Working with Data in Step Definitions

Jason Roberts
.NET MVP

@robertsjason dontcodetired.com

Overview

- Step argument conversion
- Built-in automatic / custom transforms
- Automatic enum conversion
- Strongly-typed step table data
- Dynamic step table data
- Custom data transforms
- “3 days ago” → DateTime
- Automatically applying custom transforms
- Passing data between steps definitions

```
When I take 100 damage
```

```
[When(@"I take (.*) damage")]
```

```
public void WhenITakeDamage(int damage)
```

Step Argument Conversion

Plain text in feature file

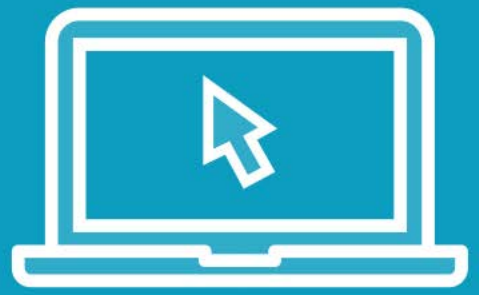
Parameters match pieces of text

Plain text (string) → .NET parameter data type

Step Argument Conversion Precedence

No Conversion	Custom Step Argument Transformation	Standard (Inbuilt) Conversion
object, string	**Parameter type matches a defined custom transform**	**Convert.ChangeType()** **Text → enum value** **Text → GUID** **E.g. int damage**

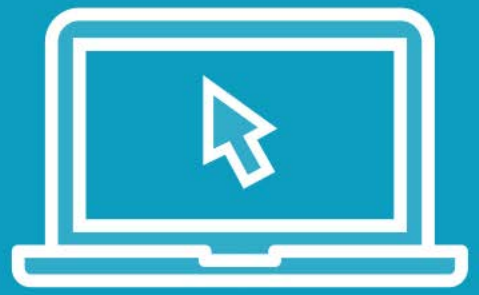
Demo



Additions to the PlayerCharacter Class

- New MagicalItem class
- New Weapon class
- New CharacterClass enum
- New PlayerCharacter methods / properties

Demo



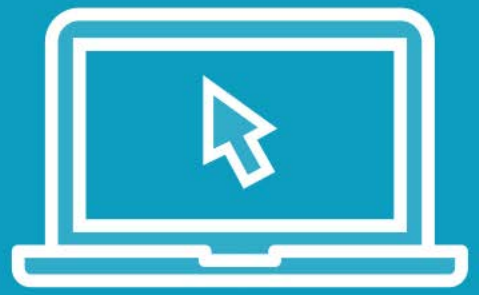
Automatic Enum Conversion

New scenario “Healers restore all health”

Parameterized CharacterClass enum parameter

Standard conversion

Demo



Strongly-Typed Step Table Data

GivenIHaveTheFollowingAttributes code

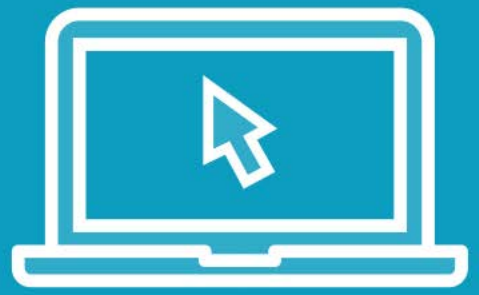
New PlayerAttributes class in test project

Using TechTalk.SpecFlow.Assist;

Extension methods on Table class

CreateInstance<PlayerAttribute>()

Demo



Dynamic Step Table Data

Reduce need for extra test classes

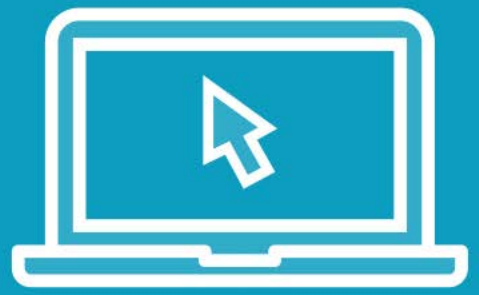
Modify GivenIHaveTheFollowingAttributes

Use dynamic C#

SpecFlow.Assist.Dynamic NuGet package

table.CreateDynamicInstance()

