# Tips for Writing a Programming Book

In this article, I want to go through the process of writing a book, to give others who are thinking of writing a general idea of what kinds of things you will need to do. This is the article I wish I had read before starting out. Some of this will be applicable to fiction writing, but there are better resources if that's what you're interested in. I can only share what I personally know.

## Before-You-Start Checklist

Here's a short list-based summary of many of the things I'll discuss in this article. These are all things to think about before you start writing and throughout the process:

1. Do you have enough material for a book? Write out an outline to make sure. Iterate over it until it's quite detailed.
2. Quash self-doubt! If no one else has written this book, then it's up to you!
3. Do you want or need to go with self-publishing?
4. Who will be your editor? (You do need one.)
5. Should you do a print edition? What's the market for that?
6. When are you going to make the time to write?
7. Do you have a place you can write uninterrupted?
8. Will your family support the time commitment?
9. Does your day job have a moonlighting policy? Follow it!
10. Don't use DRM!
11. Familiarize yourself with the various publishers' processes. Do some dry runs. Figure out what's going to work and what's not.
12. How are you going to market your book? Get started on laying the groundwork for that early. Start tweeting and blogging in anticipation.
13. When marketing your book, remember to Always Be Providing Value.
14. Price sanely and fairly. Chances are that in unless you're fulfilling a unique, high-in-demand niche, you're not going to make a lot of money, certainly not enough to quit your day job. Optimize for the number of sales, and be competitive with others in your genre. The satisfaction of producing something meaningful and sharing your knowledge with others is worth the effort you are putting in.

Writing a book is not easy. Just because you know stuff does not mean you can write a book about it. But it

is a prerequisite. I'm the author of two non-fiction computer programming books. The first, C# 4 How-To was published in March 2010 by Sams. My most recent book is Writing High-Performance .NET Code. This was self-published, for reasons I'll get to later.

After the experience of writing my first book, I promised myself and whoever would listen that I would never write another book. The process was too grueling for the payoff. The money just isn't there. Sure, you do get some, but if you have a well-paying full-time job, it's a drop in the bucket. You should not write books for riches–chances are you won't get them.

The rest of this article is part advice, part journal. Get out of this what you will.

## Inspiration

At my day job in Bing, I've spent the last few years becoming something of an expert in the area of .NET performance. I wrote a very significant part of the Bing query-processing stack in .NET and performance is obviously a vital consideration in everything it does. I had the benefit of an amazing mentor (see my book's acknowledgements [PDF]) for a number of years, and the experience we gained as a team was considerable. Many of the bits of knowledge we had to apply were things that were not available via Internet search, or if they were, lacked so much context that it was worse than useless.

One day, at a dinner with my brother-in-law, we got to talking tech and our work projects in generalities, and he casually mentioned that I should write a book about these performance lessons. This wasn't the first time the thought had come to me, but this time I actually thought it through for a while. I mulled on it for a few weeks before writing out a simple outline, at which point I knew that I had enough material for a book.

I wasn't sure how long it would take, or what my coworkers would think, so I kept it to myself and close family.

## Doubts

Everyone has self-doubt. I certainly did. Through much of the writing process, I constantly wondered to myself whether I should be the person writing this book. I am not on the CLR team; I work on a specialized project; I'm fairly young compared to the experts I look up to: who am *I* to be writing a book on .NET Performance? Et cetera.

There is a fairly common psychological condition called Imposter Syndrome, which explains these doubts. Basically, most people at one point or another feel like a fraud in their success.

The fact is, I realized, that someone ought to have written a book like I was, and that someone might as well be me. As long as I got good editing, feedback, and technical help, there was no reason I could not produce a very high-quality textbook on .NET performance, despite my worries.

So I pressed ahead.

Another doubt was the scope of the book. My knowledge is heavily weighted towards vanilla servers, pure-.NET. No WCF, ASP.Net, WPF, etc. I am familiar with those, but they're not my bread-and-butter, so to speak. Would people read a performance book that didn't cover the trendy buzzwords?

I decided that they would. There was no book that covered the fundamentals of .NET performance engineering in the way I was going to. This was an important niche, and it was actually an advantage to not cover the higher-layer technologies. It gave the book laser-like focus, and allowed it to explain the fundamental techniques in a way that a widely scoped manual would lose. There was no book that explained the fundamental costs and benefits of using .NET from a performance point of view, regardless of code library.

It was like a really good cook book that would teach you fundamental cooking techniques, but the book would not have thousands of recipes. Once you had the techniques, you could apply them on any recipe from other books.

