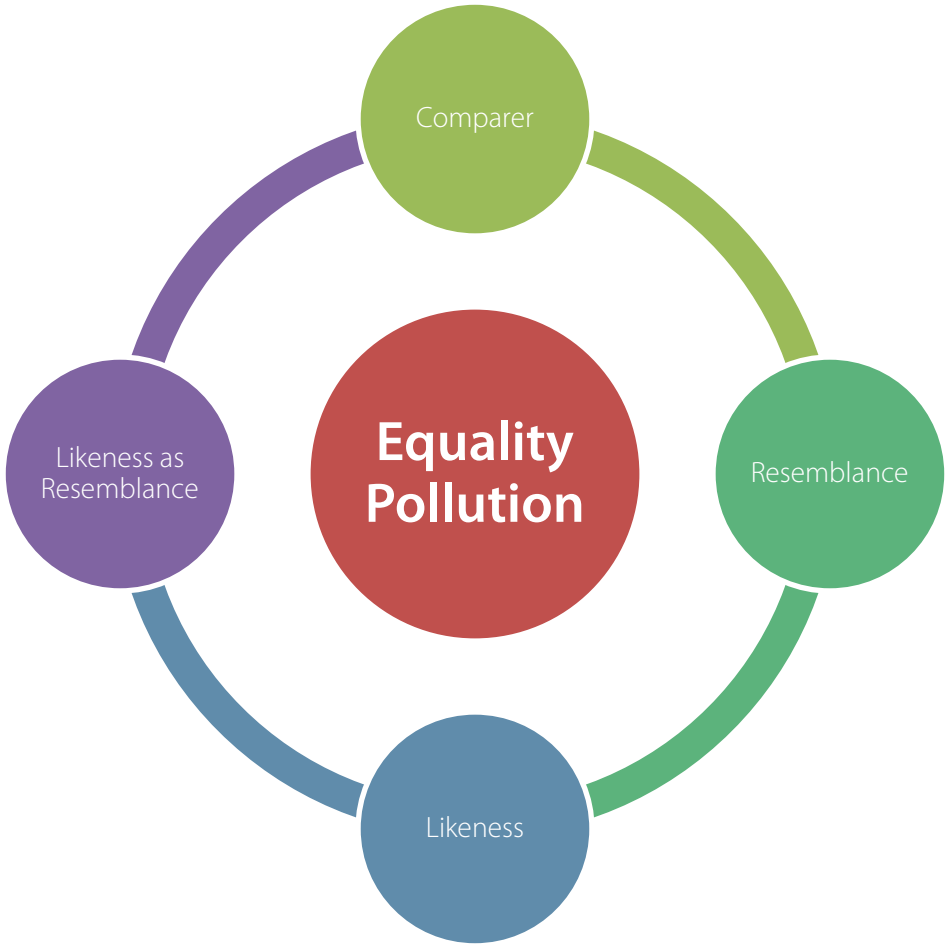
Advanced Unit Testing
Test-Specific Identity

