 3 survive
[InlineData("", false)] // -> 2 survive
[InlineData(null, false)] // -> 2 survive
[InlineData("123", true)] // -> 1 survive
[InlineData("12345", false)] // -> 0 survive
public void ValidationRule_works(string input, bool expected) =>
_sut.IsValid(input).Should().Be(expected);
[Theory]
[InlineData(5, 5, true)] // addition
[InlineData(15, 5, true)] // subtraction
[InlineData(2, 5, true)] // multiplication
[InlineData(20, 2, true)] // division
[InlineData(5, 4, true)] // modulo
[InlineData(1, 1, false)] // other combinations
public void AnotherValidation_works(int a, int b, bool expected) =>
_sut.AnotherValidation(a, b).Should().Be(expected);
[Theory]
[InlineData(3, 3)]
public void Looping_works(int input, int expected) =>
_sut.Looping(input).Should().Be(expected);
[Theory]
[InlineData(true, true, true)]
[InlineData(true, false, true)]
[InlineData(false, true, true)]
[InlineData(false, false, false)]
public void Logical_works(bool a, bool b, bool expected) =>
_sut.Logical(a, b).Should().Be(expected);
// https://github.com/stryker-mutator/stryker-handbook/blob/master/mutator-types.md#method-expression
[Theory]
[MemberData(nameof(GetFirstData), parameters: 1)] // list with length 1: First() == Last()
[MemberData(nameof(GetFirstData), parameters: 2)] // list with 2 different entries
public void GetFirst_works(IEnumerable<int> collection, int expected) =>
_sut.GetFirst(collection).Should().Be(expected);
public static IEnumerable<object[]> GetFirstData(int numberOfTests)
{
var allData = new List<object[]>
{
new object[] {new List<int> {1}, 1},
new object[] {new List<int> {1, 2}, 1},
};
return allData.Take(numberOfTests);
}
Unit Test Coverage
Coverage Tree
Hot Spots
All Tests in All Sc
Coverage tree has excluded nodes. Show all nod
Type to search
Symbol
Coverage (%)
Uncover
total 100% 0/41
DemoLib 100% 0/41
DemoLib 100% 0/41
Rules 100% 0/41
GetFirst 100% 0/1
Logical(100% 0/7
IsValid(100% 0/10
Loopin(100% 0/10
Anothe 100% 0/13

Example

```
public class Rules
{
  public bool IsValid(string s)
  {
    if (string.IsNullOrWhiteSpace(s))
    {
      return false;
    }

    if (s.Length > 3)
    {
      return false;
    }

    return true;
  }
}
```

```
public class RulesTests
{
  private readonly Rules _sut;

  public RulesTests() => _sut = new Rules();

  [Theory]
  [InlineData("", false)]
  [InlineData("a", true)]
  [InlineData("12345", true)]
  public void Validation_works(
    string input, bool expected) =>
    _sut.IsValid(input).Should().Be(expected);
}
```

- 100% coverage...
- but, are we covering all corner cases?

Let’s create some mutants!

Let’s change

```
if (s.Length > 3)
```

to

```
if (s.Length < 3)  // <- this is a "MUTANT"

if (s.Length >= 3) // <- this is another "MUTANT"

if (s.Length <= 3) // <- ...and another "MUTANT"
```

Do we still have the same code coverage?

Concept

- Production code is modified (by the mutation testing framework)
- Test suite is run

Did any mutants survive?

- If all mutants die, the test suite is fine 👌
- But if some mutants survive, the tests are not covering all cases 👀
  - 👉 take a closer look

Many mutation frameworks generate an interactive html report

Example mutations

https://stryker-mutator.io/docs/stryker-net/mutations/

Mutations

Stryker supports a variety of mutators, which are listed below. In parentheses the names of correspondent mutations are specified, which you might need for the `exclude-mutations` section of the configuration.

Do you have a suggestion for a (new) mutator? Feel free to create an issue!