## Why Self-Publishing

I probably could have gotten a publisher if I had asked. I had already published a book via Sams, and while it wasn't a best-seller, I did earn back my advance and a little more. However, royalty rates truly are abysmal. 10% of the wholesale price is fairly typical. Wholesale is usually 50% of the cover price. So for a $40 book, you can expect to earn about $2 per copy, less on an eBook. The advantage of having a publisher is that they will get you into physical stores, which is where I saw most of my sales for my first book. However, that was 5 years ago. The market has changed dramatically in that time. Amazon is the king of book selling, by a LONG shot.

So I went with self-publishing for a number of reasons:

- I could earn much higher royalties, even if I priced the book much lower.
- I knew someone who could be technical editor.
- I have a couple of editors in the family who could help with grammar and style.
- I had experience and so knew basically what kinds of formatting I would need, what parts of the book I would need–all the technical details.
- I was willing to do all of the pre-publication grunt work myself.
- I have an artist in the family who could do a cover for me. Heck, even if I didn't, I know Photoshop. I could probably whip up something better than most of the bland covers that seem de riguer for programming books these days. Admittedly, my sister did a *much* better job on the cover than I could have.

Self-publishing has definitely paid off for me. Benefits I've noticed:

- A bigger sense of accomplishment. I didn't just write the book. I managed it from beginning to end. I made it a business.

- I feel much more in control of my own product. I can update it at will. I have more direct contact with many readers on Twitter and other places.
- I'm not going to quit my day job anytime soon, but in the first three months of my book being out, I made more money than in all 4 years my previous book was on the market.

## Writing

The hardest part of any large project is just getting started. The first thing I did was flesh out my draft outline to list all of the chapters I thought I would have, then within those chapters, individual topics.

Then I just started writing. I don't remember if I started with the Introduction or Chapter 1, but I did start more-or-less at the beginning. At the start, I did not concern myself with grammatical correctness, precise descriptions, or even necessarily getting all the details I wanted. I just tried to get all of the ideas onto the page. As I thought of related ideas, I would write them into my outline, or later in the document itself with a TODO prefix and an explanation of what I was thinking.

I pretty much wrote the whole book linearly like this. Not perfectly, of course. I jumped around a bit as necessary. Sometimes, sections belonged in a different order, or in a different chapter entirely. When I got to a part where I knew I needed a code sample, sometimes I just wouldn't be in the mood to work on code or the debugger, so I would put a TODO in the chapter with an explanation of what the code needed to do. Then I moved onto writing something else.

One of the things I had to decide fairly early on was my writing style. Was it going to be "folksy" or "clinical"? How much like a textbook did I want it to read? My natural style is fairly informal. I started writing the book just like I write a blog entry, but I realized, especially later during editing, that while this is mostly ok, you do need to tighten up the writing a little bit. Some things just require more clarity or a precise explanation.

Another important decision I made early on was to take a stand, be opinionated in what I recommend. So much programming documentation is clinical and does not take a firm stand on what you should do in a given situation. I wanted to make sure my own personal voice rang through the text.

This isn't just about the language and grammar being used–it applies especially to the content. In a blog entry, if you don't want to explain a piece of prerequisite knowledge, you can just link to an existing source somewhere on the Internet. Book readers do not want that. If there's something they need to know, you need to provide it to them (within reason of course–it's fair to state knowledge assumptions up front).

The increased need for formality also forces you to really get your facts right. There is nothing like teaching others to really get you to learn the material well. This is true for EVERYTHING, but especially if you're writing a book. There is a huge difference between knowing in your head how something works, and being able to write it down in a coherent way. Of all the chapters in my book that I knew I had to get right, Chapter 2 – Garbage Collection is the one that needed to be the most perfected. I knew garbage collection details pretty well. I have nearly weekly interaction with the people responsible for maintaining and developing it, but it was still surprising how many nit-picky details I needed to tweak after going through the content with the

owner of the GC.

I did not have daily or weekly writing goals at first, but as I got towards the end, I did start challenging myself to writing a thousand words per writing session. This can actually be pretty tough when you realize you need to do more research, or write a code sample. Those take time. My only goal was the rough timeline. I knew I wanted to publish in the summer of 2014, so I planned out what had to be done by when, and gave myself about two months for editing and finalization.

Resist the urge to pad your text. This is really easy to do in a programming text. Just throw some more code on the page, add some extra topics, and you can create pages out of thin air! Don't do this! Yes, find good, relevant content that *should* be in your book, but only add it with the right motivation. It's unlikely your outline will contain everything that needs to be in the book—you will certainly forget something. Do research, brainstorming, take a step away from the project, ask your technical editor what's missing, etc. Add content for good reasons that fit the scope of your book and leave it at that.

