The Pragmatic Programmers

函数式编程入门：使用Elixir

Learn Functional Programming with Elixir

[匈] Ulisses Almeida 著

杜万 译

華中科技大學出版社
http://www.hustp.com

函数式编程入门：使用Elixir

Learn Functional Programming with Elixir

[匈] Ulisses Almeida 著

杜万 译

华中科技大学出版社
中国 · 武汉

内容简介

函数式编程具有代码简洁、开发速度快、易理解、易维护、扩展性强的特点，在某些领域可以解决让命令式编程头痛的问题，具有广泛的应用场景和良好的发展前景。本书是函数式编程的零基础教程，以 Elixir 为例讲解函数式编程与命令式编程的区别，帮助读者掌握函数式编程的基本概念和思想（如不可变值、显式数据转换、模式匹配、递归函数、高阶函数、多态等），避免新手常犯的错误。本书尤其适合对 Elixir 感兴趣且无函数式编程基础的读者入门学习。

图书在版编目（CIP）数据

函数式编程入门：使用 Elixir / (匈) 乌利斯 • 阿尔梅达著；杜万译.
—武汉：华中科技大学出版社，2020.5
ISBN 978-7-5680-6171-1
Ⅰ. ①函… Ⅱ. ①乌… ②杜… Ⅲ. ①程序语言－程序设计 Ⅳ. ①TP312
中国版本图书馆 CIP 数据核字(2020)第 076769 号

湖北省版权局著作权合同登记 图字：17-2020-039 号

书　　名　函数式编程入门：使用 Elixir
　　Hanshushi Biancheng Rumen: Shiyong Elixir
作　　者　[匈] Ulisses Almeida
译　　者　杜　万

策划编辑　徐定翔
责任编辑　李　昊
责任监印　徐　露

出版发行　华中科技大学出版社(中国 • 武汉)
　　武汉市东湖新技术开发区华工科技园（邮编：430223　电话：027-81321913）
录　　排　华中科技大学惠友文印中心
印　　刷　湖北新华印务有限公司
开　　本　787mm × 960mm　1/16
印　　张　12.75
字　　数　220 千字
版　　次　2020 年 5 月第 1 版第 1 次印刷
定　　价　66.80 元

本书若有印装质量问题，请向出版社营销中心调换
全国免费服务热线: 400-6679-118　竭诚为您服务

读者赞誉

Early Praise for *Functional Programming with Elixir*

学习函数式编程需要运用新的思维方式，不能急于求成。这是一本值得慢慢阅读，仔细品味的书，它清楚地阐述了函数式编程的本质，而 Elixir 很适合用来学习这种编程风格。

➤ **Kim Shrier**，独立软件开发者

我几年前就发现熟练运用函数式编程和并发编程是甄别优秀程序员的一个标准。Elixir 是一门现代函数式编程语言，具有开发并发程序的独特能力。本书很适合用来学习函数式编程的思维方式。

➤ **Nigel Lowry**，Lemmata 公司董事兼首席顾问

对希望学习函数式编程的程序员来说，这是一本很好的书。作者生动有趣、引人入胜的讲解方式令人印象深刻。

➤ **Carlos Souza**，Pluralsight 公司程序员

本书非常适合初学者学习函数式编程和 Elixir，它将为你进一步学习 OTP、

Phoenix、元编程奠定良好的基础。

➤ **Gábor László Hajba**，EBCONT 公司高级顾问

本书内容紧凑，行文流畅，示例代码清晰易懂，是一本学习 Elixir 基础知识不可多得的好书。

➤ **Stefan Wintermeyer**，Wintermeyer 咨询公司创始人

译序

自我的上一本译作《Elixir 程序设计》出版已经过去了近四年。Elixir 也从 1.2 版本更新到了 1.10 版本。官方一直保持着每半年更新一个大版本的节奏。在这些更新版本中，有关语法的变化越来越少，针对库、工具链、使用体验、性能的更新越来越多，特别是 1.9 版本，José Valim 声称 Release 是最后一个计划中的特性。我真为 Elixir 的日臻完善而感到高兴。

很多人把 Elixir 比作 Erlang 平台的 Ruby。诚然，Elixir 的作者和贡献者从 Ruby 身上借用了许多设计。Ruby 的编程体验可以说是令人惊艳的，其动态、简洁、元编程都是 Java、Golang、Python 这些同时期的编程语言所不具备的。当然，Ruby 在性能和并发编程方面也有不足。Elixir 选择将其基座造在 BEAM（Erlang VM）之上，BEAM 以 9 个 9 的可用性（31 毫秒/年的宕机时间）而著称。就并发而言，Actor 模型曾经是 Erlang 的优势之一，但今天 Rust 的 Actix 和 Java 的 Vert.x 性能测评甚至比 Erlang 的还要好。Erlang 的真正优势在于抢占式调度带来的低延时和软实时性。Elixir 的设计目标是更高的可扩展性、更高的生产力，同时保持跟 Erlang 生态圈的兼容性。

