Everything you
need to know to
get started!

KITABU

NANDO VIEIRA

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

CHAPTER 1

Getting Started

This guide is designed for beginners who want to get started with Kitabu from scratch. Kitabu uses Ruby to generate e-books in different formats by using Markdown.

But before doing it so, you need to have some things installed:

- The Ruby interpreter version 3.2 or newer.
- The PrinceXML converter version 14.0 or newer.

Installing Ruby

To install Ruby, consider using RVM or rbenv, both available for Mac OSX and Linux distros. If you're running a Windows, well, I can't help you. I don't even know if Kitabu runs over Windows boxes, so if you find any bugs, make sure you let me know.

Installing PrinceXML

PrinceXML is the best HTML to PDF converter available. You can use advanced CSS features to style your book in any way you want. But good things don’t come for free, and PrinceXML is no exception. The Professional License, which you grant you a installation on a single computer by a single user costs 495USD. If you don’t like the price tag, consider using DocRaptor when you’re ready to publish your book.

To install PrinceXML, go to the website and download the correct version for your platform; you can choose from Mac OSX, to Linux and Windows.

Installing Calibre

Calibre is a e-book manager that, among other features, allows converting ePub files into .mobi files.

To install Calibre, go to the website and download the correct version for your platform; you can choose from Mac OSX, to Linux and Windows.

CHAPTER 2

Creating Chapters

You can create chapters by having multiple files or directories. They’re alphabetically sorted, so make sure you use a prefixed file name like `01_Introduction.md` as the file name.

If you’re going to write a long book, make sure you use the directory organization. This way you can have smaller text files, which will be easier to read and change as you go. A file structure suggestion for a book about Ruby on Rails would be:

```
getting-started-with-rails
├── text
    └── 01_Guide_Assumptions.md
    └── 02_Whats_Rails.md
    └── 03_Creating_A_New_Project
        └── 01_Installing_Rails.md
        └── 02_Creating_The_Blog_Application.md
    └── 04_Hello_Rails
        └── 01_Starting_Up_The_Web_Server.md
        └── 02_Say_Hello_Rails.md
        └── 03_Setting_The_Application_Home_Page.md
    └── ...
```

Notice that the file name does not need to be readable, but it will make your life easier.

CHAPTER 3

Syntax Highlighting

What about the syntax

Kitabu uses Route as the syntax highlight formatter. It emits an output compatible with stylesheets designed for pygments, the Python library used by many.

To highlight a code block, use the fenced block syntax. The following example would be formatted as Ruby.

~~~~
```ruby
class User
  attr_accessor :name, :email

  def initialize(name, email)
    @name = name
    @email = email
  end
end
```
~~~~

The output would be something like this:

```ruby
class User
  attr_accessor :name, :email

  def initialize(name, email)
    @name = name
    @email = email
  end
end
```

! Here’s a tip!

If you’re using Sublime Text, make sure you install the Markdown Extended plugin; it enables code syntax highlighting on your Markdown files.

You can also provide inline options such as line numbers and inline rendering.

~~~
```ruby?line_numbers=1
class User
  attr_accessor :name, :email

  def initialize(name, email)
    @name = name
    @email = email
  end
end
```
~~~

This would be rendered like this:

```ruby
class User
  attr_accessor :name, :email

  def initialize(name, email)
    @name = name
    @email = email
  end
end
```

You can select the theme by using the configuration `theme` on your `config/kitabu.yml`. The available themes are:

- thankful_eyes
- colorful
- base16
- base16.dark
- base16.light
- base16.solarized
- base16.solarized.dark
- base16.solarized.light
- base16.monokai
- base16.monokai.light
- base16.monokai.dark
- github
- github.dark
- github.light
- igorpro
- monokai
- molokai
- monokai.sublime
- gruvbox
- gruvbox.light
- gruvbox.dark
- tulip
- pastie
- bw
- magritte

Lexers

Rouge comes with dozens of lexers. Check out this list, generated dynamically when you export your e-book.

