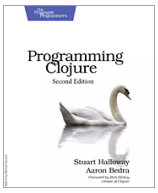

Programming Clojure
Stuart Halloway, Aaron Beedra
Second Edition April 2012
ISBN 978-1-934356-86-9
268 Seiten, broschiert
EUR 29.00
http://www.oreilly.de/catalog/9781934356869/

Buchbesprechung

Der erste angenehme Eindruck bei diesem Buch ist die seine Gestaltung im Rahmen der Reihe Pragmatic Bookshelf: In grosser Schrift und typografisch übersichtlich gestaltet, bekommt man den Stoff in zehn gleichgroßen, lesegerechten Kapiteln (jeweils 30-40 Seiten) ohne viel Redundanz aber dennoch leichtverständlich dargestellt. Die anfänglichen Zweifel z.B., ob es denn mehr als eine bloße Spielerei ist, ein LISP in der Java VM zu implementieren, verfliegen so sehr schnell, da man eindrucksvoll und knapp vorgeführt bekommt, wie reibungslos die Interoperabilität zwischen diesen beiden grundverschiedenen Spracharchitekturen hergestellt wurde.

Das erste Kapitel gibt einen sehr gut gemachten Schnellüberblick über die Besonderheiten von Clojure und die Interaktionsmöglichkeiten mit Java, d.h. die Verwendung von Spezial-Variablen, einer Demonstration von Shared-State-Funktionen und das Einbinden von Clojure Libraries.

In der zweiten Auflage wurde leider das Beispielprojekt Lancet gestrichen, das im Laufe des Buches Stück für Stück programmiert werden sollte – wahrscheinlich weil das tool *Leiningen* inzwischen diese Aufgabe übernimmt. Dies ist aber trotzdem schade, denn gerade dies machte die Erstauflage für mich sehr eindrucksvoll, wenngleich auch im Verlaufe des Lancet-Projekts hier noch einige Ungereimtheiten aufgetaucht waren. Stattdessen wurde diesmal im letzten Kapitel eine vollständige Beispielapplikation eingeführt und auch in Kapitel neun ein Praxisbeispiel eingefügt. Ein ähnliches kleines Software-Projekt wie in der Erstauflage aber, das sich parallel mit dem Fortgang des Buches fortentwickelt, wäre dennoch mein großer Wunsch.

Im zweiten Kapitel werden die grundlegenden Elemente von Clojure vorgestellt: Datenstrukturen, Interpretermakros, Funktionen und die Bindung von Werten an Symbole innerhalb bestimmter Namensräume. Kurz werden Kontrollstrukturen beschrieben und auf die Besonderheiten der Vermeidung von Schleifen in funktionaler Programmierung eingegangen. Zum Ende des Kapitels wird erörtert, wie in Clojure-Code Metadaten eingebaut werden können. Alles ist sehr interessant und gut geschrieben, jedoch sollte man hier unbedingt über etwas LISP-Kenntnisse verfügen, um den Gesamtkontext verstehen zu können,

von dem hier gehandelt wird.

Der Anfang des dritten Kapitels der Erstauflage wurde hier auf einen Unterabschnitt im zweiten Unterabschnitt eingedampft. Der Rest des Kapitels taucht völlig umgearbeitet im neuen neunten Kapitel am Ende des Buches auf, was eine grosse Verbesserung im Vergleich zur Erstauflage darstellt.

Das neue dritte, vormals vierte Kapitel behandelt Folgen wie z.B. Listen, Vektoren, Hashs usw. und was man damit anstellen kann. Sehr wichtig ist die Erkenntnis, dass man Folgen einsetzen kann, um Schleifen zu vermeiden. Vorgestellt werden die allgemeinen Funktionen der Folgen-Bibliothek und die Funktionen für bestimmte Folgen-Typen. Es wird darauf eingegangen, wie Folgen „lazy“ evaluiert werden und wie man Java-Datenstrukturen mit Folgen-Funktionen bearbeiten kann.

Im vierten Kapitel der Zweitauflage, das Techniken funktionaler Programmierung behandelt, erwartet man allgemeines Lehrbuchwissen, das man bereits von anderer Stelle kennt. Hier jedoch wird ganz besonders auf die Tücken und Beschränkungen eingegangen, die Clojure aufgrund seiner Einbettung in die Java VM notwendigerweise bietet. So kommt es z.B. bei tiefer Rekursion in Clojure/Java schnell zu einem Stack-Overflow, was sich jedoch durch Anwendung bestimmter Programmiertechniken und speziell zur Vermeidung dieses Problems entwickelter Clojure-Befehle umgehen lässt.