## Tools

I used Word 2013 for the entire process. This has its pros and cons.

Pros:

- Advanced formatting abilities. (It's not perfect for advanced layout requirements, but for a book like mine, it was sufficient.)
- Great for producing a print-ready PDF.
- Great collaboration abilities, especially via OneDrive and Office Online.
- Ability to edit anywhere I was, on any computer.

Cons:

- eBooks support very limited formatting. In particular, the Amazon Kindle conversion was a nightmare. I had to remove a LOT of formatting. (Tables and multiple fonts per paragraph are the two big ones that come to mind.)
- When it came time for the EPUB conversion, which is used for every online retailer except Amazon, it required an entire day to clean up the HTML from the conversion process. I had to fix a lot of styles and makes tons of tweaks. To the point where it is now easier to make corrections in multiple documents rather than make them in the master document and then redo the conversions.

I stored the book on OneDrive while I was actively working on it, which allowed me to edit anywhere I was. When it came time to collaborate with editors, this made it very easy to share with others who could add comments directly in the document. I made weekly, sometimes daily, backups to another drive at home, which was further backed up by Carbonite.

For screenshots, I used the built-in Windows utility Snipping Tool. This mostly worked, but you have to keep

in mind that print is 300 DPI, while most screens are 96 DPI, which means the images are smaller than you think. In some cases, I used Photoshop Elements to increase the size of and resample small images to make them suitable for printing.

For code samples, I used Visual Studio 2012 Ultimate. I used Subversion on my local Synology NAS to store my code samples. Once the book was done, I also added all graphics, manuscripts, and all other book artifacts as well. The Synology has a publicly accessible DNS that I could use from my laptop away from home.

For eBook editing, I used Calibre, which has both library management and eBook editing tools.

When it came time to upload to online retailers, I first ran EPUBCHECK on the final EPUB files to ensure they passed with 0 errors.

## Time Scheduling

Given that I have a full-time job and a family with a small daughter, as well as other commitments (including a musical, where I was playing in the orchestra), finding time to work on the book was a problem. My wife and I instituted a “night off” every week to work on personal projects—after dinner, we have no more responsibilities for the rest of the night. We don’t have to help with dishes, bath time, reading stories, or interacting at all. I utilized my night for working on the book. I say goodnight, kiss and hug my daughter, and then I can isolate myself in the house or go somewhere else. The time was 100% my own. Typically this amounted to about 4 hours. For the rest of the week, I would be lucky to get an hour or two a night. But the 4 hour block really helped–it was enough time that I could focus on big issues.

Towards the end, perhaps the last month and a half, one night a week wasn’t sufficient either. There was so much to do that I had to start taking most of all my Saturdays to work on the book or I would never get done.

In the end, the book took about 10 months, start to end, and that was mostly working nights and weekends. And the book isn’t even that long! Don’t underestimate how much work it is.

## Setting

For writing, by far the best place for me to work was somewhere outside the home that had some kind of ambient noise. I worked at the local library, Starbucks, and even McDonald’s a couple of times (they were open later than the others). There is something about the noise level that can help you concentrate more than being at home. If you have to work at home, you can try any number of ambient noise tracks on YouTube.

The times where I did work at home, I usually put in headphones and listened to classical music.

For code and debugging samples, it was much easier to just do that at home where I have two large

screens.

## Editing

I am lucky. I knew someone who would be a great technical editor. He was on the CLR team before joining my team and he became a great mentor. I asked if he would be my technical editor, and he readily agreed. This meant that my technical content was in good hands. He wouldn't let anything egregious slip by.

