SYSDEV2 documentation *Release*

anonymous

Mar 14, 2023

Table of Contents

Index 9

```
customer facing add form
class frontend.add_form.BaseAddForm ( page: str, fields: Tuple[str, str], title: str, order_id: str |
None = None)
    Bases: object
    base form for adding things to the db
    __init__ ( page: str, fields: Tuple[str, str], title: str, order_id: str | None = None ) \rightarrow None
    __weakref__
         list of weak references to the object (if defined)
    add ( inputs: list ) \rightarrow None
         add the item to backend
         Parameters inputs (list) – inputs from the form
    cancel() \rightarrow None
         cancel action to go back to main page, on cancel button
    destroy()
         destroy root error message
    destroy_both()
         destroy error message and input window
    fetch (entries: tuple ) → None
         get input text to list to input to backend
         Parameters entries (tuple) – tkinter input fields, .get() transforms to strings
    init_ui ( root: Frame, fields: tuple ) → tuple
         initialise what is shown in the window namely labels and input boxes
         Parameters
                         • root (Frame) – root frame
                         • fields (tuple) - fields to be labled
         Returns
                      _description_
         Return type tuple
    message ( message: str, command: object )
         displays message to user
         Parameters
                        • message (str) - message to be displayed of the user
```

• **command** (object) – command after pressing the ok button

return_back () → None

```
go back to main page
base order/order page ui - NOTE: this is hideously long - as there is no way to make it shorter
without either tkinter breaking or having lots of duplicate code
class frontend.base_page.orderListForm ( page_type: str )
    Bases: object
    base order page
    __init__ ( page\_type: str ) \rightarrow None
         constructure for the order page class,
             shows everything seen in the page when it first pops up
        understandably its not the most pythonic
             but I would rather make the app more usable
                  as everything can be controlled from two pages
      weakref
        list of weak references to the object (if defined)
    create_items_tree ( listframe: object ) → Treeview
        create list tree full of items :param listframe: frame to put list tree into :type listframe: object
                     tree view full of items
        Return type ttk.Treeview
    create_menu_tree ( listframe: object ) → Treeview
        create tkinter treeview of menu
        Parameters listframe (object) – tk inter frame to put list tree in
        Returns
                     list tree
        Return type ttk. Treeview
    create_order_tree ( listframe: object ) → Treeview
        create list tree for orders :param listframe: frame to put list tree into :type listframe: object
                     list tree full of orders
        Returns
        Return type ttk.Treeview
    delete_menu_item_backend(item: str) \rightarrow None
         deete an item from the db
    delete_order_backend ( id_value: str ) → None
        delete an order from the order DB :param id_value: selected order id from listtree :type
        id value: str
    populate_items\_tree(order_id: str) \rightarrow None
        get items from order and populate tree :param order_id: order id to populate :type order_id:
        string
    populate_listree ( listtree: Treeview ) → None
        populate a tree view with data :param listtree: tk tree view to populate :type listtree: ttk.Tree-
        view :param page: what type of page its in,
               to tell the function what type of data to populate with.
    update_menu_item_backend ( item: str, new_price: str ) → None
         update the price of an item in the backend
```

```
update order backend (order id: int, customer: str, location: str) \rightarrow None
        backend methods to update an item in the db :param id: order if :type id: int :param
        customer: name of customer (unique) :type customer: str :param location: location placed
        with order (unique) :type location: str
common utils code used in the front end
frontend.front_end_utils.configure_listree (listtree: Treeview, listframe: Frame) 

Tree-
view
    configure a tree view for the main pages
    Parameters
                   • listtree (ttk. Treeview) - tree view to configure
                   • listframe (Frame) – frame which tree veiw is placed
    Returns
                configured tree view
    Return type ttk. Treeview
frontend.front_end_utils.create_cmdframe ( detailsframe: Frame ) → Frame
    command frame to put buttons into
    Parameters detailsframe (Frame) - control panel to place into
                a command frame
    Returns
    Return type Frame
frontend.front_end_utils.create_list_frame (baseframe: Frame, column: int = 0, row: int
= 1) \rightarrow Frame
    create a frame for tree views
    Parameters
                   • baseframe (Frame) – base window frame of put into
                   • column (int, optional) - column to place this into in the baseframe.
                     Defaults to 0.
                   • row (int, optional) - row to place this into in the baseframe. Defaults to 1.
    Returns
                frame for tree views
    Return type Frame
frontend.handle_exceptions.handle_3words_exceptions (func: object )
    handle exceptions from the what three words api
    Parameters func (function) – function to be decorated
    Returns
               decorator
frontend.handle_exceptions.handle_db_exceptions (func: object)
    decorator to handle exceptions from the database
    Parameters func (function) – function to be decorated
    Returns
               decorator
class frontend.pop_up.UpdateMsg ( message: str )
    Bases: object
    display message when something is succesfully updated or otherwise
     _init__ ( message: str ) \rightarrow None
        constructor for class for displaying update messages
        Parameters message (str) – message to be displayed
```

