kescher

Oct 31, 2020

Contents:

kescher is an accounting tool for small businesses. It has various helpers and interfaces to be used with different bank statement formats, VAT tariffs and accounting philosophies.

At westnetz we pay taxes on cash basis and are not required to create annual balances but report our netincome to the tax office. Knowing that this use case is special, the tool currently is optimized to do that. It might be possible to use *kescher* for other accounting methods, e.g. accrual basis, or without having to pay the VAT (small business operator regulations in Germany). But you should be aware, that errors and bugs in that case might not be covered by tests.

CHAPTER 1

Getting started

1.1 Quick Tour

Let's take the quick tour. We will use the samples and fixtures from the automated software tests to play around a bit with the tool.

The test data is not shipped with the program, therefore you have to get it from the repository on github. Navigate to *kescher/tests/fixtures* and download:

- accounts.yaml
- journal.csv

Create a new directory, and put the files there

```
$ mkdir kescher_quick_tour
$ cd kescher_quick_tour
$ cp ~/Downloads/accounts.yaml .
$ cp ~/Downloads/journal.csv
```

Now it's time to initialize the *kescher*:

```
$ kescher init
```

If that fails: Check if you have installed the tool. Have a look at the *Installation* section to find out how to do that.

If everything worked, your directory should look like this now:

```
$ ls -w 0
accounts.yaml
journal.csv
kescher.db
kescher.log
```

Have a look at the default accounts chart in the *accounts.yaml* file to see how we will structure our accounts. After you did that, it's time to import it into the *kescher*.

```
$ kescher import-accounts accounts.yaml
```

Also import the journal. In the real world, this is your bank statement.

```
$ kescher import-journal journal.csv
```

1.2 Design

This document describes the design principles behind kescher, it is supposed to help you to understand how and why things work as they do.

1.3 Installation

Currently this package is not available on the PythonPackageIndex (PyPI).

Note: *kescher* is still under heavy development. Even the stable versions are not fit for production use yet!

kescher is developed with and for Python3.8. It might work in older versions. Incompatibility with Python versions <3.8 will not be fixed and not considered bugs!

1.3.1 Prerequisites

Install Python3.8 with your favourite package manager. It is recommended to use a virtual environment helper such as virtualenv.

```
$ mkvirtualenv kescher
```

Installation with *poetry* is desired. Therefore you have to install it via pip.

```
$ pip install poetry
```

1.3.2 Stable Version

The latest stable version can be optained by using the *releases* tab on github. Fetch the latest zip file and extract it.

1.3.3 Latest Version

Obtain the latest version from github, by cloning the repository:

```
$ git clone https://github.com/westnetz/kescher
```

1.3.4 Installation

After cloning the repository change your directory to it and install *kescher*:

```
$ poetry install
```

1.4 Quickstart

This chapter covers the steps needed in order to set up *kescher* to start accounting.

1.4.1 Initialization

It is useful to have a directory dedicated to kescher. Documents, bank statements, etc. can and should be stored here.

Note: It is not recommended to import documents from outside the kescher directory, as this is can cause problems when using exports.

Create the kescher working directory

```
$ mkdir accounting_2020
$ cd accounting_2020
```

Next, you initialize the kescher in this directory. This will create the database *kescher.db* and a log file *kescher.log* here.

1.4.2 Account Creation

The easiest way to set up your chart of accounts is by creating a yaml file with the following structure

```
Customers:
  - "1000"
  - "1001"
  - "1002"
VAT:
  - VAT_in
  - VAT_out
Expenses:
  - Internet
  - Power
  - Rent
  - Phone
  - Aquisition:
    - Hardware
    - Materials
Revenue:
  - Internet_Connections
```

This will create four base accounts (Customers, VAT, Expenses, Revenue) with child accounts. The base accounts are not meant to be used to directly assign bookings to, but rather to allow for easier balance aggregation.

1.4.3 Invoice Import

Invoices can be imported directly from a rechnung working directory. Of course it is possible to import invoices from another invoicing tool, but currently only the format *rechnung* uses is supported (yaml file for data and a pdf as the real document).

```
$ kescher import-invoices --nested ../rechnung_2020/invoices cid total\_gross date
```

This will import all invoices from rechnung located in the given directory. This will create virtual bookings for each invoice. These virtual bookings are virtual as they are not the basis for tax calculations. Non existing customer accounts will be created (yet, not assigned to the *Customer* parent account!). The pdf documents are imported as documents, ready to be assigned to entries in your journal.

