Appsilon

Getting The Most Out Of Test-Driven Development For Shiny

Jakub Sobolewski

1. What is TDD?
2. How to do TDD?
3. Why use TDD?

What is TDD?

What is TDD?

🏛️ A design technique.

What is TDD?

🏛️ A design technique.

📈 A guided evolution.

What is TDD?

🏛️ A design technique.

📈 A guided evolution.

🧪 Tests first.

3 steps of TDD

❌ Red

✅ Green

🔁 Refactor

3 steps of TDD

❌ **Red**

Write a failing test first. ⛔

✅ Green

🔁 Refactor

3 steps of TDD

❌ Red

✅ **Green**

Write the production code.

🔁 Refactor

3 steps of TDD

❌ Red

✅ **Green**

Write the production code.

Only as much as needed to make tests pass. ✋

🔁 Refactor

3 steps of TDD

❌ Red

✅ Green

🔁 **Refactor**

Don’t worry about breaking things.

3 steps of TDD

❌ Red

✅ Green

🔁 **Refactor**

Don’t worry about breaking things.

Make code more readable and maintainable. 💄

Example

❌ Red

sum.R

```
#
```

test-sum.R

```
describe("my_sum", {
  it("should calculate the sum for multiple numbers", {
    fail("Not implemented")
  })
})
```

```
── Failure: my_sum: should calculate the sum for multiple numbers
──────────────
Not implemented

Error:
! Test failed
```

❌ Red

sum.R

```r
#
```

test-sum.R

```r
describe("my_sum", {
  it("should calculate the sum for multiple numbers", {
    # Arrange
    numbers <- c(1, 2)

    # Act
    result <- my_sum(numbers)

    # Assert
    expect_equal(result, 3)
  })
})
```

```
── Error: my_sum: should calculate the sum for multiple numbers
────────────────

Error in `my_sum(numbers)`: could not find function "my_sum"

Error:
! Test failed
```

❌ Red

sum.R

```r
#
```

test-sum.R

```r
describe("my_sum", {
  it("should calculate the sum for multiple numbers", {
    # Arrange
    numbers <- c(1, 2)

    # Act
    result <- my_sum(numbers)

    # Assert
    expect_equal(result, 3)
  })
})
```

```
── Error: my_sum: should calculate the sum for multiple numbers
────────────────

Error in `my_sum(numbers)`: could not find function "my_sum"

Error:
! Test failed
```

❌ Red

sum.R

```r
#
```

test-sum.R

```r
describe("my_sum", {
  it("should calculate the sum for multiple numbers", {
    # Arrange
    numbers <- c(1, 2)

    # Act
    result <- my_sum(numbers)

    # Assert
    expect_equal(result, 3)
  })
})
```

```
── Error: my_sum: should calculate the sum for multiple numbers
────────────────

Error in `my_sum(numbers)`: could not find function "my_sum"

Error:
! Test failed
```

❌ Red

sum.R

```
#
```

test-sum.R

```r
describe("my_sum", {
  it("should calculate the sum for multiple numbers", {
    # Arrange
    numbers <- c(1, 2)

    # Act
    result <- my_sum(numbers)

    # Assert
    expect_equal(result, 3)
  })
})
```

```
── Error: my_sum: should calculate the sum for multiple numbers
────────────────
Error in `my_sum(numbers)`: could not find function "my_sum"
Error:
! Test failed
```

❌ Red

sum.R

```r
my_sum <- function(x) {

}
```

test-sum.R

```r
describe("my_sum", {
  it("should calculate the sum for multiple numbers", {
    # Arrange
    numbers <- c(1, 2)

    # Act
    result <- my_sum(numbers)

    # Assert
    expect_equal(result, 3)
  })
})
```

```
── Failure: my_sum: should calculate the sum for multiple numbers
──────────────

`result` not equal to 3.
target is NULL, current is numeric

Error:
! Test failed
```

✅ Green

sum.R

```r
my_sum <- function(x) {
  x[1] + x[2]
}
```

test-sum.R

```r
describe("my_sum", {
  it("should calculate the sum for multiple numbers", {
    # Arrange
    numbers <- c(1, 2)

    # Act
    result <- my_sum(numbers)

    # Assert
    expect_equal(result, 3)
  })
})
```