3

```
weakref
         list of weak references to the object (if defined)
    destroy ( ) \rightarrow None
         destroy window
utils code for the backend
class backend.common.DBClass
    Bases: object
    class used by all backend classes for interacting with sqllite
    \_\_sql_attempt ( sql: str ) \rightarrow list
         wrapper for a sql attempt to connect/commit to db (private)
         Parameters sql(str) - sql query in a string
         Returns
                      return from db
         Return type list
     _weakref__
         list of weak references to the object (if defined)
    execute_sql (sql: str) \rightarrow list
         wrapper for a sql attempt with try/except
         Parameters sql(str) - sql in string
                      return from db
         Returns
         Return type list
    init_tables()
         creates tables if hasnt already been created
backend.common.coordinates_to_words ( lat: str, lon: str ) \rightarrow dict
    convert to co ords to words using the what three words api
    Parameters
                    • lat (str) – latitude
                     • lon (str) – longitude
    Returns
                 threewords location string
    Return type dict
backend.common.words_to_coordinates ( words: str ) \rightarrow str
    turn what three words location into latitude and location
    Parameters words (str) – what three words as a string with each word separated with a.
    Returns
                 co ords location in a string e.e. (1.1,1.1)
    Return type str
handler class for returning objects for interacting with the db or select all type functions
class backend.main.Backend
    Bases: object
    handle returning of objects for interacting with the database
     _weakref_
         list of weak references to the object (if defined)
```

```
existing item ( name: str ) \rightarrow ExistingMenuItem
        instantiates an existing item object
        Parameters name (str) – of existing menu item
        Returns
                     instantiates object
        Return type ExistingMenuItem
    existing_order ( order_id: str ) → ExistingOrder
        instantiates na existing order object
        Parameters order_id (int) - order id of exisitng order
        Returns
                     instantiated order object
        Return type ExistingOrder
    classmethod init_db() \rightarrow None
        initialise the db
    new_item ( name: str, price: str ) → NewMenuItem
        instantiates a new item object
                        • name (str) - Name of new menu item
        Parameters
                        • price (int) - Price
        Returns
                     instanciated object
        Return type NewMenuItem
    new_order ( customer: str, location_co_ords: str ) → NewOrder
        instantiates a new order object
                        • customer (str) – customer username
        Parameters
                        • location_co_ords (str) - latitude and longitude separated by
        Returns
                     instantiated new order object
        Return type NewOrder
    classmethod view_menu() → list
        shows alll menu items and their prices
        Returns
                     list returned by db of all menu items i.e. [("hotdog", 1)]
        Return type list
    classmethod view_orders ( ) → list
        show all orders from the db (no items attatched)
        Returns
                     list returned by db of all orders
        Return type list
all interactions with the menu
class backend.menu.ExistingMenuItem ( name: str )
    Bases: Menu
    instantiates a new menu item object
      _init__ ( name: str ) \rightarrow None
        init method for a new menu item
```

