Vaadin TestBench Testing Practices.

Glossary:

Test(s)	-	Vaadin TestBench IDE .html test
Test suite	-	Vaadin TestBench IDE .html test suite file

Recording tests

Vaadin TestBench tests are recorded using the Vaadin TestBench IDE plugin for FireFox3.

If a test is long it would be advisable to make the test in multiple parts and put them in order as a test suite. This will help in debugging and finding where a test fails much easier.

Test suites should always be combined from an empty test suite if there is a need to reorganize test order. This is due to the IDE not supporting reordering of test cases.

Good practices

Before beginning testing it would be good to define standards and naming conventions for tests. This will make maintenance easier and the test cases more readable.

When a bug is found in the code or UI. If possible create a test that finds the bug and fails. After this, fix the bug so that the test passes. This way you get a new test that helps against regression and you get more code coverage.

All tests (cases and suites) should be made with the system initialized to a predetermined state.

Rules of Thumb:

1. Always test using sandbox data
   Test cases should work on sandbox data that is initialized for each test (suite) so that there are no dependencies.
2. Stand-alone tests (Atomic tests)
   Minimize interaction between tests.
3. Conditions
   Prefer waiting for conditions to occur instead of using pauses in tests.
4. Prefer smaller tests
   Separate tests per business module. Each test should only test one part of the system.
5. Single responsibility
   One behavior, multiple methods → One test
   One method, multiple behaviors → Multiple tests
6. Tests should be self-descriptive
   The names of the test cases should be informative and descriptive.
7. Use consistent test data
   Test data should be consistent for all runs.
8. Isolated tests (test suites)
   Tests (test suites) should not be depend on the state of a previous test (suite). (no state sharing)
   Different execution order => same results.
9. Environment isolation (Mock)
   Tests should be isolated from environmental influences.

To take into consideration when creating test cases/suites

All tests (test suites) should be consistent. Multiple invocations on a test (test suite) should consistently return true or consistently return false if no changes have been made. This means that tests (test suites) can not depend on the environment (different dates, operating systems and hardware configurations) or use methods that return random data.

Tests (test suites) should only test behaviors not methods. This means that one test (test suite) may have multiple assertions. This often makes tests more readable and easier to understand.

The order of execution of tests (test suites) should not/cannot matter. Tests (test suites) should be independent from one another and changes in one test (test suite) cannot affect the result of other tests (test suites). When tests (test suites) share states it becomes more difficult to find the cause of a failure, because a test can fail even though behavior of tested functionality is correct.

Common actions used often or in multiple test cases should be moved to separate html test files. This also makes the code easier to reuse.

Testing should be fully automated and not require manual involvement. Special environments should be generated automatically before tests (test suites) are run.

Production code should not include in itself any test logic.

Required database and other setups for tests should be done using a special servlet that is deployed on the development server with the test build. Note that this servlet should never be added to a release build (not even as a disabled module).