

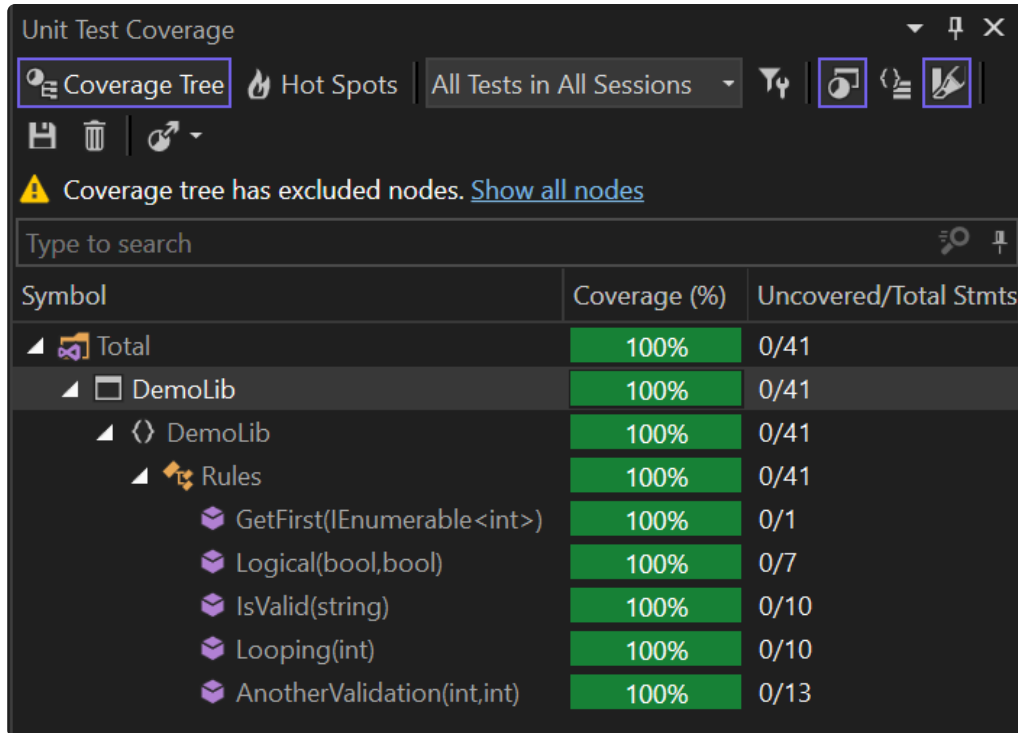
Mutation Testing

PATRICK DRECHSLER



Code Coverage 100%

what more do you want?



The screenshot shows the 'Unit Test Coverage' window. At the top, there are tabs for 'Coverage Tree' (selected), 'Hot Spots', and a dropdown menu set to 'All Tests in All Sessions'. Below the tabs are icons for saving, deleting, and refreshing. A warning message states: 'Coverage tree has excluded nodes. [Show all nodes](#)'. Below this is a search bar with the placeholder text 'Type to search'. The main area displays a table with three columns: 'Symbol', 'Coverage (%)', and 'Uncovered/Total Stmts'.

Symbol	Coverage (%)	Uncovered/Total Stmts
▲ Total	100%	0/41
▲ DemoLib	100%	0/41
▲ DemoLib	100%	0/41
▲ Rules	100%	0/41
GetFirst(IEnumerable<int>)	100%	0/1
Logical(bool,bool)	100%	0/7
IsValid(string)	100%	0/10
Looping(int)	100%	0/10
AnotherValidation(int,int)	100%	0/13

