Developer Tips: C# Extension Methods on Interfaces

By Andrew Hinkle

I write my code with dependency injection in mind. Given that extension methods are static and I've only ever seen them implemented on concrete classes I've avoided them for a while. While watching "[C# Extension Methods by Elton Stoneman](https://app.pluralsight.com/library/courses/csharp-extension-methods/table-of-contents)" PluralSight course I learned that you may apply extension methods to interfaces. This was reinforced with the article "[Extension Methods Guidelines in C# .NET by Michael Shpilt](https://michaelscodingspot.com/extension-methods/)". This has brought extension methods back into my toolkit.

Elton gave a great example of a use case involving design patterns. His example involved three different repositories that shared an interface that implemented a method called "GetItems". He continued that he wanted to add a new method "DistinctItems" to the repository interface, but did not want to create an implementation for each repository class. Instead he created an extension method on the repository interface. Cool!

I highly recommend watching Elton's PluralSight course for more information and advanced examples. All of his examples are followed up with unit tests and design pattern best practices in mind.

I expect a "DistinctItems" method to be performant and efficient, so at some point I'd expect the implementation to be unique in each repository. Instead I liked the idea of adding the "DistinctItems" method as an extension of IEnumerable<T> where T is an Item in the following example. Below is an implementation of an extension method on an interface. The code base can be found in full along with another example in my github tips: <https://github.com/penblade/Tips/tree/master/Tips.ExtensionMethods>.

# Project: Tips.ExtensionMethods.ExtendInterfaces

## Item

This is the model the repositories will return



## IRepository

The interface does not contain the method "DistinctItems". We'll extend the interface next.



## IEnumerableItemExtension

I liked this implementation of LINQ to get distinct items.



# Project: Tips.ExtensionMethods.ExtendInterfaces.Test

## ITestItemFactory

Even though we're in a test project, I still follow dependency injection principles and created an interface for the test item factory used in the mock repository factory.



## TestItemFactory

I created an item factory to group the test data together for easier maintenance.



## MockRepositoryFactory

This factory creates the mock repositories with the test items given the test item factory.



## RepositoryTests

Test method "GetItemsDistinctTest" tests the LINQ method implementation on its own. "GetItemsDistinctExtensionTest" tests the extension method.



## Conclusion

The code base can be found in full along with another example in my github tips: <https://github.com/penblade/Tips/tree/master/Tips.ExtensionMethods>.

While you can't mock static extension methods on concrete classes, you can mock them on interfaces and that adds another tool in your toolbox when working with dependency injection. How do you use extension methods? Did you know you could create extension methods on interfaces? How do you test your extension methods?