

# PeepCode Author Guide

*A workflow built for developers*

by Geoffrey Grosenbach

## CONTENTS

|    |                              |    |           |
|----|------------------------------|----|-----------|
| 4  | Royalties                    | 20 | Workflow  |
| 4  | Sales Goal of 1,000          | 20 | Proofs    |
| 5  | Technicalities               | 21 | Revisions |
| 6  | Practical and Timely Content |    |           |
| 6  | Practical Content            |    |           |
| 7  | Informative Communication    |    |           |
| 9  | Entertain                    |    |           |
| 10 | Concerns                     |    |           |
| 11 | Markup                       |    |           |
| 11 | Sample                       |    |           |
| 13 | File Organization            |    |           |
| 16 | Titles                       |    |           |
| 16 | Paragraphs                   |    |           |
| 17 | Lists                        |    |           |
| 17 | Tables                       |    |           |
| 18 | Other Standard Tags          |    |           |
| 18 | Special Tags                 |    |           |

## QUICK START

PeepCode Press (<http://peepcode.com>) is a boutique publisher designed from scratch to provide you with a smooth workflow for producing practical, fabulously styled PDF minibooks.

Here's a summary of what we'll talk about:

- You receive a 50% royalty on the purchase price of PDFs and screencasts that you write. We pay a \$2,000 advance when the book is published and quarterly royalties thereafter.
- We want short, useful, focused material. Length is unimportant. We'll add diagrams and graphics to enhance the communication.
- Use Textile to write your document. Put code in the `code` directory and reference it from your book.
- You'll commit content to our Git repository and we'll generate drafts periodically.

We eat our own dog food. This document was produced using the exact workflow you will use to write your book.

# Royalties

## CHAPTER 1

We pay 50% of the purchase price, after payment processing fees have been subtracted.

## Sales Goal of 1,000

Our goal is to sell at least 1,000 copies of your book (a reasonable target after a few months of sales). If this happens, you'll receive about US\$3,000.

The first \$2,000 of this will come when your book is published, and the remainder will be paid quarterly in January, April, July, and October.

Most PeepCode books sell more than 1,000 copies, and some have sold 3,000 copies.

PeepCode products can be purchased in several ways:

- Full retail value for \$9
- As part of a 5-credit or 10-credit purchase (discounted)
- As part of an Unlimited subscription (access to all PeepCode content for a year). You receive a 50% royalty based on the average number of items downloaded by subscribers.
- Bulk sales to companies, teams, schools, etc.

# Technicalities

- A limited number of copies will be given away for free in order to promote the book. No royalties will be paid from these copies.
- Some copies will be sold through affiliates that take a percentage of sales. Authors will receive 50% of the revenue that PeepCode receives after paying affiliate fees.

Royalties are paid quarterly (every 3 months). We're also working on a real-time reporting system so you can track sales of your books.

## Updates

Updates and revisions to the same book are provided to paid customers at no extra charge. There is no obligation to update or revise your book after the initial release, however, it often helps to sell more copies if the book is kept up to date.

## No DRM

We do our best to combat piracy, but it will inevitably happen. We strongly believe that legitimate paid customers should not be penalized for the actions of pirates, so our policy is to avoid using DRM or other technologies that handicap legitimate customers from experiencing the content.

However, it should be noted that PDF books will be saved in a non-editable way in order to comply with the licensing agreement for some of the typefaces used.

# Practical and Timely Content

## CHAPTER 2

Anyone can ask you to write about a topic, save it as a PDF, and sell it. Our aim is to do more. We want to make 50 pages that are as useful as 200 pages. We want them to be practical, informative, and entertaining, in that order.

We're not trying to build a system where anyone can upload a PDF. We want to publish only high-quality content from capable authors. In addition, we will contribute graphic design, diagramming, and editing to every book.

We're not concerned whether or not you have written a book previously. Show us a few blog posts you've written or send a few pages of your proposed book and we'll be glad to take a look at it.

Although we may add multimedia to some books, you will never be required to produce audio or video unless you want to. You will also be given the opportunity to comment on content that we add to your book before it is published.