Groovy `groovy`
The Groovy programming language (http://www.groovy-lang.org/)

Graphql `graphql`
GraphQL

Gradle `gradle`
A powerful build system for the JVM

ArmAsm `armasm`
Arm assembly syntax

Hack `hack`
The Hack programming language (hacklang.org)

PHP `php`
The PHP scripting language (php.net)

Io `io`
The IO programming language (http://iolanguage.com)

Prolog `prolog`
The Prolog programming language (http://en.wikipedia.org/wiki/Prolog)

Prometheus `prometheus`
prometheus

Augeas `augeas`
The Augeas programming language (augeas.net)

Awk `awk`
pattern-directed scanning and processing language

Batchfile `batchfile`
Windows Batch File

.properties `properties`
.properties config files for Java

Protobuf `protobuf`
Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data

Q `q`
The Q programming language (kx.com)

QML `qml`
QML, a UI markup language

BBCBASIC `bbcbasic`
BBC BASIC syntax

Haml `haml`
The Haml templating system for Ruby (haml.info)

R `r`
The R statistics language (r-project.org)

JavaScript `javascript`
JavaScript, the browser scripting language

Sass `sass`

The Sass stylesheet language language (sass-lang.com)

Racket `racket`

Racket is a Lisp descended from Scheme (racket-lang.org)

BibTeX `bibtex`

BibTeX

ReasonML `reasonml`

New syntax on top of OCaml ecosystem (reasonml.github.io)

Rego `rego`

The Rego open-policy-agent (OPA) policy language (openpolicyagent.org)

ReScript `rescript`

The ReScript programming language (rescript-lang.org)

RML `rml`

A system agnostic domain-specific language for runtime monitoring and verification (https://rmlatdibris.github.io/)

BIML `biml`

BIML, Business Intelligence Markup Language

Handlebars `handlebars`

the Handlebars and Mustache templating languages

C# `csharp`

a multi-paradigm language targeting .NET

BPF `bpf`

BPF bytecode syntax

Haskell `haskell`

The Haskell programming language (haskell.org)

Robot Framework `robot_framework`

Robot Framework is a generic open source automation testing framework (robotframework.org)

Brainfuck `brainfuck`

The Brainfuck programming language

Haxe `haxe`

Haxe Cross-platform Toolkit (http://haxe.org)

BrightScript `brightscript`

BrightScript Programming Language (https://developer.roku.com/en-ca/docs/references/brightscript/language/brightscript-language-reference.md)

Hashicorp Configuration Language `hcl`

Hashicorp Configuration Language, used by Terraform and other Hashicorp tools

HLSL `hlsl`

HLSL, the High Level Shading Language for DirectX (docs.microsoft.com/en-us/windows/win32/direct3dhlsl/dx-graphics-hlsl)

HOCON `hocon`

Human-Optimized Config Object Notation (https://github.com/lightbend/config)

HQL `hql`

Hive Query Language SQL dialect

SQL `sql`

Structured Query Language, for relational databases

XML `xml`

XML

1C (BSL) `bsl`

The 1C:Enterprise programming language

Rust `rust`

The Rust programming language (rust-lang.org)

HTTP `http`

http requests and responses

SAS `sas`

SAS (Statistical Analysis Software)

Ceylon `ceylon`

Say more, more clearly.

XPath `xpath`

XML Path Language (XPath) 3.1

CFScript `cfscript`

CFScript, the CFML scripting language

HyLang `hylang`

The HyLang programming language (hylang.org)

Cisco IOS `cisco_ios`

Cisco IOS configuration lexer

YAML `yaml`

Yaml Ain’t Markup Language (yaml.org)

Clean `clean`

The Clean programming language (clean.cs.ru.nl)

IDL `idlang`

Interactive Data Language

Idris `idris`

The Idris programming language (idris-lang.org)

Clojure `clojure`

The Clojure programming language (clojure.org)

CMake `cmake`

The cross-platform, open-source build system

IgorPro `igorpro`

WaveMetrics Igor Pro

Scala `scala`

The Scala programming language (scala-lang.org)

CMHG `cmhg`

RISC OS C module header generator source file

CODEOWNERS `codeowners`

Code Owners syntax (https://docs.gitlab.com/ee/user/project/codeowners/reference.html)

CoffeeScript `coffeescript`

The Coffeescript programming language (coffeescript.org)

Scheme `scheme`

The Scheme variant of Lisp

SCSS `scss`

SCSS stylesheets (sass-lang.com)

sed `sed`

sed, the ultimate stream editor

INI `ini`

the INI configuration format

Irb `irb`

Shell sessions in IRB or Pry

Irb_output `irb_output`

Common Lisp `common_lisp`

The Common Lisp variant of Lisp (common-lisp.net)

Isabelle `isabelle`

Isabelle theories (isabelle.in.tum.de)

Sieve `sieve`

mail filtering language

Slice `slice`

Specification Language for Ice (doc.zeroc.com)

Slim `slim`

The Slim template language

Smalltalk `smalltalk`

The Smalltalk programming language

Config File `conf`

A generic lexer for configuration files

Console `console`

A generic lexer for shell sessions. Accepts ?lang and ?output lexer options, a ?prompt option, ?comments to enable # comments, and ?error to handle error messages.

