Build and Source Control - Support #6858

rework existing testcases to use ABLUnit

10/18/2022 06:56 PM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Marian Edu	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

Related issues:

Related to Build and Source Control - Feature #6183: design and implement (as...	**WIP**

History

#1 - 11/21/2022 06:22 AM - Marian Edu

- Status changed from New to WIP

We're working on this one and migrate existing unit tests (no user interaction) to new ones using ABLUnit, is there a new bazaar repository that we should use for this (currently we have a separate branch in our local git repo)?

#2 - 11/21/2022 08:40 AM - Greg Shah

I prefer to migrate these "in place". It is already confusing to have multiple testcase projects (old and busted vs new hotness) and adding one more will make it more messy.

The modified tests are intended to be a full replacement for the original, so there is no reason we can't do this.

#3 - 11/21/2022 09:13 AM - Marian Edu

Greg Shah wrote:

> I prefer to migrate these "in place". It is already confusing to have multiple testcase projects (old and busted vs new hotness) and adding one more will make it more messy.
>
> The modified tests are intended to be a full replacement for the original, so there is no reason we can't do this.

I've probably confused 'old and busted' and 'new hotness', thought what we were using before is (going to be) the old and busted one and hence expected a new repo to show up :)

There are other things that just unit tests in testcases repo, some tests require user interaction so most probably we will need to isolate those out of the regular unit tests and we've also used the repo to simply reproduce issues reported in <large_customer_gui_app> so we can debug those easily in a less complex environment. Beside that there are some PASOE related stuff (PSDOE project, service definitions), SoapUI tests and results, database definition files and other stuff... it might be a good time to re-structure things there a bit?

#4 - 11/21/2022 09:24 AM - Greg Shah

- Related to Feature #6183: design and implement (as needed) the approach for writing automated 4GL compatibility tests added

#5 - 11/21/2022 09:30 AM - Greg Shah

Just to remind everyone: the "master plan" for testing can be found in Testing Process and those pages/tasks linked from there.

> some tests require user interaction so most probably we will need to isolate those out of the regular unit tests

Yes, this makes sense. It can still be in the same repo, but we will automate ChUI using the "Harness" and GUI using Sikuli. I'd like to come up with a convention to separate these cleanly without removing them from the repo.

> we've also used the repo to simply reproduce issues reported in <large_customer_gui_app> so we can debug those easily in a less complex environment

We will want these to be automated for regression testing purposes. I expect these will usually be interactive tests. They can go into the same separated interactive areas.

> Beside that there are some PASOE related stuff (PSDOE project, service definitions), SoapUI tests and results, database definition files and other stuff...

All of this probably does need to be there. .df files need to be there to enable conversion. We will need dump files to re-import as the test database.

Dominik has suggested that the system will be more flexible and easier to manage if each group of tests has its own database/data. I think that is probably right, although it will bring extra configuration and processing to separate everything. I'm open to discussing this.

We need to test SOAP, REST, webspeed and appserver just like anything else. We just will handle those using the Harness rather than ABLUnit.

> it might be a good time to re-structure things there a bit?

Yes, now is the time. But I do want to keep "one repo to rule them all".

#6 - 11/21/2022 01:25 PM - Vladimir Tsichevski

I've started reworking the sources abl_unit/ subdirectory into a "pure ABLUnit". Also I found some obvious bugs in the source structure.

#7 - 11/21/2022 01:37 PM - Marian Edu

Vladimir Tsichevski wrote:

> I've started reworking the sources abl_unit/ subdirectory into a "pure ABLUnit". Also I found some obvious bugs in the source structure.

I don't see what is to rework there, those were simply used to check the ABLUnit behaviour when errors(AppError, AssertionFailedError or plain return error - SysError) are thrown from before/setup/after.

If we're more to work on this rewrite then please split the workload in separate tasks to avoid confusion, I have nothing against if you offer to take over the task all together just please `assign to self` ;)

#8 - 11/21/2022 01:40 PM - Greg Shah

Vladimir is only working on the ablunit parts. He is adding some tests at a minimum. One can think of that work as being part of #3827, not this task.