Elixir 的官方定义为：一种用于构建**可伸缩**、**可维护**应用的**动态**、**函数式编程语言**（Elixir is a dynamic, functional language designed for building scalable and maintainable applications）。下面我们就来谈谈 Elixir 与众不同的地方。

相比于大家熟悉的面向对象编程（OOP），函数式编程（FP）更强调程序执行的结果而不是过程，它倡导利用若干简单的执行单元渐进地、逐层地完成运算，而不是设计一个复杂的执行过程。每个函数的执行结果只依赖于函数的参数，而不受其他数据的影响。严格的函数式语言要求函数必须无副作用。

Elixir 的函数式编程特性包括数据不可变、模式匹配、管道等。数据不可变要求每次都通过创建新的数据来修改已有的数据。正是这一点保证了传递的参数是完全不可变的。模式匹配让我们用新的视角去看待赋值和判断，它不仅能对数据结构进行解构，还能够根据传递的参数对方法逻辑进行拆分，使得代码更简洁。管道是一种类似链式调用的语法糖，它可以让数据的变化和流动变得更清晰。Elixir 的这些语法特性是非常直观的，初学者可以非常轻松地入门并写出清晰且易于维护的代码。

Elixir 是一种强动态类型语言，它的数据类型都是在运行时才推断出来的。你也可以使用类型规格（typespec）在编译期间声明函数的签名和自定义类型，使用类型规格声明函数后，Erlang 的工具就会对源代码进行静态的类型检查，提前发现类型不一致的问题。这样做的好处是，你既可以获得静态类型语言的大部分优势，又不会失去动态类型所带来的灵活性。

大多数基于解释执行的动态语言都支持 eval 函数，eval 提供另一种动态的、执行一段运行时才能确定的代码片段。通常来说程序只能操作数据，这种通过程序修改程序的方式称为元编程。Elixir 通过宏开放了对 AST（抽象语法树）的操作能力，而不是像 C 语言中的宏利用编译器将代码文本直接替换。元编程也是 Ruby 和 Rust 的重要语言特性，它能提供强大的表达能力，让程序（尤其是框架代码）变得更简洁。

我最近两年一直在从事与 FaaS 有关的研发工作，如今容器技术发展得如火如荼，无服务计算（Serverless）方兴未艾，“以应用为中心”成为一种新的架构理念。从云原生生态的角度看，Docker、K8S、Serverless 等一系列基础设施

都在以与语言无关的方式回答可伸缩和可维护的问题。反观 Elixir，开箱即用、完整的构建和发布工具链、面向高并发的 Actor 模型，以及构建大型可伸缩、支持热更新的 OTP 框架，都让它显得小而美。Elixir 本地开发和部署到云端的版本无差异，它从语言层面、原生工具层面就开始考虑这些问题，在业务逻辑和健壮应用之间没有脱节，不需要学额外的框架，更不需要熟悉复杂的第三方平台。

掌握 Elixir 也许目前还不能给你的简历添彩，让你在职场获得更高的溢价，但学习 Elixir 可以让你以不同的视角去看待函数、可变性、并发、高可用。软件工程最大的挑战是，在持续满足业务复杂度的同时，保持工程的可维护性。而 Elixir 给出了从语言层面出发的系统性解法。虽然这门语言尚未流行起来，但这一点也不能掩盖它的优秀。

感谢编辑徐定翔给予的信任和耐心，我才得以顺利完成本书翻译。也感谢 Elixir 上海社区的同学们，特别是组织者 Tony，正因为有你们，我们可以在漫漫长路上结伴同行。由于水平和时间有限，书中难免有疏漏，敬请广大读者批评指正。

杜万

2020 年 2 月 22 日于上海

目录

Contents

前言

Preface

小时候，我喜欢玩电子闯关游戏，比如《超级马里奥兄弟》《金刚》《狮子王》《阿拉丁》。我可以毫不费力地在这些游戏间切换，因为它们都是 2D 游戏，有相同的游戏模式：向右移动，跳上平台，避免被敌人击中。

在编程语言之间切换是类似的。我的工作需要在 Ruby、JavaScript、CoffeeScript、Java、Python、Objective-C 之间切换。虽然这些语言有很大的差异，但它们在某些方面却是相似的：它们都是面向对象的。

但是从闯关游戏切换到格斗游戏就不一样了。虽然格斗游戏也是 2D 的，但是玩法完全不同。格斗游戏不需要向右越过障碍物，而是在有限的空间内击打敌人。我需要学习新东西来掌握格斗游戏的玩法。

这也是我刚开始切换到函数式编程时的感受。对象和方法在哪里？原来的思维模式害我不停犯错。我不能再像以前那样编程了，必须改变思路。

切换到新的编程范式需要你用不同的方式思考，否则就会遇到麻烦。

我建议读者在学习函数式编程之前先清空大脑。这本书将让你从全新的角度看待以往的代码。如今，大多数主流语言都支持一些函数式编程思想，即使你不会立刻使用 Elixir，你也一样可以运用函数式编程的思想。

本书适合你吗
Is This Book for You?

