# Controlling Test Execution and Running Additional Code

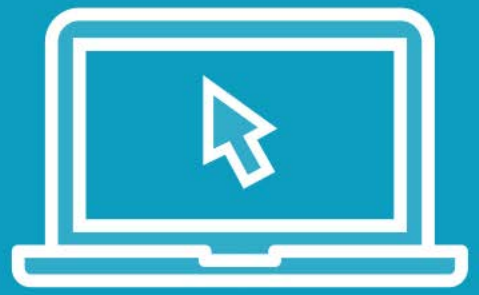

**Jason Roberts**
.NET MVP

@robertsjason dontcodetired.com

# Overview

- Using @tags to execute subsets of tests
- Restricting step execution with scoped bindings
- Scoped binding rules
- Scoped bindings for team workflow
- Running additional code with hooks
- Executing code before/after each scenario
- Considerations for parallel test execution
- Parallel test execution in action

# Demo



## Using Tags to Execute Subsets of Tests

Add @elf tag to specific scenarios

Build

Test Explorer

Traits view

Class view with filter

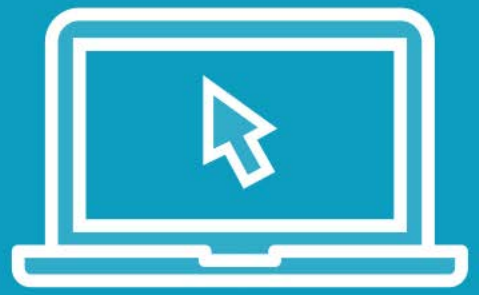
Trait : “elf”

Run all will only run “elf” tests

Special ignore tag

@elf @ignore

# Demo



## Restricting Step Execution with Scoped Bindings

Different "I take (.*) damage" for elves

New WhenITakeDamageAsAnElf method

Ambiguous step definition error

Add [Scope]

Feature, Scenario, tag

Scope step definition to only scenarios tagged with @elf

```
[Scope(Tag="elf",
       Feature="PlayerCharacter",
       Scenario="Health reduction")]


[Scope(Tag = "elf")]
[Scope(Feature = "PlayerCharacter")]
[Scope(Scenario = "Health reduction")]
```

# Combining Scoped Binding Criteria

**Multiple criteria on single [Scope] → AND**

**Multiple [Scope] attributes → OR**

# Multiple Matches

Matches regular step definition and scoped step definition

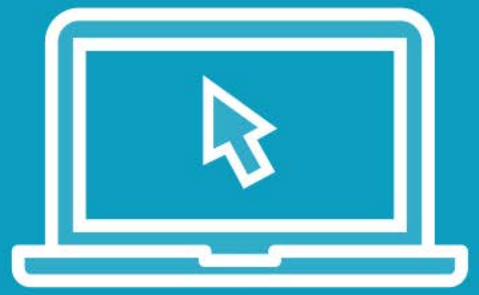
**Scoped definition will be used**

Matches >1 scoped step definition

**Most restrictive step will be used**

Be careful with scoped bindings as they can introduce complexity and maintenance overheads.