Test passed 🥳

❌ Red

sum.R

```r
my_sum <- function(x) {
  x[1] + x[2]
}
```

test-sum.R

```r
  it("should calculate the sum for one number", {
    # Arrange
    numbers <- c(1)

    # Act
    result <- my_sum(numbers)

    # Assert
    expect_equal(result, 1)
  })
})
```

```
Test passed 🎊
── Failure: my_sum: should calculate the sum for one number
──────────────────
`result` not equal to 1.
1/1 mismatches
[1] NA - 1 == NA

Error:
! Test failed
```

✅ Green

sum.R

```r
my_sum <- function(x) {
  sum <- 0
  for (number in x) {
    sum <- sum + number
  }
  sum
}
```

test-sum.R

```r
describe("my_sum", {
  it("should calculate the sum for multiple numbers", {
    # Arrange
    numbers <- c(1, 2)

    # Act
    result <- my_sum(numbers)

    # Assert
    expect_equal(result, 3)
  })

  it("should calculate the sum for one number", {
    # Arrange
    numbers <- c(1)
```

```
Test passed 😀
Test passed 🥳
```

🔁 Refactor

sum.R

```r
my_sum <- function(x) {
  base::sum(x)
}
```

test-sum.R

```r
describe("my_sum", {
  it("should calculate the sum for multiple numbers", {
    # Arrange
    numbers <- c(1, 2)

    # Act
    result <- my_sum(numbers)

    # Assert
    expect_equal(result, 3)
  })

  it("should calculate the sum for one number", {
    # Arrange
```

```
Test passed 🎉
Test passed 🎉
```

😏 That was easy.

😏 That was easy.

How do we use it for more complex problems?

2 approaches

📈 Bottom-up

📉 Top-down

2 approaches

📈 **Bottom-up**

Start small. 🧱

📉 Top-down

2 approaches

📈 **Bottom-up**

Start small. 🧱

Use existing pieces of codebase. 🧩

📉 Top-down

2 approaches

📈 **Bottom-up**

Start small. 🧱

Use existing pieces of codebase. 🧩

A design emerges. 🏛️

📉 Top-down

2 approaches

📈 Bottom-up

📉 **Top-down**

List high-level objectives first. 🎯

2 approaches

📈 Bottom-up

📉 **Top-down**

List high-level objectives first. 🎯

Describe behavior from user perspective. 🙌

2 approaches

📈 Bottom-up

📉 **Top-down**

List high-level objectives first. 🎯

Describe behavior from user perspective. 🙌

Fill in the gaps until tests pass. 🧩

🏎️ We do what helps us move fast.

🧪 Tests are the driving force.

🧪 Tests are the driving force.

They tell us if the code does what we think it does.

From requirements to tests

🙋‍♀️ 💬

🙋‍♂️ 💬

➡️

✅ Pass

✅ Pass

👍 Examples help us understand the problem.

👎 Abstract descriptions don’t.

Acceptance criteria

A list of verifiable statements.

📝 Acceptance criteria

A list of verifiable statements.

- ☑️ Something should do something.

Acceptance criteria

A list of verifiable statements.

- ☑ Something should do something.
- ☑ Something should do the other thing.

Acceptance criteria

A list of verifiable statements.

- ☑ Something should do something.
- ☑ Something should do the other thing.

```
describe("something", {
  it("should do something", {
    # Arrange

    # Act

    # Assert
  })

  it("should do the other thing", {

  })
})
```

📝 Test cases

🎯 Specific for low-level (unit) tests.

💬 Abstract for high-level (acceptance) tests.

📝 Test cases

🎯 **Specific for low-level (unit) tests.**

~~✅ It should work correctly.~~

💬 Abstract for high-level (acceptance) tests.

Test cases

🎯 **Specific for low-level (unit) tests.**

- ~~It should work correctly.~~
- It should throw an error when the table has 1 row.
- Abstract for high-level (acceptance) tests.

Test cases

🎯 Specific for low-level (unit) tests.

💬 **Abstract for high-level (acceptance) tests.**

☑️ ~~It should render the table after clicking the “Go” button.~~

📝 Test cases

🎯 Specific for low-level (unit) tests.

💬 **Abstract for high-level (acceptance) tests.**

- ~~It should render the table after clicking the “Go” button.~~
- It should show the summary.

Tests show we solved the problem.

📝 And document what we solved.

Drive communication with a client.

📊 Via executable examples on real data.

TDD x Shiny

TDD x Shiny

📦 The testable unit of an app is a Shiny module.

TDD x Shiny

📦 The testable unit of an app is a Shiny module.

✨ It represents a feature.

TDD x Shiny

📦 The testable unit of an app is a Shiny module.

✨ It represents a feature.

🖥️ It can be run as a standalone app.