Mark Seemann
http://blog.ploeh.dk

Overview


Comparer
Resemblance
Likeness
Likeness as Resemblance
Equality Pollution

Equality Pollution

Task

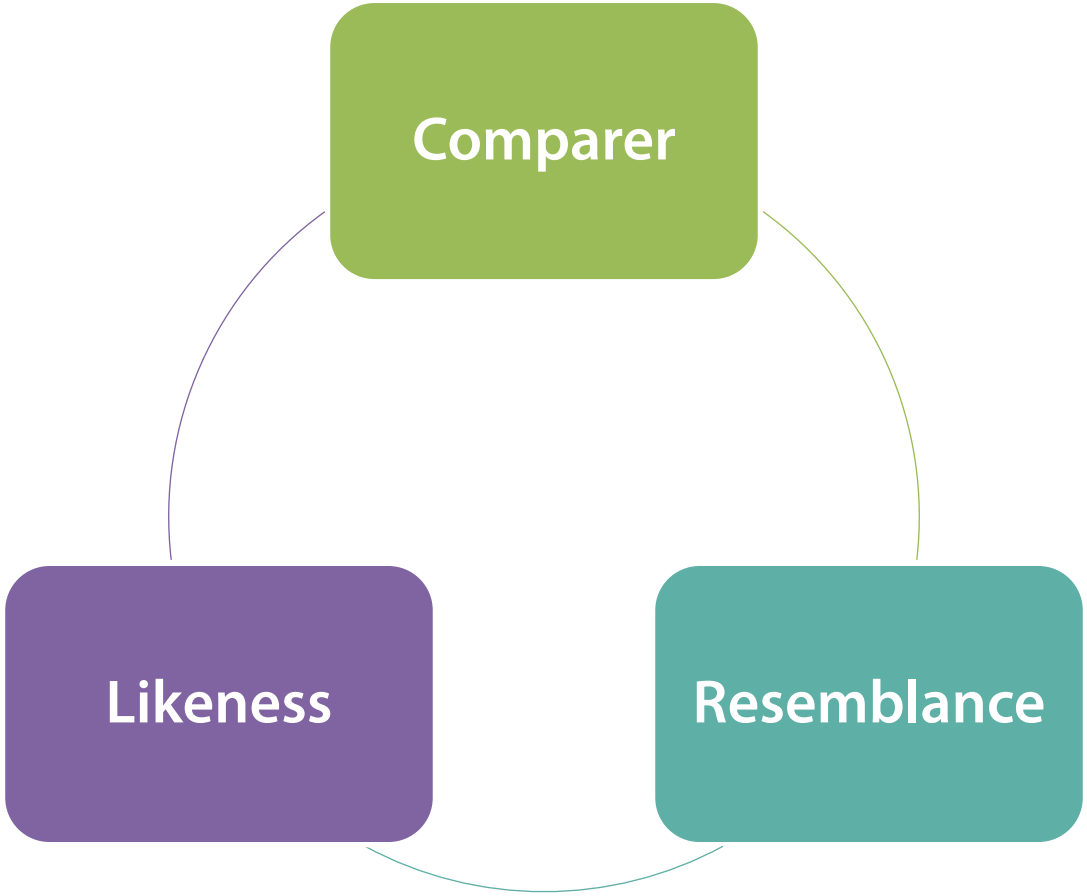
Compare expected and actual values according to properties

Problem

Entity

Service

Test-Specific Equality



IEqualityComparer<T>

xUnit.net

```
Assert.Equal<T>(T expected, T actual, IEqualityComparer<T> comparer);
Assert.Equal<T>(IEnumerable<T> expected, IEnumerable<T> actual,
    IEqualityComparer<T> comparer);
```

Everything else

```
Assert.IsTrue(new MyComparer().Equals(expected, actual));
```

Write your own extensions

Assertion Roulette?

```
[Theory]
[InlineData(1, 1, 1, 1, 1)]
[InlineData(2, 1, 1, 1, 1)]
[InlineData(2, 2, 1, 1, 1)]
[InlineData(2, 2, 2, 1, 1)]
[InlineData(2, 2, 2, 2, 1)]
[InlineData(2, 2, 2, 2, 2)]
public void VisitBasketItemReturnsCorrectResult(
    int threshold, int rate, int subtotal, int unitPrice, int quantity)
{
    var sut = new VolumeDiscountVisitor(threshold, rate, subtotal);

    var basketItem =
        new BasketItem("Dummy", unitPrice, quantity);
    var actual = sut.Visit(basketItem);

    var vd = Assert.IsAssignableFrom<VolumeDiscountVisitor>(actual);
    Assert.Equal(threshold, vd.Threshold);
    Assert.Equal(rate, vd.Rate);
    Assert.Equal(subtotal + basketItem.Total, vd.Subtotal);
}
```