If you want to import invoices from another tool, you can also convert the necessary data to the desired data, an store them in a directory and use the *–flat* option for the import. A minimal invoice to be imported into *kescher* would look like this:

```
cid: "1000"
total_gross: 24.00
date: "2020-02-11"
```

1.4.4 Journal Import

1.5 kescher

1.5.1 kescher package

Subpackages

kescher.tests package

Submodules

kescher.tests.test_cli module

This module contains the tests for cli commands. It asserts that the command line interface works as intended.

kescher.tests.test_cli.**iterate_accounts**(*accounts*)
Helper function to iterate to ease iterating through the accounts.

kescher.tests.test_cli.**test_auto_vat**()
Asserts the auto-vat command executes without error. The results are checked separately with the show_saldo test command.

kescher.tests.test_cli.**test_import_accounts**()
Asserts that a set of test accounts are imported correctly.

kescher.tests.test_cli.**test_import_document**()
Asserts that document import from a flat directory structure works correctly.

kescher.tests.test_cli.**test_import_invoices**()
Asserts that invoices are imported correctly, connected pdf documents are imported as well and the command exits with 0 as exit code. The invoices import is perfomed for the nested and flat directory structure.

kescher.tests.test_cli.**test_import_journal**()
Asserts that a test journal is imported correctly and the import exits with 0 as exit code.

kescher.tests.test_cli.**test_init_creates_db_and_log**()
Asserts that a database file and a log file is created and the init command exits with 0 as exit_code.

kescher.tests.test_cli.**test_init_debug_writes_to_log**()
Asserts that the –debug flag turns on debug messages to be written to the log.

kescher.tests.test_cli.**test_show_accounts**(*expected_accounts*)
Asserts the list of accounts matches the expected accounts.

kescher.tests.test_cli.**test_show_booked_entry**(*entry_3_booked*)
Asserts, that after the remains of entry 3 where booked to one account, the journalentry should display these bookings.

kescher.tests.test_cli.**test_show_entry**(*entry_3*)
Asserts, that the filter results matches expectations of content and formatting. The called journalentry should be displayed in one table, and below all corresponding entries.

kescher.tests.test_cli.**test_show_journal_errors**()
Test if the column filter returns only the desired columns (on exact match!).

kescher.tests.test_cli.**test_show_journal_filtered**(*journal_filtered_klausi_meyer*)
Test if the column filter returns only the desired columns (on exact match!).

kescher.tests.test_cli.**test_show_saldo**()
This test show the saldo of the specified account in the specified range of time.

Module contents

Submodules

kescher.booking module

kescher.booking.**auto_book_vat**(*percentage*, *vat_in_acc*, *vat_out_acc*)
This function automatically books VAT for all journal entries.

TODO: Filtering by date/range and check for existing bookings to prevent double booking.

kescher.booking.**book_entry**(*value*, *comment*, *journalentry_id*, *account_name*, *force*)
Book a journalentry to some account.

kescher.booking.**get_account_saldo**(*account*, *start_date=None*, *end_date=None*, *with_virtual=False*)
Sums all bookings for the given account in the given timeframe to return 1 decimal (or two, if also the virtual bookings shall be considered).

If the selected account is parent to other accounts, the total of all child accounts will be returned.

kescher.cli module

kescher.database module

kescher.database.**get_db**()

kescher.importers module

Home of all importers.

Importers are bulk operators. I.e. they read yaml or csv data and put them into the database in bulks. This is necessary to get bank data, as well as the account structure easily into the database, without having to set up everything by hand or one by one.

class kescher.importers.**AccountImporter**(*account_file*)
Bases: *kescher.importers.Importer*

The AccountImporter is a helper to set up your accounts (Kontenrahmen). As this can easily done in a yaml file. This yaml file is then loaded into the AccountImporter which will automagically create all accounts accordingly.

import_accounts()
This wrapper function is to to be called from external functions or __call__(). After importing the accounts it deletes the class instance and closes the db connection, to ensure the db connection is not left open, and then not run again.

class kescher.importers.**DocumentImporter**(*path*, *flat=True*)
Bases: *kescher.importers.Importer*

The DocumentImporter assists for bulk importing documents.

EXTENSION = '.pdf'

import_documents()

class kescher.importers.**Importer**
Bases: object

Importer base class creates the logger instance and the import_date.

class kescher.importers.**InvoiceImporter**(*path*, *account_key*, *amount_key*, *date_key*, *flat=False*)
Bases: *kescher.importers.Importer*

EXTENSION = '.yaml'

import_invoices()

class kescher.importers.**JournalImporter**(*csv_file*, *delimiter=';'*, *quotechar='"'*)
Bases: *kescher.importers.Importer*

import_rows()
This functions reads all rows and creates temp objects in a list, after all lines were read sucessfully, the objects are saved.

kescher.logging module

kescher.logging.**setup_logging**(*cwd*)
Creates and returnes a logger with the default logging level "INFO" in the current working directory.

kescher.models module

class kescher.models.**Account**(**args*, ***kwargs*)
Bases: *kescher.models.BaseModel*

Accounts are used to structure bookings e.g. by type, customer, etc.

