Module Interface Specifications

Purpose of this document:

The purpose of this document is to describe the ways in which the various modules of BeetBook interact with one another and to provide context to the various function calls that are invoked during those interactions.

App.py

The App class is the only module intended to be directly interacted with by the FrontEnd/User. It contains all the necessary attributes needed to support the intended functionality of BeetBook.

check_address_book_id(self,book_id):
Given an integer value for (book_id), checks whether there is an open AddressBook denoted by book_id and returns True/False

check_entry_id(self, book_id, entry_id):
Given integer values for (book_id, entry_id), checks whether there is an Entry denoted by entry_id in the AddressBook denoted by book_id

get_default_fields:
Returns a list of strings which denotes the default fields when creating a new entry.

create_address_book(self, address_book_name):
Takes in a string denoted by address_book_name, creates a new empty AddressBook and sets its name to address_book_name, then returns its newly assigned address_book_id

open_address_book(self, filename = ''):
Takes in a string indicating the absolute or relative path to an existing AddressBook, opens the AddressBook and returns its newly assigned address_book_id

close_address_book(self, address_book_id = -1, force = False):
Takes in an integer values (address_book_id) and Boolean (force). If force is true, the AddressBook denoted by address_book_id will be closed and return will be True. If force is false, the AddressBook denoted by address_book_id will be checked for changes since the

last save. If changes are detected, False is returned. If no changes are detected, the AddressBook is closed and True is returned

save_address_book(self, address_book_id = -1, filename = “”):
Takes in an integer value (address_book_id) representing an open address book and an optional string (filename). If the filename is not an empty string, the filepath of the AddressBook denoted by address_book_id is reassigned. If the filepath is an empty string, the AddressBook is saved to its default file location and True is returned. If the AddressBook has no default file location and a file location was not provided in filename, False is returned.

create_entry(self, address_book_id = -1, entry_data = None):
Takes in an integer value (address_book_id) representing an open address book and a dictionary (entry_data) representing an Entry. A new Entry object is created from entry_data and assigned to the AddressBook denoted by address_book_id. Upon success, True is returned. If an error is encountered, False is returned.

delete_entry(self, address_book_id = -1, entry_id = -1):
Takes in an integer value (address_book_id) representing an open address book and an integer (entry_id) representing an Entry within the open address book. The Entry denoted by entry_id is deleted from the AddressBook denoted by address_book_id. Upon success, True is returned. If an error is encountered, False is returned.

update_entry(self,: entry_data = {}):
Takes in a dictionary (entry_data) representing an existing Entry within an existing AddressBook. Updates the existing Entry within the existing AddressBook with the data contained within entry_data. Returns True on success and False if an error is encountered.

get_address_book(self, address_book_id = -1):
Takes in an integer value (address_book_id) representing an open address book and returns a JSON representation of the AddressBook. Returns False if an error is encoutered.

import_tsv(self, address_book_id = -1, filename = ''):
Takes in an integer value (address_book_id) representing an open address book and a string (filename) representing the absolute or relative path to a TSV file. Imports the TSV file into the existing AddressBook denoted by address_book_id. Returns true upon success and False if an error is encountered.

sort_address_book(self, address_book_id = -1, key = 'last_name', orderBy = 'asc'):
Takes in an integer value (address_book_id) representing an open address book and two strings (last_name) and (orderBy) representing the desired sort order. Changes the default sort order in the AddressBook denoted by address_book_id and resorts the AddressBook. Returns True on success and False if an error is encountered
update_book_name(self,address_book_id = -1, new_book_name = ''):
Takes in an integer value (address_book_id) representing an open address book and a string (new_book_name) representing the desired book name. Updates the book name in

the AddressBook denoted by address_book_id. Returns True on success and False if an error is encountered.

get_open_address_books(self):
Returns the dictionary App class attribute self.open_address_books

AddressBook.py

The AddressBook class contains all of the necessary functionality to create and maintain a single instance of an AddressBook. It is not intended for direct access by the User/FrontEnd

check_entry_id(self, entry_id):
Takes in an integer(entry_id) and returns True/False if the Entry denoted by entry_id exists within the current AddressBook

def __hash__(self):
Custom hash function. Used in determining if changes have been made to an AddressBook during normal user operation and before save. Returns an integer representing the current AddressBook hash

def get_file_name(self):
Returns a string representing the current AddressBook's local file location

set_file_name(self, filename = ''):
Takes in a string(filename) and redefines the AddressBook attribute as filename. Returns True on success and False upon Failure.

add_entry(self, entry_object = None):
Takes in an Entry Object (entry_object), assigns the values entry_id, address_book_idand adds it to the dictionary of current entries within the AddressBook. Returns True on success and False if an error is encountered

delete_entry(self, entry_id = -1):
Takes in an integer(entry_id) representing an existing entry within the current AddressBook. If the Entry exists, it is deleted and True is returned. False is returned if the Entry does not exist.

update_entry(self, entry_data = {}):
Takes in a dictionary(entry_data) representing an existing Entry object within the Current AddressBook. Updates the existing Entry Object with the data stored in entry_data. True is returned upon success and False is returned if an error is encoutered.

import_tsv(self, filename = '', address_book_id = -1):
Takes in a string(filename) representing the relative or absolute path of a TSV file to be imported and an integer(address_book_id) representing the id of the current AddressBook.

