CultureMesh Android Documentation

Release 0.2.0-beta

CultureMesh and Stanford CodeTheChange

Aug 27, 2018

Contents:

CHAPTER 1

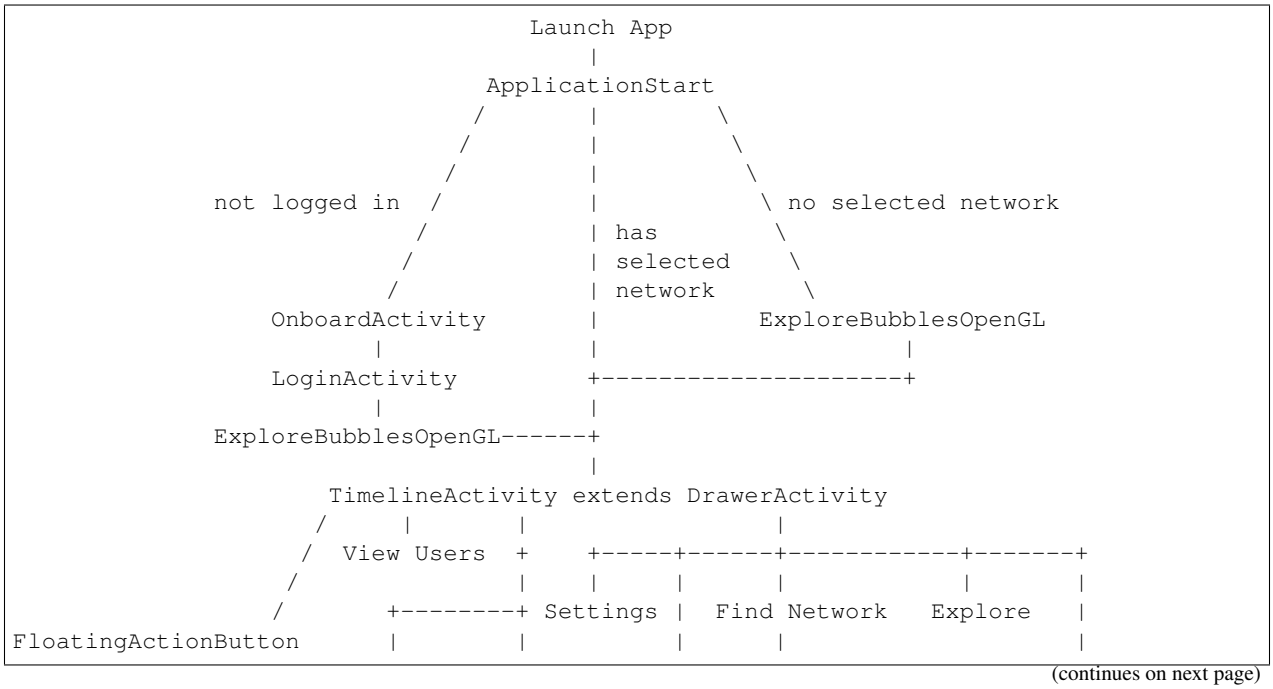
Introduction

This is the developer documentation for the Android app for CultureMesh. This app remains in development, so this documentation is geared toward developers, not end users.

This documentation and the source code for the app were created by Stanford University's Code the Change for CultureMesh.

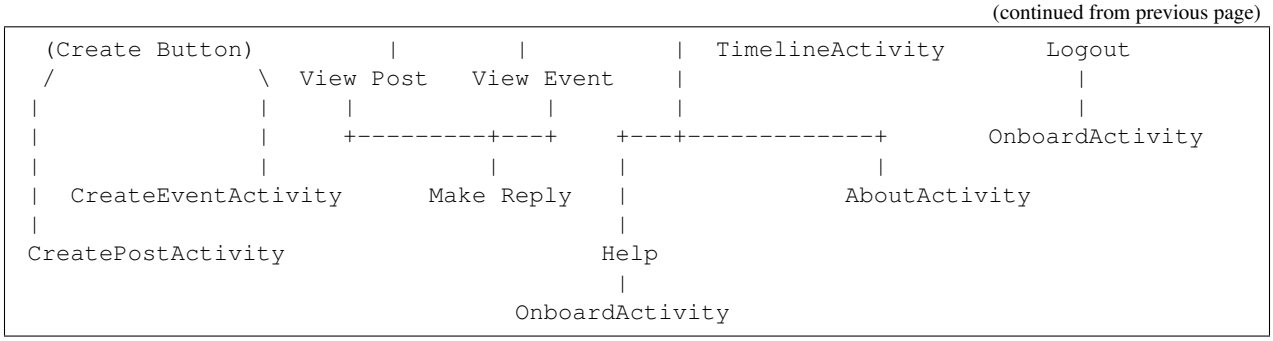
1.1 User Interaction Flow

The below diagram illustrates roughly how users move through the app and between different activities.