本书是专门写给函数式编程和 Elixir 的初学者看的。如果你学过其他编程语言，或者具备最基本的编程知识，比如编写简单的程序、调试错误、运行终端命令等，你就能阅读本书。请放心，我们将从最基本的概念讲起。

如果你是有经验的面向对象编程程序员，或者正在找工作的大学生，希望找到一本清晰阐述函数式编程概念的书，那么这本书就非常适合你。

这本书里有什么
What's in This Book?

本书讲解函数式编程的概念和基本的 Elixir 编程知识，全书共分 7 章。

第 1 章介绍函数式编程的主要概念，以及函数式编程思维的重要性。

第 2 章从头开始学习 Elixir，讲解简单的表达式、模块及函数（包括匿名和具名函数）。

第 3 章学习使用函数创建条件代码，讲解模式匹配在函数式编程中的核心作用。

第 4 章讲解递归函数，你将学习用递归完成重复的任务。

第 5 章学习可以接收或返回函数的函数（高阶函数），我们将借助高阶函数编写更简洁的代码。

第 6 章学习编写较大的应用程序，我们将使用 Elixir 进行数据建模、创建约束、实现多态。

第 7 章学习处理非纯函数，介绍四种处理策略并分析它们的优缺点。

书后有两个附录。附录 1 丰富书中的游戏示例；附录 2 是各章习题的答案。

选择 Elixir
Using Elixir

Elixir 是在 Erlang VM 中运行的函数式编程语言，Erlang VM 是运行分布式系统的绝佳环境。Elixir 具有简洁的语法、充满活力的社区，以及丰富的开发工具。选择 Elixir 能让我们专心学习函数式编程的重点。

安装 Elixir
Installing Elixir

安装 Elixir 时会默认安装 Erlang，因为 Elixir 需要 Erlang 来运行。按照 Elixir 官方安装指南进行安装即可。[1]该指南说明了所有主流操作系统安装 Elixir 的方法。请最好安装最新的 Elixir 版本（不低于 1.6.0），以便使用书中的示例。

运行代码
Running the Code

有些示例需要你在终端输入命令，它们看起来像这样：

```
$ elixir -v
Erlang/OTP 20 [erts-9.2] [source] [64-bit] [smp:4:4] [ds:4:4:10]
[async-threads:10] [hipe] [kernel-poll:false]
Elixir 1.6.0 (compiled with OTP 19)
```

在 `$` 符号后面输入命令 `elixir -v`，然后按 `Enter` 键以查看结果。执行该命令将显示你安装的 Elixir 版本。

我们还将使用一些 Elixir 的终端工具，尤其是在第 6 章。许多示例都会用到 Elixir 的交互式 shell，IEx。尝试一下：

1 https://elixir-lang.org/install.html

```
$ iex
Erlang/OTP 20 [erts-9.2] [source] [64-bit] [smp:4:4] [ds:4:4:10]
[async-threads:10] [hipe] [kernel-poll:false]
Interactive Elixir (1.6.0) - press Ctrl+C to exit (type h() ENTER for help)
iex(1)>
```

这个交互式 shell 非常适合用来尝试 Elixir 代码和调试系统。在 `iex>`提示符后输入要运行代码，然后按 `Enter` 键查看结果。例如：

```
iex> IO.puts "Hello, World"
Hello, World
:ok
```

在 IEx 中，按 `Tab` 键可以实现自动补全功能；按两次 `Ctrl + C` 则会退出。

本书的示例代码采用如下格式：

introduction/hello_world.exs
```
IO.puts "Hello, World!"
```

第一行是文件名，扩展名为 `exs`（脚本文件）或 `ex`（可编译的源文件）。可以在终端执行文件运行代码，如下所示：

```
$ elixir hello_world.exs
Hello, World!
```

掌握以上内容，你就可以使用 Elixir 运行书中的所有示例了。

在线资源
Online Resources

读者可以在本书网站上找到所有示例、提交勘误、开展讨论。[2]你可以在本书的 Elixir 社区论坛上与我和其他读者联系。[3]

2 https://pragprog.com/book/cdc-elixir/learn-functional-programming-with-elixir
3 https://elixirforum.com/t/learn-functional-programming-with-elixir-pragprog/5114

第 1 章

函数思想
Thinking Functionally

编程范式正在发生变化。如果这句话没让你感到吃惊，让我换一个说法：**编程的经典规则正在发生变化**。这种事可不常见，一旦发生就会出大事。

新编程语言不断涌现，老的编程语言不断被淘汰。造成新语言出现的原因有很多，例如出现新的问题（移动开发催生了 Swift）、出现新的要求（对性能的要求催生了 C 语言）、出现新的需求（跨硬件平台催生了可移植的 Java）。

而当编程范式发生变化时，情况就更严重了。

1.1 为什么需要函数式编程
Why Functional?

编程范式包括构建软件的法则和设计原则。范式的改变是一件严肃的事情。这意味着我们构建软件的方法并不能满足现在的需求。我们需要快速、可靠地处理多任务和海量数据。由于 CPU 的性能很难再大幅度地提升，我们不能再埋头编写代码并等待更快的 CPU 问世。解决办法是使用多核，甚至多机来