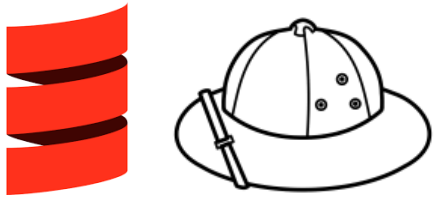
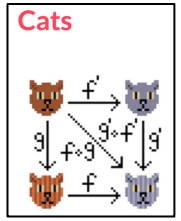
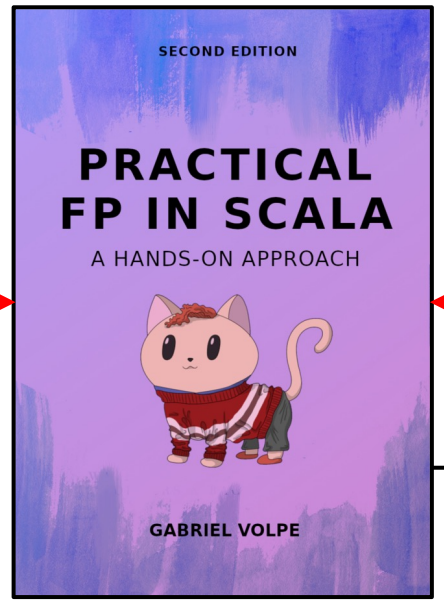




a sighting of

sequence

in



by

Gabrielλ Volpe
@volpegabriel87

slides by

@philip_schwarz

FP Illuminated

http://fpilluminated.com/

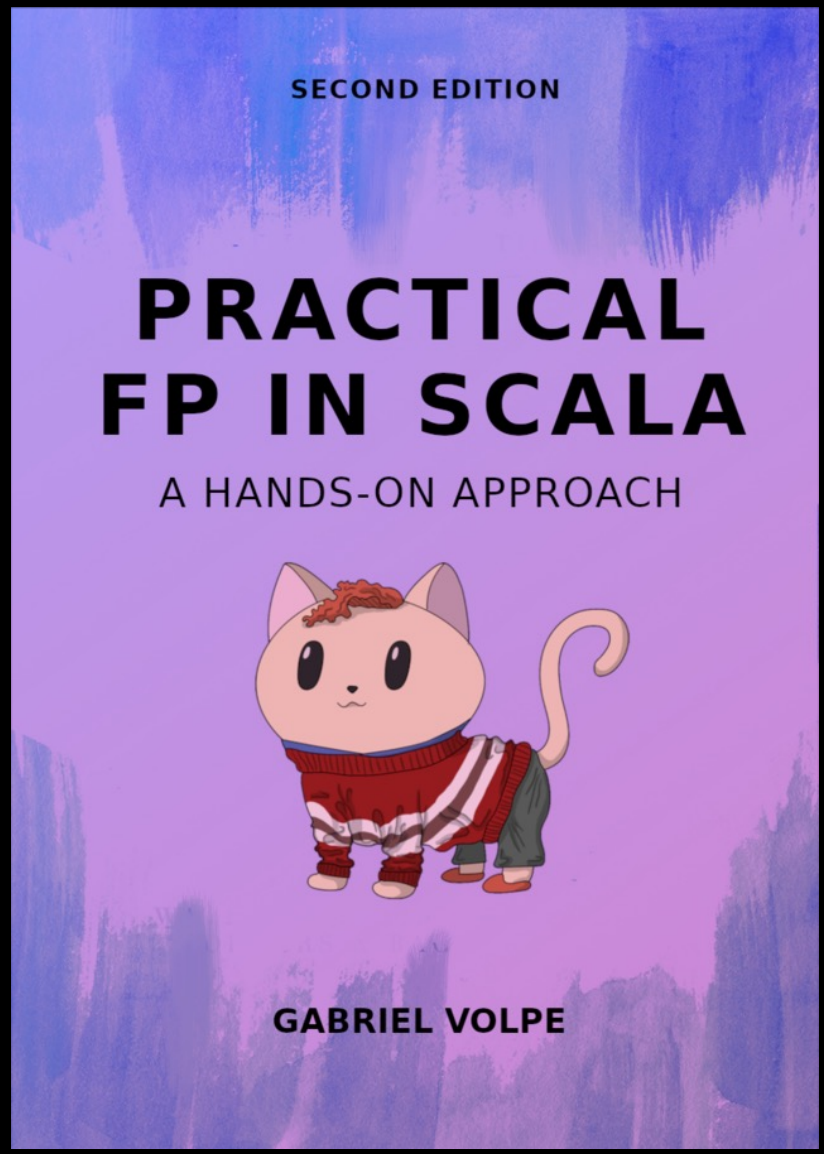
```scala
final case class CartRoutes[F[_]: JsonDecoder: Monad](
  shoppingCartService: ShoppingCart[F]
) extends Http4sDsl[F] {

  private[routes] val prefixPath = "/cart"

  private val httpRoutes: AuthedRoutes[CommonUser, F] =
    AuthedRoutes.of {

      // Get shopping cart
      case GET -> Root as user =>
        Ok(shoppingCartService.get(user.value.id))

      // Add items to the cart
      case authorisedRequest @ POST -> Root as user =>
        authorisedRequest.req.asJsonDecode[Cart].flatMap { shoppingCart =>
          shoppingCart.items
            .map { case (itemId, quantity) =>
              shoppingCartService.add(user.value.id, itemId, quantity)
            }
            .toList
            .sequence *> Created()
        }
```



