collecting_society - Webfrontend #701

Fix tests

11/05/2018 03:58 PM - Alexander Blum

Status:	Erledigt	**Start date:**	
Priority:	Dringend	**Due date:**	
Assignee:	Thomas Mielke	**% Done:**	100%
Category:		**Estimated time:**	2.00 hours
Target version:	Repertoire 2) Testing phase II	**Spent time:**	18.00 hours

Description

Especially unit/function tests fail due to the changes in Tdb.transaction().
Find out, how to talk with the db now (maybe a transaction needs to be opend on test init).

Associated revisions

Revision 89b40d99 - 06/29/2019 05:27 PM - Thomas Mielke

repaired functional tests: added database connection to FunctionalTestBase class; ref #701

Revision 89b40d99 - 06/29/2019 05:27 PM - Thomas Mielke

repaired functional tests: added database connection to FunctionalTestBase class; ref #701

Revision 537d279e - 09/22/2019 04:53 AM - Alexander Blum

ref #701: removes scenario test data (master only)

Revision 537d279e - 09/22/2019 04:53 AM - Alexander Blum

ref #701: removes scenario test data (master only)

Revision c55d488d - 09/22/2019 04:55 AM - Alexander Blum

fixes #701: fixes tests

Revision c55d488d - 09/22/2019 04:55 AM - Alexander Blum

fixes #701: fixes tests

Revision 1b88c205 - 09/22/2019 05:01 AM - Alexander Blum

fixes #701: fixes tests, adds test data class

Revision 1b88c205 - 09/22/2019 05:01 AM - Alexander Blum

fixes #701: fixes tests, adds test data class

Revision 706eb000 - 09/22/2019 07:46 AM - Alexander Blum

ref #701: fixes containers using test database in jenkins

Revision 706eb000 - 09/22/2019 07:46 AM - Alexander Blum

ref #701: fixes containers using test database in jenkins

Revision af9f78eb - 09/22/2019 07:54 AM - Alexander Blum

ref #701: fixes open transactions for test server

Revision af9f78eb - 09/22/2019 07:54 AM - Alexander Blum

ref #701: fixes open transactions for test server

History

#1 - 03/14/2019 12:54 AM - Alexander Blum

- Estimated time set to 2.00

#2 - 03/17/2019 09:58 AM - Alexander Blum

- Priority changed from Normal to Hoch

#3 - 03/17/2019 10:07 AM - Alexander Blum

- Priority changed from Hoch to Dringend

#4 - 06/29/2019 05:29 PM - Thomas Mielke

functional tests now have db support again; test_login_with_right_credentials() throws exception, which is obviously not database-related (acl?).

unit tests still need to be fixed.

#5 - 06/29/2019 07:26 PM - Thomas Mielke

Concerning the failing functional test test_login_with_right_credentials():

User meik@c3s.cc is obviously part of demo data, but not test data. I tried replacing it with user '1@rep.test' but then the assert won't fit:

self.assertIn('The resource was found at', res2.body)

No idea what the original test author intended with this. Tests should be inline-documented on a line-by-line basis...

#6 - 06/29/2019 08:55 PM - Alexander Blum

> test_login_with_right_credentials()

The name describes the purpose, what is to be tested. So all what should
be done here is exactly that: that a login with right credentials works as intended.

> self.assertIn('The resource was found at', res2.body)

The result should be the backend html - but there is some stuff going on before the backend: there's a redirect to '/dashboard', which is redirected to '/repertoire', which is redirected to '/repertoire/dashboard' - just have a look at the views and/or your browser network request log to understand this:

- /dashboard will be needed, if there are more plugins.
- The redirect from /repertoire to /repertoire/dashboard is just to have some entry point, which could be pointed elsewhere in the future (e.g. due to user settings)

So, in a nutshell, the author of the code is just testing, if the request results in a redirection response via the message in the html body. Probably there's no redirect, when the authentication failed. This test could be optimized though, to be more explicit.

#7 - 06/29/2019 09:03 PM - Alexander Blum

e.g. an obvious better way would be to test for the response code instead of possibly unreliable human readable messages.