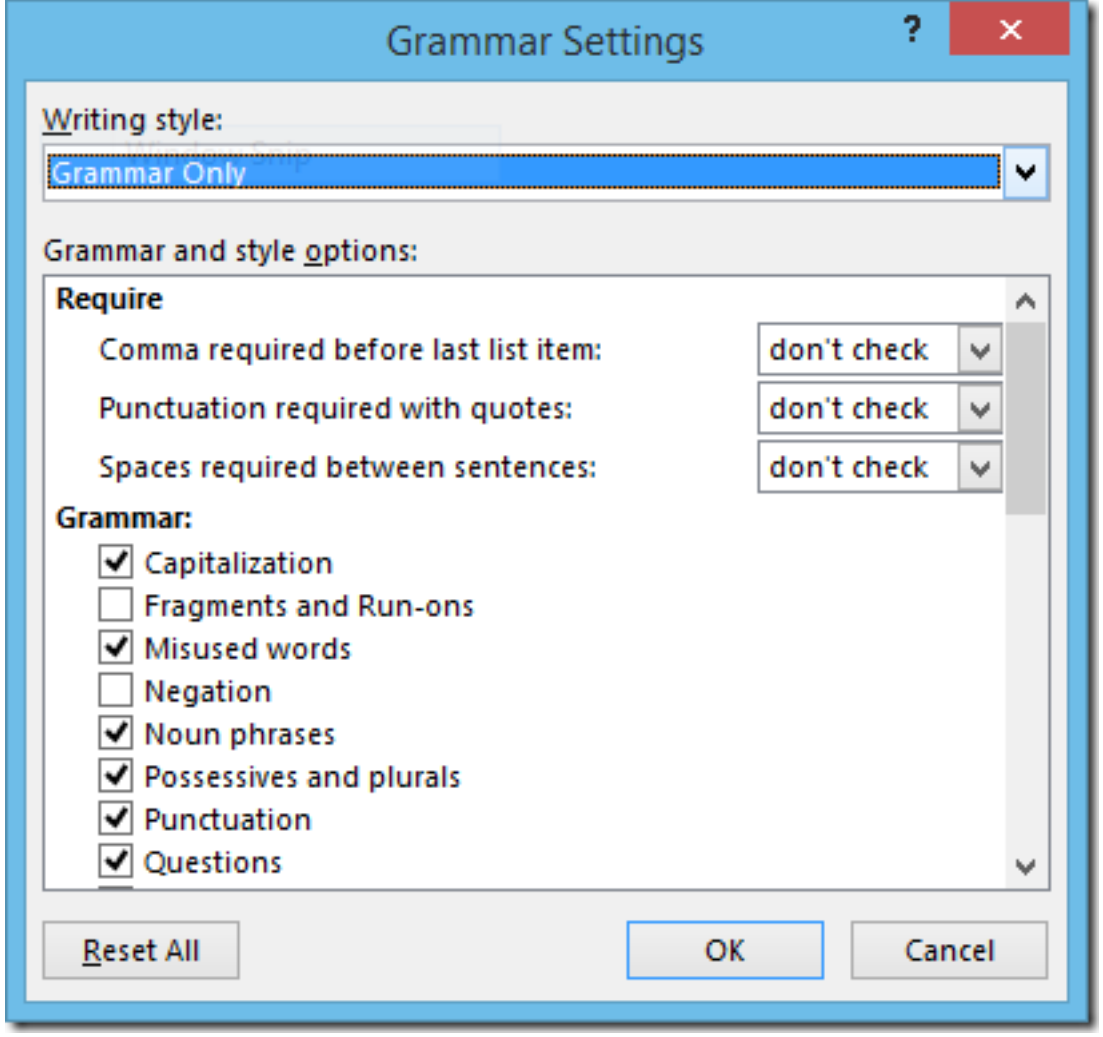
Since I also had a relationship with people on the CLR team, I asked some of them to proofread smaller portions of the book, in particular Chapter 2, which I consider the most important part.

Once I was done with the first draft, I gave the whole thing to my TE. After he had it for a few weeks, we got together in person and walked through a major portion of the manuscript. I came away with about three pages of notes. Everything from minor technical errors, need for a source, a better wording, to a better way of summarizing content in the chapters and at the end of the book. Not all of them were big changes, but some were. A couple involved major restructuring. This was about a month worth of work.

For grammar, style, and formatting, I asked my wife and father to review. My wife is an amazing editor. She has worked at a couple of jobs that required very precise abilities in analyzing documents, so she was the perfect person for this job. I also asked my dad to take a look as well, and between them I got a lot of good feedback.

To allow them to edit, I shared the document with them via OneDrive and asked them to only leave comments—not change the body text. Once I resolved their comment, I deleted it.

One other thing I did was a bit a crazy. I turned on ALL of Word's grammar suggestions and analyzed the final manuscript. Don't do this unless you like pain. If you do like pain, then go to Options | Proofing | When correcting spelling and grammar in Word | Settings..., which brings up this window:




Turn on the settings you want and prepare to be nitpicked to death. I turned it off after a while, but it was helpful, and helped forestall some comments from my wife.

One of the most common things that was corrected by my editors was the use of contractions. If you look in the final book text, you will find very, very few contractions. Contractions are a very informal way of writing. Fine for a blog entry, but in a book, they do stand out a bit. I only left them in when they sounded better.

There are dozens of nitpicky details you have to take care of. It's hard to think of them all right now, but I strongly suggest you look at another book in the same genre you're thinking of writing and analyze all aspects of it, especially the parts that aren't just paragraphs of text. Things like:

- Heading Capitalization.
- Number of heading levels.
- Structure of lists. Sentences or not? Periods at the end or not? My rule was that each list had to be internally consistent, but not necessarily consistent with other lists. Catching all of these can be hard.
- Sentences that are too long, too short, oddly phrased, confusing, etc.
- Colloquialisms.