Not DAMP

Concrete Comparer

```
public class VolumeDiscountVisitorComparer :
    IEqualityComparer<VolumeDiscountVisitor>
{
    public bool Equals(VolumeDiscountVisitor x, VolumeDiscountVisitor y)
    {
        return object.Equals(x.Threshold, y.Threshold)
            && object.Equals(x.Rate, y.Rate)
            && object.Equals(x.Subtotal, y.Subtotal);
    }

    public int GetHashCode(VolumeDiscountVisitor obj)
    {
        return
            obj.Threshold.GetHashCode() ^
            obj.Rate.GetHashCode() ^
            obj.Subtotal.GetHashCode();
    }
}
```

Refactored test

IBasketVisitor

```
[Theory]
[InlineData(1, 1, 1, 1, 1)]
[InlineData(2, 1, 1, 1, 1)]
[InlineData(2, 2, 1, 1, 1)]
[InlineData(2, 2, 2, 1, 1)]
[InlineData(2, 2, 2, 2, 1)]
[InlineData(2, 2, 2, 2, 2)]
public void VisitBasketItemReturnsCorrectResult(
    int threshold, int rate, int subtotal, int unitPrice, int quantity)
{
    var sut = new VolumeDiscountVisitor(threshold, rate, subtotal);

    var basketItem =
        new BasketItem("Dummy", unitPrice, quantity);
    var actual = sut.Visit(basketItem);

    var expected = new VolumeDiscountVisitor(
        threshold, rate, subtotal + basketItem.Total);
    var vd = Assert.IsAssignableFrom<VolumeDiscountVisitor>(actual);
    Assert.Equal(expected, vd, new VolumeDiscountVisitorComparer());
}
```

Interface Comparer

```
public class VolumeDiscountVisitorComparer :
    IEqualityComparer<VolumeDiscountVisitor>,
    IEqualityComparer<IBasketVisitor>
{
    public bool Equals(VolumeDiscountVisitor x, VolumeDiscountVisitor y)
    {
        return object.Equals(x.Threshold, y.Threshold)
            && object.Equals(x.Rate, y.Rate)
            && object.Equals(x.Subtotal, y.Subtotal);
    }

    public bool Equals(IBasketVisitor x, IBasketVisitor y)
    {
        var vdvX = x as VolumeDiscountVisitor;
        var vdvY = y as VolumeDiscountVisitor;
        return vdvX != null && vdvY != null && this.Equals(vdvX, vdvY);
    }

    // GetHashCode implementation goes here...
}
```

Refactored test

```
[Theory]
[InlineData(1, 1, 1, 1, 1)]
[InlineData(2, 1, 1, 1, 1)]
[InlineData(2, 2, 1, 1, 1)]
[InlineData(2, 2, 2, 1, 1)]
[InlineData(2, 2, 2, 2, 1)]
[InlineData(2, 2, 2, 2, 2)]
public void VisitBasketItemReturnsCorrectResult(
    int threshold, int rate, int subtotal, int unitPrice, int quantity)
{
    var sut = new VolumeDiscountVisitor(threshold, rate, subtotal);

    var basketItem =
        new BasketItem("Dummy", unitPrice, quantity);
    var actual = sut.Visit(basketItem);

    var expected = new VolumeDiscountVisitor(
        threshold, rate, subtotal + basketItem.Total);
    Assert.Equal(expected, actual, new VolumeDiscountVisitorComparer());
}
```

Refactored test

```
[Theory]
[InlineData(1, 1, 1, 1, 1)]
[InlineData(2, 1, 1, 1, 1)]
[InlineData(2, 2, 1, 1, 1)]
[InlineData(2, 2, 2, 1, 1)]
[InlineData(2, 2, 2, 2, 1)]
[InlineData(2, 2, 2, 2, 2)]
public void VisitBasketItemReturnsCorrectResult(
    int threshold, int rate, int subtotal, int unitPrice, int quantity)
{
    var sut = new VolumeDiscountVisitor(threshold, rate, subtotal);

    var basketItem =
        new BasketItem("Dummy", unitPrice, quantity);
    var actual = sut.Visit(basketItem);

    var expected = sut.WithSubtotal(subtotal + basketItem.Total);
    Assert.Equal(expected, actual, new VolumeDiscountVisitorComparer());
}
```