If you haven't stumbled upon it yet, this is the response object you are dealing with in webtest.

#8 - 06/29/2019 09:09 PM - Alexander Blum

ah - and the very proper way to prevent such credentials from being invalid after some time is to explicitly create an account via setUp and maybe delete it with tearDown (it should not have any side effects on other tests at least). It's dependency on some other import somewhere else is unneccessary, tests should contain themself as far as possible.

#9 - 06/30/2019 06:10 PM - Thomas Mielke

I wasn't able to get something beyond a 403 here and I'm giving up on it now:

res3 = res2.follow().follow().follow()

There must be some difference between real browser requests and the WebTest framework behavior.

#10 - 07/01/2019 12:42 AM - Alexander Blum

Thomas Mielke wrote:

> beyond a 403

There is nothing beyond a 403, that's a 'forbidden' response. The login does not work then. Credentials? Test DB Setup?

#11 - 07/01/2019 12:14 PM - Thomas Mielke

Alexander Blum wrote:

> Thomas Mielke wrote:
>
> > beyond a 403
>
> There is nothing beyond a 403, that's a 'forbidden' response. The login does not work then. Credentials? Test DB Setup?

But login does work with a real browser. This is just looking for phantom bugs.

If the login failed, "Login failed" would have appeared as in the negative login test above this test.

Someone should 'fix' this test who has an agenda and sees some meaning in the way the test works.

#12 - 07/01/2019 06:59 PM - Thomas Mielke

Today I tried to make progress with unit tests. But again had no luck: there is an exception in the decorator code of transaction in models/base.py and I don't have any clue how to solve it. This is code without any inline comment.

```
    def decorator(func):
        @wraps(func)
        def wrapper(*args, **kwargs):
            _db = Tdb._db
            _user = user or 0
            _retry = Tdb._retry or 0
            _readonly = readonly
            if 'request' in kwargs:
                _readonly = not (
                    kwargs['request'].method
                    in ('PUT', 'POST', 'DELETE', 'PATCH'))
            _tdbg(func, "WRAP", None, 1)

            for count in range(_retry, 0, -1):
                if closed():
                    _tdbg(func, "CONNECT")
                    with Transaction().start(_db, 0):
                        Cache.clean(_db)
                        pool = Pool(Tdb._db)
                        User = pool.get('res.user')
                        _context = User.get_preferences(context_only=True)
                        _context.update(context or {})
                    Transaction().start(
                        _db, _user, readonly=_readonly, context=_context,
                        close=False)

                cursor = Transaction().cursor
                try:
                    _tdbg(func, "CALL", "Try %s, Cursor %s" %
                          (_retry + 1 - count, id(cursor)))
                    result = func(*args, **kwargs)
                    if not _readonly:
                        _tdbg(func, "COMMIT", "Try %s, Cursor %s" %
                              (_retry + 1 - count, id(cursor)))
                        cursor.commit() # <-- inner exception here: "AttributeError: 'NoneType' object has no attribute 'commit'"
```

when called after the unit tests:

```
    cursor = Transaction().cursor
    if cursor and not Transaction().cursor._conn.closed:
        Cache.resets(Tdb._db)
        Transaction().stop() # <-- exception on database close code lent from close_db_connection()
```

Why does this feel like punishment? :)

#13 - 07/03/2019 02:45 PM - Thomas Mielke

- Status changed from Neu to In Bearbeitung

- Assignee set to Alexander Blum

- % Done changed from 0 to 50

In Transaction.stop() the cursor is set to None but is being used thereafter in the decorator function at https://github.com/C3S/collecting_society.portal/blob/develop/collecting_society_portal/models/base.py#L201

This construct seems pretty obscure to me as I'm not familiar with any of the concepts in use here.

PS: don't decorators violate pythons 'make it explicit' rule?

#14 - 07/04/2019 05:06 PM - Thomas Mielke

Maybe we should use Proteus for database access in unit tests?

#15 - 07/04/2019 07:53 PM - Alexander Blum

> This is just looking for phantom bugs.

Probably not - maybe it's just, because your browser does some automagical things.

> If the login failed, "Login failed" would have appeared as in the negative login test above this test.

