Full Stack Development
Sesi 12

TESTING

Why Testing

Tests are a great debugging aid. Writing a test that produces the invalid behavior we are seeing helps us look at our code from a different perspective. In addition, once we have the test passing we have ensured that we will not re-introduce this bug again (at least in that particular way).

Tests are also an excellent source of documentation. Since they have to deal with expected inputs and outputs reading a test suite will clarify what the code under test is expected to do. This will illuminate unclear segments of the documentation that we have written (or in simple cases, even substitute for it).

Finally, tests can be a wonderful exploratory aid - sketching out how we want to interact with our code before we write it reveals simpler APIs, and helps us paper over the internal complexity of a domain. “Test Driven Development” is the ultimate commitment to this process. In TDD we first write tests to cover our code’s functionality and only then do we write that code.

Tests will make it obvious:

- When code isn’t working,
- What code is broken, and
- Why we wrote this code in the first place.
- Every time we go to add a feature to our application, fix a bug or change some code we should make sure our code is adequately covered by tests and that the tests all pass after we’re done.

Do and Don’t

Do:

- Add tests to cover the basic functionality of your code.
- Add tests to cover as many corner/edge cases of your code that you can think of.
- Add tests to cover the corner/edge cases you didn’t think of after you go back and fix them.
- Remind your coding peers to adequately test their code.
- Bug people about code that doesn’t pass tests.

Do Not:

- Commit code without tests.
- Commit code that doesn’t pass or breaks tests.
- Change your tests so your code passes without fixing the problem.

External References

Code for today's lecture - Visit Here