Vladimir: If there is any other rework of those tests which affects this task, please let us know the proposed details.

#9 - 11/21/2022 02:04 PM - Vladimir Tsichevski

Marian Edu wrote:

> Vladimir Tsichevski wrote:
>
> > I've started reworking the sources abl_unit/ subdirectory into a "pure ABLUnit". Also I found some obvious bugs in the source structure.
>
> I don't see what is to rework there, those were simply used to check the ABLUnit behaviour when errors(AppError, AssertionFailedError or plain return error - SysError) are thrown from before/setup/after.

1. Bugs: the abl_unit/procedure_annotations/test_suite_no_explicit_errors.p refers to non-existing procedures
2. Based on custom logging procedures defined in common/log_hlp.i: uses MESSAGE to output logs. Must be based in OpenEdge.Core.Assert methods.

> If we're more to work on this rewrite then please split the workload in separate tasks to avoid confusion, I have nothing against if you offer to take over the task all together just please `assign to self` ;)

At the moment, I intend to use this tests mainly to test and debug the FWD ABLUnit engine itself. Currently some features like emitting @MESSAGE@s to GUI do not let the all tests to finish, so I want to eliminate them ASAP.

I can temporarily re-assign this task to myself until the abl_unit/ part is reworked or until I meet problem which require your assistance.

Or, we can play this piano "four hands" :-)

I also took a look at other parts of this testcases project, and found hard to rework most of them, partly because they are not documented, and I cannot get what they are supposed to test and how :-(.

#10 - 11/21/2022 02:09 PM - Vladimir Tsichevski

Greg Shah wrote:

> Vladimir is only working on the ablunit parts. He is adding some tests at a minimum. One can think of that work as being part of #3827, not this task.

Yes, and it is very convenient to use this abl_unit/ part as a test suite for #3827.

> Vladimir: If there is any other rework of those tests which affects this task, please let us know the proposed details.

I noted some issues in the previous comment, will add as soon as I discover other issues.

#11 - 11/21/2022 02:21 PM - Marian Edu

Vladimir Tsichevski wrote:

1. Bugs: the abl_unit/procedure_annotations/test_suite_no_explicit_errors.p refers to non-existing procedures

Yeah, just replace `annotations` with `procedure_annotations` - most probably after refactor when OO tests were added

1. Based on custom logging procedures defined in common/log_hlp.i: uses MESSAGE to output logs. Must be based in OpenEdge.Core.Assert methods.

Well yes, this is what all this rewrite is all about isn't it? The tests were supposed to be ran on FWD before any ABLUnit support.

> If we're more to work on this rewrite then please split the workload in separate tasks to avoid confusion, I have nothing against if you offer to take over the task all together just please `assign to self` ;)

> At the moment, I intend to use this tests mainly to test and debug the FWD ABLUnit engine itself. Currently some features like emitting @MESSAGE@s to GUI do not let the all tests to finish, so I want to eliminate them ASAP.

Again, those were not meant to test ABLUnit since there wasn't anything to test so it was simply a 'discovery' phase, feel free to use/rewrite or build any tests you need there.

> I can temporarily re-assign this task to myself until the abl_unit/ part is reworked or until I meet problem which require your assistance.

> Or, we can play this piano "four hands" :-)

If abl_unit is all you need there then it's all yours, as said those weren't really tests but mere discovery and I still don't see how do you plan to test the ABLUnit using ABLUnit but have fun with it :)

> I also took a look at other parts of this testcases project, and found hard to rework most of them, partly because they are not documented, and I cannot get what they are supposed to test and how :-(.

You made my day, normally all projects I've seen so far are fully documented - well maybe the code not 100% but tests are always cristal clear :)

#12 - 12/15/2022 08:21 AM - Vladimir Tsichevski

Multiple files in the testcases project do not convert with FWD. Is this expected?

