

Test-driven development quick reference guide

Gunjan Doshi [<http://www.gunjandoshi.com>]

Email: Gunjan At instrumentalservices.com

<p>What is a unit test? A 'unit test' tests very specific, much focused, single aspect of functionality.</p> <p>Characteristics of a unit test:</p> <ul style="list-style-type: none">▪ Focused, making only one claim at a time.▪ Fast to run.▪ Independent of<ul style="list-style-type: none">○ Each other.○ The environment.○ The order in which it is run.▪ Same high quality as the production code.▪ Automatic. <p>When it is not a unit test? A test that does not operate in isolation is not a unit test. It is safe to assume that a test that connects to the network or a database or a real file is not a unit test. Use mock objects or stubs to test in isolation.</p> <p>Being test-driven</p> <ul style="list-style-type: none">▪ Being test-driven is not about testing, but about evolving the design to meet the requirements.▪ Each unit test corresponds to a single requirement that the code must satisfy.	<p>Test-Driven Cycle:</p> <ul style="list-style-type: none">▪ Prepare a list of test cases.▪ Follow the TDD Rhythm:<ul style="list-style-type: none">○ Pick a test to implement.○ Write a failing test.○ Quickly, make the test green.○ Refactor to eliminate duplication. <p>How do I pick the next test to implement? Pick a test that</p> <ul style="list-style-type: none">▪ You are confident that you can implement.▪ You know is the next logical step and will give you the most learning. <p>Before writing the next test, ask:</p> <ul style="list-style-type: none">▪ Is there any duplication? Do I need to refactor first?▪ What are the correct inputs? How am I going to check for the effect of these inputs?▪ Where does the operation belong? <p>What should I test? Test everything that could possibly break.</p> <p>What not to test?</p> <ul style="list-style-type: none">▪ Do not test getters and setters.▪ It is almost impossible to unit test the GUI, so keep the GUI as thin as possible.	<p>Should I test private methods? Testing the public interface of a class should be sufficient. Public interface should invoke the private methods</p> <p>Things to always do:</p> <ul style="list-style-type: none">▪ Write test first.▪ Maintain a to-do list.▪ Write new code, only if, there is a failing automated unit test.▪ Run all the tests all the time and not just a single isolated test.▪ Keep the test code of the highest quality without any code smells. <p>Things to never do:</p> <ul style="list-style-type: none">▪ Write multiple failing tests at a time.▪ Write new test without eliminating duplication.▪ Commit code with failing tests. <p>What is a to-do list?</p> <ul style="list-style-type: none">▪ To-do list contains<ul style="list-style-type: none">○ Examples of test cases.○ Any refactoring that needs to be done.▪ To-do list should be kept current. At any time, it should tell the following:<ul style="list-style-type: none">○ Current test or refactoring being carried out.○ Pending tests.○ Pending refactorings.
--	--	--