Example

❌ Red

dataset_summary.R

```
#
```

test-dataset_summary.R

```
describe("dataset summary", {
  it("should show the summary", {
    fail("Not implemented")
  })
})
```

```
── Failure: dataset summary: should show the summary
──────────────────────────────────────────────────
Not implemented

Error:
! Test failed
```

❌ Red

dataset_summary.R

```r
#
```

test-dataset_summary.R

```r
describe("dataset summary", {
  it("should show the summary", {
    # Arrange
    app <- shinyApp(
      ui = dataset_summary_ui(NULL),
      server = \(input, output) dataset_summary_server(NULL)
    )
    page <- DatasetSummary$new(app)

    # Act

    # Assert
    page$expect_summary_visible()
    fail("Not implemented")
```

```
── Error: dataset summary: should show the summary
──────────────────────────────
Error in `dataset_summary_ui(NULL)`: could not find function
"dataset_summary_ui"
Backtrace:
    ■
 1. └─shiny::shinyApp( ... )
 2.   └─shiny:::uiHttpHandler(ui, uiPattern)
 3.     └─base::force(ui)

Error:
! Test failed
```

❌ Red

dataset_summary.R

```r
#
```

test-dataset_summary.R

```r
describe("dataset summary", {
  it("should show the summary", {
    # Arrange
    app <- shinyApp(
      ui = dataset_summary_ui(NULL),
      server = \(input, output) dataset_summary_server(NULL)
    )
    page <- DatasetSummary$new(app)

    # Act

    # Assert
    page$expect_summary_visible()
    fail("Not implemented")
  })
```

```
── Error: dataset summary: should show the summary ──────────────────────────
Error in `dataset_summary_ui(NULL)`: could not find function
"dataset_summary_ui"
Backtrace:
    ▆
 1. └─shiny::shinyApp( ... )
 2.   └─shiny:::uiHttpHandler(ui, uiPattern)
 3.     └─base::force(ui)

Error:
! Test failed
```

❌ Red

dataset_summary.R

```r
#
```

test-dataset_summary.R

```r
    # Arrange
    app <- shinyApp(
      ui = dataset_summary_ui(NULL),
      server = \(input, output) dataset_summary_server(NULL)
    )
    page <- DatasetSummary$new(app)

    # Act

    # Assert
    page$expect_summary_visible()
    fail("Not implemented")
  })
})
```

```
── Error: dataset summary: should show the summary
────────────────────────────────
Error in `dataset_summary_ui(NULL)`: could not find function
"dataset_summary_ui"
Backtrace:
    ■
 1. └─shiny::shinyApp( ... )
 2.   └─shiny:::uiHttpHandler(ui, uiPattern)
 3.     └─base::force(ui)

Error:
! Test failed
```

❌ Red

dataset_summary.R

```r
#
```

test-dataset_summary.R

```r
    # Arrange
    app <- shinyApp(
      ui = dataset_summary_ui(NULL),
      server = \(input, output) dataset_summary_server(NULL)
    )
    page <- DatasetSummary$new(app)

    # Act

    # Assert
    page$expect_summary_visible()
    fail("Not implemented")
  })
})
```

```
── Error: dataset summary: should show the summary
────────────────────────────────
Error in `dataset_summary_ui(NULL)`: could not find function
"dataset_summary_ui"
Backtrace:
    ■
 1. └─shiny::shinyApp( ... )
 2.   └─shiny:::uiHttpHandler(ui, uiPattern)
 3.     └─base::force(ui)

Error:
! Test failed
```

❌ Red

dataset_summary.R

```r
dataset_summary_ui <- function(id) {
  fluidPage()
}

dataset_summary_server <- function(id) {
  moduleServer(id, function(input, output, session) {

  })
}
```

test-dataset_summary.R

```r
describe("dataset summary", {
  it("should show the summary", {
    # Arrange
    app <- shinyApp(
      ui = dataset_summary_ui(NULL),
      server = function(input, output) dataset_summary_server
    )
    page <- DatasetSummary$new(app)

    # Act

    # Assert
    page$expect_summary_visible()
    fail("Not implemented")
```

```
── Error: dataset summary: should show the summary
────────────────────────────────

Error in `eval(code, test_env)`: object 'DatasetSummary' not
found

Error:
! Test failed
```