gvolpe / **pfps-shopping-cart**

pfps-shopping-cart / modules / core / src / main / scala / shop / services / **ShoppingCart.scala**

with some minor renaming, to ease comprehension for anyone lacking context – see repo for original

```
28          // Add items to the cart
            case authorisedRequest @ POST -> Root as user =>
              authorisedRequest.req.asJsonDecode[Cart].flatMap { shoppingCart =>
31              shoppingCart.items
32                .map {
33                  case (itemId, quantity) =>
34                    shoppingCartService.add(user.value.id, itemId, quantity)
35                }
36                .toList
37                .sequence *> Created()
38            }
```

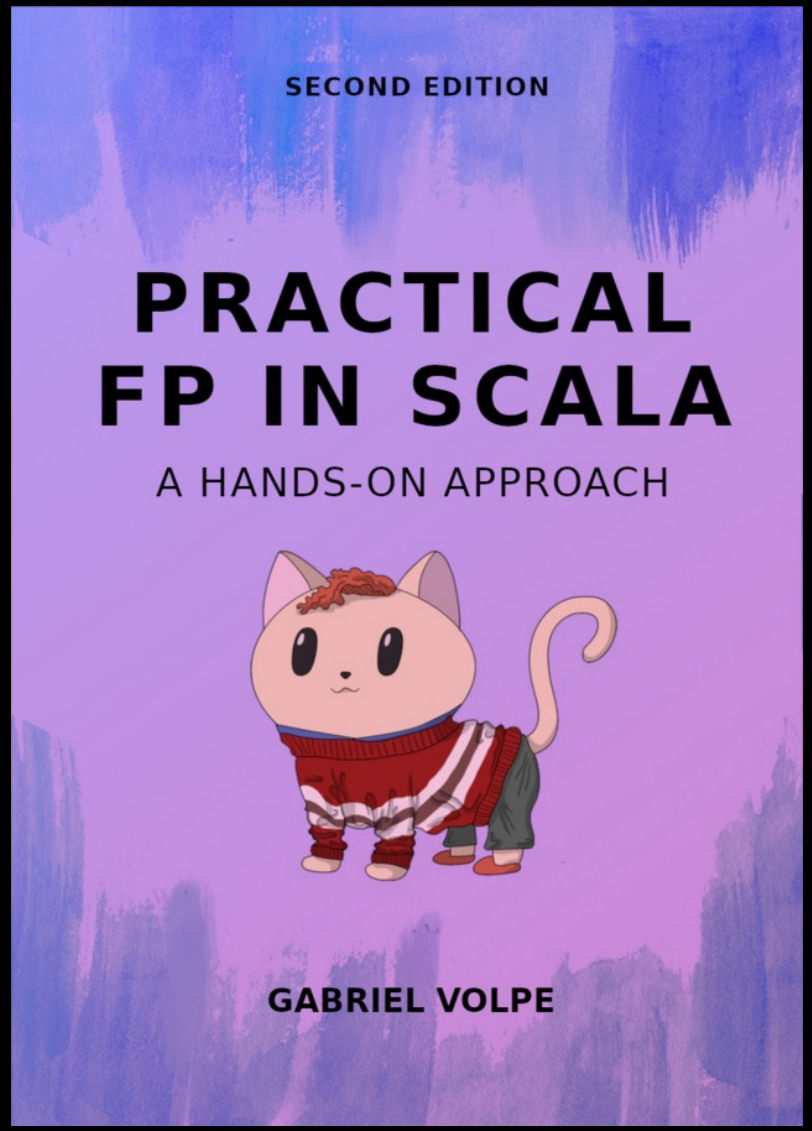
List[F[Unit]]