Arithmetic Operators (*arithmetic*)

Live Demo: Mutation Testing in C#

Other languages

Frameworks are available for many languages:

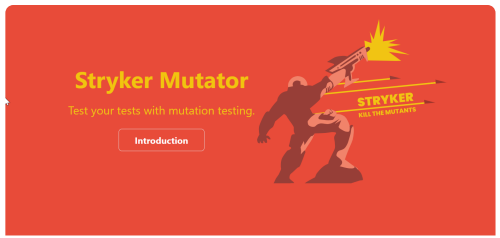
- Java (using PIT)
- Scala (using Stryker4s)
- C# (using Stryker.NET)
- Javascript/Typescript (using StrykerJS)
- Python (using Cosmic Ray or mutmut)
- Haskell (using MuCheck)
- ...

: Example project in the repo

pitest.org

Real world mutation testing

PIT is a state of the art mutation testing system, providing gold standard test coverage for Java and the jvm. It's fast, scalable and integrates with modern test and build tooling.



Getting started with Stryker

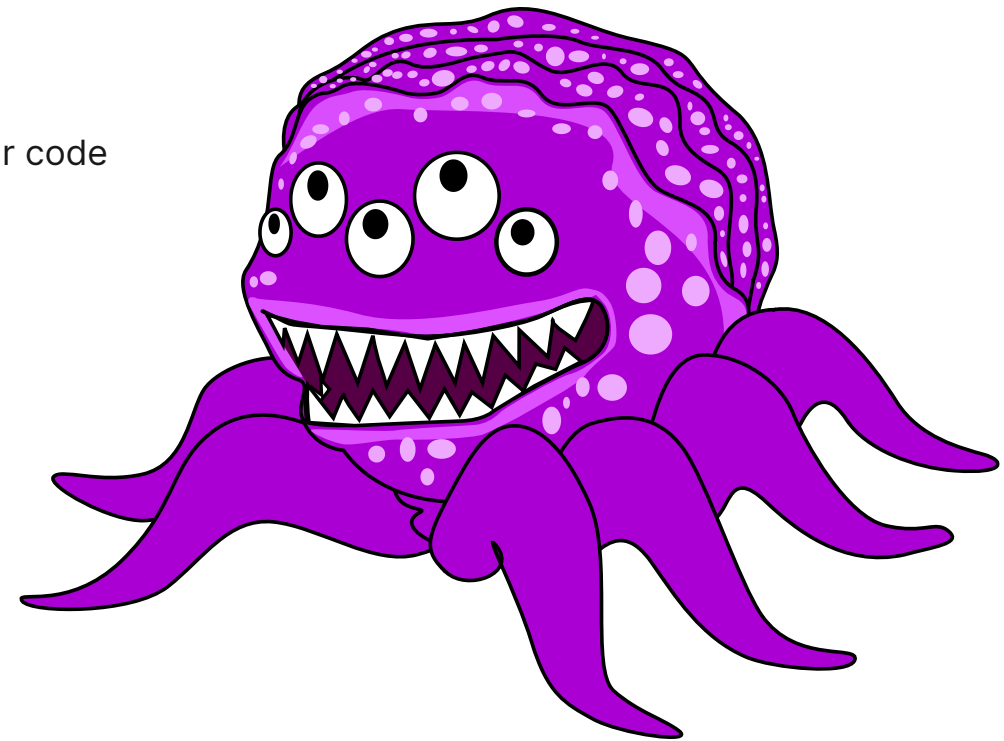
mutmut - python mutation tester

Hackage :: [Package]

MuCheck: Automated Mutation Testing

Mutation Testing: Summary

- requires decent test coverage
- modifies production code to find corner cases
- requires a lot of resources: use deliberately!
- Don’t include it in your CI/CD pipeline
- Use it as an exploratory tool to find bugs in your code
- Use it to find critical bugs in your code



The End

Any Questions?