

how to create an object using another as a source in Autofixture?

Asked 10 years, 6 months ago Modified 10 years, 6 months ago

Viewed 193 times



I have something like this:

1



```
public class ModelEntity : Entity
{
    public override int Id { get; set; }

    public string FileName { get; set; }
}

public class DataTransferObject
{
    public int Id { get; set; }

    public string FileName { get; set; }
}
```

And I would like to do something like this:

```
var model = _fixture.Create<ModelEntity>();
var dto = _fixture.Create<DataTransferObject>
().FillWith(model);
```

Right now I am doing the following but I am not sure if is the right way to do it

```
var model = _fixture.Create<ModelEntity>();
var dto =
```

```
model.AsSource().OfLikeness<DataTransferObject>
().CreateProxy();
```

autofixture

Share

Improve this question

Follow

asked Jun 18, 2014 at 23:27



Jairo Alfaro

343 ● 1 ● 9

1 Answer

Sorted by: Highest score (default)



2



AutoFixture doesn't have a feature like that, but I think there's something better to be learned from this:

AutoFixture was originally built as a tool for Test-Driven Development (TDD), and TDD is all about **feedback**. In the spirit of [GOOS](#), you should listen to your tests. If the tests are hard to write, you should consider your API design. **AutoFixture tends to amplify that sort of feedback**, and it may also be the case here.

It sounds like you need to be able to populate a `DataTransferObject` with values from a `ModelEntity` instance. Could this suggest that some sort of **mapping** would be a valuable addition to your API?

Depending on how these types are already coupled, you *could* consider adding a projection method to your `ModelEntity` class:

```
public class ModelEntity : Entity
{
    public override int Id { get; set; }

    public string FileName { get; set; }

    public DataTransferObject
    ToDataTransferObject()
    {
        return new DataTransferObject
        {
            Id = this.Id,
            FileName = this.FileName
        };
    }
}
```

However, the disadvantage of this approach is that it couples those two types to each other.

If you find that undesirable, you could instead introduce a dedicated Mapper Service, which can map a `ModelEntity` instance to a `DataTransferObject` object - and perhaps vice versa.

If, for some unfathomable reason, you don't want to introduce such a Mapper into your System Under Test, you can still add it as a reusable Service in your test project.

If you don't wish to write such a Mapper yourself, you could consider using something like [AutoMapper](#) for that purpose.

Follow



Ruben Bartelink

61.8k ● 31 ● 194 ● 260

answered Jun 19, 2014 at 6:10



Mark Seemann

233k ● 49 ● 446 ● 769
