Plumbing

To Recap...

We have discussed most of the core elements of a custom test automation framework:

- A testrunner to collect, run, and report on tests. Pytest is the glue that holds together the test automation framework, and calls on pytest are scattered throughout the framework. Pytest is how the framework is invoked and run. When you integrate your framework in a build system or CI pipeline, you are calling pytest from the command line.
- The framework contains some specialized tools that are used by the tests to interact with applications under test or perform other test-related logic.
- The framework has wrappers for the application to act as a sort of internal interface for the test cases.
- We have data models that keep data out of the test case files. These data models parametrization of the test cases, so that a few general test cases can be dynamically expanded into instances for each data member.
- We've tried hard to refine the test cases to the core test-specific logic and abstract out anything that doesn't belong in the test cases.

All of this has been the foundation of the most important part of a custom framework: the business models.

Plumbing

Everything we've discussed so far has been the basic structure setup for the meat of test automation: business process abstractions that support end-to-end automation.

Our app wrappers support extensive deep functional tests. Our business abstractions will let us define happy paths through our functional affordances so that we can focus on user journeys.