## Practical Content

Tech books are a financial investment for most developers. They buy books in the hope that they will learn skills that will help them complete a project, get a raise, or land a better job. They want to spend \$9 and learn \$90.

Your job is to teach them those skills. A developer once wrote "I'm hoping to finish this project and you mentioned that your next screencast will cover *[a programming topic]*. Will it be released before Friday?"

Although most developers will not schedule their release cycle around the PeepCode publishing calendar, they do count on learning valuable skills that they will put to immediate use.

## Useful Examples

Use real-world examples when possible. Many books have short snippets that illustrate syntax or style. Instead of using the ubiquitous `foo` and `bar`, pull a short example from an application you're working on.

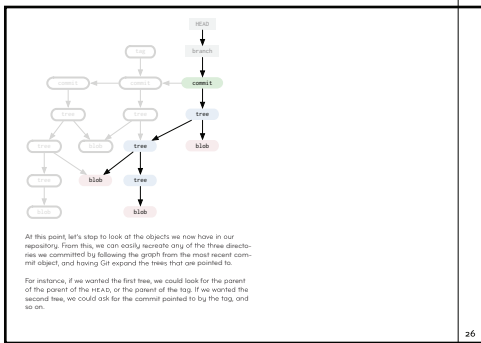
Of course, this isn't always an option. Some snippets may be too short to be useful or the best way to communicate may be to use an example devoid of greater meaning. That's OK.

## Informative Communication

A book must communicate well in order to be practical. PeepCode Press was built to give you many options for communicating your ideas.

## Best Medium

Choose the best medium for explaining the topic at hand. If a diagram can explain it best, we'd be glad to contribute one. Color, graphics, and hyperlinks can also be used in a PDF. We can bundle a short screencast for download alongside a book.



We're here to illustrate and enhance your minibook. We will also design additional graphics for topics that are already communicated in the text (but will always get your approval before the final release). We can also purchase stock photography or consult the owners of other photographs you want to use throughout your book.

## No Rug

Don't sweep the difficult concepts under the rug. Teaching a topic is the best way to learn, and you'll encounter bugs or oddities along the way. Figure out why it works that way and propose a workaround.

For example, the *Rails Page, Action, and Fragment Caching* screencast talks about how some expiration methods take a `String` argument and others take a `Hash`. This makes little sense, but it's the way it works and confounded me for a good 30 minutes. Mentioning it has saved many people 30 minutes (or days) of confusion.

Even if you include just a few such examples in your book, people will feel that they have received good value and will look forward to your next book with eager anticipation.

## No Padding

Write as much as is necessary to communicate the information. Print

publishers need authors to write a certain number of pages so the book can be sufficiently wide on the bookstore shelf. Minibooks have no such constraint and can establish their length more naturally.

## Assume a Level of Experience

Assume previous experience instead of covering every topic from the beginner level. Pick up any book on Javascript and you'll find a chapter or appendix that explains the basics of Javascript syntax. The minibook format makes that unnecessary.

Instead, assume your readers have a basic knowledge of the topic and write to that level. Prerequisites will be mentioned when the book is sold.

Very few books are aimed at a market of advanced developers, but the minibook makes that possible. If you plan to write a series, you're free to address topics in separate non-sequential minibooks. For example, the *Capistrano Concepts* screencast assumed that readers had an existing server and didn't show how to install a database, configure a webserver, etc. A subsequent screencast may show how to do those things.

---

If you do happen to write a series of minibooks on a topic, there is a possibility that they could be bundled into an anthology and sold as a paper book later on. Writing minibooks frees you from needing to make a commitment toward an entire book and helps to fund your writing during this process.

---

## Entertain

It's possible to write a book that is practical, informative, and humor-

ous. Although technical content can often be stale, a little humor can keep it interesting. Humor can be expressed in code examples, the phrasing of sentences, or real-life anecdotes.

We also want to craft the visual design of the book to reflect your sense of humor or the culture of the code you are writing about. For example, the Merb framework is often represented by a pug dog and the Nginx webserver is represented by a Russian star. These graphical elements could be incorporated into the design of the book.