A lot of this stuff you will be absolutely blind to. You MUST have another person help you find a lot of it. It's also helpful if that person is familiar with the genre or is at least well-read and understands the language very well. Don't ask your buddy who reads < 1 book a year to proofread.

## Formatting

If you're writing a book that is guaranteed to never be printed, then your primary concern with formatting should be how little of it you can get away with. Use one or two fonts. Use as few header styles as possible. There are plenty of formatting guides out there. One that I read and was very helpful was Mark Coker's Smashwords Style Guide.

Right up until the first week of publication, I was planning on keeping this eBook-only. As soon as I told people it was available, though, some people asked about a print edition, and I realized I would need to do that.

Formatting for print is significantly more work. I wanted a number of things:

- Bold, obvious styles for chapter titles, section headers, and more. Some of this includes underlining.
- Graphics with captions throughout the book.
- Different font and background shading for code samples.
- Different font for code elements cited in paragraphs.
- Appropriate white space after tables, code samples, section titles.
- Appropriate white space before section titles.
- Special callouts and borders for anecdotes and tips.
- Page numbers, with alternating sides for even/odd.
- Page headers consisting of the chapter number and title.
- Chapters always start on the right side (I had originally planned on some graphics that bled to the edge, but this would require a more expensive printing option, so I abandoned it)

Word was great for all of this. If you're not familiar with these features, it can take a bit to make it work, and it probably doesn't have the power of a true desktop publishing application, but it worked great for me.

There are a lot more things I could have done. Take a look at another programming book and check out the advanced formatting and features they include. Most of it is unnecessary, but it can add a certain flair.

## Cover Design

I knew almost from the beginning that I wanted the cover to have gears on it (see the introduction to the book for an explanation why), and have a very distinctive look. Most programming books are very bland. I wanted mine to stand out and still be professional-looking.

I am fortunate to have an artist sister, Claire Watson, who was willing to do some Photoshop work for me. She has a great eye for design, and after I told her my general ideas, gave her some sample stock photos, she produced no less than 30 variations of multiple cover designs. The one I ultimately chose was a variation of her very first design, and I love it.

You can find artwork and jewelry by my sister Claire at http://www.bluekittycreations.co.uk/.

## Code Samples

I used Visual Studio 2012 Ultimate for the code samples. I could have used the 2013 version, but decided 2012 was a safer bet for minimum hassle for the majority of readers.

There were a few challenges with code samples:

- Keeping them extremely short while still demonstrating the point I'm trying to convey.
- Line length, especially on an eBook is an issue. Small screens and limited resolution play havoc with code formatting. If you're on a really small or old device like an iPhone or original Kindle, then sorry, I did not bother trying to format for your device—it doesn't even make sense at that point. I optimized for devices like the Kindle Fire and better.

## Moonlighting Policy

Before I completed the book, I checked and double-checked the current moonlighting policy of Microsoft, which does allow me to write and publish books without prior approval. The only thing that can get me in trouble is revealing proprietary information, and I was very careful about this, especially when talking about .NET and GC internals (I had the GC owner review the whole chapter, and she did ask me to remove some small details that are prone to change in the future). So no worries here about publishing.

## Prepping for Publish

If you thought you were done when the manuscript was complete, edited, and finalized, then you are in for a surprise. Depending on how complicated your manuscript is, you still have days worth of work ahead of you, so plan accordingly.

At some point I set a hard deadline for myself: July 12. That was the day it was going to go live on at least Amazon.com. The weeks before that, I basically worked every night and all weekend to complete as much of the final edits that I could and get final feedback from my editors, both technical and grammar. Then I took off work on Thursday and Friday to work on the eBook conversion. I knew this was going to be a chore, but I had no idea how grueling it would be. I understand now the value that professional publishers, editors, typesetters, etc. have. I decided to do the Amazon Kindle conversion first. For this, I just needed to upload the Word document to Amazon and their system would check it and provide a list of errors to me. I could also proofread it on various Kindle simulators, which would attempt to show me what it would look like. This was very helpful.

The single biggest problem from this process was that paragraphs with different fonts in them did not render properly. I used different font styles for code keywords within body text and this was throwing off the whole system. So I had to go fix a bunch of styles in a copy of my manuscript. I didn't want to modify the original document because I knew that I would want those different styles for EPUB and print.

I also quickly realized that tables, despite being supported in all eBook formats will not work except for the most trivial of tables. It just doesn't look good. This was too bad, because I had spent quite a while converting some bullets to tables. I spent a few hours converting back.