DoesNotExist
alias of AccountDoesNotExist

account_entries

child_accounts

id = <AutoField: Account.id>

name = <CharField: Account.name>

parent = <ForeignKeyField: Account.parent>

parent_id = <ForeignKeyField: Account.parent>

updated_at = <DateTimeField: Account.updated_at>

class kescher.models.**BaseModel**(**args*, ***kwargs*)
Bases: peewee.Model

DoesNotExist
alias of BaseModelDoesNotExist

id = <AutoField: BaseModel.id>

save(**args*, ***kwargs*)

updated_at = <DateTimeField: BaseModel.updated_at>

class kescher.models.**Booking**(**args*, ***kwargs*)
Bases: *kescher.models.BaseModel*

A booking references a (partial) amount of a JournalEntry to an Account.

DoesNotExist
alias of BookingDoesNotExist

account = <ForeignKeyField: Booking.account>

account_id = <ForeignKeyField: Booking.account>

comment = <CharField: Booking.comment>

id = <AutoField: Booking.id>

journalentry = <ForeignKeyField: Booking.journalentry>

journalentry_id = <ForeignKeyField: Booking.journalentry>

updated_at = <DateTimeField: Booking.updated_at>

value = <DecimalField: Booking.value>

class kescher.models.**Document**(**args*, ***kwargs*)
Bases: *kescher.models.BaseModel*

A Document is an invoice/receipt which reasons a JournalEntry.

DoesNotExist
alias of DocumentDoesNotExist

content = <TextField: Document.content>

hash = <CharField: Document.hash>

id = <AutoField: Document.id>

journal_entries

static make_hash(*path*)

path = <CharField: Document.path>

updated_at = <DateTimeField: Document.updated_at>

class kescher.models.**JournalEntry**(**args*, ***kwargs*)
Bases: *kescher.models.BaseModel*

A JournalEntry is one row (line) in your imported journal (bank statement).

DoesNotExist
alias of JournalEntryDoesNotExist

account_entries

balance = <DecimalField: JournalEntry.balance>

date = <DateField: JournalEntry.date>

document = <ForeignKeyField: JournalEntry.document>

document_id = <ForeignKeyField: JournalEntry.document>

id = <AutoField: JournalEntry.id>

imported_at = <DateTimeField: JournalEntry.imported_at>

receiver = <CharField: JournalEntry.receiver>

sender = <CharField: JournalEntry.sender>

subject = <CharField: JournalEntry.subject>

updated_at = <DateTimeField: JournalEntry.updated_at>

value = <DecimalField: JournalEntry.value>

class kescher.models.**PathField**(*null=False*, *index=False*, *unique=False*, *column_name=None*, *default=None*, *primary_key=False*, *constraints=None*, *sequence=None*, *collation=None*, *unindexed=False*, *choices=None*, *help_text=None*, *verbose_name=None*, *index_type=None*, *db_column=None*, *_hidden=False*)
Bases: peewee.Field

db_value(*value*)

field_type = 'TEXT'

python_value(*value*)

class kescher.models.**VirtualBooking**(**args*, ***kwargs*)
Bases: *kescher.models.BaseModel*

To allow for cash accounting, all invoices are to be created as virtual account entries.

DoesNotExist
alias of VirtualBookingDoesNotExist

account = <ForeignKeyField: VirtualBooking.account>

account_id = <ForeignKeyField: VirtualBooking.account>

comment = <CharField: VirtualBooking.comment>

date = <DateField: VirtualBooking.date>

document = <ForeignKeyField: VirtualBooking.document>

```
    document_id = <ForeignKeyField:  VirtualBooking.document>
    id = <AutoField:  VirtualBooking.id>
    updated_at = <DateTimeField:  VirtualBooking.updated_at>
    value = <DecimalField:  VirtualBooking.value>
kescher.models.create_tables()
```

kescher.sanitizers module

This script sanitizes CSV files downloaded from Postbank (Germany). It strips useless characters, and converts the numbers to a reasonable and usable format.

exception kescher.sanitizers.**CsvHeaderError**
Bases: Exception

class kescher.sanitizers.**PostBankCsvParser**(*q*)
Bases: object

amount_to_decimal(*value*)

expected_header = ['Buchungsdatum', 'Wertstellung', 'Umsatzart', 'Buchungsdetails', 'A

get_entry(*row*)

header_ok(*header*)

sanitize_subject(*subject*)

Module contents

- genindex
- modindex
- search

CHAPTER 2

Authors, Copyright, License

kescher is developed by Florian Rämisch on behalf of *Westnetz w.V.* an independent not-for-profit internet service provider (ISP). It is licensed under the GNU General Public License v3. Copyright is by *Florian Rämisch, 2020*.

Python Module Index

k

Index

I

J

K

M

N

P

R

S

T

U

V