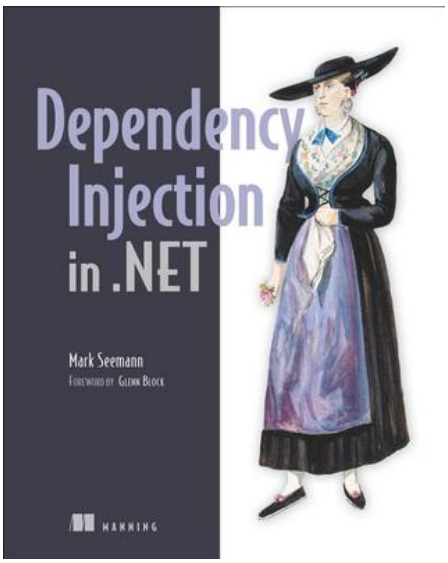

Dependency Injection in .NET

Mark Seemann

Dependency Injection in .NET

Mark Seemann

Dependency Injection in .NET Mark Seemann
Summary

Dependency Injection in .NET, winner of the 2013 Jolt Awards for Productivity, presents core DI patterns in plain C#, so you'll fully understand how DI works, covers integration with standard Microsoft technologies like ASP.NET MVC, and teaches you to use DI frameworks like Structure Map, Castle Windsor, and Unity.

About the Technology Dependency Injection is a great way to reduce tight coupling between software components. Instead of hard-coding dependencies, such as specifying a database driver, you inject a list of services that a component may need. The services are then connected by a third party. This technique enables you to better manage future changes and other complexity in your software.

About this Book*Dependency Injection in .NET* introduces DI and provides a practical guide for applying it in .NET applications. The book presents the core patterns in plain C#, so you'll fully understand how DI works. Then you'll learn to integrate DI with standard Microsoft technologies like ASP.NET MVC, and to use DI frameworks like StructureMap, Castle Windsor, and Unity. By the end of the book, you'll be comfortable applying this powerful technique in your everyday .NET development.

This book is written for C# developers. No previous experience with DI or DI frameworks is required.

Purchase of the print book comes with an offer of a free PDF, ePub, and Kindle eBook from Manning. Also available is all code from the book.

Winner of 2013 Jolt Awards: The Best Books—one of five notable books every serious programmer should read.

What's Inside

Many C#-based examples A catalog of DI patterns and anti-patterns Using both Microsoft and open source DI frameworks

Tabel of Contents

Dependency Injection in .NET Details

Date	:	Published September 28th 2011 by Manning Publications
ISBN	:	9781935182504
Author	:	Mark Seemann

Format : Paperback 584 pages
Genre : Computer Science, Programming, Science, Software, Technology

From Reader Review Dependency Injection in .NET for online ebook

Brett says

Seemann did a fantastic job presenting Dependency Injection in concept and practice with several different .NET frameworks. Perfect length and structure for a quick read, but also deep enough for my long-term reference on the subject.

Wish all my technical literature was produced like this.

Christian Dalager says

Not just a good DI Container book, but a great OO software design book, building on top of other great classics.

Even in the last chapters going through the different containers, Seemann continues to provide examples that makes you smarter.

And never have I built up an appetite like that for Sauce Bearnaise in a book about programming.

I'll keep this close to my desk.

Ireney Berezniak says

Dependency Injection in .NET is a fine book on the exact same subject of its title. It seems Mark Seemann and I have become practitioners of DI techniques in a similar manner: at first, as away of improving testability. Through exploration, this has led to a more pure DI practice, where dependency injection has become the prime focus, and not testability, though testability certainly gains in a major way from adoption of DI, of course.

As such, this title does not concern itself with unit testing. It strives to explain DI in its purest form and patterns like "bastard injection" are considered to be an anti-pattern, and rightly so. However, from another perspective, "bastard injection" is considered a valuable pattern. I've employed it for the first time in my current legacy project as a way of improving testability, and to use it as a stepping stone towards implementation of true DI, which has been impossible to accomplish from the outset due to the scope and constraints of the projects.