What do you mean?

#16 - 07/04/2019 08:00 PM - Alexander Blum

1.

cursor.commit() # <-- inner exception here: "AttributeError: 'NoneType' object has no attribute 'commit'"

2.

```
if cursor and not Transaction().cursor._conn.closed:
    Cache.resets(Tdb._db)
    Transaction().stop() # <-- exception on database close code lent from close_db_connection()
```

The question is, why in 1. there is no cursor anymore, although 2. explicitly checks for its existence. Obviously it gets destructed inbetween, or the check is faulty?

#17 - 07/04/2019 08:03 PM - Alexander Blum

Thomas Mielke wrote:

> In Transaction.stop() the cursor is set to None but is being used thereafter in the decorator function

Which function being decorated? Asked the other way around: why is the transaction stopped before all transactions are finished?

> PS: don't decorators violate pythons 'make it explicit' rule?

If it would, there would be no decorators in python ;) The code gets way cleaner for certain situations using them.

#18 - 07/04/2019 08:08 PM - Alexander Blum

Thomas Mielke wrote:

> Maybe we should use Proteus for database access in unit tests?

Unittests in pyramid should never test the database, but the app methods. If you want to test Tdb and the model wrapper classes, it would be funny to circumvent Tdb and the model wrapper classes ;) Anyway, it's a different syntax (even less powerful) and cannot replace the tryton pool objects used in the model wrapper.

#19 - 07/04/2019 08:13 PM - Alexander Blum

Alexander Blum wrote:

> why is the transaction stopped before all transactions are finished?

More explicitly: the transaction shall not be stopped, before the transactions have finished - which means probably, that all decorated functions must have returned.

You might also switch on tdb debugging in development.ini (outputs to /ado/tmp/transaction.log), maybe it could help?

#20 - 07/05/2019 02:02 PM - Thomas Mielke

Alexander Blum wrote:

> If the login failed, "Login failed" would have appeared as in the negative login test above this test.

What do you mean?

https://github.com/C3S/collecting_society.portal/blob/89b40d9902e102c9d0d8f9e185b1890a9d9facac/collecting_society_portal/tests/functional/login.py#L71

#21 - 07/05/2019 02:05 PM - Thomas Mielke

Alexander Blum wrote:

> Thomas Mielke wrote:
>
> > PS: don't decorators violate pythons 'make it explicit' rule?
>
> If it would, there would be no decorators in python ;) The code gets way cleaner for certain situations using them.

If you hide all the ugly stuff, the code looks cleaner for sure! B)

#22 - 07/05/2019 02:08 PM - Alexander Blum

Thomas Mielke wrote:

> Alexander Blum wrote:
>
> > If the login failed, "Login failed" would have appeared as in the negative login test above this test.
>
> What do you mean?

> https://github.com/C3S/collecting_society.portal/blob/89b40d9902e102c9d0d8f9e185b1890a9d9facac/collecting_society_portal/tests/functional/login.py#L71

But it does not fail? Or what are the implications?

#23 - 07/05/2019 02:12 PM - Thomas Mielke

Alexander Blum wrote:

> Thomas Mielke wrote:
>
> > Maybe we should use Proteus for database access in unit tests?
>
> Unittests in pyramid should never test the database, but the app methods.

So, we should simulate access to the Company object in unit tests?

#24 - 07/05/2019 02:15 PM - Thomas Mielke

Alexander Blum wrote:

> Thomas Mielke wrote:
>
> > Alexander Blum wrote:
> >
> > > If the login failed, "Login failed" would have appeared as in the negative login test above this test.
> >
> > What do you mean?
>
> https://github.com/C3S/collecting_society.portal/blob/89b40d9902e102c9d0d8f9e185b1890a9d9facac/collecting_society_portal/tests/functional/login.py#L71

> But it does not fail? Or what are the implications?

No, it does not fail. It just gets a 403 somewhere, but doesn't harm a real browser on its way through the redirects, finally getting to the dashboard.