#13 - 12/15/2022 08:23 AM - Greg Shah

Yes. At any given point in our development, new testcases are written for features we have no implemented yet. It is the nature of the beast.

#14 - 12/15/2022 09:49 AM - Vladimir Tsichevski

Greg Shah wrote:

> Yes. At any given point in our development, new testcases are written for features we have no implemented yet. It is the nature of the beast.

So, we better create and keep the list of things which compiles and which of them work. Otherwise we cannot detect any regressions with this project.

#15 - 01/09/2023 10:00 AM - Marian Edu

Greg, we have a number of things to discuss here.

- First, not all tests are really unit-tests - some of them really need user interaction, other tests needs an application server environment (either on the client or the server side) so won't work in an automated testing pipeline. How should we organise the tests there? Are we grouping those in unittests, ui, appsrv or something along that line?
- Tests that needs database connection, I see the previous database was renamed but that is not an issue as we can use that database. The idea was to clean/reset database content on each test so we can expect the same results on each run, not sure if the current database is being used for something else and needs the content to be preserved as-is for whatever reason?
- I would rather not recreate the database on each test - load the df on each run, not sure if even possible in FWD right now.
- Some tests needs multiple databases - security tables, not sure how to handle those cases just yet.
- Output to files (profiler/log-manager/write-xml/json) we can compare the output to some presets when same result is expected on each run, sometimes things are different (like timestamps) so we have to only see if the content 'matches' - beside the output might be slightly different in FWD, is using `matches` good enough here? For some cases we don't even test the output, guess those 'tests' were used only to generate some output to serve for the initial implementation so were not meant to be actual tests.

Otherwise the rewrite is going painful slow, not sure how the test engine will work in FWD so we're trying to make the test methods that way so we can't assume any execution order - our current tests were just going linear from start/end so 'state' was preserved in between (unless specifically reset). Wonder if we can somehow get out hands on Vladimir's implementation to see how things are working in FWD?

#16 - 01/09/2023 12:08 PM - Vladimir Tsichevski

Marian Edu wrote:

> not sure how the test engine will work in FWD so we're trying to make the test methods that way so we can't assume any execution order - our current tests were just going linear from start/end so 'state' was preserved in between (unless specifically reset).

Generally, unit tests must be executed in random order. The @BeforeClass / @AfterClass methods are assumed to assure the universe state is reset before each test. I suppose, in FWD test engine tests are executed in order of appearance, but I need to check this.

> Wonder if we can somehow get out hands on Vladimir's implementation to see how things are working in FWD?

I think, I will change to our renovated trunk, then make a #3827 branch repository, so it will be available to any interested party.

#17 - 01/10/2023 07:48 AM - Vladimir Tsichevski

Vladimir Tsichevski wrote:

> > Wonder if we can somehow get out hands on Vladimir's implementation to see how things are working in FWD?

I created a branch repository 3827a based on the trunk rev. 14472, and committed all current changes as rev. 14473.

#18 - 01/10/2023 07:53 AM - Greg Shah

When I made an entry in this task on 2022-12-15 at 09:55, I accidentally set it to private.

Here it is. It relates to the question about how to sub-set the files for conversion.

> For now, the simplest idea is to create file lists for conversion of sub-sets of the testcases project. We already have some of these defined (see file-cvt-list.txt.*). When #6256 is complete, then we will be able to more cleanly define "apps" as convertable sub-sets.

#19 - 01/10/2023 08:07 AM - Greg Shah

- First, not all tests are really unit-tests - some of them really need user interaction, other tests needs an application server environment (either on the client or the server side) so won't work in an automated testing pipeline.

For the ones that need user interaction, we will automate the tests using the Harness (for ChUI) or Sikuli (for GUI).

For appserver, it may be reasonable to use unit testing because our test engine assumes that the FWD server is running. The test engine client is a FWD client and calls over to the server. If this is not enough, we can discuss how we can expose some management capability, but it seems like we can handle this case with ABLUnit.

> How should we organise the tests there? Are we grouping those in unittests, ui, appsrv or something along that line?