Demo



Multi-Column Step Table Data

New scenario: Total magical power

Table of magical items

Weakly-typed code

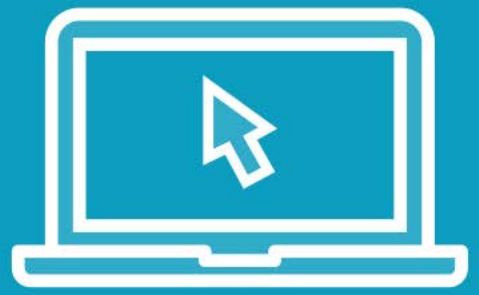
Strongly-typed version

CreateSet<MagicalItem>()

Dynamic version

CreateDynamicSet()

Demo



Custom Data Conversions

New scenario: “Reading a restore health scroll when over tired has no effect”

Parameterized step definition

DateTime parameter

Capture text “3 days ago”

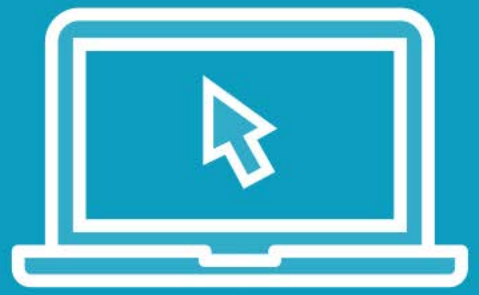
"I last slept (.* days ago)"

New CustomConversions class

Method to perform conversion

[StepArgumentTransformation(...)]

Demo



Automatically Applying Custom Transforms

Scenario “Weapons are worth money”

