

# Dependencies & Layers



# Layers

Standard dependency injection is clunky:

- doesn't scale
- becomes hard to debug/understand
- needs serious discipline to do well
- instantiates services at the point of call
- can easily leak resources

```
val subscriptionService = ZIO.succeed(
  UserSubscription.create(
    EmailService.create(),
    UserDatabase.create(
      ConnectionPool.create(10)
    )
  )
)

def subscribe(user: User): ZIO[Any, Throwable, Unit] =
  for {
    sub <- subscriptionService
    _ <- sub.subscribeUser(user)
  } yield ()
```

# Layers

## Alternative: ZIO dependencies

- mark them as needed when they are needed
- pass them once at the "end of the world"

```
def subscribe_v2(user: User): ZIO[UserSubscription, Throwable, Unit] =  
  for {  
    sub <- ZIO.service[UserSubscription]  
    _ <- sub.subscribeUser(user)  
  } yield ()  
  
val program_v2 = subscribe_v2(User("Daniel", "daniel@rockthejvm.com")).provide(...)
```

## Benefits:

- business logic uncluttered by instantiation of dependencies
- resources used once at the end of the application
- logic easily testable by passing a different resource implementation (e.g. mocked one)

# Layers API

Like regular ZIOs

```
ZLayer.succeed(ConnectionPool.create(10))
```

Create a layer out of function arguments (magic macros)

```
ZLayer.fromFunction(UserDatabase.create _)
```

explicit eta-expansion needed in Scala 2

## Composing layers

- vertical: consumes dependencies of the first, produces values of the last
- horizontal: consumes dependencies of both, produces values of both

```
val databaseLayerFull: ZLayer[Any, Nothing, UserDatabase] =  
  connectionPoolLayer >>> databaseLayer  
val subscriptionRequirementsLayer: ZLayer[Any, Nothing, UserDatabase & EmailService] =  
  databaseLayerFull ++ emailServiceLayer
```

# Layers API

## Magic auto-wiring

- macro-based inspection of type signatures, graph connections
- rich compiler errors for missing layers, duplicate layers
- pretty-printed graph representation (text or Mermaid links)

```
val runnableProgram_v2 = program_v2.provide(  
  UserSubscription.live,  
  EmailService.live,  
  UserDatabase.live,  
  ConnectionPool.live(10),  
  // ZIO will tell you if you're missing a layer  
  // and if you have multiple layers of the same type  
  // and tell you the dependency graph!  
  ZLayer.Debug.mermaid  
)
```

**ZIO rocks**