For the most part, however, I have been lucky to begin projects with DI as part of the architecture from the very start, or in other cases where the project has been well in progress when I joined the team, refactoring it to adopt DI proved relatively simple. I have had experience with Autofac, Unity, and Castle Windsor, DI frameworks which this book covers, and a few others besides. Mark Seeman covers the same topics of each framework, providing a clear insight into each framework and its capabilities, which is quite handy for anyone looking to implement it. Ninject, the framework that emerged as a favourite of mine, unfortunately is

not covered in this book.

Despite the fact that I am familiar with DI and have been practicing it for years myself, I often enjoy titles such as these, which help me to solidify my own understanding, or fill gaps in my own knowledge. Most of such books do not provide anything more then validation of my own proficiency, but this one has exposed something that has been missing from my awareness of DI framework capabilities: interception. For whatever reason, it have not across DI-based interception solutions, or if I did, I must have ignored them. This could be due to the fact that some frameworks, like Autofac, do not support interception, and others, like Unity or Ninject, require extensions to enable it. When I needed to address certain cross cutting concerns, I typically looked to aspect frameworks, like PostSharp. Either way, this has been the revelation of the book for me, and raised my rating from the 3 start "I liked it" to the 4 start "I really liked it"!

DHEERAJ Kumar says

This book is a rare gem, a must must read for OOP developers. Every line you read will make you smarter developer.
It will not only help you understanding Dependency Injection to the deepest, but also help you making better architecture your applications which would adhere to the principles and best practices.

I would recommend few prerequisites for beginners to make most of this book.
1. Understanding of C#
2. Basic understanding of Design Patterns.(Decorator, AbstractFactory, Facade, Singleton)
3. Understanding of OOP Concepts.
4. Understanding of Solid Principles.

Happy Coding!

Oleksandr Bilyk says

We have situation when in .NET software companies as I currently inquired (asked colleagues and read different blogs):
- 50% of people don't care about Dependency Injection
- 70% of people don't know the difference between Dependency Injection and Dependency Injection Container, Inversion of Control, Dependency Inversion. People just don't see the difference :)
- 80% of people don't understand what dependency injection is but strongly think that they using it.
- 90% of people don't understand that Microsoft Extensibility Framework usage for DI causes "Control Fake" antimatter.
- Maximum 5% know tow to use it in different environments: Console Application, Win Forms, WPF, ASP.NET Web Forms, ASP.NET MVC Rather, PowerShell, and different Java Script mixes like TypeScript + React.
- 1% of posts understand how build abstract transactions in domain layer.

Situation should be changed and at least 20% of developers in different projects should read this book better from cover to cover.

I think this book should be bigger and contain more complex samples from Domain area where people
- build layers
- .NET Core Dependency Injection Container.
- Entity Framewokr Core with Dependency Injection.
- Consider WPF Extensible Object as one very powerful pattern for DI that allows to associate contexts to objects as alternative to Ambient Context pattern.
- Compare System.Transactions.TransactionScope as not universal Ambient Context instrument for domain Transactions.
- Consider Sprint.NET Transactions as in DI context and try to build light weight alternative http://www.springframework.net/doc-1....

Anyway this book is #1 if we are talking about .NET.

Ant says

We use dependency injection at work, and while it's easy enough to follow the predefined patterns, and accepting that it was so we could unit test the code we were writing, I really had little idea as to what it meant beyond injecting dependencies into constructors of classes implemented on abstract interfaces. This book opened my eyes to what it is; an entire code lifestyle that goes beyond unit tests.

Seeman takes the reader through many well defined examples to hammer home the value of inversion of control, and how it assists in keeping to various Object Oriented design principles which in turn allow for extensible code in the ways of avoiding tight coupling.

The style of the book is light in spite of the pretty heady topic. He moves slow, ensuring you understand each principle, and reiterates points with similar examples from different angles. Throughout the book he constantly uses analogies to cooking which keeps it fun.

The book is separated into well defined sections which can be read in any order, though front to back may be the best if you are starting from absolute scratch. He also touches upon various design patterns which inversion of control makes use of.