5

```
Parameters name (str) – unque name for menu item already in db
    property price: int
         fetches price of a menu item from the db
         Returns
                      price of menu item
         Return type int
    \textbf{save} \ (\ ) \to None
         save the menu item stored in the object to the db
class backend.menu.Menu ( name: str | None = None )
    Bases: DBClass
    SUperclass with basic methods for interacting with the menu
     __init__ ( name: str \mid None = None ) \rightarrow None
    delete\_from\_db() \rightarrow None
         Delete teh menu item (stored in the object) from the db
    view_menu() \rightarrow list
         view the menu as it currently is in the db
                      e.g. [("name1", price)]
         Returns
         Return type list
class backend.menu.NewMenuItem ( name: str, price_input: str )
    Bases: Menu
    instantiates a new menu item object
    __init__ ( name: str, price_input: str ) \rightarrow None
         constructor for a new menu item
         Parameters
                         • name (str) – name of menu item (must be unique!)
                         • price (int) - price fo menu item
    save ( ) \rightarrow None
         save the menu item stored in the object
all interactions with orders
class backend.order.ExistingOrder ( order_id: str )
    Bases: Order
    class for all interactions with existing orders
    __init__ ( order_id: str ) \rightarrow None
         constructor for existing order, found by id
    __set_date() \rightarrow datetime
         get the date of the order from the db and
             overwrite the date in the super class,
         if not found set to none, added to super class both classes needed set order
                      date of the order
         Returns
         Return type datetime
    add_items ( name: str, quantity: int ) \rightarrow None
         add a given quantity of an item to the db
```

```
Parameters
                        • name (str) – name of menu item (must be unique!)
                        • quantity (int) – amount of the item to add the db
    property location_co_ords
         converts what three words to co ords
         Returns
                      lat and long in a string e.g. "1,1"
         Return type str
    remove_items ( name: str ) \rightarrow None
         remove an item and all of its quantities from the db
         Parameters name (str) – name of menu item (must be unique!)
    update_order () \rightarrow None
         updates order in db with new values stored in object
class backend.order.NewOrder ( customer: str, location_co_ords: str )
    Bases: Order
    new order object which can later be added to db
    __init__ ( customer: str, location_co_ords: str ) \rightarrow None
         constructor for a new order
         Parameters
                        • customer (str) – custoemr name (must be unique)
                        • location_co_ords (str) - lat and long in a string e.g. "1,1"
    property location_words: str
         converts co ords to what three words
         Returns
                      what threee words str separated by . e.g. "hello.my.name"
         Return type str
    property order_id: int
         find next available order id for order to take
                      id of the orders
         Returns
         Return type int
    save ( ) \rightarrow None
         save teh order stored in the object to the database
class backend.order.Order
    Bases: DBClass
    super class for all order interactions
    \_init\_() \rightarrow None
         constructor for superclass
    delete ( ) \rightarrow None
         delete the order stored in the object from the db
    get\_total() \rightarrow int
         return total cost of items in an order
                      total cost of items in an order
         Returns
         Return type int
```

7

custom exceptions used by both front end and backend whislt this technically counts as higher coupling, I would argue that this would normally be imported in a package and therfore would not be a problem, I cannot so instead have produced this. It could have been avoided by misusing python exceptions but their was less pythonic than making custom ones and sufferign the higher coupling

```
exception custom.exceptions.NoKeyError
Bases: Exception
raised when no key is found in the environment
__weakref__
list of weak references to the object (if defined)

exception custom.exceptions.WhatThreeWordsError
Bases: Exception
raised when there is a problem with the what three words api
__weakref__
list of weak references to the object (if defined)
```

- genindex
- modindex
- search

Symbols	В
init() (backend.menu.ExistingMenuItem	Backend (class in backend.main), 4
method), 5	backend.common
init() (backend.menu.Menu method), 6	module, 4
init() (backend.menu.NewMenuItem	backend.main
method), 6	module, 4
init() (backend.order.ExistingOrder	backend.menu
method), 6	module, 5
init() (backend.order.NewOrder method), 7	backend.order
init() (backend.order.Order method), 7	module, 6
init() (frontend.add_form.BaseAddForm method), 1	BaseAddForm (class in frontend.add_form), 1
init() (frontend.base_page.orderListForm	С
method), 2	
init() (frontend.pop_up.UpdateMsg	cancel() (frontend.add_form.BaseAddForm
method), 3	method), 1
set_date() (backend.order.ExistingOrder	configure_listree() (in module frontend.fron-
method), 6	t_end_utils), 3
sql_attempt() (backend.common.DBClass	coordinates_to_words() (in module backend
method), 4	common), 4
weakref (backend.common.DBClass	create_cmdframe() (in module frontend.fron-
attribute), 4	t_end_utils), 3
weakref (backend.main.Backend attribute),	create_items_tree() (frontend.base_page.or-
4	derListForm method), 2
weakref (custom.exceptions.NoKeyError	create_list_frame() (in module frontend.fron-
attribute), 8	t_end_utils), 3
weakref (custom.exceptions.WhatThree-	create_menu_tree() (frontend.base_page.or- derListForm method), 2
WordsError attribute), 8	create_order_tree() (frontend.base_page.or-
weakref (frontend.add_for-	derListForm method), 2
m.BaseAddForm attribute), 1	
weakref (frontend.base_page.orderList-	custom.exceptions module, 8
Form attribute), 2	module, o
weakref (frontend.pop_up.UpdateMsg	D
attribute), 4	
	DBClass (class in backend.common), 4
A	delete() (backend.order.Order method), 7
add() (frontend.add_form.BaseAddForm	delete_from_db() (backend.menu.Menu
method), 1	method), 6
add_items() (backend.order.ExistingOrder	delete_menu_item_backend() (frontend.base
method), 6	page.orderListForm method), 2
	delete_order_backend() (frontend.base
	page.orderListForm method), 2

