

ZIO Error Handling



Error Handling

Attempt: wrap an expression that might throw

```
val failedWithThrowable = ZIO.attempt(throw new RuntimeException("Boom!"))
```

Catch/catchAll: process potential errors

```
val catchError = anAttempt.catchAll(e => ZIO.succeed(s"Recovered from $e"))
```

Fold/foldZIO: process both success and failure

```
val handleBoth = anAttempt.foldZIO(  
  ex => ZIO.succeed(s"Something bad happened: $ex"),  
  value => ZIO.succeed(s"Length of the string was $value")  
)
```

Conversions between Try/Either/Option to ZIO

```
val aTryToZIO: ZIO[Any, Throwable, Int] = ZIO.fromTry(Try(42 / 0))  
val anEitherToZIO: ZIO[Any, Int, String] = ZIO.fromEither(Right("Good!"))  
val anOption: ZIO[Any, Option[Nothing], Int] = ZIO.fromOption(Some(42))
```

Errors and Defects

- Errors: expected failures, present in the type signature
- Defects: unforeseen failures, not present in the type signature

Turn failures into defects

```
val failedDefect: ZIO[Any, Nothing, String] = failedEffect.orDie
```

Narrow failure type, leave the rest as defects

```
failedEffect.refineOrDie[IOException] {  
  case e: IOException => e  
  case _: NoRouteToHostException => new IOException(s"No route to host")  
}
```

Treat failure causes, including defects

```
val foldedWithCause = failedInt.foldCauseZIO(  
  cause => ZIO.succeed(s"this failed with ${cause.defects}"),  
  value => ZIO.succeed(s"this succeeded with $value")  
)
```

ZIO rocks