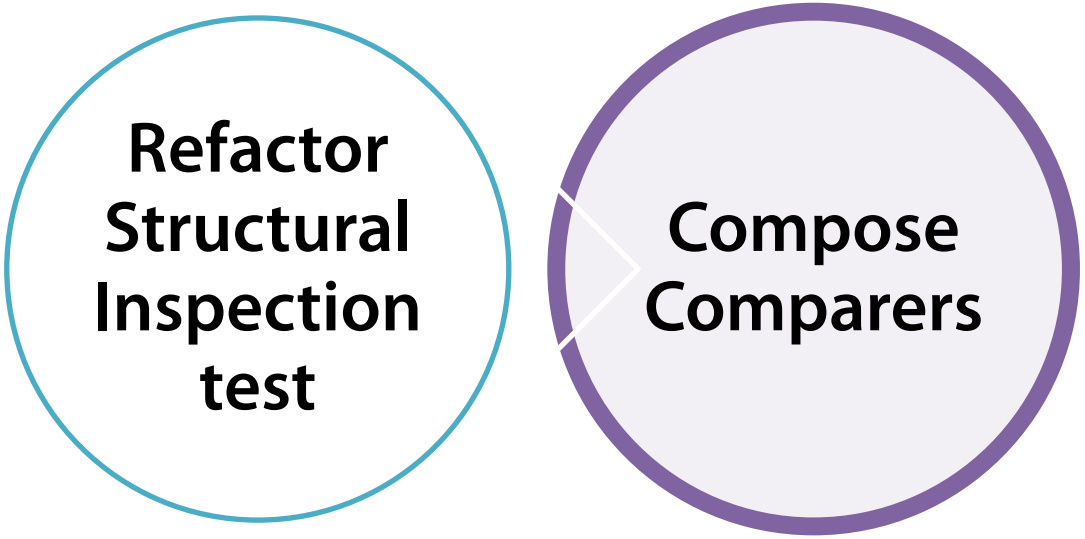
Refactored test

```
[Theory]
[InlineData(1, 1, 1, 1, 1)]
[InlineData(2, 1, 1, 1, 1)]
[InlineData(2, 2, 1, 1, 1)]
[InlineData(2, 2, 2, 1, 1)]
[InlineData(2, 2, 2, 2, 1)]
[InlineData(2, 2, 2, 2, 2)]
public void VisitBasketItemReturnsCorrectResult(
    int threshold, int rate, int subtotal, int unitPrice, int quantity)
{
    var sut = new VolumeDiscountVisitor(threshold, rate, subtotal);

    var basketItem =
        new BasketItem("Dummy", unitPrice, quantity);
    var actual = sut.Visit(basketItem);

    Assert.Equal(
        sut.WithSubtotal(subtotal + basketItem.Total),
        actual,
        new VolumeDiscountVisitorComparer());
}
```

Demo



Demo recap

BasketVisitorPipeComparer compares contained IBasketVisitor Inspection Properties

It uses a CompositeEqualityComparer<IBasketVisitor> to compare visitors

The Composite contains three 'leaf' comparers

The comparison result is true if at least one of the 'leaf' comparers return true

Resemblance

Make a derived class

- In test project

Override Equals

- And GetHashCode

Use derived instance as expected value

- Invoke Equals to compare against actual

Non-sealed classes only

Unit test this method

```
[HttpPost]
public ViewResult Post(BookingViewModel model)
{
    this.channel.Send(model.MakeReservation());
    return this.View("Receipt", model);
}
```

MakeReservertion

```
public RequestReservationCommand MakeReservation()
{
    return new RequestReservationCommand(
        this.Date,
        this.Email,
        this.Name,
        this.Quantity);
}
```

RequestReservationCommand constructor

```
public RequestReservationCommand(
    DateTime date,
    string email,
    string name,
    int quantity)
{
    this.date = date;
    this.email = email;
    this.name = name;
    this.quantity = quantity;
    this.id = Guid.NewGuid();
}
```

Unit test this method

```
[HttpPost]
public ViewResult Post(BookingViewModel model)
{
    this.channel.Send(model.MakeReservation());
    return this.View("Receipt", model);
}
```

RequestReservationCommand

Value Object identity?

