

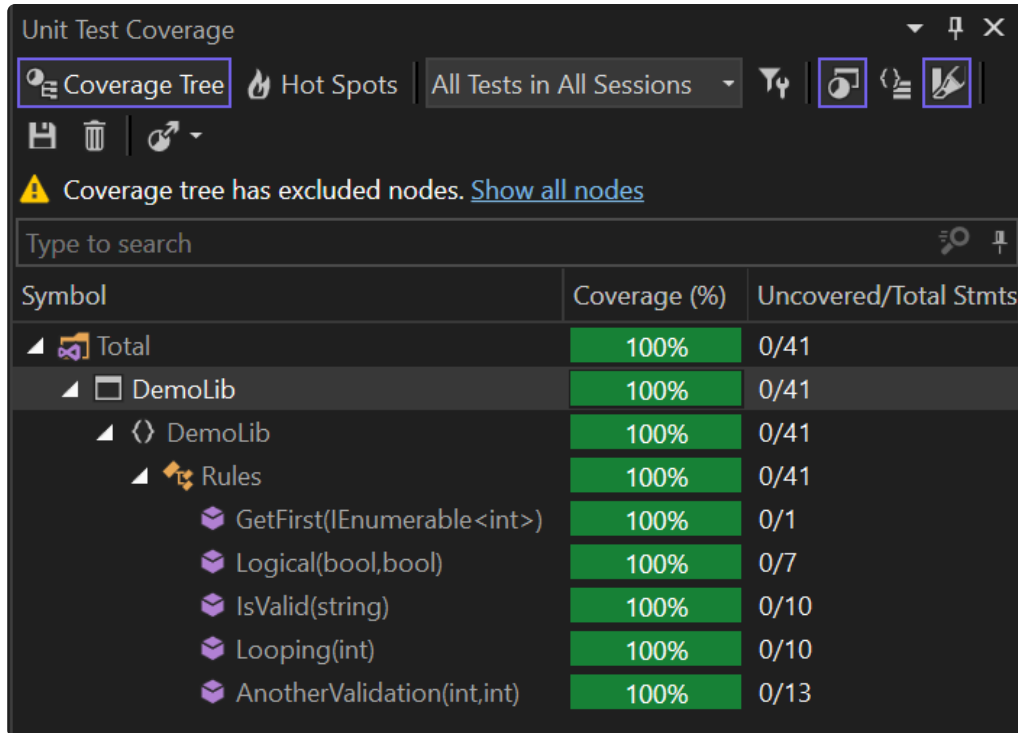
Mutation Testing

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

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:

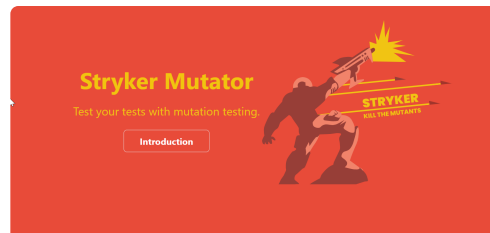
- 🐙 Java (using PIT)
- Scala (using Stryker4s)
- 🐙 C# (using Stryker.NET)
- 🐙 Javascript/Typescript (using StrykerJS)
- 🐙 Python (using Cosmic Ray or mutmut)
- Haskell (using MuCheck)
- ...

🐙: 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

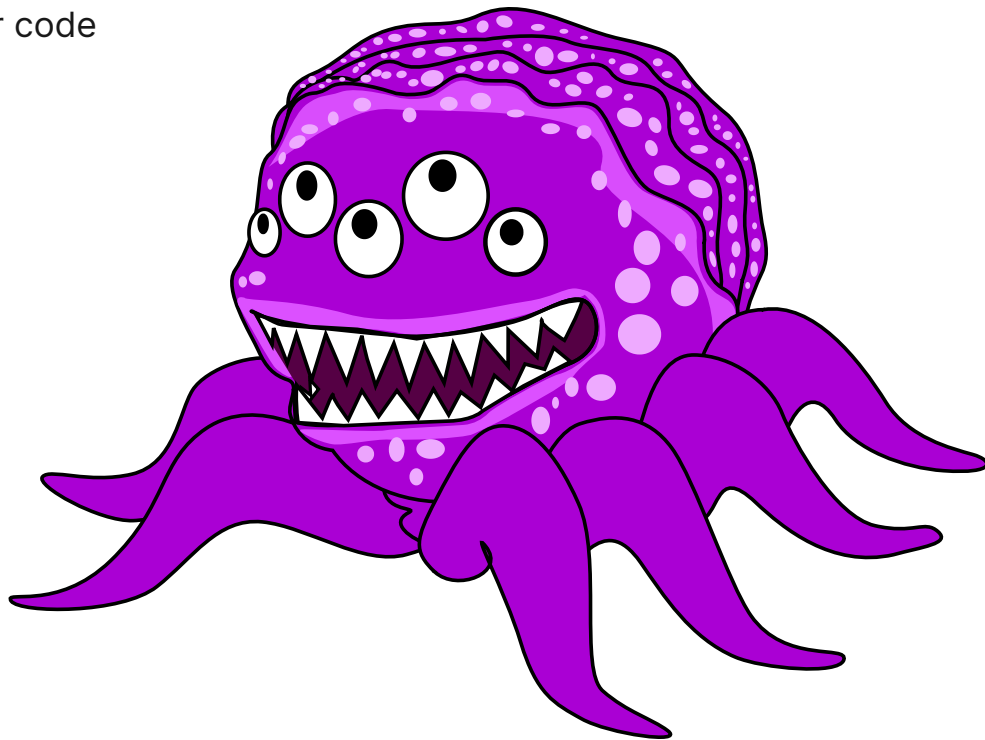
» Hackage :: [Package]

MuCheck: Automated Mutation Testing



Mutation Testing: Summary

- Mutation testing is a process of testing code for unintended side effects
- Don't include it in your CI/CD pipeline (it's not a low resource process)!
- 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?