Parameterized step definition

Replace: Table table

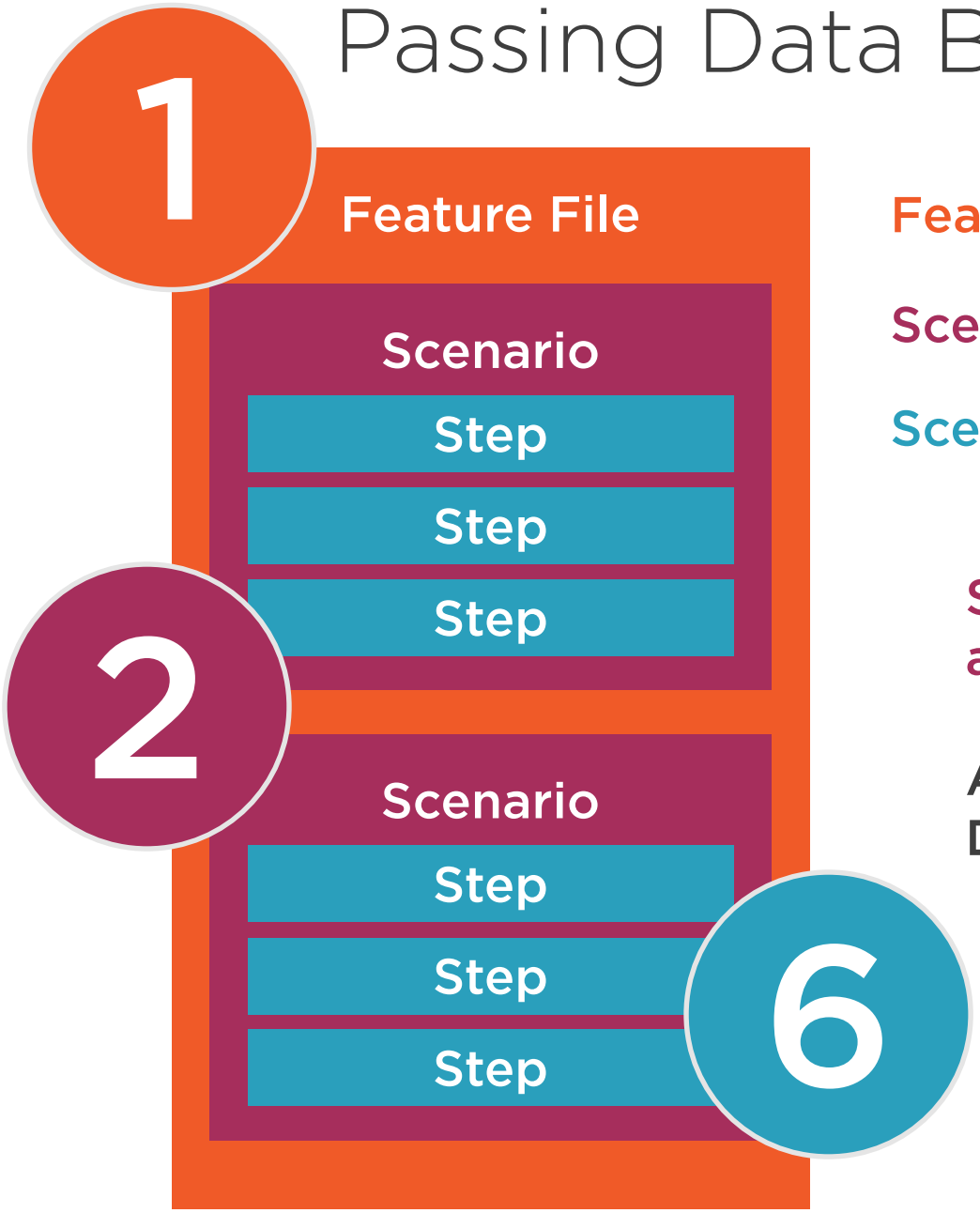
With: IEnumerable<Weapon> weapons

Add WeaponsTransformation

Passing Data Between Step Definitions

Step Class Fields or Properties	SpecFlow Provided Context Objects	Custom Context Object Injection
Multi class bindings	**Convenience**	**Additional class**
Static/parallel	**Thread safe versions**	**Strongly typed**
	Weakly-typed dictionary	**Thread safe**

Passing Data Between Step Definitions



Feature Context

Scenario Context

Scenario Step Context

Scenario Context can be used to share data between all the steps of a single executing scenario

All 3 context types provide a weakly typed Dictionary<string, object>

ScenarioContext (Not Thread Safe)

```
[Given(@"I have an Amulet with a power of (.*)")]
public void GivenIHaveAnAmuletWithAPowerOf(int power)
{
    ScenarioContext.Current["power"] = power;
}
```

```
[Then(@"The Amulet power should not be reduced")]
public void ThenTheAmuletPowerShouldNotBeReduced()
{
    int expectedPower = (int) ScenarioContext.Current["power"];
}
```

ScenarioContext (Thread Safe)

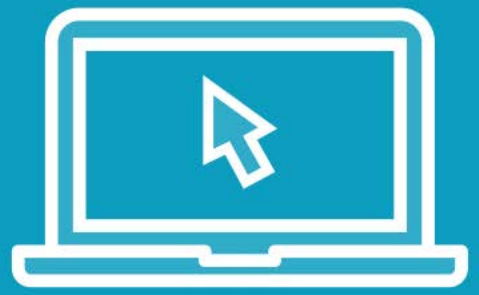
```
[Given(@"I have an Amulet with a power of (.*)")]
public void GivenIHaveAnAmuletWithAPowerOf(int power)
{
    this.ScenarioContext["power"] = power;
}
```

```
[Then(@"The Amulet power should not be reduced")]
public void ThenTheAmuletPowerShouldNotBeReduced()
{
    int expectedPower = (int) this.ScenarioContext["power"];
}
```

ScenarioContext (Thread Safe)

```
[Binding]
public class PlayerCharacterSteps : TechTalk.SpecFlow.Steps
{
    // Step definitions
}
```

Demo



Using Context Injection

- Strongly typed
- Scenario “Elf race characters don't lose magical item power”
- “Given I’m a new player” separate step class
- Null reference exception
- Store value from Given step
- Retrieve value in Then step
- New PlayerCharacterStepsContext class
- Add constructor to steps classes
- Instance provided by SpecFlow

Summary

Step argument conversion

Built-in automatic / custom transforms

Automatic enum conversion

Strongly-typed step table data

Dynamic step table data

[StepArgumentTransformation]

“3 days ago” → DateTime

IEnumerable<Weapon>

ScenarioContext and context injection

Next:

Controlling Test Execution and Running Additional Code