Yes, this seems reasonable. Within those categories we will need more functional definitions of what is under test.

- Tests that needs database connection, I see the previous database was renamed but that is not an issue as we can use that database. The idea was to clean/reset database content on each test so we can expect the same results on each run, not sure if the current database is being used for something else and needs the content to be preserved as-is for whatever reason?

I don't know what dependencies there are on the existing database. Dominik suggested that it is easier to maintain over time if the database is small and specific to the tests being run. This seems like a reasonable conclusion. That would mean that we would implement small dedicated databases for groups of related tests. This means the schemata would be dedicated and there would be a set of dump files that are dedicated.

- I would rather not recreate the database on each test - load the df on each run, not sure if even possible in FWD right now.

I would think that we would script the create/import of the specific database as a precondition to that test set, not for specific tests.

- Some tests needs multiple databases - security tables, not sure how to handle those cases just yet.

I think these would just be sets of databases dedicated to the set of tests.

- Output to files (profiler/log-manager/write-xml/json) we can compare the output to some presets when same result is expected on each run, sometimes things are different (like timestamps) so we have to only see if the content 'matches' - beside the output might be slightly different in FWD, is using `matches` good enough here?

I not sure we want to encode the comparison logic using matches.

In the Harness we have some useful text report comparison tools. For example, we can define repeating or fixed areas of a text file which can be ignored, including dealing some some pagination concepts. These are used to test old-school ChUI reports, but it may be pretty close to what we need here. The question is how we expose it. Perhaps we have some generation steps that are converted 4GL code. Then we define some post-conditions that have direct Java usage to call the Harness jars to test the output.

> For some cases we don't even test the output, guess those 'tests' were used only to generate some output to serve for the initial implementation so were not meant to be actual tests.

I think we should expand the output testing in these cases so that they can be used as real tests and not just examples.

#20 - 01/10/2023 08:24 AM - Greg Shah

Wonder if we can somehow get out hands on Vladimir's implementation to see how things are working in FWD?

I think, I will change to our renovated trunk, then make a #3827 branch repository, so it will be available to any interested party.

I have pushed 3827a to xfer in /opt/fwd/3827a/.

#21 - 03/24/2023 07:13 AM - Marian Edu

- % Done changed from 0 to 90

Most of the unit tests that we've previously created were converted to ABLUnit format - test classes used whenever possible, test procedures when definitions not allowed in a class.

We've reorganised a bit the folder structure:

- all test moved inside the `tests` folder but otherwise keep the existing folder structure
- all test classes/procedures names starts with `Test`
- topic specific helper/support code are under a separate `support` sub-folder under the test topic
- generic helper/support code under project root folder (this was added to `PROPATH`)
- all old tests that were converted were removed (only unit test)
- all other tests that require user interaction of any kind (not unit tests) were kept in place - to be decided if need to be converted to UI tests (sikuli)

This is the conversion status as of today.

Topic	Converted
copy_lob	100%
data_relation	100%
data_source	100%
dataset	100%
error_handling	100%
i18n	100%
log_manager	100%
oo	~50%
primitive_type	100%
profiler	100%
query	~75%
table	100%
security	~70%

#22 - 09/06/2023 07:33 AM - Greg Shah

Is everything complete in this task?

#23 - 09/06/2023 07:36 AM - Marian Edu

Greg Shah wrote:

> Is everything complete in this task?

The only thing remaining there is related to 'security' where we need multiple databases, this wasn't (yet) sorted out, I'll see what we can do about it - we can try connect at runtime to swap between those databases, not sure now how this will work in FWD but I don't see any other way atm.

#24 - 04/22/2024 02:36 AM - Marian Edu

- % Done changed from 90 to 100

- Status changed from WIP to Review

This can be probably closed since all the conversion is actually done, it's just we do not have all the different database security scenarios setup to be used in ABLUnit but otherwise the tests were converted.

#25 - 04/22/2024 12:48 PM - Greg Shah

- Status changed from Review to Closed