Extracted from:

Web Development with Clojure, Third Edition

Build Large, Maintainable Web Applications Interactively

This PDF file contains pages extracted from *Web Development with Clojure, Third Edition*, published by the Pragmatic Bookshelf. For more information or to purchase a paperback or PDF copy, please visit http://www.pragprog.com.

Note: This extract contains some colored text (particularly in code listing). This is available only in online versions of the books. The printed versions are black and white. Pagination might vary between the online and printed versions; the content is otherwise identical.

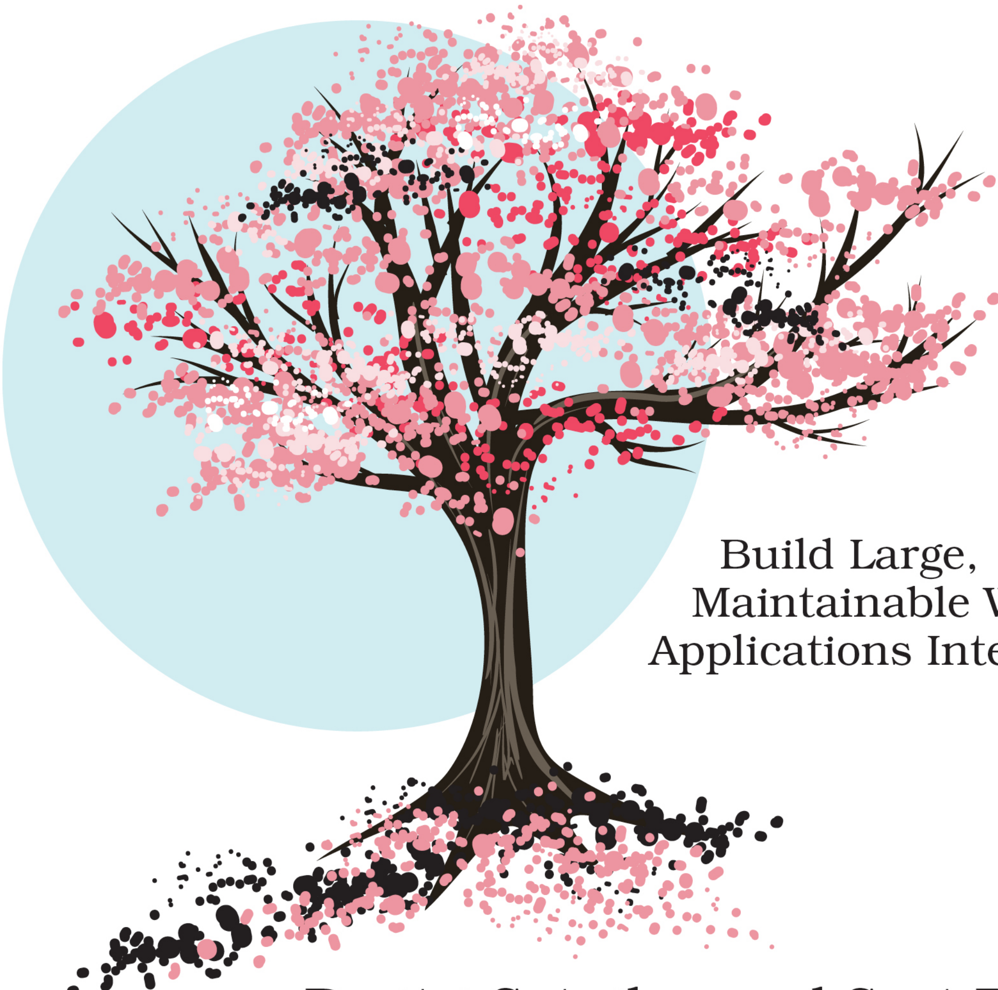
The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Web Development with Clojure

3rd Edition



Build Large, Maintainable Web Applications Interactively

Dmitri Sotnikov and Scot Brown

edited by Michael Swaine

Web Development with Clojure, Third Edition

Build Large, Maintainable Web Applications Interactively

Dmitri Sotnikov
Scot Brown

The Pragmatic Bookshelf
Raleigh, North Carolina

For our complete catalog of hands-on, practical, and Pragmatic content for software developers, please visit *https://pragprog.com*.

The team that produced this book includes:

CEO: Dave Rankin
COO: Janet Furlow
Managing Editor: Tammy Coron
Development Editor: Michael Swaine
Copy Editor: L. Sakhi MacMillan
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics
Founders: Andy Hunt and Dave Thomas

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

ISBN-13: 978-1-68050-682-2
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—July 2021

Introduction

The cover of this book has a bonsai tree on it. We chose it to represent elegance and simplicity because these qualities make Clojure such an attractive language. A good software project is like a bonsai. You have to meticulously craft it to take the shape you want, and the tool you use should make it a pleasant experience. We hope to convince you here that Clojure is that tool.

