Commander

Command Line Interface Functional Testing

George Lesica - November 2020

“Software and cathedrals are much the same: first we build them, then we pray.”

Anonymous

Unit Testing

How it works

- Break software into self-contained “units”
- Test each one in isolation
- Units can be functions, classes, methods, or whatever makes sense

Strengths

- Write tests as you write code
- Mapping from failure to bug is usually easy
- Testing requires little domain knowledge

Weaknesses

- White box
- Bugs occur at interfaces
- Users don’t use one unit at a time

Functional Testing

How it works

- Run the program, see if it works
- Test from the user’s perspective
- Verify functionality, not implementation

Strengths and weaknesses

- More “practical”
- Require domain knowledge
- Verify the user experience
- Take longer to run

“More than the act of testing, the act of designing tests is one of the best bug preventers known.”

Boris Beizer

Commander

Introduction

- https://github.com/commander-cli/commander
- Written in Go
- Provide input, assert output
- Allows “gold file” testing
- Tests written in YAML or JSON

Example 1

```
tests:
  hello world:
    command: python -c “print(‘hello world’)”
    stdout: hello world
```

Run Example 1

```
➜ ./commander test example-1.yaml
Starting test file example-1.yaml...

✓ [local] hello world

Duration: 0.058s
Count: 1, Failed: 0, Skipped: 0
```

More Resources

- https://github.com/commander-cli/commander
- https://github.com/TravisWheelerLab/InstitutionalMemory/wiki/Functional-Testing-with-Commander
- https://github.com/glesica/commander-tutorial