Im fünften Kapitel wird die Programmierung nebenläufiger Prozesse beschrieben und auf die vier in Clojure zur Verfügung stehenden Verfahren eingegangen. Dabei wurden die einleitenden Abschnitte im Vergleich zur Vorauflage neu geschrieben und modernisiert. Nebenläufigkeit mit refs, atoms und agents werden dabei in klarer, sehr gut verständlicher Form beschrieben. Bei der Implementierung von Nebenläufigkeit mittels vars und Java locks wird es dann jedoch wieder etwas komplizierter und vorkenntnisabhängiger. Am Ende des Kapitels wird ein snake-Spiel mittels nebeläufiger Programmierung implementiert. Außerdem wird das Lancet-Projekt um eine weitere Ausbaustufe erweitert, wobei der locking-Mechanismus vorgestellt wird.

Das sechste Kapitel über Protokolle und Datentypen ist in der zweiten Auflage neu hinzugekommen. Hier erfahren wir, wie man Java-Klassen erweitert, ohne den Beschränkungen zu unterliegen, die Interfaces mit sich bringen, dass diese nämlich nicht erweitert werden können, ohne sie neu zu schreiben. Hierzu dienen in Clojure die Mechanismen von protocls und datatypes, die im Kapitel anhand einer Reimplemaentation der slurp- und spit-Daten-I/O-Funktionen dargestellt werden. Am Ende des Kapitels sehen wir in einem zweiten Beispiel, wie ein Midi-Protokol mithilfe von records für Noten implementiert wird und dann mittels reify zu einer Apparatur für Zufallsmusik umgestaltet wird.

Das siebte Kapitel behandelt die Programmierung von Macros: wann sie verwendet werden sollen, Expandierung zum Testen von Macros, die Verwendung von Templates, Quotes/Unquotes und automatisch generierten eindeutigen Symbolnamen. Außerdem wird eine Typologie der Arten von Macros gegeben.

Das achte Kapitel behandelt schließlich den Einsatz von Multimethoden an zwei Beispielen: der Implementierung einer polymorphen print-Funktion, die klassenabhängig unterschiedliche Datentypen drucken kann und einigen Funktionen zur unterschiedlichen Handhabung von Spar- vs. Girokonten in Bezug auf Zinssatz und Kontoführungsgebühren, die statt Klassen Ad-Hoc-Typen zur Fallunterscheidung verwenden. Es folgen Hinweise darüber, wann Multimethoden anzuwenden sind, untersucht anhand von Beispielen aus den Clojure-Bibliotheken.

Das neunte Kapitel beinhaltet einen Großteil des Inhalts von Kapitel 3 der Erstauflage, allerdings in etwas veränderter Reihenfolge und mit einigen Ergänzungen. Der erste Abschnitt befasst sich mit dem Exception Handling, gefolgt von einem neuen Abschnitt über erweiterte Integeroperatortypen und deren Vorteile gegenüber den Java-Klassen. Es folgen die schon aus der Erstauflage bekannten Abschnitte über Peformance-Optimierung und das Erzeugen von Java-Klassen, wobei dessen letzter Abschnitt überarbeitet wurde. Ganz Neu ist hingegen ein umfangreiches Praxisbeispiel, in dem ein Programm entwickelt wird, das die Erreichbarkeit von Webseiten überprüft.

Im letzen Kapitel wird ein umfangreiches Projekt realisiert, anhand dessen auf wichtige Aspekte, wie die Auswahl von Tools, die Gestaltung eines sinnvollen Workflows, die Durchführung von Tests, die Reimplementierung einer Benutzerschnittstelle und das Deployment der Anwendung eingegangen wird. Durchgeführt wird das Ganze anhand eines Mastermind-Games, das über eine Weboberfläche bedient werden kann.

Fazit: Ein schnell lesbares Buch, da verständlich, spannend und gut geschrieben, welches man mit viel Gewinn lesen kann, vorausgesetzt man ist kein völliger Anfänger und bereits ein wenig mit den Grundlagen von Lisp und Java vertraut. Ein Buch, an dem es nichts auszusetzen gibt, ausser dass es nach 250 Seiten leider schon zu Ende ist, obwohl man immer noch gerne weiterlesen würde. Wie bei den meisten Titeln der Reihe Pragmatic Programmers wird dem Leser hier durch wirklich sinnvoller Konzentration auf das für die Praxis Wichtigste ein hervorragener Ersteinstieg in eine aktuelle Thematik geboten. Hut ab vor den Erfindern dieses Formats.