SEPTIMAL MIND

IZUMI 1.0

YOUR NEXT SCALA STACK

WHO WE ARE

- Pavel and Kai
- Septimal Mind, an Irish consultancy
- Engineer’s productivity is our primary concern
- Love Scala and pure FP
- Scala contributors

WHAT WE DO

- We identify productivity issues in SDLC pipeline
- We build tools to address these issues

WHAT WE USE

- We've been using Scala for many years
- Bifuctors are beneficial, we've been using Scalactic
- We've adopted ZIO since first public alpha
  - It closed the everlasting question of error encoding

OPEN QUESTION

How do we design
complex but extensible
FP applications?

(Hundreds/thousands of components)

THE ANSWER

Modularity and modules

HOW TO GET THERE?

- Dependency Injection
- Tagless Final

WE MADE OUR OWN "NANOFRAMEWORK", IZUMI

- DIStage: Advanced Dependency Injection
- BIO: Tagless Final for Bifunctors and Trifunctors

Also:

- LogStage: Zero-Effort Structural logging
- etc

WHAT IS BIO?

- A set of typeclasses for Bifunctors and Trifunctors
- Like Cats but for F[+_, +_] and F[-_, +_, +_]
- Should be incorporated into ZIO Prelude in foreseeable future

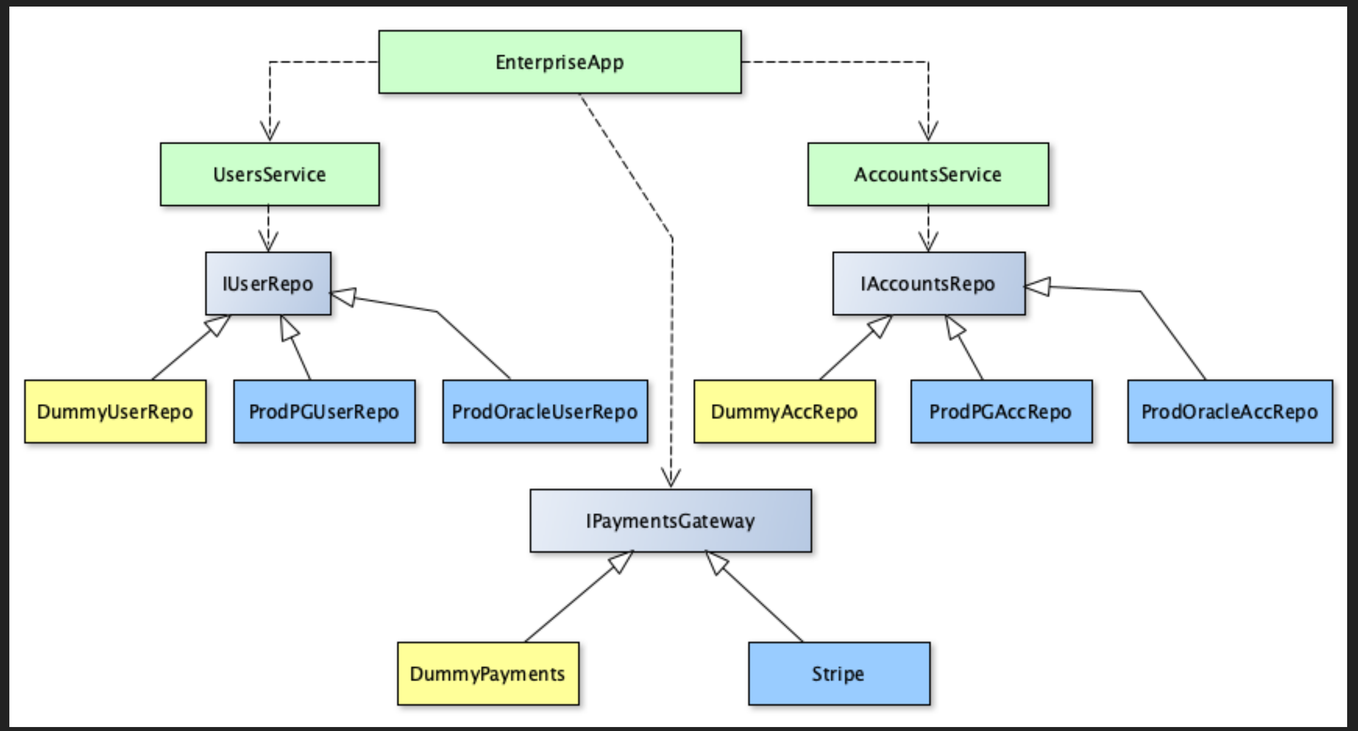
WHAT IS DISTAGE?

- Your regular Dependency Injection library
- Supports F[_]
- Unique feature: configurable applications
  - **Since 1.0: State-of-the-art compile-time verification**
- Many features: resources/lifecycles, integration checks, etc
- The most advanced DI/Module system available

WHAT IS A CONFIGURABLE APP?

