distage

Modern Staged Dependency Injection for Scala

Modular Functional Programming
with
Context Minimization
through
Garbage Collection

Septimal Mind Ltd

team@7mind.io

DI is outdated and doesn't compose with FP?

Many Scala folks think that:

1. DI is heavy and slow
   - *"tests initialize longer than they execute"*
2. DI is unsafe
   - *"my program may compile but fail on startup after a huge delay"*
3. DI doesn't work for modern FP code
   - *"we cannot inject* `IO[_, _]` *into* `Repository[_[_, _]]`*"*
4. DI is full of magic and error-prone
   - *"I've read 80% out of 5000-page Spring manual but still don't understand why do I need to put these twelwe annotations here. Also I've tried Guice but it failed with 10-megabytes stack after five minutes and 300 retries of database connection instantiation"*

TLDR

```scala
import distage._, scalaz.zio.IO

trait Repository[F[_, _]] {}
class ProductionRepository[F[_, _]]() extends Repository[F]
class DummyRepository[F[_, _]]() extends Repository[F]
class App[F[_, _]](repository: Repository[F]) { def run = ??? }

class MyAppProd[F[_, _]] extends PluginDef {
  make[Repository[F]].from[ProductionRepository[F]]
  make[App[F]]
}
class Main[F[_, _]] { def main(args: Array[String]): Unit = {
  Injector()
    .produce(MyAppProd[F], roots = Set(DIKey.get[App[F]]))
    .run { app: App[F] => app.run() }
}}
object Main[IO]
```

distage: overview

1. ??
2. ???

Garbage Collection for better and faster tests

1. ??
2. ???

Garbage Collection for deployment: flexible monoliths

1. ??
2. ???

Config support

1. ??
2. ???

Run-time and compile-time injection

1. ??
2. ???

. . . and even more

1. ??
2. ???

How it works: Plans

1. ??
2. ???

Plans: introspection

1. ??
2. ???

Plans: extensibility

1. ??
2. ???

distage

7mind Stack

distage: status and things to do

distage is:

1. ready to use,
2. in real production,
3. all Run-time features are available,
4. all Compile-time features except of full Producer are available.

Our plans:

1. `ProducerF[_]` — Producer within a monad,
2. New Roles API,
3. Scala.js support,
4. Compile-time Producer,
5. Isolated Classloaders for Roles (in future),
6. Check our GitHub: https://github.com/pshirshov/izumi-r2.

distage is just a part of our stack

We have a vision backed by our tools:

1. Idealingua: transport and codec agnostic gRPC alternative with rich modeling language,
2. LogStage: zero-cost logging framework,
3. *Fusional Programming and Design* guidelines. We love both FP and OOP,
4. *Continous Delivery* guidelines for Role-based process,
5. *Percept-Plan-Execute* Generative Programming approach, abstract machine and computational model. Addresses Project Planning (see Operations Research). Examples: orchestration, build systems.

Altogether these things already allowed us to significantly reduce development costs and delivery time for our client.

More slides to follow.

You use Guice?
Switch to distage!

Teaser: LogStage

A log call . . .

```
log.info(s"$user logged in with $sessionId!")
```

. . . may be rendered as a text like `17:05:18 UserService.login user=John Doe logged in with sessionId=DEADBEEF!`

. . . or a structured JSON:

```
{
  "user": "John Doe",
  "sessionId": "DEADBEEF",
  "_template": "$user logged in with $sessionId!",
  "_location": "UserService.scala:265",
  "_context": "UserService.login",
}
```

Teaser: Idealingua

```
id UserId { uid: str }
data User {  name: str /* ... more fields */ }
data PublicUser {
 + InternalUser
 - SecurityAttributes
}
adt Failure = NotFound | UnknownFailure
service Service {
  def getUser(id: UserId): User !! Failure
}
```

1. Convenient Data and Interface Definition Language,
2. Extensible, transport-agnostic, abstracted from wire format,
3. JSON + HTTP / WebSocket at the moment,
4. C#, go, Scala, TypeScript at the moment,
5. Better than gRPC / REST / Swagger/ etc.

Thank you for your attention

distage website: https://izumi.7mind.io/

We're looking for clients, contributors, adopters and colleagues ;)

About the author:

1. coding for 18 years, 10 years of hands-on commercial engineering experience,
2. has been leading a cluster orchestration team in Yandex, "the Russian Google",
3. implemented "*Interstellar Spaceship*" – an orchestration solution to manage 50K+ physical machines across 6 datacenters,
4. Owns an Irish R&D company, https://7mind.io,
5. Contact: team@7mind.io,
6. Github: https://github.com/pshirshov
7. Download slides: https://github.com/7mind/slides/

Thank you for your attention

distage website: https://izumi.7mind.io/

We're looking for clients, contributors, adopters and colleagues ;)

About the author:

1. coding for 18 years, 10 years of hands-on commercial engineering experience,
2. has been leading a cluster orchestration team in Yandex, "the Russian Google",
3. implemented "*Interstellar Spaceship*" — an orchestration solution to manage 50K+ physical machines across 6 datacenters,
4. Owns an Irish R&D company, https://7mind.io,
5. Contact: team@7mind.io,
6. Github: https://github.com/pshirshov
7. Download slides: https://github.com/7mind/slides/