Rules.cs - Demolib

```
1 namespace Demolib;
2
3 public class Rules
4 {
5     1 reference | Patrick Drechslar, 6 days ago | 1 author, 1 change
6     public bool IsValid(string s)
7     {
8         if (string.IsNullOrEmpty(s))
9         {
10             return false;
11         }
12         if (s.Length > 3)
13         {
14             return false;
15         }
16         else
17         {
18             return true;
19         }
20     }
21
22     1 reference | Patrick Drechslar, 6 days ago | 1 author, 1 change
23     public bool AnotherValidation(int a, int b)
24     {
25         if (a + b == 10) return true;
26         if (a - b == 10) return true;
27         if (a * b == 10) return true;
28         if (a / b == 10) return true;
29         if (a % b == 1) return true;
30         return false;
31     }
32
33     1 reference | Patrick Drechslar, 6 days ago | 1 author, 1 change
34     public int Looping(int a)
35     {
36         var result = 0;
37         for (int i = 0; i < a; i++)
38         {
39             result++;
40         }
41         return result;
42     }
43
44     1 reference | Patrick Drechslar, 6 days ago | 1 author, 1 change
45     public bool Logical(bool b1, bool b2)
46     {
47         if (b1 && b2) return true;
48         if (b1 || b2) return true;
49         return false;
50     }
51
52     1 reference | Patrick Drechslar, 6 days ago | 1 author, 1 change
53     public int GetFirst(IEnumerable<int> numbers) => numbers.First();
54 }
```

RulesTests.cs - DemolibTests

```
1 public RulesTests() => _sut = new Rules();
2
3 // Include each 'InlineData' line one by one, run 'dotnet stryker' and see how it changes the outcome.
4 [Theory]
5 [InlineData("a", true)] // -> 3 survive
6 [InlineData("", false)] // -> 2 survive
7 [InlineData(null, false)] // -> 2 survive
8 [InlineData("123", true)] // -> 1 survive
9 [InlineData("12345", false)] // -> 0 survive
10
11 public void ValidationRule_works(string input, bool expected) =>
12     _sut.IsValid(input).Should().Be(expected);
13
14 [Theory]
15 [InlineData(5, 5, true)] // addition
16 [InlineData(15, 5, true)] // subtraction
17 [InlineData(2, 5, true)] // multiplication
18 [InlineData(20, 2, true)] // division
19 [InlineData(5, 4, true)] // modulo
20 [InlineData(1, 1, false)] // other combinations
21
22 0 references | Patrick Drechslar, 3 hours ago | 2 authors, 3 changes
23 public void AnotherValidation_works(int a, int b, bool expected) =>
24     _sut.AnotherValidation(a, b).Should().Be(expected);
25
26 [Theory]
27 [InlineData(3, 3)]
28
29 0 references | Patrick Drechslar, 3 days ago | 1 author, 2 changes
30 public void Looping_works(int input, int expected) =>
31     _sut.Looping(input).Should().Be(expected);
32
33 [Theory]
34 [InlineData(true, true, true)]
35 [InlineData(true, false, true)]
36 [InlineData(false, true, true)]
37 [InlineData(false, false, false)]
38
39 0 references | Patrick Drechslar, 3 hours ago | 2 authors, 3 changes
40 public void Logical_works(bool a, bool b, bool expected) =>
41     _sut.Logical(a, b).Should().Be(expected);
42
43 // https://github.com/stryker-mutator/stryker-handbook/blob/master/mutator-types.md#method-expression
44 [Theory]
45 [MemberData(nameof(GetFirstData), params parameters:1)] // list with length 1: First() = Last()
46 [MemberData(nameof(GetFirstData), params parameters:2)] // list with 2 different entries
47
48 0 references | Patrick Drechslar, 3 hours ago | 2 authors, 3 changes
49 public void GetFirst_works(IEnumerable<int> collection, int expected) =>
50     _sut.GetFirst(collection).Should().Be(expected);
51
52 2 references | Patrick Drechslar, 3 days ago | 1 author, 2 changes
53 public static IEnumerable<object[]> GetFirstData(int numberOfTests)
54 {
55     var allData = new List<object[]>()
56     {
57         new object[] { new List<int> { 1 }, 1 },
58         new object[] { new List<int> { 1, 2 }, 1 },
59     };
60     return allData.Take(numberOfTests);
61 }
62
63
64
65
```

Unit Test Coverage

Coverage Tree | Hot Spots | All Tests in All S...

