Travel Blog Documentation

Group: Broke Boys Inc

Amimul Bhuiyan, Julian Zelazny, Daniel Castro, Michael Desena

CSC 440-001-Fall 2021:Software Engineering.

Nov 30, 2021

Introduction

- We decided to use the JavaDoc Style format when writing our documentation which goes as follows:
    - ***+remove(stuff: String): boolean***
    - The example above means that remove is a public method (because of the plus) and it has one String parameter called stuff. Finally, it returns a boolean value

- **Agreed-Upon Coding Style**
    - Our agreed-upon coding styles are pretty standard:
        - Frontend and Backend are kept totally separate. Backend classes are inside the **backendoperations** folder which are called by the frontend whenever needed.

- **Important Note**
    - Note that the following documentation is done on the backend classes only and not really the front end since the frontend mostly consists of just displaying what the backend has done. None of the frontend features like buttons, were technically created by us and were created by the JavaFX framework and thus, their documentation is already given.

Class: DatabaseCredentials

Description: Just holds the credentials needed to access the database on the local host

Variables

- durl: String
- dusername: String
- dpassword: String

- durl holds the url of the database that this software should connect to.
- dusername holds the username of the database that this software should connect to.
- dpassword holds the password of the database that this software should connect to

Methods

- ***+getURL(): String***
    - Used to get the url needed in order to connect to the database.
    - Returns a String containing the database url

- ***+getUsername(): String***
    - Used to get the username needed in order to connect to the database.
    - Returns a String containing the username

- ***+getPassword(): String***
    - Used to get the password needed in order to connect to the database.
    - Returns a String containing the password

Class: AccountBackend

Description: This class is used to perform the backend operations for the login screen. Thus, its methods allow the user to basically login to the system or create an account

Variables

- durl: String
- dusername: String
- dpassword: String
- dbConnectionError: Runnable

- durl holds the url of the database that this software should connect to.
- dusername holds the username of the database that this software should connect to.
- dpassword holds the password of the database that this software should connect to
- dbConnectionError This is a method that holds a list of operations to be performed if there is a database connection issue

Methods

- ***+AccountBackend(databaseConnectionError: Runnable)***
    - Default Constructor. Initializes the class. Takes in a Runnable variable which is used to perform certain actions if there is a database connection error
    - databaseConnectionError → A runnable function used to perform a specific operation upon the failure to connect to a database

- ***+userLogin(usern: String, passw: String): String[]***
    - Used to validate the user login details and see if the user is registered. If the user is registered, it returns their username followed by their account type
    - usern → The username of the user trying to login
    - passw → The password of the user trying to login
    - Returns a String integer of size 2 with the first value containing a

- ***-validateLoginDetails(usern: String, passw: String): boolean***
    - This method is used by the **userLogin** method. This is the one that actually connects to the database and tries to find the specified user.
    - usern → The username of the user trying to login which is provided by the **userLogin** method
    - passw → The password of the user trying to login which is provided by the **userLogin** method
    - Returns true if the user login credentials are found. False if otherwise

- ***+createAccount(usern: String, passw: String): String***
    - Creates a new account for a new user.
    - usern → Desired username for the new account
    - pass → Desired password for the new account
    - Returns a String with one or the four following messages:
        - Username or Password is too long. Keep UNDER 20 characters!
        - Username or Password is too short. Keep OVER 1 character!
        - Account Successfully Created
        - Username is taken

- ***-accountCreationProcess(usern: String, passw: String): boolean***
    - This method is used by the **createAccount** method to actually connect to the database and enter the new login information for the new user.
    - usern → The desired username for the new account which is provided by the **createAccount** method.
    - passw → The desired password for the new account which is provided by the **createAccount** method.

Class: SearchScreenBackend

Description: Used to perform the backend operations for the search screen class which includes: getting a list of all posts, getting a list of posts specified by a certain country,

Variables

- durl: String
- dusername: String
- dpassword: String
- dbConnectionError: Runnable

- durl holds the url of the database that this software should connect to.
- dusername holds the username of the database that this software should connect to.
- dpassword holds the password of the database that this software should connect to
- dbConnectionError This is a method that holds a list of operations to be performed if there is a database connection issue

Methods

- ***+SearchScreenBackend(databaseConnectionError: Runnable)***
    - Default Constructor. Initializes the class. Takes in a Runnable variable which is used to perform certain actions if there is a database connection error
    - databaseConnectionError → A runnable function used to perform a specific operation upon the failure to connect to a database

- ***+getAllPosts(): LinkedList<MinimalPostDetails>***
    - This method is used to get a linkedList of minimalPostDetails object which just holds the post id, author, and post title
    - Returns a LinkedList of objects of type MinimalPostDetails.

- ***+getPostByCountry(specifiedCountry: String): LinkedList<MinimalPostDetails>***
    - This method is used to get all posts about specific countries.
    - specifiedCountry → The country that the posts should be about
    - Returns a linked list containing MinimalPostDetails Objects of all the posts about the specified country.