What You Need

This book is aimed at readers of all levels. While having some basic proficiency with functional programming will be helpful, it's by no means required to follow the material presented. If you're not a Clojure user already, this book is a good starting point since it focuses on applying the language to solve concrete problems. This means we'll focus on the small set of language features needed to build web applications.

Why Clojure?

Clojure is a small language whose primary goals are simplicity and correctness. As a functional language, it emphasizes immutability and declarative programming. As you'll see, these features make it easy and idiomatic to write clean and correct code.

Web development has many languages to choose from and as many opinions on what makes a language "good." Some languages are simple but verbose. You've probably heard people say that verbosity doesn't matter—that if two languages are Turing complete, anything that can be written in one language can also be written in the other with a bit of extra code. We think that's missing the point.

The real question isn't whether something can be expressed in principle; it's how well the language maps to the problem being solved. One language lets you think in terms of your problem domain, while another forces you to translate the problem to its constructs.

The latter is often tedious and rarely enjoyable. You end up writing a lot of boilerplate code and constantly repeating yourself. There's a certain irony in having to write repetitive code.

At the other extreme, some languages are concise because they provide many different tools for solving problems. Unfortunately, this vast array of tools brings different problems.

The more features a language has, the more things you have to keep in your head to work with the language effectively. Soon we find ourselves constantly expending mental overhead thinking about all the different features and how they interact with one another.

What really matters is whether you can use a language without thinking about it. When a language is lacking in expressiveness, you become acutely aware that you're writing code that you shouldn't be. On the other hand, when a language has too many features, it can feel overwhelming and it's easy to get distracted playing with them.

To make an analogy with mathematics, understanding a few fundamental theorems and their implications is far more useful than rote memorization of specific formulas.

This is where Clojure comes in. It allows us to easily derive a solution to a particular problem from a small set of general patterns. All you need to become productive is to learn a few simple concepts and a bit of syntax. These concepts can then be combined in myriad ways to solve all kinds of problems.

Why Make Web Apps in Clojure?

Clojure boasts tens of thousands of users across hundreds of companies; it's used in a wide range of settings, including banks and hospitals. Clojure is likely the most popular Lisp dialect today for starting new development. It has proven itself in serious production systems, and the feedback from users has been overwhelmingly positive.

Because web development is one of the major domains for using Clojure, several popular libraries and frameworks have sprouted and matured in this area. In this book we'll primarily focus on the Luminus stack. The following chapters will teach you how to use Clojure and Luminus to build web applications effectively.

Many platforms are available for doing web development, so why should you choose Clojure over other options?

Well, consider those options. Many popular platforms force you to make trade-offs. Some platforms lack performance, others require a lot of boilerplate, and others lack the infrastructure necessary for real-world applications.

Clojure addresses the questions of performance and infrastructure by being a hosted language. The Java Virtual Machine (JVM) is a mature and highly performant environment with great tooling and deployment options. Clojure brings expressive power akin to that of Ruby and Python to this excellent platform. When working with Clojure you won't have to worry about being limited by your runtime when your application grows.

Additionally, Clojure isn't limited to the server. It can be compiled into JavaScript that is on par with popular front-end frameworks. Most web platforms require to learn and write JavaScript as well. With Clojure, you have a common toolset for both client and server.

The most common way to handle the boilerplate in web applications is by using a framework. Examples include Ruby on Rails, Django, and Spring. The frameworks provide the canned functionality needed for building a modern site.

The benefits these frameworks offer also come with inherent costs. Since many operations are done implicitly, you have to memorize what effects any action might have. This opaqueness makes your code more difficult to reason about. When you need to do something that's at odds with the framework's design, it can quickly become awkward and difficult. You might have to dive deep into the internals of that framework and hack around the expected behaviors.

Instead of using frameworks, Clojure makes a number of powerful libraries available, and we can put these libraries together in a way that makes sense for our particular project. As you'll see, we manage to avoid having to write boilerplate, while retaining the code clarity we desire. As you read on, we think you'll agree that this model has clear advantages over the framework-based approach.

Our goal is to give you both a solid understanding of the Clojure web stack and the expertise to quickly and easily build web applications using it. The following chapters will guide you all the way from setting up your development environment to creating a complete real-world application. We'll show you what's available and then guide you in structuring your application using the current best practices.

How to Read This Book

Before you get started, we should cover some conventions used in this book.

Throughout the book, we iteratively improve the applications we're working on with instructiveness as our primary goal. You'll see the top-level directory of files change periodically. This is intentional and is done to provide different stages of development for a single app. We recommend working along with your own version of the application and trying different variations of the examples provided. This will help you to get a feel for how things work and what alternatives are possible to what we present here. If you encounter any issues, you can compare your code against our version. The source code used throughout the book can be found on the Pragmatic Programming website.[1]

We strongly recommend experimentation in the REPL. This is by far the best way to develop your skill with Clojure.

1. https://pragprog.com/titles/dswdcloj3/web-development-with-clojure-third-edition/