It was invaluable going through the book on the various Kindle simulators. I got to see a lot of things that just didn't work on a small screen. It's always a challenge putting a well-formatted book into an eBook, especially one with tons of code samples. Some Kindles displayed the shaded backgrounds, others didn't. None will use different fonts, so you don't get the benefits of uniform spacing for code samples. All of this I just let slide–I figure if you're reading a coding book on a Kindle, you kind of know what you're getting. I didn't even bother proofreading (beyond curiosity) on the older Kindles or iPhone Kindle App. You deserve what you get if you try it. Sorry.

By Friday night, I hit the Publish button on Amazon KDP. A few hours later, it was up for sale.

On Saturday, I went through the EPUB conversion process. This went a bit faster than the Kindle process, but was still a little bit labor intensive. I used Calibre to convert the Word document to EPUB format, which is really just a ZIP file containing HTML, styles, and images. Then I used in the Calibre eBook editor to modify the HTML as necessary to fix up styles and formatting inconsistencies. This took a few hours and required a bit of hand-editing of styles and individual locations throughout each chapter.

Once all of that was done, the EPUB could be uploaded to all of the retailers besides Amazon.

By the end of Saturday, I was published on all major eBook platforms.

On Monday when I announced it to my work colleagues, a bunch of people said they wanted a print edition, which up until this point, I was not sure I was going to do. But given that response, I instantly realized that a dead tree version of a programming book is still in-demand. It's the code samples—some people just prefer a physical book for code.

So I spent the next week getting it ready for publication via CreateSpace. Basically, this was just ensuring that all of the formatting I wanted (see above) was present and correct. My first proof copy actually had some bad fonts for most of the code samples–I had selected Courier New, and it just didn't look as good as Consolas, so that was the major fix before publication.

I ordered a single proof copy, made the font change (and a few other minor fixes as well), and then hit the publish button. Within a day or so it was on Amazon for sale. A couple of days after that, it was matched up with the existing Kindle book so people could easily see both editions. (The matching process is normally automatic, but if it doesn't happen for you, you can contact KDP support and let them know the ISBN/ASIN numbers and they can match them up for you.)

A note on DRM (Digital Rights Management): **Avoid it**. Unfortunately, your book is going to be hacked, stolen, put on Pirate Bay, whatever, anyway. DRM only punishes the law-abiding consumer. Just don't, and make peace with the fact that people will steal things. If you sell to organizations, then it is fair to ask them

to pay for multiple copies for multiple people, but don't try to enforce this. It's a losing battle and isn't going to help your bottom line much anyway.

## Pricing

By the end of Friday, I had the Kindle ready to go. I just had to decide on markets and pricing. I went with $9.99 because that's the maximum price you're allowed to do on KDP and still get a 70% royalty rate. This was actually disappointing to me. For me, the sweet spot in pricing would have been around $15. This is competitive with existing programming eBooks while being slightly cheaper. I had a fear that if I priced it too low, I wouldn't have credibility compared to more expensive offerings. I don't worry about this now. The book has taken off and gotten enough reviews that it can stand on its own, and the price now only helps.

For the print edition, I set the price much more in line with current offerings (slightly cheaper, of course). I also made EU prices similar to other books in those markets, but I have since changed them to be tied to US price and the exchange rate. This makes my book VERY competitive in those markets, and it has definitely netted me more sales.

## Marketing

I tried to have a comprehensive marketing plan, but I don't know how well I followed it. There were so many things to do that a lot of things just fell through the cracks for weeks. However, when you self-publish, time doesn't really matter. Yes, more sales in a short period of time can lead to follow-on sales as the book will get featured in some places, especially on Amazon, but in the long run, you can try lots of things and let your book's demand naturally grow.

Here is a short summary of what I did:

- Reddit – I had no idea how much good this would do for me, but it really did. I didn't just post a link to my book and beg people to buy it—that's guaranteed to fail on any social media site, but especially on Reddit. Instead, I discussed my own background and motivation for the book. Just those facts contained enough interesting tidbits that people wanted to know more. My biggest single-day spike in sales came from that thread. Plus it led to a lot of interesting discussion. The point here is that I tried to provide value external to my book.
- Twitter – I was never a big Twitter user, and I'm still not, but I decided to make some efforts and reach out to people, participate in discussions, comment on tech news, share personal tidbits, and more. Yes, I plug my book, especially retweeting other people's comments, but I don't make my Twitter feed exclusively book plugs—that's annoying. Remember, try to provide value.
- Free Samples – The book's site contains a PDF of the book up through Chapter 1. People can see the full Table of Contents and get a feel for what's in the book, without me giving it all away. I believe a free sample is critical. I even went further, releasing the entirety of Chapter 5 (Class Design and General Coding) as an article on CodeProject. It won Best C# Article for August and is ranked as one of the Top 5 articles posted on the site.
- More articles on CodeProject, such as this one about object allocation. That article starts with some-

thing that's in the book, but I provide a lot more information and examples that are not in the book. This article won Best C# Article of September. (Sadly, I didn't have an article for October, but I'll work on another one soon.)