## Concerns

Cite the sources of your information (use hyperlinks or list them in a bibliography at the end of the book). Don't plagiarize. Content from photo sharing services can sometimes be used, but we will research the individual licenses to make sure we are in compliance.

# Markup

## CHAPTER 3

Our workflow is built around the easy, plain-text Textile (<http://textism.com/tools/textile>) formatting syntax. Markdown (<http://daringfireball.net/projects/markdown>) is also possible but was made to look like plain-text email and isn't as well suited for some of the specific tagging that we need to covert text into a PDF.

Among other benefits, you can use any text editor you are familiar with to write your book. Some also have built-in features for working with Textile. You can also use TextMate's automated web-preview to see basic styling as you write.

## Sample

Here is a short Textile sample with extra PeepCode tags:

h1. Tools Chapter

This chapter is about the tools you will use, inspired by the person who said:

bq. A sharp knife is safer than a dull one.

h2. Razor Selection

It's important to choose a good "razor":<http://www.gillette.com>. The terms `_blade_` and `_razor_` are used interchangeably.

We urge you to consider these points:

- \* Price
- \* Style

## \* Durability

note. A razor is irrelevant for those who prefer to be unshaven.

### h3. Electric Razor

Here's some code illustrating the use of an electric razor.

```
ruby. electric_project/blade.rb#preferred_section
```

First, open the @razor.rb@ file and issue the @rake@ command.

### h4. Samurai Blade

For those who can afford it, a samurai blade is the best.

```
!samurai_blade.eps(Illustration of a Samurai blade)!
```

To summarize, see this table:

|                                    |
|------------------------------------|
| Blade Type Description             |
| Straight Takes skill to use        |
| Electric Very convenient           |
| Sword Recommended only for experts |

You'll note that there are

- Titles: h1, h2, etc.
- Textile links.
- Inline code surrounded by the @ symbol.
- The special `note.` tag for a highlighted tip.
- Code inserted from elsewhere via tags like `ruby..` The optional part after the hashmark identifies a single section from the code file.
- Images with a caption in parentheses.

- Tables with column titles in the first row.

## File Organization

You should have been given access to a PeepCode repository at GitHub. If not, contact **peepcode@topfunky.com** with your GitHub (<http://github.com>) username and we will set one up for you.

The general layout of your repository will look like this:

```
shavingbook/  
  artwork # For the cover, diagrams, etc.  
  code    # Single files or entire projects  
  layout  # InDesign files  
  output  # Generated drafts  
  sidebars # Textile snippets for sidebars  
  text    # Textile and Markdown files
```

## Text

The files in the `text` directory of your repository will be combined into a single document for printing. You can prepend their names with a number to order them, or just rely on automatic alphabetical sorting (Ruby's `sort` method is used). Apart from the prefix, the names of the files are irrelevant to the final output, so use names that make sense to you.

```
text/  
  1-shaving-basics.textile  
  2-shaving-tools.textile  
  3-soaps-and-lathers.textile  
  4-recovering-from-mishaps.textile
```

# Code

The `code` directory holds single files or entire projects that you will reference in your book.

## MARKUP FOR CODE

Source code is kept separate from your text document and is inserted with a few special tags. This means you can edit example code in your favorite IDE, then come back to your text document to write about it.

A textile tag identifies the kind of code that is being included. Follow that with the path to the file, relative to the `code` directory.

Here's how to insert an entire Ruby document into your book:

```
ruby. demo/app/models/farm.rb
```

All paths are relative to the `code` directory, so you can start with the name of the project or file as listed inside the `code` directory. You can organize the `code` directory however you want to. Put entire Rails applications in there, single files, or whatever makes the most sense to you.

If you want to include only a snippet from a file, mark off your code with `BEGIN` and `END` blocks, preceded by the single-line comment mark for the language in question:

```
# BEGIN special  
map.resources :farms  
# END special
```

Then include it into your document with the hashmark syntax:

```
ruby. demo/config/routes.rb#special
```

Available languages are:

- ruby
- css
- html
- javascript
- xml
- yaml
- shell (for code entered on the terminal)
- code (for other languages not listed here, or plain text such as logfiles)

## Artwork

Images can be placed with the standard Textile image tag (exclamation points). Words in parentheses will be used as a caption.

```
!razor.eps(A razor)!
```

Images will be searched for relative to the `artwork` directory.

