Build and Source Control - Feature #1887

Feature # 1886 (Closed): implement a unit testing framework

research unit testing frameworks and make an appropriate selection

10/31/2012 04:47 PM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Greg Shah	**% Done:**	100%
Category:		**Estimated time:**	16.00 hours
Target version:			
billable:	No	**vendor_id:**	GCD

Description

History

#1 - 10/31/2012 04:53 PM - Greg Shah

- Estimated time set to 16.00

JUnit is popular and well supported, so it should be on the short list. We are open to other choices as well if there are compelling reasons.

A VERY IMPORTANT consideration: in accommodating unit testing, we will NOT:
- make massive changes to our current classes
- choose completely different designs for future work
- implement abstraction layers whose only purpose is to enable testing frameworks

Any unit testing framework must be able to work within the above constraints.

#2 - 11/16/2016 01:16 PM - Greg Shah

- Target version deleted (24)

#3 - 03/03/2021 04:15 PM - Greg Shah

The most obvious frameworks to consider are JUnit and TestNG.

Comparing JUnit and TestNG:

https://www.baeldung.com/junit-vs-testng
https://www.javacodegeeks.com/2016/09/junit-vs-testng-testing-framework-choose.html

A quick summary:

- Both frameworks have the same core features and both are used enough to be reasonable choices.
- JUnit
  - Was made first and is much more popular.
  - Has caught up with TestNG from a feature perspective in most cases.
  - Is better for:
    - running tests
    - community/support
- TestNG
  - Has the benefit of being designed later and taking lessions from early JUnit versions.
  - It is less popular which means it is not as well supported.
  - Is better for:
    - test grouping
    - provides hooks for suite and group level pre/post processing
    - a limited form of dependencies (still not very good, IMO)

- parallelism
- built-in reports for the output

Functionally, one would probably lean toward TestNG but JUnit is really universal so there is a definite advantage there.

Integration Testing

A separate question is whether JUnit or TestNG are also suitable for integration testing. A few articles:

https://stackoverflow.com/questions/2584629/how-can-i-specify-junit-test-dependencies
https://github.com/junit-team/junit4/wiki/Assumptions-with-assume
https://stackoverflow.com/questions/284774/can-we-use-junit-for-automated-integration-testing

My assessment is **NO**. Although there are some lightweight test dependency capabilities and TestNG has grouping + parallelism, neither framework is really designed for sequences of tests, let alone the multi-user complex business flows that we need. The Harness is far superior for that kind of testing.

I am OK using one of these frameworks for pure unit tests in FWD. But we won't be using them for more complex test scenarios.

#4 - 03/08/2021 07:09 AM - Greg Shah

- % Done changed from 0 to 100

- Status changed from New to Closed

- Start date deleted (10/31/2012)

- Assignee set to Greg Shah

Both Hynek and Vladimir have voiced their preference for JUnit. I have no specific requirements that suggest TestNG is a better match for us, so we are moving ahead with JUnit.