Smarty `smarty`

Smarty Template Engine

SML `sml`

Standard ML

Coq `coq`

Coq (coq.inria.fr)

ISBL `isbl`

The ISBL programming language

SPARQL `sparql`

Semantic Query Language, for RDF data

J `j`
The J programming language (jsoftware.com)

SQF `sqf`
Status Quo Function, a Real Virtuality engine scripting language

SSH Config File `ssh`
A lexer for SSH configuration files

C++ `cpp`
The C++ programming language

Stan `stan`
Stan Modeling Language (mc-stan.org)

Crystal `crystal`
Crystal The Programming Language (crystal-lang.org)

Stata `stata`
The Stata programming language (www.stata.com)

SuperCollider `supercollider`
A cross-platform interpreted programming language for sound synthesis, algorithmic composition, and realtime performance

Janet `janet`
The Janet programming language (janet-lang.org)

Svelte `svelte`
Svelte single-file components (https://svelte.dev/)

Swift `swift`
Multi paradigm, compiled programming language developed by Apple for iOS and OS X development. (developer.apple.com/swift)

Markdown `markdown`
Markdown, a light-weight markup language for authors

Systemd `systemd`
A lexer for systemd unit files

Jinja `jinja`
Django/Jinja template engine (jinja.pocoo.org)

Syzlang `syzlang`
Syscall description language used by syzkaller

JSL `jsl`
The JMP Scripting Language (JSL) (jmp.com)

Json-doc `json-doc`
JavaScript Object Notation with extensions for documentation

Jsonnet `jsonnet`
An elegant, formally-specified config language for JSON

Jsp `jsp`
JSP

JSX `jsx`
An XML-like syntax extension to JavaScript (facebook.github.io/jsx/)

csvs `csvs`

The CSV Schema Language (digital-preservation.github.io)

CUDA `cuda`

Compute Unified Device Architecture, used for programming with NVIDIA GPU

Cypher `cypher`

The Cypher query language (neo4j.com/docs/cypher-manual)

Julia `julia`

The Julia programming language

Cython `cython`

Cython and Pyrex source code (cython.org)

Python `python`

The Python programming language (python.org)

Kotlin `kotlin`

Kotlin Programming Language (http://kotlinlang.org)

TAP `tap`

Test Anything Protocol

Lasso `lasso`

The Lasso programming language (lassosoft.com)

Tcl `tcl`

The Tool Command Language (tcl.tk)

Syzprog `syzprog`

Program description language used by syzkaller

Terraform `terraform`

Terraform HCL Interpolations

Lean `lean`

The Lean programming language (leanprover.github.io)

D `d`

The D programming language(dlang.org)

Liquid `liquid`

Liquid is a templating engine for Ruby (liquidmarkup.org)

TeX `tex`

The TeX typesetting system

TOML `toml`

the TOML configuration format (https://github.com/toml-lang/toml)

TSX `tsx`

TypeScript-compatible JSX (www.typescriptlang.org/docs/handbook/jsx.html)

TTCN3 `ttcn3`

The TTCN3 programming language (ttcn-3.org)

Tulip `tulip`

the tulip programming language (twitter.com/tuliplang)

Dafny `dafny`
The Dafny programming language (github.com/dafny-lang/dafny)

C `c`
The C programming language

Literate CoffeeScript `literate_coffeescript`
Literate coffeescript

Literate Haskell `literate_haskell`
Literate haskell

Twig `twig`
Twig template engine (twig.sensiolabs.org)

Turtle/TriG `turtle`
Terse RDF Triple Language, TriG

LiveScript `livescript`
LiveScript, a language which compiles to JavaScript (livescript.net)

TypeScript `typescript`
TypeScript, a superset of JavaScript (https://www.typescriptlang.org/)

Vala `vala`
A programming language similar to csharp.