Entity identity?

Unit test with no DSL

```
[Theory, AutoWebData]
public void PostSendsOnChannel(
    [Frozen]Mock<IChannel<RequestReservationCommand>> channelMock,
    BookingController sut,
    BookingViewModel model)
{
    sut.Post(model);

    var expected = model.MakeReservation();
    channelMock.Verify(c =>
        c.Send(It.Is<RequestReservationCommand>(cmd =>
            cmd.Date == expected.Date &&
            cmd.Email == expected.Email &&
            cmd.Name == expected.Name &&
            cmd.Quantity == expected.Quantity)));
}
```

Static helper method

```
private static bool Equals(
    RequestReservationCommand expected,
    RequestReservationCommand actual)
{
    return actual.Date == expected.Date &&
        actual.Email == expected.Email &&
        actual.Name == expected.Name &&
        actual.Quantity == expected.Quantity;
}
```

Unit test with static helper method

```
[Theory, AutoWebData]
public void PostSendsOnChannel(
    [Frozen]Mock<IChannel<RequestReservationCommand>> channelMock,
    BookingController sut,
    BookingViewModel model)
{
    sut.Post(model);

    var expected = model.MakeReservation();
    channelMock.Verify(c =>
        c.Send(It.Is<RequestReservationCommand>(cmd =>
            Equals(expected, cmd))));
}
```

Not DAMP

Resemblance

```
internal class RequestReservationCommandResemblance :
    RequestReservationCommand
{
    public RequestReservationCommandResemblance(
        RequestReservationCommand source) : base(
            source.Date, source.Email,
            source.Name, source.Quantity) { }

    public override bool Equals(object obj)
    {
        var other = obj as RequestReservationCommand;
        if (other != null)
            return object.Equals(this.Date, other.Date)
                && object.Equals(this.Email, other.Email)
                && object.Equals(this.Name, other.Name)
                && object.Equals(this.Quantity, other.Quantity);
        return base.Equals(obj);
    }
}
```

Don't forget GetHashCode!

Unit test with Resemblance

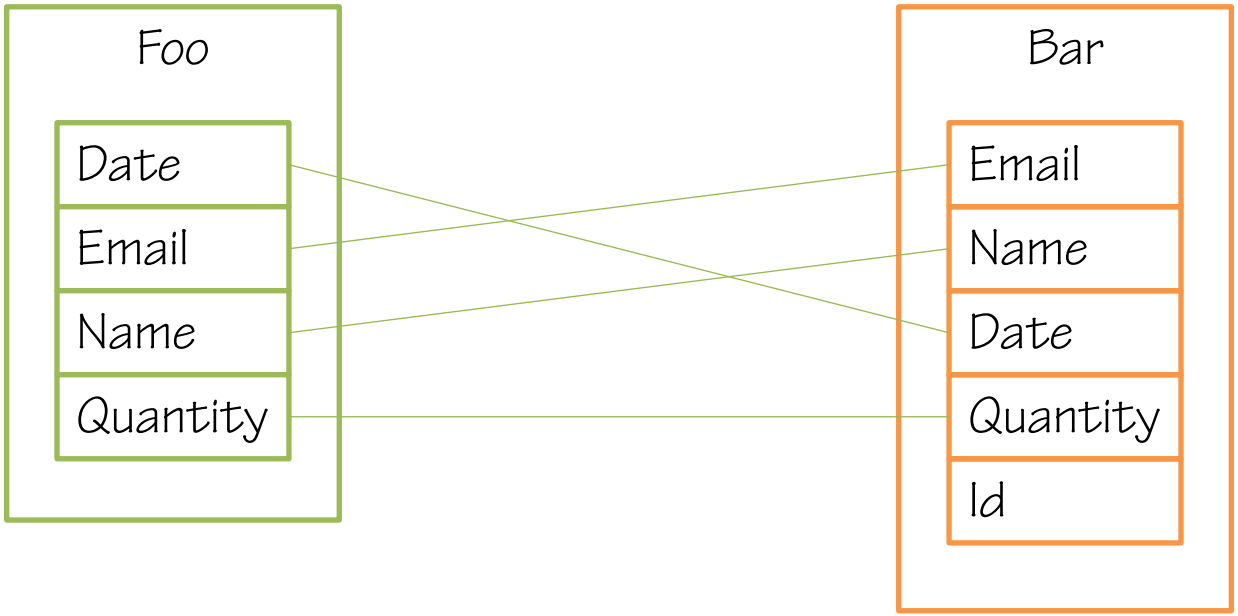
```
[Theory, AutoWebData]
public void PostSendsOnChannel(
    [Frozen]Mock<IChannel<RequestReservationCommand>> channelMock,
    BookingController sut,
    BookingViewModel model)
{
    sut.Post(model);

    var expected = new RequestReservationCommandResemblance(
        model.MakeReservation());
    channelMock.Verify(c => c.Send(expected));
}
```