One grudge I had was that many of his abstractions were abstract classes which were sometimes difficult to determine as such when looking at a page of code examples. While Interfaces & abstract classes can be used more or less interchangeably as the base construct for DI implementations, for example purposes I feel Interfaces would have been easier to identify on the page, but that may be my own limitations.

In all a great book which I will be returning to for reference. It clearly explains ways to overcome common issues such as circular dependencies & the best form of injection to use for a given situation. A great education and a great reference.

Mateusz St?pniak says

The amazing AutoFixture library creator shares some of his deep knowledge. I enjoyed culinary metaphors.

Nikita Skurat says

This book will be very helpful for any .NET developer (and not only .NET), even for those, who've been facing DI challenges in many projects from their experience.
It contains a very detailed overview of DI, its parts, how it can be implemented, what questions developers usually face when applying these technics. And it also contains a description of several most used containers available for free.
I found reading this book really useful for me, I could see what mistakes i did in the past and which things I must be careful with while working with DI. A must-read.

Johnny Graber says

If you are interested in the concept of dependency injection and inversion of control you definitely should read this book by Mark Seemann. He explains the architectural concepts and ideas behind in the same easy to understand way as he explains the different DI containers and how to structure your code in a better way.

The theoretical parts are still up-to-date and help you to solve your problems. The specific containers however didn't stay frozen in time and evolved. Before selecting one over the other you should check if the described limitations are still an issue.

Amy Gilchrist Thorne says

This book was pretty helpful in clearing up some of my confusions about dependency injection frameworks.

I already was aware of the basics of constructor injection, but was experiencing some pain around problems like constructor over-injection and lots of 1-1 mapping between interfaces and implementations. This book explores ways to fix these issues and many others.

Here are some topics the book covered that I really appreciated:

* DI patterns, including the different types of injection, and Ambient Context
* DI antipatterns, including Service Locator
* DI refactorings, including Abstract Factory and dealing with cyclic dependencies
* Object composition, including implementing the composition root for common types of .NET applications
* Interception

I particularly liked the way the Interception chapter went over Decorators and how to configure them, and then moved into how to implement and configure interception.

Gabriel says

I like to read books cover to cover, and as an experienced developer I found this one extremely verbose. It could have cut down the same info to half the size. If you use it more as a reference, the redudancy might be helpful.

The book did open my eyes to the concept of Composition Root, and the role of Abstract Factories that basically bridge the gap between your static object graph created at application start (the objects that would typically be Singletons in less enlightened architectures) and the runtime objects that flow through it.

The section about applying DI to various .NET frameworks is a very good reference that could save a lot of headaches. The same can be said to the chapters on the various DI containers, it's basically the most exhaustive comparison that has ever been done, you won't find the equivalent online.

Overall, due to the absence of competition I have to recommend it, but there's definitely room for a better book on dependency injection in .NET (2nd edition is in the works, who knows...).

Tuan Truong says

My programming bible.

Matt says

I worked for a while this fall changing our C# web projects (ASP.NET Web API v2) to use dependency injection. I'd flipped through this book before, but it was much more relevant now.

This is a well written book that drives home the concepts at play for DI, and traps to avoid. I found the material helpful. It helped frame how I think about the structure of the code, and the DI work itself (apart from this book) has been paying dividends.

Andrew says

Despite the second edition not being completed yet, this is an absolutely invaluable resource to any programmer interested in Object Oriented Design. The concepts are clearly communicated with focused examples that are general enough to be actually practically useful.

This book is a must-read.

Δημ?τρης Κ?κκινος **says**

A very well structured book. Despite the fact that it addresses a specific subject (dependency injection), it provides a much wider and deeper insight to the structuring of certain parts of an OO project.
It paints the whole picture of dependency injection (composition, lifetime management, interception) with a lot of examples and many principles to support them.
Additionally, it pays off to have this book as a reference, because it provides a lot of material for many well known DI containers.
Also, it has a whole section on how to avoid bad practices in this domain, which helps to get a better idea and develop intuition about dependency injection.