VCL: Varnish Configuration Language `vcl`
The configuration language for Varnish HTTP Cache (varnish-cache.org)

Dart `dart`
The Dart programming language (dart.dev)

Visual Basic `vb`
Visual Basic

Datastudio `datastudio`
Datastudio scripting language

LLVM `llvm`
The LLVM Compiler Infrastructure (http://llvm.org/)

Lua `lua`
Lua (http://www.lua.org)

diff `diff`
Lexes unified diffs or patches

digdag `digdag`
A simple, open source, multi-cloud workflow engine (https://www.digdag.io/)

Velocity `velocity`
Velocity is a Java-based template engine (velocity.apache.org)

HTML `html`
HTML, the markup language of the web

VHDL 2008 `vhdl`
Very High Speed Integrated Circuit Hardware Description Language

Lustre `lustre`
The Lustre programming language (Verimag)

Lutin `lutin`
The Lutin programming language (Verimag)

M68k `m68k`
Motorola 68k Assembler

Magik `magik`
Smallworld Magik

Vue `vue`
Vue.js single-file components

Verilog and System Verilog `verilog`
The System Verilog hardware description language

Wollok `wollok`
Wollok lang

VimL `viml`
VimL, the scripting language for the Vim editor (vim.org)

Apache `apache`
configuration files for Apache web server

Xojo `xojo`
Xojo

Make `make`
Makefile syntax

Mason `mason`
The HTML::Mason framework (https://metacpan.org/pod/HTML::Mason)

Docker `docker`
Dockerfile syntax

DOT `dot`
graph description language

Perl `perl`
The Perl scripting language (perl.org)

ECL `ecl`
Enterprise Control Language (hpccsystems.com)

Mathematica `mathematica`
Wolfram Mathematica, the world’s definitive system for modern technical computing.

EEX `eex`
Embedded Elixir

Elixir `elixir`
Elixir language (elixir-lang.org)

Eiffel `eiffel`
Eiffel programming language

MATLAB `matlab`
Matlab

XQuery `xquery`
XQuery 3.1: An XML Query Language

Meson `meson`
Meson’s specification language (mesonbuild.com)

YANG `yang`

Lexer for the YANG 1.1 modeling language (RFC7950)

MiniZinc `minizinc`

MiniZinc is a free and open-source constraint modeling language (minizinc.org)

MoonScript `moonscript`

Moonscript (http://www.moonscript.org)

Elm `elm`

The Elm programming language (http://elm-lang.org/)

Mosel `mosel`

An optimization language used by Fico's Xpress.

Email `email`

An email message

EPP `epp`

Embedded Puppet template files

Puppet `puppet`

The Puppet configuration management language (puppetlabs.org)

Ruby `ruby`

The Ruby programming language (ruby-lang.org)

Erlang `erlang`

The Erlang programming language (erlang.org)

Plain Text `plaintext`

A boring lexer that doesn't highlight anything

MessageTrans `msgtrans`

RISC OS message translator messages file

MXML `mxml`

MXML

Zig `zig`

The Zig programming language (ziglang.org)

Nasm `nasm`

Netwide Assembler

NesAsm `nesasm`

Nesasm3 assembly (6502 asm)

nginx `nginx`

configuration files for the nginx web server (nginx.org)

Nial `nial`

The Nial programming language (nial-array-language.org)

Factor `factor`

Factor, the practical stack language (factorcode.org)

Nim `nim`

The Nim programming language (http://nim-lang.org/)

Fluent `fluent`

Fluent localization files

Fortran `fortran`
Fortran 2008 (free-form)

Nix `nix`
The Nix expression language (https://nixos.org/nix/manual/#ch-expression-language)

FreeFEM `freefem`
The FreeFEM programming language (freefem.org)

FSharp `fsharp`
F# (fsharp.net)

Java `java`
The Java programming language (java.com)

GDScript `gdscript`
The Godot Engine programming language (https://godotengine.org/)

CSS `css`
Cascading Style Sheets, used to style web pages

Objective-C `objective_c`
an extension of C commonly used to write Apple software

GHC Cmm (C--) `ghc-cmm`
GHC Cmm is the intermediate representation of the GHC Haskell compiler

Escape `escape`
A generic lexer for including escaped content - see Formatter.enable_escape!

Objective-C++ `objective_cpp`
an extension of C++ uncommonly used to write Apple software

OCaml `ocaml`
Objective Caml (ocaml.org)

GHC Core `ghc-core`
Intermediate representation of the GHC Haskell compiler.

OCL `ocl`
OMG Object Constraint Language (omg.org/spec/OCL)

OpenEdge ABL `openedge`
The OpenEdge ABL programming language

Gherkin `gherkin`
A business-readable spec DSL (github.com/cucumber/cucumber/wiki/Gherkin)

ABAP `abap`
SAP - Advanced Business Application Programming

OpenType Feature File
`opentype_feature_file`
Feature specifications for an OpenType font (adobe-type-tools.github.io/afdko)

Pascal `pascal`
a procedural programming language commonly used as a teaching language.