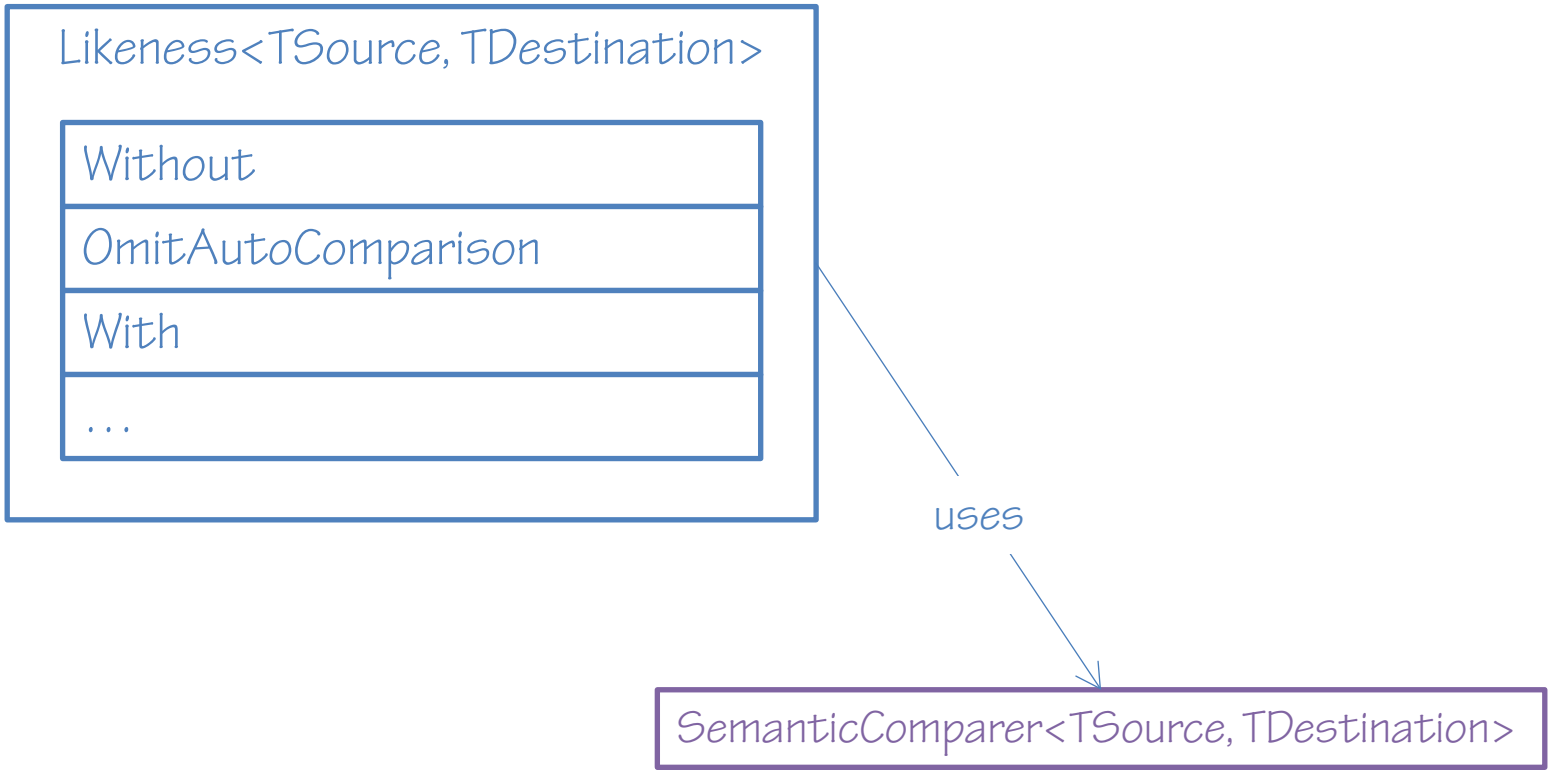
Unit test with Resemblance and extension method

```
[Theory, AutoWebData]
public void PostSendsOnChannel(
    [Frozen]Mock<IChannel<RequestReservationCommand>> channelMock,
    BookingController sut,
    BookingViewModel model)
{
    sut.Post(model);
    var expected = model.MakeReservation().ToResemblance();
    channelMock.Verify(c => c.Send(expected));
}
```