destroy() (frontend.add_form.BaseAddForm	M
method), 1 destroy() (frontend.pop_up.UpdateMsg	M main
method), 4	module, 1
destroy_both() (frontend.add_for-	Menu (class in backend.menu), 6
m.BaseAddForm method), 1	message() (frontend.add_form.BaseAddForm method), 1
E	module
execute_sql() (backend.common.DBClass method), 4	backend.common, 4 backend.main, 4
existing_item() (backend.main.Backend method), 5	backend.menu, 5 backend.order, 6
existing_order() (backend.main.Backend method), 5	custom.exceptions, 8 frontend.add_form, 1
ExistingMenuItem (class in backend.menu), 5 ExistingOrder (class in backend.order), 6	frontend.base_page, 2 frontend.front_end_utils, 3
F	frontend.handle_exceptions, 3 frontend.pop_up, 3
fetch() (frontend.add_form.BaseAddForm	main, 1
method), 1 frontend.add_form	N
module, 1	new_item() (backend.main.Backend method), 5
frontend.base_page	new_order() (backend.main.Backend method),
module, 2	5
frontend.front_end_utils	NewMenuItem (class in backend.menu), 6
module, 3	NewOrder (class in backend.order), 7
frontend.handle_exceptions module, 3	NoKeyError, 8
frontend.pop_up	0
module, 3	Order (class in backend.order), 7
G	order_id (backend.order.NewOrder property),
get_total() (backend.order.Order method), 7	orderListForm (class in frontend.base_page), 2
Н	Р
handle_3words_exceptions() (in module fron- tend.handle_exceptions), 3	populate_items_tree() (frontend.base_page.or- derListForm method), 2
handle_db_exceptions() (in module frontend.handle_exceptions), 3	<pre>populate_listree() (frontend.base_page.or-</pre>
ı	price (backend.menu.ExistingMenuItem property), 6
init_db() (backend.main.Backend class	•
method), 5	R
init_tables() (backend.common.DBClass method), 4	remove_items() (backend.order.ExistingOrder method), 7
init_ui() (frontend.add_form.BaseAddForm method), 1	return_back() (frontend.add_for- m.BaseAddForm method), 2
L	S
location_co_ords (backend.order.ExistingOrder	save() (backend.menu.ExistingMenuItem
property), 7	method), 6
location_words (backend.order.NewOrder property), 7	save() (backend.menu.NewMenuItem method)
	save() (backend.order.NewOrder method), 7 set_order_date() (backend.order.Order method), 8

10 Index

U

```
update_menu_item_backend() (frontend.base_-
page.orderListForm method), 2
update_order() (backend.order.ExistingOrder
method), 7
update_order_backend() (frontend.base_-
page.orderListForm method), 3
UpdateMsg (class in frontend.pop_up), 3
```

V

```
view_menu() (backend.main.Backend class method), 5
view_menu() (backend.menu.Menu method), 6
view_order_items() (backend.order.Order method), 8
view_orders() (backend.main.Backend class method), 5
view_orders() (backend.order.Order method), 8
```

W

WhatThreeWordsError, 8 words_to_coordinates() (in module backend.common), 4

Index 11