Imports the entries contained within the TSV file into the current AddressBook. Returns True on success and False if an error is encountered.

export_tsv(self, filename = '', entry_ids = []):
Takes in a string(filename) representing the relative or absolute path of a TSV file to be exported to and an optional list of integers(entry_ids) representing entry_ids within the current AddressBook to be exported. If the entry_ids list is empty all Entries are exported. If entry_ids is not empty, the specified Entries are exported. Returns True on success and False if an error is encountered.

import_json(self, filename = '',address_book_count = -1):
Takes in a string(filename) representing the relative or absolute path of a JSON file to be imported and an integer(address_book_id) representing the id of the current AddressBook. Imports the metadata and Entries from the file denoted by filename. Returns True on success and False if an error is encountered.

save(self, filename = ''):
Takes in an optional string(filename) representing the absolute or relative path for the AddressBook to be saved to. If the filename is not empty, the AddressBook attribute self.filename is updated to filename. The current AddressBook is then saved to the file path specified in the AddressBook attribute self.filename. True is returned on success and False if an error is encountered.

sanatize_for_export(self,data_dict):
Takes in a dictionary(data_dict) representing an Entry object that is destined to be exported to a TSV. The data_dict is then sanitized of all app specific information such as entry_id and address_book_id as well as features yet to be implemented. The sanitized data_dict is then returned.

sanatize_for_save(self,data_dict):
Takes in a dictionary(data_dict) representing an Entry object that is destined to be saved to JSON. The data_dict is then sanitized of all instance specific information such as entry_id and address_book_id. The sanitized data_dict is then returned.

sort(self, key = 'last_name', orderBy = 'asc'):
Takes in optional strings (key) and (orderBy) representing the desired AddressBook sort order. The relevant self.meta attributes are updated and the AddressBook is then resorted based on those values

sort_helper(self, entry,key):
Takes in an Entry object (entry) and returns an information necessary to perform the sort operation.

Entry.py

The Entry class contains all of the necessary functionality to create and maintain a single instance of a contact entry. It is not intended for direct access by the User/FrontEnd

def __hash__(self):
Custom hash function. Used in determining if changes have been made to an Entry during normal user operation and before save. Returns an integer representing the current Entry hash

Main.js

The main electron process. Handles the creation of windows, inner process communication, and starting the python api server.

getFilePath(filename):
Given a filename, this function gets the full path of the file and returns it.

getScriptPath():
Finds and returns the python executable.

createBackendProc():
Starts the backend server by calling the python process, and establishing process logs.

exitBackendProc():
Kills the backend server.

createMainWindow():
Creates the main window, sets attributes of that window, and loads in home page html.

createBookWindow(bookName, parentWindow):
Given the book name and the parent window, this function creates the address book window, sets attributes of that window, and enables navigation menu options for that window.

setBookWindowOptions(parentWindow, bookWindow, bookContents):
Given the parent window, the address book window, and the address book content, the function sets the necessary content for that address book on the page.

createNameModal():
Creates the new book name modal that appears when creating a new address book.

createCustomFieldModal():
This is a function for a new field to be added in a future release.

exportFile():
Exports to .tsv when called by menu.

importFile():
Imports from .json when called by menu.

openFile():
Opens a .json when called by menu and buttons.

openBook(filename):
Given a filename, this function opens route to backend and returns the id of the opened address book.

getBook(id):
Given an id, this function returns the book as a json object.

createMenu():
Creates a menu based on the mainMenuTemplate.

Book.html

The html page that includes all displayed elements in the address book window, the styles for those elements, and javascript functions.

sendEntry(entry):
Given an entry, this function makes a call to the api to edit entry.

sendSort(id, key, order):
Given an id, key, and order, this function makes a call to the api to sort.

update_sort():
Updates the sort parameters for the current selection.

pageFill(data):
Given a json object known as data, this function fill the page with the given content.

pageUpdate():
Calls the api to update a page.

cancelToggleDynamic(id):
Handles the onclick for the cancel button functionality.

cancelStylingAppear(id):
Creates styling for individual cancel and delete button ids.

editToggle(id):
Handles onclick for the edit button functionality.

closeBook(id, force = false):
Given an id and a force, this function makes a call to the api to close.

newEntry(id, entry):
Given an id and an entry, this function makes a call to the api for a new entry.

getBook(id):

Given an id, this function makes a call to the api to get the book.

deleteEntry(book_id, entry_id):

Given a book_id and an entry_id, this function makes a call to the api to delete an entry.

submitForm():

Handles the onclick for the form functionality.

sendDelete(id):

Given an id, this function tells the backend to delete.

saveBook(id, filename=’’, book_name=’’):

Given an id, filename, and book_name, this function makes a call to the api to save a book.

importBook(filename, id):

Given a filename and an id, this function makes a call to the api to import a book.

hideWindow():

This function refocuses the users screen to the main window.

exportBook(book_id, filename, entry_ids):

Given a book_id, filename, and entry_ids, this function makes a call to the api to export a book.