Likeness


Foo
Date
Email
Name
Quantity
Bar
Email
Name
Date
Quantity
Id

SemanticComparison



http://bit.ly/10UmRh9

Unit testing MakeReservation

```
[Theory]
[InlineData("2013-03-28", "foo@ploeh.dk", "Foo", 1)]
[InlineData("2013-03-27", "bar@fnaah.dk", "Bar", 9)]
public void MakeReservationReturnsCorrectResult(
    string date, string email, string name, int quantity)
{
    var sut = new BookingViewModel {
        Date = DateTime.Parse(date), Email = email,
        Name = name, Quantity = quantity };

    RequestReservationCommand actual = sut.MakeReservation();

    Assert.Equal(sut.Date, actual.Date);
    Assert.Equal(sut.Email, actual.Email);
    Assert.Equal(sut.Name, actual.Name);
    Assert.Equal(sut.Quantity, actual.Quantity);
}
```

Testing with Likeness

```
[Theory]
[InlineData("2013-03-28", "foo@ploeh.dk", "Foo", 1)]
[InlineData("2013-03-27", "bar@fnaah.dk", "Bar", 9)]
public void MakeReservationReturnsCorrectResult(
    string date, string email, string name, int quantity)
{
    var sut = new BookingViewModel {
        Date = DateTime.Parse(date), Email = email,
        Name = name, Quantity = quantity };

    RequestReservationCommand actual = sut.MakeReservation();

    var expected = sut
        .AsSource().OfLikeness<RequestReservationCommand>()
        .Without(d => d.Id);
    expected.ShouldEqual(actual);
}
```

Likeness as Resemblance

```
public class Likeness<TSource, TDestination>
{
    public TDestination CreateProxy();
}
```

