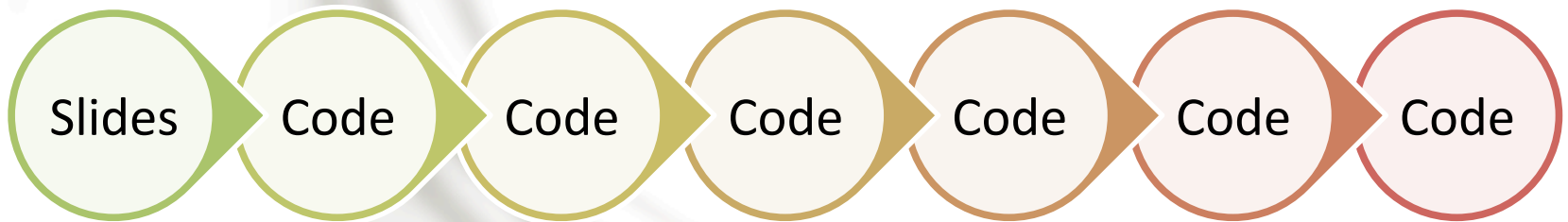




Bending AutoFixture to your will

Mark Seemann  
@ploeh

# Agenda



# Goals

AutoFixture  
provides  
feedback –  
listen to it

You can  
modify  
AutoFixture's  
behavior to  
fit your needs

Favor  
conventions  
over specific  
modifications

# Test Data Builder

```
public class ContactBuilder
{
    private readonly string n;
    private readonly string p;

    public ContactBuilder() : this("Jane Doe", "55512345") { }

    private ContactBuilder(string name, string phone)
    {
        this.n = name;
        this.p = phone;
    }

    public ContactBuilder WithName(string name)
    {
        return new ContactBuilder(name, this.p);
    }

    public ContactBuilder WithPhone(string phone)
    {
        return new ContactBuilder(this.n, phone);
    }

    public Contact Build()
    {
        return new Contact { Name = this.n, Phone = this.p };
    }
}
```

# ContactBuilder examples

```
var contact = new ContactBuilder().Build();
```

```
var contact = new ContactBuilder().WithName("John Doe").Build();
```

```
var contact = new ContactBuilder().WithPhone("12345678").Build();
```

# Test Data Builder

A dream to  
use

A pain to  
maintain

A TDB  
for  
every  
SUT



# AutoFixture



Takes away the pain  
of maintaining TDBs



Reflection-based



Convention-based

The background of the slide features a grayscale image of several interlocking gears of different sizes, creating a mechanical or industrial aesthetic. The gears are slightly out of focus, giving a sense of depth.

Replace ContactBuilder with AutoFixture

**DEMO**

# What just happened?



The background of the slide features a grayscale illustration of several interlocking gears of different sizes, creating a mechanical and industrial aesthetic.

Customize the Fixture

**DEMO**

# What just happened?

The phone rule  
was  
encapsulated in  
a class



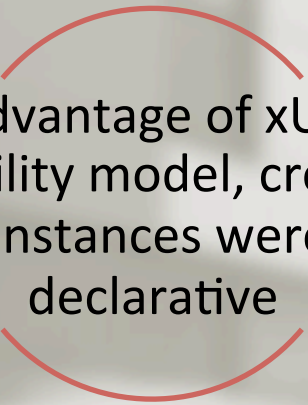
This provides a  
central place to  
manage the rule

The background of the slide features a faint, grayscale illustration of several interlocking gears of different sizes, creating a mechanical or industrial aesthetic.

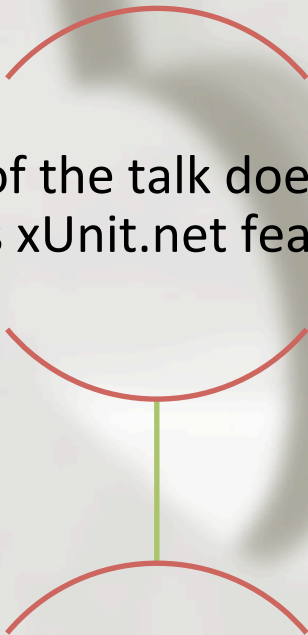
Move towards a declarative style

**DEMO**


# What just happened?



Taking advantage of xUnit.net's  
extensibility model, creation of  
object instances were made  
declarative



The rest of the talk doesn't hinge  
on this xUnit.net feature at



It's just a cool thing to do

The background of the slide features a faded, grayscale illustration of several interlocking mechanical gears of different sizes, creating a sense of complex machinery or a system in motion.

Create unique phone numbers

**DEMO**

# What just happened?

Phone  
numbers  
were made  
unique to a  
Fixture

Partial delegation  
to Fixture's ability  
to create unique  
numbers

The background of the slide features a grayscale illustration of several interlocking gears of different sizes, creating a mechanical or industrial aesthetic. The gears are rendered with a soft, slightly blurred effect, giving a sense of depth and movement.

Auto-mocking

**DEMO**

# What just happened?



The background of the slide features a grayscale illustration of several interlocking gears of different sizes, creating a mechanical or industrial aesthetic. The gears are rendered with a soft, slightly blurred effect, giving a sense of depth and movement.

Circular references

**DEMO**

# What just happened?

Circular references  
are a design smell



However, they can  
be dealt with by  
changing the  
behavior of Fixture

The background of the slide features a faint, grayscale illustration of several interlocking gears of different sizes, creating a mechanical or industrial aesthetic.

Conventions for phone numbers

**DEMO**

# What just happened?

Manual edit using the  
Fluid Builder API

Convention based on  
PropertyInfo



Value resolved via int



Value resolved via  
RangedNumberRequest

# To go

If AutoFixture  
can't create an  
instance of your  
API, it's  
probably your  
fault

Use  
AutoFixture's  
many  
extensibility  
points to make  
it behave like  
you'd like

Strive towards a  
single set of  
conventions per  
test project



## AutoFixture

- <http://autofixture.codeplex.com/>
- <http://stackoverflow.com/questions/tagged/autofixture>
- Nuget: AutoFixture



## Mark Seemann

- <http://blog.ploeh.dk/>
- @ploeh



## Nikos Baxevanis

- <http://www.nikosbaxevanis.com/bonus-bits/>
- @nikosbaxevanis



## Enrico Campidoglio

- <http://megakemp.wordpress.com/>
- @ecampidoglio