

Testing: Services & Dependencies



Testing With Dependencies

Dependencies & layers allow you to pass a "mocked" dependency to ZIOs

```
def spec = suite("A user survey application should...")(
  test("normalize user names") {
    val surveyPreliminaryLogic = for {
      db <- ZIO.service[Database[String, String]]
      _ <- db.put("123", "Daniel")
      username <- db.get("123")
      normalized <- normalizeUsername(username)
    } yield normalized

    assertZIO(surveyPreliminaryLogic)(Assertion.equalTo("DANIEL"))
  }
).provide(ZLayer.fromZIO(mockedDatabase))
```



provide this mocked service to all tests requiring it

Testing Services

```
def spec = suite("Checking built-in test services")(
  test("ZIO console application") {
    val logicUnderTest: Task[Vector[String]] = for {
      _ <- TestConsole.feedLines("Daniel")
      _ <- DummyConsoleApplication.welcomeUser()
      output <- TestConsole.output
    } yield assert(...)
  },

  test("ZIO clock") {
    val parallelEffect = for {
      fiber <- ZIO.sleep(5.minutes).timeout(1.minute).fork
      _ <- TestClock.adjust(1.minute)
      result <- fiber.join
    } yield assert(...)
  },

  test("ZIO Random") {
    val effect = for {
      _ <- TestRandom.feedInts(3,4,1,2)
      value <- Random.nextInt
    } yield assert(...)
  }
)
```

TestConsole

- "type" lines in
- fetch all output

TestClock

- manipulate time

TestRandom

- control randomness

Test Aspects

Descriptions of how a test should run

- `timeout(duration)`
- `eventually` - retries until successful
- `nonFlaky(n)` - repeats n times, stops at first failure
- `repeats(n)` - same
- `retries(n)` - retries n times, stops at first success
- `debug` - prints everything in the console
- `silent` - prints nothing
- `diagnose(duration)`
- `parallel/sequential` (aspects of a SUITE)
- `ignore`
- `success` - fail all ignored tests
- `timed` - measure execution time
- `before/beforeAll` + `after/afterAll`

```
def spec = suite("Testing Aspects")(
  test("timeout aspect") {
    val effect = for {
      molFib <- computeMeaningOfLife.fork
      _ <- TestClock.adjust(3.seconds)
      v <- molFib.join
    } yield v

    assertZIO(effect)(Assertion.equalTo(42))
  } @@ timeout(10.millis) @@ timed
)
```

ZIO rocks