Dynamically emitted
Resemblance of TDestination

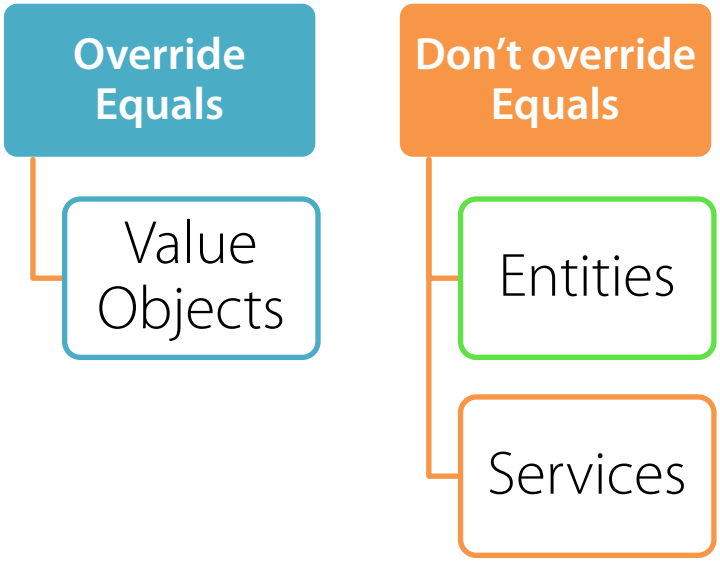
Derives from Tdestination
Overrides Equals

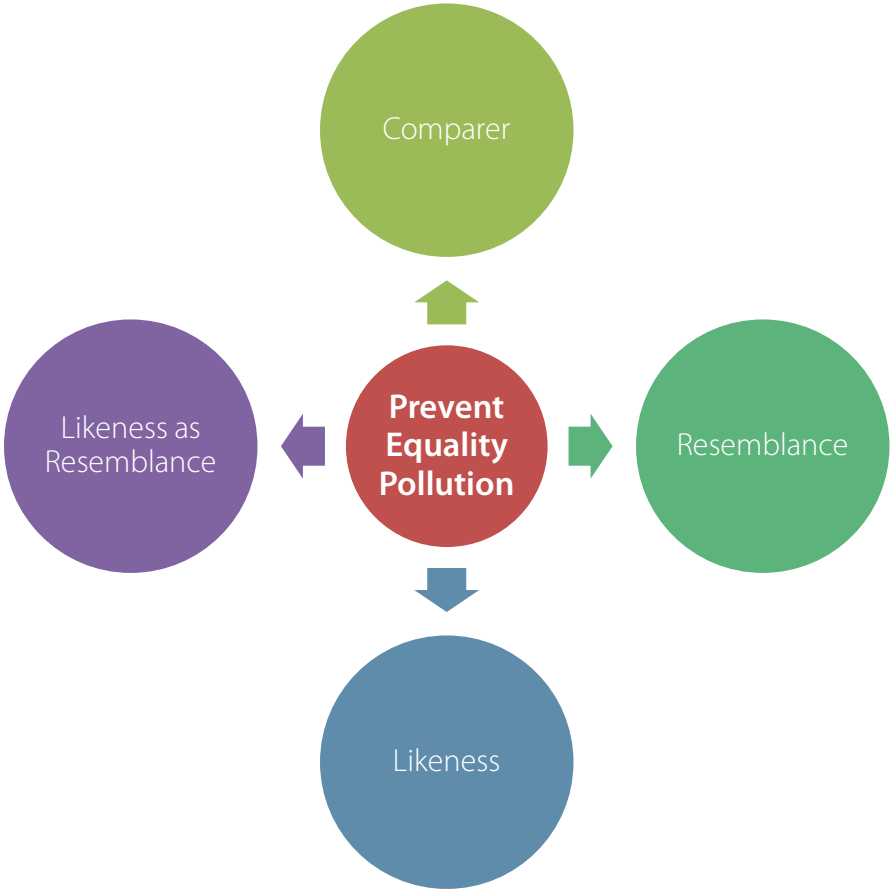
Unit test with Likeness as Resemblance

```
[Theory, AutoWebData]
public void PostSendsOnChannel(
    [Frozen]Mock<IChannel<RequestReservationCommand>> channelMock,
    BookingController sut,
    BookingViewModel model)
{
    sut.Post(model);

    var expected = model.MakeReservation()
        .AsSource().OfLikeness<RequestReservationCommand>()
        .Without(d => d.Id)
        .CreateProxy();
    channelMock.Verify(c => c.Send(expected));
}
```

Summary


Override Equals
Value Objects
Don't override Equals
Entities
Services


Comparer
Likeness as Resemblance
Prevent Equality Pollution
Resemblance
Likeness

Course Summary