❌ Red

dataset_summary.R

```r
dataset_summary_ui <- function(id) {
  fluidPage()
}

dataset_summary_server <- function(id) {
  moduleServer(id, function(input, output, session) {

  })
}
```

test-dataset_summary.R

```r
DatasetSummary <- R6::R6Class(
  classname = "DatasetSummary",
  public = list(
    driver = NULL,
    initialize = function(app) {
      self$driver <- AppDriver$new(app)
    },
    expect_summary_visible = function() {

    }
  )
)

describe("dataset summary", {
```

```
── Failure: dataset summary: should show the summary
────────────────────────────

Not implemented

Error:
! Test failed
```

❌ Red

dataset_summary.R

```r
dataset_summary_ui <- function(id) {
  fluidPage()
}

dataset_summary_server <- function(id) {
  moduleServer(id, function(input, output, session) {

  })
}
```

test-dataset_summary.R

```r
describe("dataset summary", {
  it("should show the summary", {
    # Arrange
    app <- shinyApp(
      ui = dataset_summary_ui(NULL),
      server = function(input, output) {
        data_provider <- list(get_summary = \() {
          iris
        })
        dataset_summary_server(NULL, data_provider)
      }
    )
    page <- DatasetSummary$new(app)

    # Act
```

```
── Failure: dataset summary: should show the summary
──────────────────────────────

Not implemented

Error:
! Test failed
```

❌ Red

dataset_summary.R

```r
dataset_summary_ui <- function(id) {
  fluidPage()
}

dataset_summary_server <- function(id, data_provider) {
  moduleServer(id, function(input, output, session) {

  })
}
```

test-dataset_summary.R

```r
describe("dataset summary", {
  it("should show the summary", {
    # Arrange
    app <- shinyApp(
      ui = dataset_summary_ui(NULL),
      server = function(input, output) {
        data_provider <- list(get_summary = \() {
          iris
        })
        dataset_summary_server(NULL, data_provider)
      }
    )
    page <- DatasetSummary$new(app)

    # Act
```

```
── Failure: dataset summary: should show the summary
──────────────────────────────

Not implemented

Error:
! Test failed
```

❌ Red

dataset_summary.R

```r
dataset_summary_ui <- function(id) {
  fluidPage()
}

dataset_summary_server <- function(id, data_provider) {
  moduleServer(id, function(input, output, session) {

  })
}
```

🎯 {selenider}

test-dataset_summary.R

```r
  public = list(
    driver = NULL,
    selenider = NULL,
    initialize = function(app) {
      self$driver <- AppDriver$new(app)
      self$selenider <- selenider_session(
        driver = self$driver,
        local = FALSE
      )
    },
    expect_summary_visible = function() {
      find_element(self$selenider, css = "#summary > table")
        elem_expect(is_visible) %>%
        elem_expect(\(x) has_at_least(elem_children(x), 1))
```

```
── Failure: dataset summary: should show the summary
──────────────────────────

Condition failed:
`is_visible`
ℹ `x` is not visible.
Caused by error in `is_visible()`:
! `x` does not exist in the DOM.

Where `x` is:
A selenider element selecting:
The first element with css selector "#summary > table".

Error:
```

❌ Red

dataset_summary.R

```r
dataset_summary_ui <- function(id) {
  fluidPage()
}

dataset_summary_server <- function(id, data_provider) {
  moduleServer(id, function(input, output, session) {

  })
}
```

🎯 {selenider}

test-dataset_summary.R

```r
      self$selenider <- selenider_session(
        driver = self$driver,
        local = FALSE
      )
    },
    expect_summary_visible = function() {
      find_element(self$selenider, css = "#summary > table")
        elem_expect(is_visible) %>%
        elem_expect(\(x) has_at_least(elem_children(x), 1))
    }
  )
)

describe("dataset summary", {
  it("should show the summary", {
```

```
── Failure: dataset summary: should show the summary
──────────────────────────

Condition failed:
`is_visible`
ℹ `x` is not visible.
Caused by error in `is_visible()`:
! `x` does not exist in the DOM.

Where `x` is:
A selenider element selecting:
The first element with css selector "#summary > table".

Error:
```

✅ Green

dataset_summary.R

```r
dataset_summary_ui <- function(id) {
  ns <- NS(id)
  fluidPage(
    tableOutput(ns("summary"))
  )
}

dataset_summary_server <- function(id, data_provider) {
  moduleServer(id, function(input, output, session) {
    output$summary <- renderTable({
      data_provider$get_summary()
    })
  })
}
```

test-dataset_summary.R

```r
    # Arrange
    app <- shinyApp(
      ui = dataset_summary_ui(NULL),
      server = function(input, output) {
        data_provider <- list(
          get_summary = function() {
            iris
          }
        )
        dataset_summary_server(NULL, data_provider)
      }
    )
    page <- DatasetSummary$new(app)

    # Act

    # Assert
    page$expect_summary_visible()
  })
})
```