- Letters to influential bloggers and podcasters. I knew this was a long shot. They probably get product and media pitches all the time, right? But a few did respond. Some written reviews will hopefully show up on some influential blogs. I was a guest on the fabulous .NET podcast, .NET Rocks in episode 1041. My book was also mentioned in This Week on Channel 9.
- Facebook Page – an easy way for people to follow updates on the book, reviews, articles, blog entries. It's not a super busy page, but it's a useful central place for making announcements.
- Ads – I tried advertisements on Google, Bing, Facebook, and Twitter, with very limited effect. I mostly tried out of curiosity, and shut down these experiments fairly quickly after it was clear they weren't doing much.
- Letters to family and friends – I sent an email about my book to literally everyone in my address book, not asking them to buy it, but to forward to programmer friends or colleagues of theirs.
- Ask for reviews – I'm not shy about it. Anybody who I know read it and loved it, I will ask them to leave a review on Amazon. These make a difference. I hear from a lot of people on Twitter, and I'll give them a few weeks or so, and then ask them to write a review. I never tell people to leave a five star review or give any other guidance. More reviews the better.
- Book's website – Where to get all information about the book, a complete retail location listing, a place to buy the PDF or EPUB directly from me, errata page, download the book's source code samples, and more. This site is important.
- Posters – I printed 50 copies of a small poster of my book and put it up on every floor of my building and two others near me. I'm not sure what effect this had. Probably not worth it.
- GoodReads – I made sure my book was in GoodReads, with the right cover, editions, descriptions, etc. You can add all of this yourself.
- Blogging – I started blogging again, before the book came out. This provides another outlet for providing extra value as well as places to mention the book.

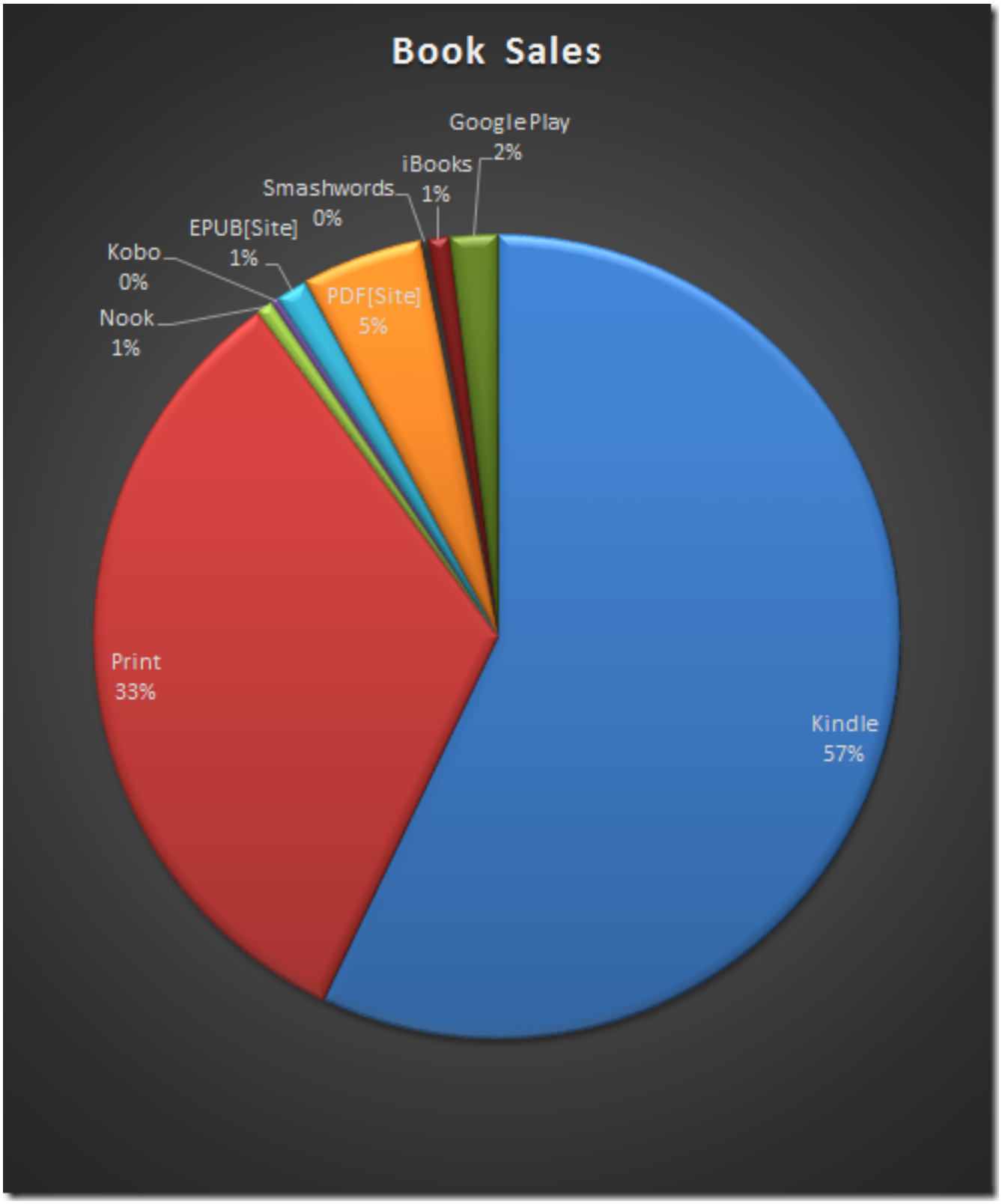
Remember to go beyond just "Please buy my book" – Always Be Providing Value.