The best format for diagrams is the vector-based EPS. We can also work with TIF or PNG for images. We prefer to use OmniGraffle for creating diagrams.

---

We're glad to create or cleanup images and diagrams for you. In fact, we will probably edit any diagram slightly to use our standard fonts and stock icon illustrations.

---

# Titles

Use `h1` for chapter titles. Any `h1` detected will automatically start a new page in the final output. If you design your minibook to be a single chapter, you don't need to use `h1` at all. This author guide doesn't use `h1` since it is only a few pages long.

`h1. Tools for Shaving`

The `h2` works best for section titles and will show up in the table of contents. Use these to mark a new topic within a chapter.

`h2. Straight Razor or Cartridge?`

You can also use `h3` and `h4`. Too many levels of hierarchy can get confusing for you and for the reader, but they are available if you need them. Titles in `h4` will be shown in small caps and are useful for marking a list of related items that have too much content for a simple bulleted list.

The *Markup* section in this document is an `h2`. The *Paragraphs* section below this paragraph is an `h3`. The *No Rug, No Padding*, and *Serialize* sections above use `h4`.

# Paragraphs

Most paragraphs in your book will just be plain paragraphs with no markup.

Some useful inline tags are

| Inline Tag | Description  |
|------------|--|
| @          | Identifies code or file-names.   |
| _          | Underscore for emphasis. Use this instead of quoting terms or special phrases. |
| *          | Asterisk for bold. Use sparingly.  |

## Lists

Lists can be constructed with asterisks.

- \* Item one
- \* Item two
- \* Item three

Produces

- Item one
- Item two
- Item three

You can also use the inline tags within lists to bold, identify code, etc.

## Tables

Textile tables are supported. You should always include titles in the first row since they will be automatically reused if the table spans several pages.

| Blade Type | Description                  |
|------------|------------------------------|
| Straight   | Takes skill to use           |
| Electric   | Very convenient              |
| Sword      | Recommended only for experts |

## Other Standard Tags

The other standard Textile and Markdown tags should just work, with the exception of image includes. A few are in the process of being implemented but should be working by the time your book is ready to sell.

## Special Tags

We've added a few tags to make the final output look better. They all consist of simple identifiers that precede a block of text.

### Note

A note is similar to a blockquote but is separated from the text with horizontal rules. Notes are useful for tips or side comments that should stand out from the rest of the text. Inline identifiers can be used within note blocks.

`note.` Scott Barron has written one of the classic books on shaving.

---

Scott Barron has written one of the classic books on shaving.

---

## Todo

A `todo` section will be displayed in bright colors. This is a good way to mark a section that you need to finish and don't want to forget about. `todo` blocks will throw a flag when publishing to ensure that the final book is not printed with things left to do.

```
todo. Write this section before publishing!
```

---

```
** TODO ** Write this section before publishing!  
** TODO **
```

---

# Workflow

## CHAPTER 4

We want to deliver cutting-edge content to as many devices as possible. For example, we're planning on a web interface that can be easily viewed on the iPhone.

## Proofs

Once you have written a few pages we will start building proofs of your book periodically. This will give you an exact copy of what the final book will look like. You will be able to download a copy from the Subversion repository for your book.

We will read your book during this process and make comments in email or in the text under an `editor` tag.

Although we don't have the full cachet of copy editor, proofreader, technical editor, managing editor, series editor, editor-in-chief, and assistant to the editor-in-chief, we will send your book to a proofreader before final publication.

You are not obligated to update your book after it is done. However, we will fix any minor spelling errors or other bugs. We may contact you if any glaring technical inaccuracies are reported by readers.

# Revisions

## CHAPTER 5

- **May 2008** Reflects new table and image tags.
- **August 9, 2007** Rewritten to use new screen-targeted format. Added notes about enhanced code includes.
- **May 30, 2007** Implemented hyperlinks. Use the Textile link syntax or just type a url and it will be turned into a clickable link in the final document.
- **May 29, 2007** Implemented code includes. See code section for details.