Test passed 🐱

🔁 Refactor – UI 🖼️

dataset_summary.R

```r
dataset_summary_ui <- function(id) {
  ns <- NS(id)
  fluidPage(
    tableOutput(ns("summary"))
  )
}

dataset_summary_server <- function(id, data_provider) {
  moduleServer(id, function(input, output, session) {
    output$summary <- renderTable({
      data_provider$get_summary()
    })
  })
}
```

test-dataset_summary.R

```r
if (interactive()) {
  shinyApp(
    ui = dataset_summary_ui(NULL),
    server = function(input, output) {
      data_provider <- list(
        get_summary = function() {
          iris
        }
      )
      dataset_summary_server(NULL, data_provider)
    }
  )
}

DatasetSummary <- R6::R6Class(
  classname = "DatasetSummary",
  public = list(
    driver = NULL,
    selenider = NULL,
    initialize = function(app) {
```

Test passed 🌈

🔁 Refactor – UI 🖼️

dataset_summary.R

```r
dataset_summary_ui <- function(id) {
  ns <- NS(id)
  fluidPage(
    tableOutput(ns("summary"))
  )
}

dataset_summary_server <- function(id, data_provider) {
  moduleServer(id, function(input, output, session) {
    output$summary <- renderTable({
      data_provider$get_summary()
    })
  })
}
```

test-dataset_summary.R

```r
if (interactive()) {
  shinyApp(
    ui = dataset_summary_ui(NULL),
    server = function(input, output) {
      data_provider <- list(
        get_summary = function() {
          iris
        }
      )
      dataset_summary_server(NULL, data_provider)
    }
  )
}

DatasetSummary <- R6::R6Class(
  classname = "DatasetSummary",
  public = list(
    driver = NULL,
    selenider = NULL,
    initialize = function(app) {
```

Test passed 🌈

🔁 Refactor – UI 🖼️

dataset_summary.R

```
1  dataset_summary_ui <- function(id) {
2    ns
3    fl
4      ta
5    )
6  }
7
8  dat
9    ns
10     o
11
12     }
13   })
14 }
```

test-dataset_summary.R

```r
if (interactive()) {
  shinyApp(
    ui = dataset_summary_ui(NULL),
    server = function(input, output) {
      data_provider <- list(
        get_summary = function() {
          iris
        }
      )
      dataset_summary_server(NULL, data_provider)
    }
  )
}

DatasetSummary <- R6::R6Class(
  classname = "DatasetSummary",
  public = list(
    driver = NULL,
    selenider = NULL,
    initialize = function(app) {
```

Test passed 🌈

Feature 👔

📦

Feature 👔

🔴 Fail

📦

Feature 👔

✅ Pass

📦

Feature 👔

✅ Pass

🔴 Fail

📦

Feature 👔

✅ Pass

✅ Pass

📦

Feature 👔

✅ Pass

✅ Pass

🔴 Fail

📦

Feature 👔

✅ Pass

✅ Pass

✅ Pass

📦 Feature 👔

✅ Pass

✅ Pass

✅ Pass

📦 Feature 👔

📦

Feature 👔

✅ Pass

✅ Pass

✅ Pass

📦

Feature 👔

🔴 Fail

📦

Feature 👔

✅ Pass

✅ Pass

✅ Pass

📦

Feature 👔

✅ Pass

Feature 👔

✅ Pass

✅ Pass

✅ Pass

Feature 👔

✅ Pass

Feature 👔

📦

Feature 👔

✅ Pass

✅ Pass

✅ Pass

📦

Feature 👔

✅ Pass

📦

Feature 👔

🔴 Fail

📦

Feature 👔

✅ Pass

✅ Pass

✅ Pass

📦

Feature 👔

✅ Pass

📦

Feature 👔

✅ Pass

📦

Feature 👔

✅ Pass

✅ Pass

✅ Pass

📦

Feature 👔

✅ Pass

📦

Feature 👔

✅ Pass

🧪 Unit tests

Why use TDD?

Why use TDD?

🏎️ Faster development cycles.

Why use TDD?

🏎️ Faster development cycles.

🧠 Understanding the problem.

Why use TDD?

🏎️ Faster development cycles.

🧠 Understanding the problem.

🛡️ Confidence that it works.

Thank you!

linkedin.com/in/jakub-sobolewski-r/

jakubsob.github.io/blog

github.com/jakubsob/cucumber