^⇧P Type Info

```
28          // Add items to the cart
            case authorisedRequest @ POST -> Root as user =>
              authorisedRequest.req.asJsonDecode[Cart].flatMap { shoppingCart =>
31              shoppingCart.items
32                .map {
33                  case (itemId, quantity) =>
34                    shoppingCartService.add(user.value.id, itemId, quantity)
35                }
36                .toList
37                .sequence *> Created()
38            }
```

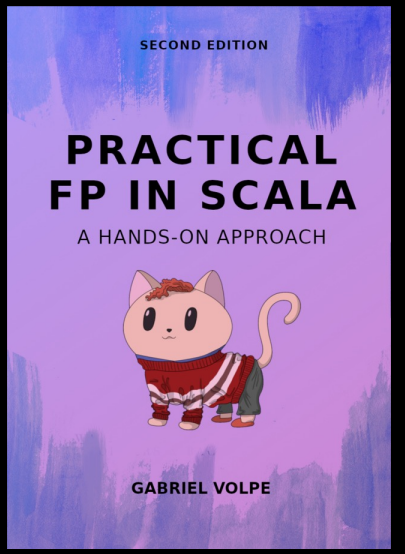
F[List[Unit]]

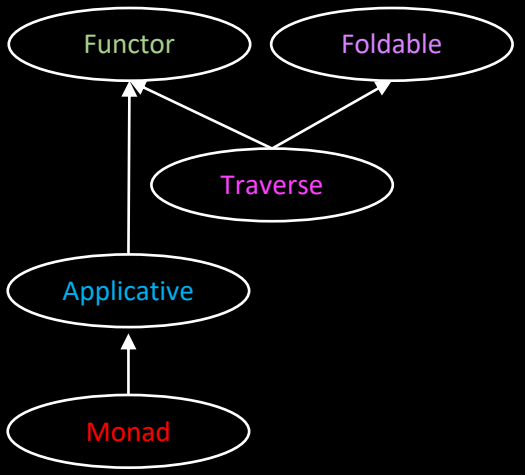
^⇧P Type Info



```scala
// Add items to the cart
case authorisedRequest @ POST -> Root as user =>
  authorisedRequest.req.asJsonDecode[Cart].flatMap { shoppingCart =>
    shoppingCart.items
      .map {
        case (itemId, quantity) =>
          shoppingCartService.add(user.value.id, itemId, quantity)
      }
      .toList
      .sequence *> Created()
  }
```

List[F[Unit]]






```scala
// Add items to the cart
case authorisedRequest @ POST -> Root as user =>
  authorisedRequest.req.asJsonDecode[Cart].flatMap { shoppingCart =>
    shoppingCart.items
      .map {
        case (itemId, quantity) =>
          shoppingCartService.add(user.value.id, itemId, quantity)
      }
      .toList
      .sequence *> Created()
  }
```

F[List[Unit]]

```scala
case class CartRoutes[F[_]: JsonDecoder: Monad]
```

```scala
List[F[Unit]] => F[List[Unit]]
```


```scala
@typeclass trait Traverse[F[_]] extends Functor[F] with Foldable[F] with …
  …
  Thread all the G effects through the F structure to invert the structure from F[G[A]] to G[F[A]].
  def sequence[G[_]: Applicative, A](fga: F[G[A]]): G[F[A]] =
    traverse(fga)(ga => ga)
```