Moving On Up With ZIO

Scala Thank You

— Creator of a beautiful language
— Focus on usability
— Something worth debating

Gmail Adam Fraser <adam.fraser@gmail.com>

Implicit Function Types and Functional Reactive Programming

2 messages

Adam Fraser <adam.fraser@gmail.com> Sun, May 6, 2018 at 12:33 PM
To: martin.odersky@epfl.ch

Hi Professor Odersky. I'm a former student in your Coursera classes on functional programming in Scala. Thanks for the great content! I was playing around with Dotty and was able to implement the Signal class you described in your lecture on functional reactive programming without any global mutable variables using implicit function types. It actually ended up being quite elegant I think. Link to the code is below if you are interested. Have a great day!

Adam

• • •

https://gist.github.com/adamgfraser/20d40b3dca6a122f7eca3df2fec6981a

martin odersky <martin.odersky@epfl.ch> Mon, May 7, 2018 at 2:38 PM
To: Adam Fraser <adam.fraser@gmail.com>

Dear Adam,

Yes, that's a great use case for implicit function types. Congrats for having it figured out so nicely.

- Martin

[Quoted text hidden]

--

Prof. Martin Odersky
LAMP/IC, EPFL

The Software Stack

If I have seen further, it is by standing on the shoulders of giants

— Isaac Newton

The Software Stack

- Solutions built on "stack" of software implemented in terms of software even further down the stack
- Web service implemented in terms of HTTP framework implemented in terms of concurrency framework
- The "onion architecture" at the level of a software community

Criteria For Good Component Of The Stack

— **Complete** - provide all functionality of domain
— **High Level** - no need to drop down to lower layers
— **Principled** - higher level layers can be built on it

The Value Of Diversity

Strong software communities have strong components across their relevant stacks:

— Too few low level components means solutions are likely to be inefficient

— Too few high level components means users have to do too much to solve their actual problems

— There is a natural tension between these because there is only so much time and attention

Where Are We?

Too much focus on low level components and not enough focus on high level components:

— Education
— Libraries
— Users

Education

Functional Programming In Scala

- Incredible courses
- Learned about the implementation of lists, lazy lists, and much more
- But never used a single web client!

Python For Everybody

— Also great courses

— No coverage of implementation

— Make web requests and extract data from them almost immediately using recommended libraries

Libraries

Effect Systems Are Low Level

- Scala community has spent a tremendous amount of time on effect systems over the last ten years
- Effect systems make it easier to solve your problems but they don't solve any problem you actually have
- Investment only valuable if we use them to build higher level solutions that are performant and ergonomic

Where Are The Missing Libraries?

— Scala libraries cover areas including effect systems, web frameworks, persistence, and serialization

— Spark not primarily a Scala library now and Akka becoming closed source

— Where are libraries for machine learning, facial recognition, web scraping, and voice cloning?

Users

User Problems

Users come to a software community because they have something they want to do:

— "I want to analyze climate change trends"
— "I want to trade crypto"
— "I want to create AI generated art"

What Users Want

Users want to operate as close to the level of their problem as possible:

— Describe data transformations in high level, compositional style

— Make trades using exchange provided API or at the least high level web framework

— Specify terms to generate art and possibly high level parameters

What Users Don't Want

Users don't want to have to drop down to lower levels or concern themselves with implementation details unless they specifically choose to:

— File interaction, sharding, or parallelism
— Serialization protocol
— Details of machine learning models

Libraries Are Leaking Implementation Details

```
import sttp.client3.{SimpleHttpClient, UriContext, basicRequest}

val client = SimpleHttpClient()
client.send(basicRequest.get(uri"https://httpbin.org/get"))
```

— Monad, Functor, and Kleisli are all implementation leaks

— User doesn't care about any of these things, they just want to solve their problem

— User should be able to define a web client without knowing anything about functional programming

"Simply Scala" by Martin Odersky

Where Do We Go From Here?

Build a flourishing set of high level libraries based on the strong foundation we have established:

— **Celebrate** successes at all levels of the stack
— **Encourage** development of higher level libraries
— **Simplify** all the things

Celebrate

— Contributions at all levels of the stack are necessary to make Scala great

— We should celebrate people working at whatever level their talent and interests lead them

— Best solutions will solve problems close to user pain points in terms of a robust set of lower level libraries

Encourage

On the margin, we should be encouraging others and pushing ourselves to move "up the stack":

— How we spend our time

— How we teach

— How we mentor others

— How we communicate

Simplify

— Simplicity isn't just good for the Scala language
— We should have libraries in complex domains
— We shouldn't have complex libraries

Scala Toolkit

— Effort to promote a set of "simple" libraries
— Motivation is in exactly the right place
— Reflects a massive failure of open source ecosystem

Not Simple Libraries

— "Beginner friendly" allows users to avoid concepts like asynchronous computations

— This means libraries have failed to provide an API that is simple for synchronous and asynchronous use cases

— It also means users don't have solutions that scale with them

Really Simple Libraries

```
object Example extends ZIOAppDefault {

  val run =
    Console.printLine("Hello, ZIO!")
}
```

— Write synchronous and asynchronous code exactly the same way

— Track capabilities including errors and resources

— No type classes or monad transformers

The Quest For Simplicity

```scala
import zio._
import zio.http._

object Example extends ZIOAppDefault {

  val run =
    Client.request("https://httpbin.org/get").provide(Client.default)
}
```

— We need libraries for complex domains, not complex libraries

— Ruthless focus on simplicity

— This is pretty good, but how could it be even better?

Simplicity In Complex Domains

- ZIO Schema is library in extremely complex domain, describing the "structure" of data as data
- Derive codecs, diff, patch, merge, and migrate based on these schemas
- Despite complexity, still provides a simple user interface

Simplicity In Complex Domains

```scala
import zio._
import zio.schema._
import zio.schema.codec._

object Example extends ZIOAppDefault {

  final case class Person(name: String, age: Int)

  val schema      = DeriveSchema.gen[Person]
  val jsonEncoder = JsonCodec.jsonEncoder(schema)
  val encoded     = jsonEncoder.encodeJson(Person("Jane Doe", 42))

  val run =
    Console.printLine(encoded) // {"name":"Jane Doe","age":42}
}
```

Simply ZIO

- ZIO is focused on moving up the stack, including streaming, web framework, and distributed computing
- Relentlessly focused on simplicity in everything we do
- Not perfect, but a journey of continuous improvement we invite you to take with us

Conclusion

— Not enough high level libraries in Scala
— What can you do to help create one?
— ZIO a great foundation to build on

Thank You

— All of you for attending
— Conference organizers for making this possible
— ZIO users for your trust and feedback
— ZIO contributors for your work and insights
— John De Goes for building an incredible community