Coverage tree has excluded nodes. Show all nodes

Type to search

Symbol	Coverage (%)	Uncovered
Demolib	100%	0/41
Rules	100%	0/41
GetFirst	100%	0/1
Logical	100%	0/7
IsValid	100%	0/10
Looping	100%	0/10
Another	100%	0/13

100% code coverage

Example

```
public class Rules
{
    public bool IsValid(string s)
    {
        if (string.IsNullOrEmpty(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/>

Arithmetic Operators
(*arithmetic*)

Equality Operators (*equality*)

Logical Operators (*logical*)

Boolean Literals (*boolean*)

Assignment Statements
(*assignment*)

Collection initialization
(*initializer*)

Removal mutators (*statement*,
block)

Unary Operators (*unary*)

Update Operators (*update*)



Checked Statements (*checked*)

Linq Methods (*ling*)

String Literals and Constants
(*string*)

Bitwise Operators (*bitwise*)

Regular Expressions (*regex*)

 Watch the video "How to test your tests in .NET" from *Nick Chapsas* 

  Stryker Mutator



 > Stryker.NET > Mutations

On this page



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:

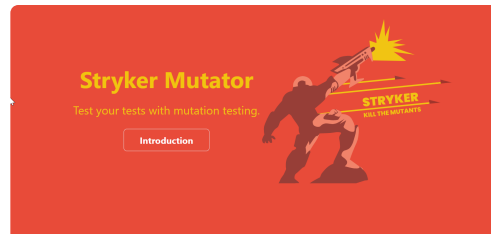
- 🐙 Java (using PIT)
- Scala (using Stryker4s)
- 🐙 C# (using Stryker.NET)
- 🐙 Javascript/Typescript (using StrykerJS)
- 🐙 Python (using Cosmic Ray or mutmut)
- Haskell (using MuCheck)
- ...

🐙: Example project in the repo



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



JavaScript and friends



C#



Scala

mutmut **mutmut - python mutation tester**

build passing docs failing codecov 81% chat 37 online

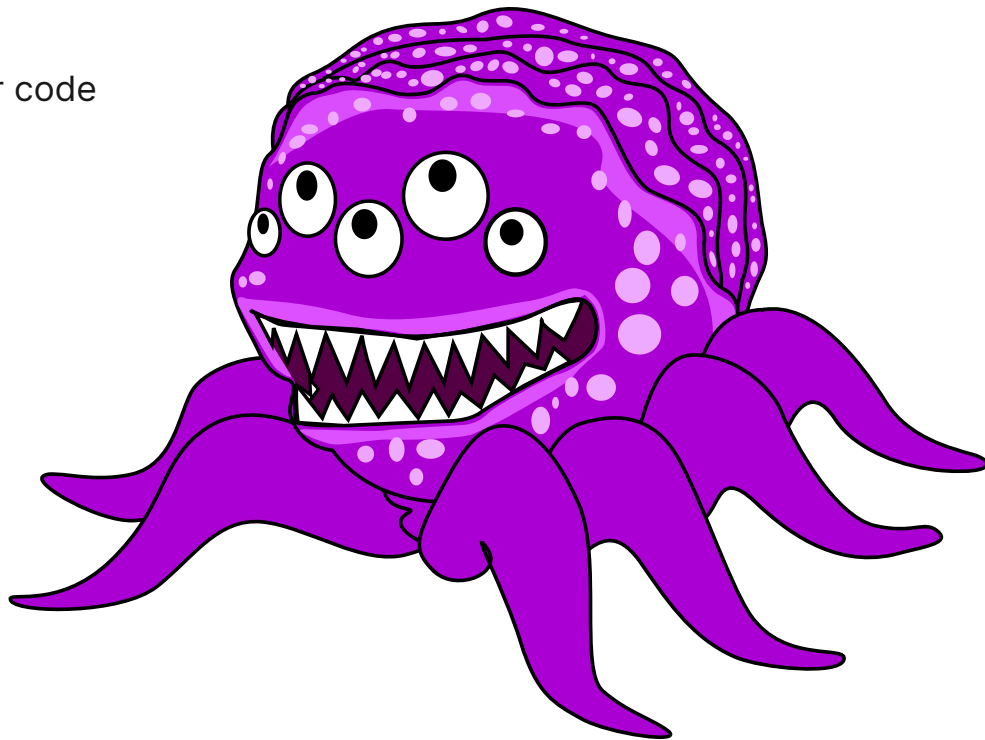
» Hackage :: [Package]

MuCheck: Automated Mutation Testing



Mutation Testing: Summary

- Mutation testing modifies production code to find corner cases
- It 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?