In the end, the buzz from the podcasts, and other online reviews that have popped up have definitely helped, but most people are finding this book just through searching Amazon. My book comes up when you search for ".NET Performance" and that's probably the most effective thing for lasting sales performance.

## Sales Results

I put my book up for sale literally everywhere I could: Kindle, CreateSpace (for print), Kobo, Nook, Smashwords, Google Play, Apple iBooks, and my own site in EPUB and PDF. Smashwords, in turn, distributes to other retailers. I wanted it available everywhere, and it mostly is. Through CreateSpace's extended distribution network, it even goes to some physical stores I believe–I know I've seen some fairly decent-sized orders for that network.

But looking at the numbers, it's interesting just how much Amazon dominates the field here. Here are the percentages:




If you add Kindle and Print (both primarily on Amazon), that's nearly 90% of all my sales coming from Amazon. The next-best thing is the PDF from the book's site. You might be tempted then to only publish on Amazon, but I think this is a mistake in the long run. Even though the percentages are small, I do get sales from the other channels. There is a psychological benefit as well—many people are happy just to see that it's available everywhere, even if they buy from one of the usual places.

Also, don't forget the international market. Nearly all of my sales via my own website are from overseas customers who can't easily use one of the big eBook retailers. By providing an EPUB and PDF to them, they can use the document however they need.

## Contact

If you have any questions about this whole process, please just let me know either in the comments or on Twitter.

---

Check out my latest book, the essential, in-depth guide to performance for all .NET developers:

**Writing High-Performance.NET Code, 2nd Edition** by Ben Watson. Available for pre-order:

- Amazon
- and more, see book site

This entry was posted in .NET, Books, Personal and tagged .net, Bing, book, editing, kindle, publishing, Tips, writing on November 10, 2014 [http://www.philosophicalgeek.com/2014/11/10/tips-for-writing-a-programming-book/] .

## 7 thoughts on “Tips for Writing a Programming Book”

important site
November 12, 2014 at 1:20 pm

You actually make it appear really easy with your
presentation but I in finding this matter to be really one thing which
I think I might by no means understand. It sort of feels too complex and extremely
broad for me. I am looking forward for your subsequent
post, I will attempt to get the hold of it!

Whitecape
April 29, 2015 at 11:35 pm

Thanks for sharing your experience, it was an delightful read. Your book is definitely a product of meticulous planning and toil on your part, and your friends and family’s contributions as well.

Adnan

January 4, 2016 at 8:31 am

Hi Ben

I have a question – my friend is thinking about writing a book on Games using Visual Basic 2012 . So will he get in trouble when he publish his book without giving notice to Microsoft Visual Basic after all he's going to use their Software and name. – You actually inspired my friend so he decided that he's going to write a programming book soon. Thanks

---

Letladi

August 10, 2016 at 10:09 am

Thank you for sharing your expriences. This was an enlightening read!

---

Aleks

January 28, 2017 at 2:44 pm

Thank you! That's very supportive!!!

---

Pingback: Seven Super Ways to Turn Programing Prowess into Passive Income – Endorsse Blog

Murv

May 10, 2017 at 3:46 am

Hey, sometimes I get a 500 site message when I browse this website. Just a heads up, regards

---