- An application may run in several "modes"
  - E.g. Purpose={Test|Prod}, Database={Dummy|Postgres}
- Modes can be chosen at application startup
- Modes may be combined
- The app should choose right component implementations
  - And block incorrect combinations

WHAT IS A CONFIGURABLE APP?



- Purpose = Production | Test
- Database = Postgres | Oracle | Dummy
- We may want to run tests with Postgres database (but never with Oracle)
- We want to never run tests with production payment service
- We may want to define defaults
  - Database is not set for Prod run => use Postgres

CONFIGURABLE APP: HOW?

There was no good way to write configurable apps

CONFIGURABLE APP: HOW?

...hard to code, hard to maintain...

...exponential amount of code paths...

...edgecases...

CONFIGURABLE APP: HOW?

- It's hard to do it
  - ... even if you don't have transitive dependencies
  - ... even if you don't have dependent constraints
- It's even harder to provide compile-time guarantees

CONFIGURABLE APP: HOW?

```
def makeUserRepo(config: Config): IUserRepo[F] = {
  config.database match {
    case OracleDb =>
      if (!config.isProd) {
        throw new RuntimeException("Oracle unsupported in test mode!")
      } else {
        new ProdOracleUserRepo[F]( /*...*/ )
      }
    case PgDb =>
      new ProdPGUserRepo[F]( /*...*/ )
    case Unset =>
      if (config.isProd) {
        throw new RuntimeException("Database is not set for prod mode!")
      } else {
        new DummyUserRepo[F]()
      }
  }
}
```

CONFIGURABLE APP: HOW?

DIStage made it possible

Though it was doing things in runtime...

We thought compile-time solution is impossible...

CONFIGURABLE APP: HOW?

DIStage since Izumi 1.0:

strong compile-time guarantees

for

configurable apps

CONFIGURABLE APP WITH DISTAGE

```
class MyAppDefinition[F[+_, +_]: TagKK] extends PluginDef {
  make[EnterpriseApp[F]]
  make[AccountsService[F]]
  make[UsersService[F]]

  make[IUserRepo[F]].from[DummyUserRepo[F]].tagged(Mode.Test)
  make[IUserRepo[F]].from[ProdPGUserRepo[F]].tagged(Db.Pg)
  make[IUserRepo[F]].from[ProdOracleUserRepo[F]].tagged(Mode.Prod, Db.Oracle)

  make[IAccountsRepo[F]].from[DummyAccRepo[F]].tagged(Mode.Test)
  make[IAccountsRepo[F]].from[ProdPGAccRepo[F]].tagged(Db.Pg)
  make[IAccountsRepo[F]].from[ProdOracleAccRepo[F]].tagged(Mode.Prod, Db.Oracle)

  make[IPaymentsGateway[F]].from[DummyPayments[F]].tagged(Mode.Test)
  make[IPaymentsGateway[F]].from[StripePayments[F]].tagged(Mode.Prod)
}
```

COMPILE-TIME SAFETY

```
// your main method

object EnterpriseMain extends RoleAppMain.LauncherBIO[zio.IO] { ... }

// in test scope

object WiringTest extends PlanCheck.Main(EnterpriseMain)
```

COMPILE-TIME SAFETY

```
// your main method

object EnterpriseMain extends RoleAppMain.LauncherBIO[zio.IO] { ... }

// in test scope

object WiringTest extends PlanCheck.Main(EnterpriseMain)
```

HOW DOES IT WORK?