#25 - 07/05/2019 02:21 PM - Alexander Blum

> If you hide all the ugly stuff, the code looks cleaner for sure! B)

:P

- https://www.python.org/dev/peps/pep-0318/#motivation
- https://www.oreilly.com/ideas/5-reasons-you-need-to-learn-to-write-python-decorators

#26 - 07/05/2019 02:26 PM - Alexander Blum

> So, we should simulate access to the Company object in unit tests?

Depends on what needs to be tested. If you want to test the model wrapper Company, there's no point in mocking the company object. If you test another function, which just needs a company as context, it's fine to mock it - and even better, as you mock away some code, which could fail but has nothing to do with the test you are writing.

#27 - 07/05/2019 02:31 PM - Alexander Blum

Thomas Mielke wrote:

> Alexander Blum wrote:
>
> > Thomas Mielke wrote:
> >
> > > Alexander Blum wrote:
> > >
> > > > If the login failed, "Login failed" would have appeared as in the negative login test above this test.

> What do you mean?

> https://github.com/C3S/collecting_society.portal/blob/89b40d9902e102c9d0d8f9e185b1890a9d9facac/collecting_society_portal/tests/functional/login.py#L71

> But it does not fail? Or what are the implications?

> No, it does not fail. It just gets a 403 somewhere, but doesn't harm a real browser on its way through the redirects, finally getting to the dashboard.

Logins with wrong credentials should definitely not give you the backend ;) I think, now I get what you mean - the login with right credentials is working, as it behaves differently than with wrong credentials.

Have you checked, where the redirections lead to?

#28 - 07/05/2019 02:43 PM - Thomas Mielke

Alexander Blum wrote:

> Have you checked, where the redirections lead to?

Leads to a 403 rightaway.

#29 - 07/05/2019 06:03 PM - Alexander Blum

TestLogin.test_login_with_right_credentials

- I encountered, that for testing the scenario_demo_data is used
- So, I don't know, which database you connected to (should have been test or test_template), but there should be no 1@rep.test user in it (by the way, you should definitely be able to connect to the test or test_template database via tryton, it's the exact same setup as c3s database)
- We probably use scenario_demo_data, because scenario_test_data lasts forever to run through, which would be bad both for execution time of the tests (as Jenkins recreates the whole thing every time) and for the VM resources (as it is executed on every push)
- The scenario_demo_data is broken with 3 errors (you said, it ran through without errors, as far as I remember? This is the first thing to check in general, if something went wrong)
- So, probably the login does not work, as the db is not setup right (have not checked it further though)
- Just to test the login itself, I created a test db with scenario_test_data - login with 1@rep.test and the current test code works fine there
- Short fix: fix scenario_demo_data (could be compared with object creation in scenario_test_data, which works fine)
- Long term goal: only scenario_master_data for the test db (no test/demo data at all) and creation of the needed context (db objects, etc) within the tests

TestHelpers.test_format_currency

- As I said, all Tdb.transaction() wrapped functions need to return before the Transaction().stop()
- Within the test, the wrapped function will not be able, to finish it's transaction, as it's transaction is stopped before it returns.

In a nutshell the wrapper works like that:

```
@Tdb.transaction()
def somefunction(...):
  pass
```

A call of somefunction(...) then will be equal to the following code (important parts from Tdb.transaction, the rest is optimization like debugging, caching, retries, etc):

```
if closed():                                  # does an "open" cursor exist?
    Transaction().start(...)                      # if not, initiate a new connection and create one
cursor = Transaction().cursor                         # the cursor, accessible from everywhere via singleton Transaction()
result = somefunction(...)                        # execute the wrapped function
if not _readonly:                             # was readonly set to False via decorator arguments?
    cursor.commit()                               # if so, commit the cursor to finally write the changes to the db
return result                                 # return, whatever the wrapped function returned
```

So, if you stop the transaction inbeteeen, it will be equal to:

```
if closed():
    Transaction().start(...)
cursor = Transaction().cursor
result = somefunction(...)
        '-> Transaction().stop() # our cursor is gone!
if not _readonly:
    cursor.commit()             # no cursor anymore!
return result
```

Is it now clear, what is happening here? Some other things to notice:

- It is not clear, why the cursor wants to be commited - no readonly=False, as far as I looked into it
- Your code contains a lot of the same stuff in Tdb.transaction(), so your code should not be neccessary

By the way, you might jump into pdb on errors by adding --pdb:

```
host$ docker-compose run --use-aliases portal bash
portal$ ado-do run-tests --path src/collecting_society.portal/collecting_society_portal/tests/unit/helpers.py:TestHelpers.test_format_currency --pdb
```

Pdbing into Tdb.transaction, I checked, where the cursor is dropped and found out, that it never gets created in the first place, as close() returns also False, when no cursor exists. This might either be a leftover from me trying to get stacked cursors to work or it assumes, that some other process is

creating the first connection (like it is done, when the app framework is initialized). It could also be, that some other function depends on closed(). I have no time to test it, but you could try defining a new function to implement the expected behaviour "does an "open" cursor exist?":

```
def open():
    if Transaction().cursor and not Transaction().cursor._conn.closed
        return True
    return False
```

And use this function instead of closed()

```
if not open():
    ...
```

At least the test is ok then, but it needs further testing for side effects (you could try to run all other tests as your first usecase for having tests ;))

#30 - 07/05/2019 06:12 PM - Alexander Blum

Ah, and also btw: To recreate the test_template db, just delete it and run some test again.

ado-do db-delete test_template

#31 - 07/06/2019 12:46 PM - Thomas Mielke

- Assignee changed from Alexander Blum to Thomas Mielke

#32 - 07/08/2019 03:58 PM - Alexander Blum

Alexander Blum wrote:

> At least the test is ok then

Oh, I meant, without the custom code around the test, just @Tdb.transaction() is enough.

#33 - 07/08/2019 06:46 PM - Thomas Mielke

Alexander Blum wrote:

TestLogin.test_login_with_right_credentials

- I encountered, that for testing the scenario_demo_data is used

This means, for tests the demo data ist used and for the demo setup the test data is used?...

- So, I don't know, which database you connected to (should have been test or test_template), but there should be no 1@rep.test user in it (by the way, you should definitely be able to connect to the test or test_template database via tryton, it's the exact same setup as c3s database)

I also tested with wilbur webuser. I have set a breakpoint in the test code to be sure and psql-ed the test db.

- We probably use scenario_demo_data, because scenario_test_data lasts forever to run through, which would be bad both for execution time of the tests (as Jenkins recreates the whole thing every time) and for the VM resources (as it is executed on every push)

There should be a lite version of test data with lesser db entries.

- The scenario_demo_data is broken with 3 errors (you said, it ran through without errors, as far as I remember? This is the first thing to check in general, if something went wrong)

I didn't build the test data till now. Confirm that 3 of 213 tests failed during db generation.

- So, probably the login does not work, as the db is not setup right (have not checked it further though)

Company and Webuser don't seem affected.

- Just to test the login itself, I created a test db with scenario_test_data - login with 1@rep.test and the current test code works fine there

Interesting.

- Short fix: fix scenario_demo_data (could be compared with object creation in scenario_test_data, which works fine)

The fix is to fix! :D

File "/ado/etc/scenario_demo_data.txt", line 143, in scenario_demo_data.txt
Failed example:
creative.save()
Exception raised:
Traceback (most recent call last):
File "/usr/lib/python2.7/doctest.py", line 1315, in __run
compileflags, 1) in test.globs
File "<doctest scenario_demo_data.txt[53]>", line 1, in <module>
creative.save()
File "/usr/local/lib/python2.7/dist-packages/proteus/__init__.py", line 695, in save
self._proxy.write([self.id], values, context)
File "/usr/local/lib/python2.7/dist-packages/proteus/config.py", line 162, in __call__
result = rpc.result(meth(*args, **kwargs))
File "/ado/src/trytond/trytond/model/modelsql.py", line 829, in write
cls._validate(sub_records, field_names=all_field_names)
File "/ado/src/trytond/trytond/model/modelstorage.py", line 1080, in _validate
error_args=error_args)

File "/ado/src/trytond/trytond/error.py", line 74, in raise_user_error
raise UserError(error)
UserError: ('UserError', ('selection_validation_record', ''))

Havent touched the creative code for ages. Haven't we?