# Demo



## Scoped Bindings for Team Workflow

Add scenarios but not implement yet

New feature for game chat

Don’t want to implement until reviewed

Add AwaitingReviewSteps class

Add [Binding] attribute

Match all steps

Run → Ambiguous step definitions

@awaitingReviewBeforeStartingWork

Add [Scope] to AwaitingReviewSteps class

# Overview of Hooks

Before test run starts

Before feature

Before scenario

Before scenario block

Before step

After step

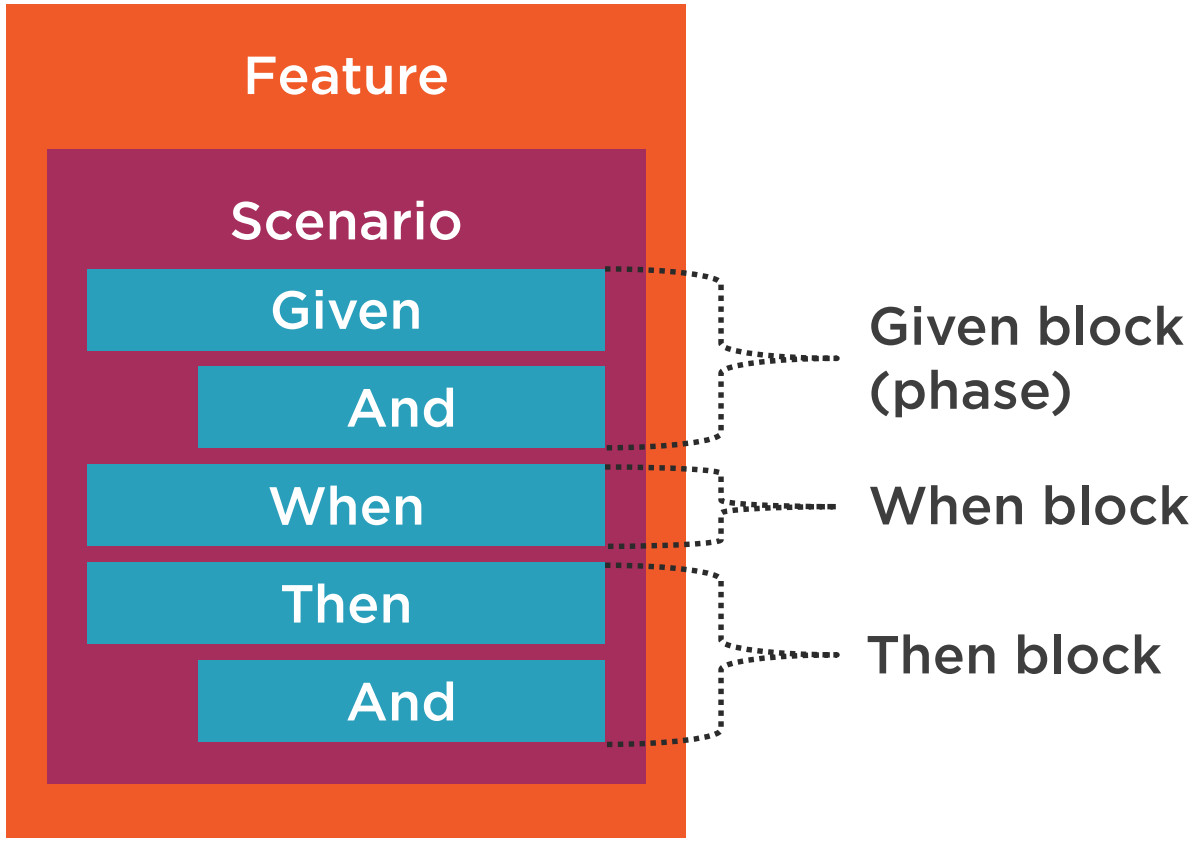
After scenario block

...

After scenario

After feature

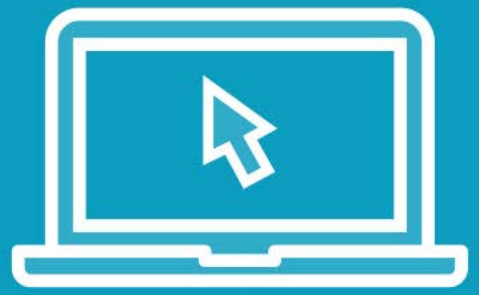
After test run ends



# Hooks Attributes

| | |
|---|---|
| Before test run starts | [BeforeTestRun] |
| Before feature | [BeforeFeature] |
| Before scenario | [BeforeScenario] or [Before] |
| Before scenario block | [BeforeScenarioBlock] |
| Before step | [BeforeStep] |
| After step | [AfterStep] |
| After scenario block | [AfterScenarioBlock] |
| ... | |
| After scenario | [AfterScenario] or [After] |
| After feature | [AfterFeature] |
| After test run ends | [AfterTestRun] |

# Demo



Executing Code Before and After Every Scenario

New Hooks class

[Binding]

[BeforeScenario] method

[AfterScenario] method

Add breakpoints

Debug all tests

ScenarioContext.Current.ScenarioInfo

: Steps

this.ScenarioContext.ScenarioInfo

[BeforeScenario("elf")]

```
[BeforeScenario]
public void BeforeScenario1(){ ... }
```
3rd

```
[BeforeScenario]
public void BeforeScenario2() { ... }
```
1st

```
[BeforeScenario]
public void BeforeScenario3() { ... }
```
2nd

# Hook Execution Ordering

**Multiple hook methods are executed in unspecified order**

**Specify order**

```
[BeforeScenario(Order = 300)]
public void BeforeScenario1(){ ... }

[BeforeScenario(Order = 200)]
public void BeforeScenario2() { ... }

[BeforeScenario(Order = 100)]
public void BeforeScenario3() { ... }
```


# Hook Execution Ordering

**Specify relative order in hook attribute**

**Default order value = 1000**

**Lower order values execute first**

# Scenario Backgrounds and [BeforeScenario]

| Background: | [BeforeScenario] | Ordering |
| --- | --- | --- |
| **Common “setup” code** | **Common “setup” code** | **[BeforeScenario]** |
| **Relevant to business** | **Not business relevant** | **Then** |
| **Not purely automation** | **Automation/technical** | **Background** |

# Considerations for Parallel Test Execution

SpecFlow v2+

Nunit v3+  xUnit.net v2+

Production code may not be thread safe

Disable parallel execution

Commercial SpecFlow+ Runner
(Memory/AppDomain isolation)

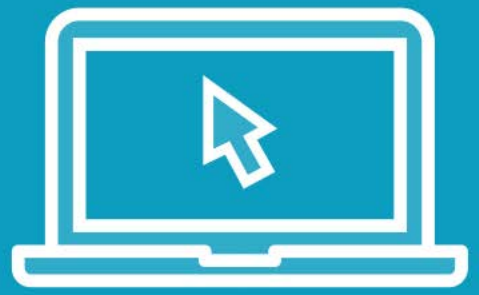
Parallelism across features

ScenarioContext.Current

Steps base class

Inject custom context class (strong typed)
or ScenarioContext class (weak typed)

# Demo



Parallel Test Execution in Action

One feature

Thread.Sleep

Run all

Split into 3 separate feature files

Run all, faster execution

Disable parallel test execution in xUnit.net

# Summary

Using @tags to execute subsets of tests

[Scope]

Tag, feature title, scenario title

Scoped binding rules(AND/OR)

Scoped bindings for team workflow

Running additional code with hooks

[BeforeScenario] [AfterScenario]

Considerations for parallel test execution

Parallel test execution in action

[CollectionBehavior]