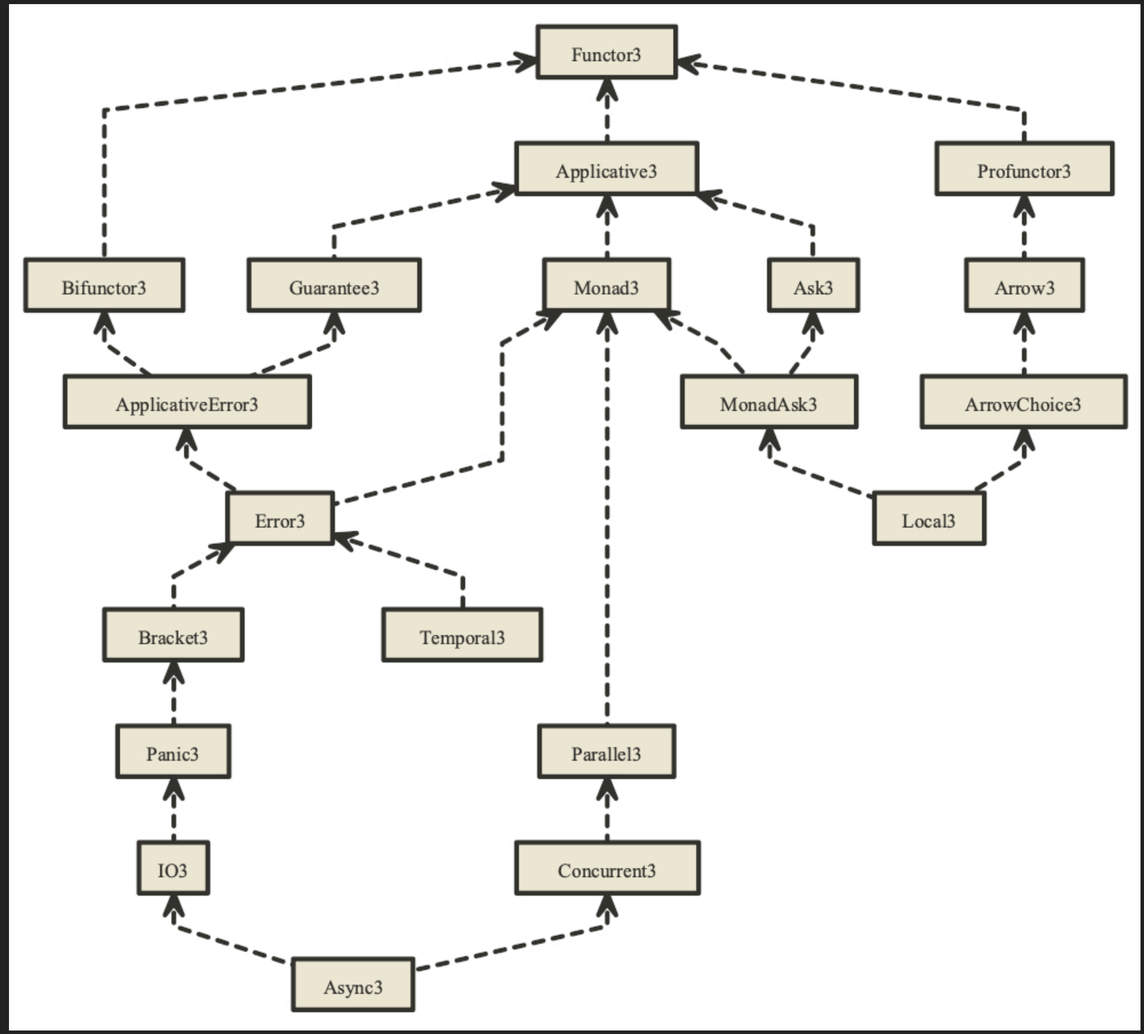
COMPILE-TIME SAFETY

```
// your main method

object EnterpriseMain extends RoleAppMain.LauncherBIO[zio.IO] { ... }

// in test scope

object WiringTest extends PlanCheck.Main(EnterpriseMain)
```

HOW DOES IT WORK?

```
abstract class PluginDef[T](implicit recompilationToken: RecompilationToken[T])
```

BIO TYPECLASSES



BIO SUMMONER

```
import izumi.functional.bio.F
```

```
class DummyUserRepo[F[+_, +_]: Monad2]
```

```
F.pure(...)
F.fail(...)
F.fromEither(...)
```

~~`Error2[F].fail(...)`~~

See also: https://www.youtube.com/watch?v=ZdGK1uedAE0&t=580s

BIO SYNTAX

```
import izumi.functional.bio.{F, Error2}
// IDE auto imports
```

```
def launchMissiles[F[+_, +_]: Error2](target: Target) =
  for {
    _ <- F.unless(storage.hasMissiles)(F.fail(AmmoDepleted))
    _ <- loadMissile(launcher, storage)
    _ <- fireMissile(launcher)
           .catchAll(_ => emergencyStop)
    _ <- F.unless(target.isAnnihilated)(launchMissiles(target))
  } yield res
```

```
import cats.syntax.all._
import cats.effect.syntax.all._
import monix.catnap.syntax._
```

COMPATIBILITY

```
import izumi.functional.bio.catz._
```

```
def http4sServer[F[+_, +_]: Async2: Fork2: UnsafeRun2] = {
  BlazeServerBuilder[F[Throwable, ?]](ec)
    .bindHttp(8080)
    .withSslContext(sslContext)
    .resource
}
```

~~ConcurrentEffect[F[Throwable, ?]]~~
~~MonadError[F[Throwable, ?], Throwable]~~

CONCLUSION

- BIO provides a great set of TF abstractions
- DIStage is great for global/static contexts
  - and ZIO's reader is perfect for local/dynamic contexts
- Izumi 1.0+ZIO is the most productive Scala stack
  - And battle-tested
- There is no excuse not to use DIStage anymore
- Consider it for your next Scala project

Thank you!

https://github.com/7mind/izumi

https://github.com/7mind/distage-example

https://twitter.com/shirshovp

https://twitter.com/kai_nyasha