TestHelpers.test_format_currency

- As I said, all Tdb.transaction() wrapped functions need to return before the Transaction().stop()

Ok, so I moved the Tdb.init() and transaction start code into setUpClass of TestHelpers to make sure the transaction is initialized before the decorator. I now get get:

Traceback (most recent call last):
File "/ado/src/collecting_society.portal/collecting_society_portal/tests/unit/models/base.py", line 32, in test_autenticate_user
webu = WebUser.authenticate("alf_imp@c3s.cc", "cc")
File "/ado/src/collecting_society.portal/collecting_society_portal/models/web_user.py", line 130, in authenticate
users = cls.get().search([('email', 'ilike', cls.escape(email))])
File "/ado/src/trytond/trytond/model/modelsql.py", line 969, in search
pool = Pool()
File "/ado/src/trytond/trytond/pool.py", line 53, in __new__
database_name = Transaction().cursor.database_name
AttributeError: 'NoneType' object has no attribute 'database_name'

> Is it now clear, what is happening here?

No. For example: Why do I have to provide the transaction code twice?

Sorry, but I didn't understand what you wanted to illustrate with your code samples.

Please mind my brain capacity! :)

I might be able to work on well designed Python code. But don't expect me to delve into flawed code not even you were able to understand completely. 8-/

> for tests the demo data ist used and for the demo setup the test data is used?...

For the normal c3s setup, scenario_test_data is used, so I would not call it "demo setup".

The purpose of scenario_demo_data was to provide a minimal set of tests for the database in a scenario style file (telling the story of what is happening within the comments) and to provide a minimal data set to be able to present the capabilities of the app ("demo").

In lack of some standardized, possibly standalone import script I wrote scenario_test_data, as we needed some more functional, parametrized import style (no story telling here) to have a lot of data for human testing. Once we have some import script, scenario_test_data should be merged into that.

To bootstrap the pyramid tests and get some coverage in a fast way, it just happened, that scenario_demo_data was chosen to have something to work with.

> I also tested with wilbur webuser. I have set a breakpoint in the test code to be sure and psql-ed the test db.

I can't tell you, what is wrong with the database - but as scenario_test_data works, the demo setup is the problem here for sure.

> There should be a lite version of test data with lesser db entries.

As I said, this is suboptimal, as the tests should not depend on external factors like previously imported data sets, but should create/delete it.

But it could be a workaround. For now, you could just rename scenario_demo_data to have a backup of it, copy the scenario_test_data to scenario_demo_data and adjust the parameter for the entry numbers on top of the file.

> Havent touched the creative code for ages. Haven't we?

The creative code went into repertoire, and we probably did a lot of changes to it.

TestHelpers.test_format_currency

> Ok, so I moved the Tdb.init() and transaction start code into setUpClass of TestHelpers to make sure the transaction is initialized before the decorator.

You don't need any of the Transaction code, just the Tdb.init() stuff, which was there already.

And you need to fix the wrong 'closed' condition within Tdb.transaction() as stated above.

> Is it now clear, what is happening here?

> No. For example: Why do I have to provide the transaction code twice?

You don't need to.

> Sorry, but I didn't understand what you wanted to illustrate with your code samples.

To provide you with a complete understanding of what @Tdb.transaction() is doing.

We should switch to an audio channel, then I might be able to explain better.

#35 - 09/16/2019 07:40 PM - Thomas Mielke

I tried around another hour without any success.

#36 - 09/17/2019 04:53 PM - Thomas Mielke

I tried around another hour without any success.

#37 - 09/22/2019 04:56 AM - Alexander Blum

- Status changed from In Bearbeitung to Erledigt

- % Done changed from 50 to 100

Applied in changeset commit:collecting_society_portal_repertoire|c55d488d3ccf4930cb5ad5c181de23e8ccab32e6.

#38 - 10/08/2019 03:37 PM - Alexander Blum

- Target version changed from 2) Testing phase II to Repertoire 2) Testing phase II

#39 - 10/08/2019 03:52 PM - Alexander Blum

- Project changed from repertoire to collecting_society