Increasing Maintainability with Shared Steps, Parameters, and Data Tables

Jason Roberts
.NET MVP

@robertsjason dontcodetired.com

Overview

Four ways to improve maintainability

Shared step definitions

Parameterized step definitions

Reducing duplicate scenarios with Scenario Outlines

Improving readability with step table data

Reducing duplicate Given steps and improving readability in scenarios

Four Ways to Improve Maintainability

Shared Step Definitions	Scenario Outlines	Scenario Step Data Tables	Scenario Backgrounds
Code reuse	**Duplication**	**Readability**	**Readability**
Parameters	**One scenario**	**Multiple And**	**Repeated Given**
	Multiple tests	**Tabular data**	**Incidental**

Parameterized Step Definitions

Parameterized step definitions allow values to be passed from scenario steps to test automation code.

Using parameterized step definitions promotes the reuse of step definition code by allowing scenario steps to share code where the only difference is in the value of data items.

```
[When(@"I take (.*) damage")]
public void WhenITakeDamage(int damage)


[When(@"I take (.*) damage from a (.*)")]
public void WhenITakeDamageFromASword(int damage,
                                      string weaponType)
```

Regular Expression Binding Parameters

Add “holes” in attribute

“Holes” take the form of regular expression matches

Values are passed to step method parameters

```
[When]
public void When_I_take_P0_damage(int p0)
public void When_I_take_DAMAGE_damage(int damage)
public void When_I_take_P0_damage_from_a_P1(
                                        int p0, string p1)
public void When_I_take_DAMAGE_damage_from_a_WEAPONTYPE(
                                    int damage, string weaponType
```

Underscore Binding Parameters

Add “holes” in uppercase

Positional: P0, P1, etc.

Named: DAMAGE, WEAPONTYPE, etc.

```
[When]
public void WhenITake_P0_Damage(int p0)
public void WhenITake_DAMAGE_DamageFromA_WEAPONTYPE(
                                    int damage, string weaponType)
```

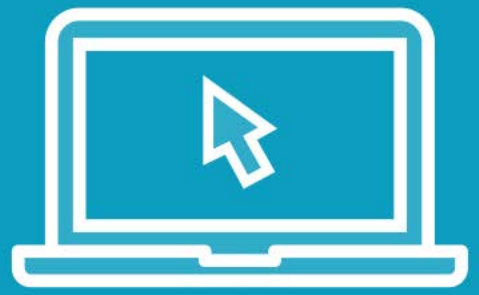
Pascal Case Binding Parameters

Add “holes” in uppercase

Positional: P0, P1, etc.

Named: DAMAGE, WEAPONTYPE, etc.

Demo



Refactoring to Use Step Parameters

- Changes to PlayerCharacter demo class
- Refactor to parameterized step definition
- “When I take 0 damage”
- “Then My health should now be 60”
- Remove duplicated step definitions

Scenario Outlines

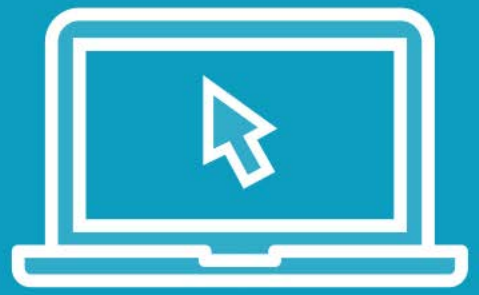
Scenario outlines allow the same basic scenario to be executed multiple times, each time with different test data.

Using scenario outlines allows the reduction or elimination of repeated scenarios where the only difference between the scenarios are the inputs or expected outcomes.

Introducing Scenario Outlines

```
Scenario Outline: Health reduction
Given I’m a new player
When I take <damage> damage
Then My health should now be <remainingHealth>
Examples:
    | damage | remainingHealth |
    | 0      | 100             |
    | 40     | 60              |
```

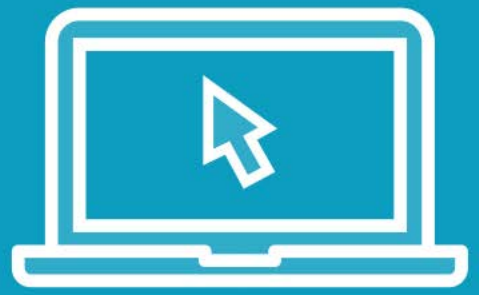
Demo



Refactoring to Use Scenario Outlines

- Refactor two scenarios
- “Taking no damage when hit doesn't affect health”
- “Starting health is reduced when hit”
- Create new Scenario Outline
- Add 2 examples
- Remove duplicated scenarios
- Run tests
- Easy to add further examples

Demo



Using Data Tables in Scenario Steps

New scenario: “Elf race characters get additional 20 damage resistance”

Implement with “And” steps

Run test

Refactor to use a step data table

Access rows from Table object*

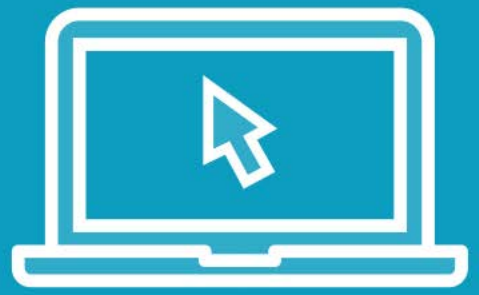
**The next module will cover more advanced / strongly typed data topics*

Scenario Backgrounds

Scenario backgrounds allow the moving of duplicated, non-essential, incidental, repeated Given steps to a common section.

Steps within the background section will be automatically executed before each scenario or scenario outline.

Demo



Creating Common Setup Code with Scenario Backgrounds

Remove duplicated “Given I'm a new player” step

Add a scenario background

Move Given step to background

Summary

Four ways to improve maintainability

(.*) P0 DAMAGE

Scenario Outline

Examples:

Multiple data items with step table data

Reducing duplicate Given steps and improving readability in scenarios

Background:

Duplicated, non-essential, incidental Given steps

Next:

Working With Data in Step Definitions