End-to-end tests rely on abstractions

```
def test_e2e(self, driver, orders):
    # disambiguate the order

    # for each item in the order:
    # find it
    # add it to the cart

    # instantiate the pricing model
    # validate the expected price with the actual price from the website

    # add the custom lettering

    # place the order
    # check the DB, verity order details and status

    # instantiate the API EOM
    # query the API for the custom lettering, verify it matches
```

Next Steps

We have discussed:

- The architecture of a simple test automation framework.
- Wrappers for applications under test.
- Business process abstractions (“models”).
- Writing end-to-end tests.

Where do you go from here?

- Add more applications.
- Wrappers for integrations and systems to support deeper checks, for example database validations.
- Containerize the test framework.
- Hook the test framework up to a CI pipeline like Travis or Jenkins.
- Improve your models! Move models into their own projects.
- Plan the next iteration of the framework.

End-to-end tests rely on abstractions

Example end-to-end test case for our pretend t-shirt commerce site:

```
def test_e2e(self, driver, orders):
    # disambiguate the order

    # for each item in the order:
    # find it
    # add it to the cart

    # instantiate the pricing model
    # validate the expected price with the actual price from the website

    # add the custom lettering

    # place the order
    # check the DB, verity order details and status

    # instantiate the API EOM
    # query the API for the custom lettering, verify it matches
```