GLSL `glsl`
The GLSL shader language

ActionScript `actionscript`
ActionScript

Go `go`
The Go programming language (http://golang.org)

shell `shell`
Various shell languages, including sh and bash

ERB `erb`
Embedded ruby template files

Ada `ada`
The Ada 2012 programming language

Plist `plist`
plist

PLSQL `plsql`
Procedural Language Structured Query Language for Oracle relational database

Pony `pony`

PostScript `postscript`
The PostScript language (adobe.com/devnet/postscript.html)

Apex `apex`
The Apex programming language (provided by salesforce)

powershell `powershell`
powershell

API Blueprint `apiblueprint`
Markdown based API description language.

Praat `praat`
The Praat scripting language (praat.org)

AppleScript `applescript`
The AppleScript scripting language by Apple Inc. (https://developer.apple.com/library/archive/documentation/AppleScript/Conceptual/AppleScriptLangGuide/introduction/ASLR_intro.html)

JSON `json`
JavaScript Object Notation (json.org)

And if what you want is not on this list, make you open a ticket on the project.

CHAPTER 4

Dynamic Content

Sometimes you may find useful to generate content dynamically. Maybe you’re going to read some configuration file, or maybe you just want to define some helpers. Kitabu has support for ERb files; all you need to do is naming your text file as `.md.erb`.

On the previous chapter, we listed all supported Rouge lexers. To do that, I created a helper that looks like this:

```ruby
# frozen_string_literal: true

module Kitabu
  # If you need to process the markdown before rendering it, you can use the
  # `before_markdown_render` hook.
  #
  # Kitabu.add_hook(:before_markdown_render) do |content|
  #   content
  # end

  # Similarly, you can manipulate the generated HTML by using the hook
  # `after_markdown_render`.
  #
  # Kitabu.add_hook(:after_markdown_render) do |content|
  #   html = Nokogiri::HTML(content)
  #
  #   # Do something with HTML here.
  #
  #   html.to_html
  # end

  module Helpers
    # This method is just a helper example.
    # Feel free to remove it once you get rid of the sample output.
    def lexers_list
      buffer = [%[<ul class="lexers two-columns">]]

      Rouge::Lexers.constants.each do |const|
        lexer = Rouge::Lexers.const_get(const)

        begin
          title = lexer.title
          tag = lexer.tag
          description = lexer.desc
        rescue StandardError
          next
        end

        buffer << "<li>"
        buffer << "<strong>#{title}</strong> "
        buffer << "<code>#{tag}</code><br>"
        buffer << "<span>#{description}</span>"
        buffer << "</li>"
      end

      buffer << "</ul>"
      buffer.join
    end
  end
end
```

To use it, I just needed to add `<%= lexers_list %>` to my text file. This allows you to create anything you need!

Escaping ERb code

If you want to write a book about Rails, you’re likely to use lots of ERb tags. In this case, make sure you escape the `<% %>` and `<%= %>` markers as `<%% %>` and `<%%= %>`; otherwise you’ll have a syntax error.

```
<%%= Date.today %>
```

CHAPTER 5

Exporting Files

You can generate files as you go. Just execute `kitabu export` from your book’s root directory.

```
$ kitabu export
** e-book has been exported
```

This command will generate all supported formats[1]. The generated files will be placed on your `output` directory; the following output list only the relevant files.

```
$ tree output
output
├── images
│   ├── kitabu.png
│   └── kitabu.svg
├── kitabu.epub
├── kitabu.html
├── kitabu.mobi
├── kitabu.pdf
├── kitabu.print.pdf
```

[1] Depend on Prince and Calibre being available on your `$PATH`. ↩

```
└── styles
    ├── epub.css
    ├── html.css
    ├── pdf.css
    └── print.css
```

This can take a while depending on your book size, but usually the process is pretty fast. If you want to generate a specific format faster, provide the `--only` flag.

```
$ kitabu export --only pdf
```

You can also automatically generate files when something changes. You can use [Guard]() for this, and Kitabu even generates a sample file for you. All you have to do is running `bundle exec guard`.

```
$ bundle exec guard
20:38:10 - INFO - Guard is now watching at 'kitabu/examples/kitabu'
** e-book has been exported
```

Exporting PDF with DocRaptor

After exporting your files, upload the `output` directory somewhere and make it available to the internet. Then you can even use curl to generate the pdf.

The command is fairly long, so I won’t try to paste it here. You can see the full command on the [README of this project]().

THIS PAGE INTENTIONALLY LEFT BLANK

Imprint

Kitabu

This guide will help you understand how all of the pieces fit together on Kitabu.

Nando Vieira