- ***+requestToBeCreator(username: String): boolean***
    - This method is to request creator privileges to certain account
    - username → The account that requested Creator privileges
    - Returns True if the request was successfully made. False if otherwise

Class: MinimalPostDetails

Description: A class just used to hold snippets of information about specific posts like postID, author, title, and country. Does not include actual posts. This is just used to list the available posts in the search screen

Variables

- postId: Integer
- posterUsername: String
- postTitle: String
- postCountry: String

- postID holds the id of the post
- posterUsername holds the author of the post
- postTitle holds the title of the post
- postCountry holds the country of the post

Methods

- +MinimalPostDetails(pid: Integer, pUsern: String, pTitle: String, pCountry: String)
    - Default constructor. Sets the id, author, title, and country of a particular post
    - pid → The post id
    - pUsern → The author of the post
    - pTitle → The title of the post
    - pCountry → The name of the country

- ***+getpostid(): Integer***
    - Used to get the id of the particular post
    - Returns an integer containing the post id

- ***+getPosterUsername(): String***
    - Used to get the name of the author
    - Returns a String containing the name of the author

- ***+getPostTitle(): String***
    - Used to get the title of the post
    - Returns a String containing the title of the post

- ***+getPostCountry(): String***
    - Used to the name of the country that the post is about

    - Returns a String containing the name of the country that this post is about

Class: ViewBlogBackend

Description: Used to perform backend operations on the ViewBlogScreen class which includes: getting the contents of a particular blog and deleting the blog.

Variables

- durl: String
- dusername: String
- dpassword: String
- dbConnectionError: Runnable

- durl holds the url of the database that this software should connect to.
- dusername holds the username of the database that this software should connect to.
- dpassword holds the password of the database that this software should connect to
- dbConnectionError This is a method that holds a list of operations to be performed if there is a database connection issue

Methods

- ***ViewBlogBackend(databaseConnectionError: Runnable)***
    - Default Constructor. Initializes the class. Takes in a Runnable variable which is used to perform certain actions if there is a database connection error
    - databaseConnectionError → A runnable function used to perform a specific operation upon the failure to connect to a database

- ***+getBlogContent(id: Integer): String***
    - Used to get the contents of the blog specified by its id
    - id → The id of the blog.
    - Returns the contents of the blog specified by the id

- ***+deleteBlogById(id: Integer)***
    - Deletes a certain blog as specified by id
    - id → The id of the blog

Class: PostBlogBackend

Description: Used to perform backend operations on the PostBlogScreen class which includes: posting the blog

Variables

- durl: String
- dusername: String
- dpassword: String
- dbConnectionError: Runnable

- durl holds the url of the database that this software should connect to.
- dusername holds the username of the database that this software should connect to.
- dpassword holds the password of the database that this software should connect to
- dbConnectionError This is a method that holds a list of operations to be performed if there is a database connection issue

Methods

- ***+PostBlogBackend(databaseConnectionError: Runnable)***
    - Default Constructor. Initializes the class. Takes in a Runnable variable which is used to perform certain actions if there is a database connection error
    - databaseConnectionError → A runnable function used to perform a specific operation upon the failure to connect to a database

- ***+postBlog(posterUsername: String, title: String, country: String, content: String)***
    - Used to save a blog to the database
    - posterUsername → Username of the person who made this post
    - title → Title of the blog
    - country → Name of the country
    - content → The contents of the blog

Class: UserRequestsBackend

Variables

- durl: String
- dusername: String
- dpassword: String
- dbConnectionError: Runnable

- durl holds the url of the database that this software should connect to.
- dusername holds the username of the database that this software should connect to.
- dpassword holds the password of the database that this software should connect to
- dbConnectionError This is a method that holds a list of operations to be performed if there is a database connection issue

Description: Used to perform backend operations on the UserRequestsScreen Class which includes: getting all the requests to be creator, Setting accounts to be a creator, and removing creator requests

Methods

- ***+UserRequestsBackend(databaseConnectionError: Runnable)***
    - Default Constructor. Initializes the class. Takes in a Runnable variable which is used to perform certain actions if there is a database connection error
    - databaseConnectionError → A runnable function used to perform a specific operation upon the failure to connect to a database

- ***+getCreatorRequests(): LinkedList<String>***
    - Gets a list of usernames of users that requested creator privileges.
    - Returns a LinkedList of user that requested creator privileges

- ***+setAccountToCreator(usern: String): boolean***
    - Grants creator privileges to a specified user
    - usern → The user who requested the creator privileges
    - Returns true if the account was successfully given creator privileges. False if otherwise

- ***+removeCreatorRequest(usern: String): boolean***
    - Removes a creator privileges request
    - usern → The user whose privileges should be removed
    - Returns true if the request was successfully removed. False if otherwise

Class: CurrentUser

Description: Used to keep track of the current user that is using the system

Variables

- username: String
- accountType: Integer

username holds the name of the current user. accountType holds the account privileges of the current user

Methods

- ***+CurrentUser()***
  - Used to initialize the class. Sets the username variable to NO CURRENT USER and accounttype variable to -99. This signals a guest account
- ***+getUsername(): String***
  - Used to get the username of the current user
  - Returns a String containing the username of the current user
- ***+getAccountType(): Integer***
  - Used to get the accountType of the current user
  - Returns an integer containing the account type of the current user
- ***+signout()***
  - Used to sign out the current user. Current username variable is set to NO CURRENT USER and accounttype variable back to -99 which signals a guest
- ***+signin(u: String, a: Integer)***
  - Records the current user that is using the system
  - u → Username of the current user
  - a → Accounttype of the current user
- ***+doesCurrentUserExist(): boolean***
  - Check to see if there is a current user or a guest account
  - Returns false if there is a guest account. True if otherwise.