(continues on next page)

(continued from previous page)



1.2 Structure of Code

1.2.1 User Interface

The screens displayed to users are called activities. Each one has an associated `*Activity.java` file that defines a `*Activity` class. For example, the timeline for a network has an associated *`TimelineActivity`* that controls it. Each activity may also include fragments (e.g. *`ListNetworksFragment`*) that define part of an activity and can be reused across multiple activities. They are also often used for parts of the activity that sometimes disappear or are exchanged with other fragments. Each activity and fragment may also have layouts defined in the `res` folder as `activity_*.xml` and `content_*.xml`.

Adapters

In some activities, large scrollable lists need to be displayed. Generating the displayable list elements (`View` s) for all the items is inefficient, so `RecyclerView` s are used, which efficiently handle generating the list using adapters. These classes (e.g. *`RVAdapter`*) provide the functionality `RecyclerView` needs to dynamically generate each displayed list element.

1.2.2 Data Models

Conceptually, the data stored with CultureMesh, including *`Place`* s, *`User`* s, *`Network`* s, and *`Event`* s are represented as models. These models are represented in JSON when in transit between the app and the server and as instances of the appropriate class (see *`models`*) within the app. The *`API`* class handles converting between object and JSON formats, often using constructors and getters within the model's class (e.g. `Post.getPostJSON`).

Places and Locations

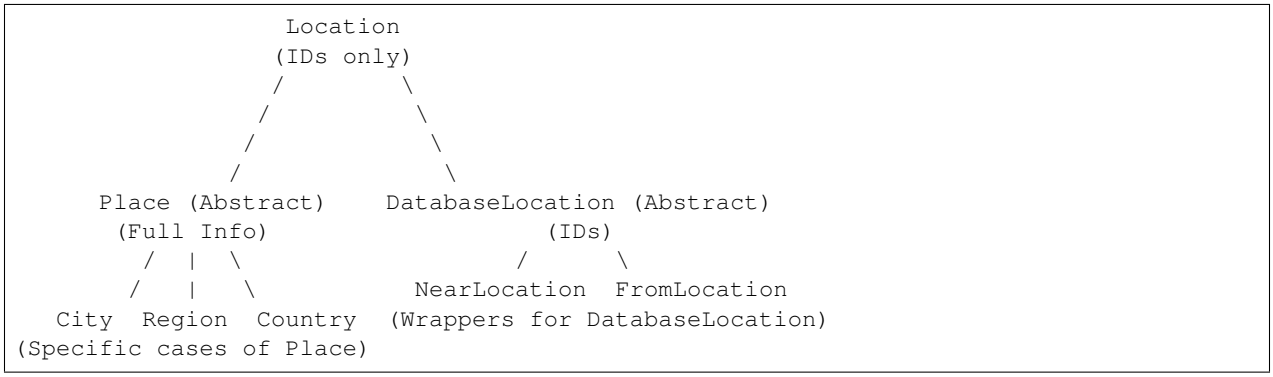
`Place` s and/or *`Location`* s are part of the definition of a *`Network`*, and they are used by themselves when displaying lists from which the user can choose parameters to narrow their search for networks.

The difference between a place and a location is not well captured in their names. A place defines a particular geographic area that is one of three types: a *`City`*, a *`Region`*, or a *`Country`*. A location can be any of those three, and includes definitions for one or more places. For example, a location might refer to San Francisco, in which case it would store `San Francisco, California, United States`.

Places can also store references to parent places, so this distinction may seem unhelpful. However, we use it because the salient difference between a location and a place is that a place is a separate model that stores all the information CultureMesh has on that place (e.g. name, population, etc.). On the other hand, a location only stores the IDs of the

places that define it. In practice, this means that places can be thought of as residing in a list of all places CultureMesh knows about, while locations are used to define networks.

Inheritance Structure



The diagram above illustrates the inheritance hierarchy consisting of classes storing location/place information. The tree rooted at *DatabaseLocation* exists because of the potential to cache data locally in a database. This would allow for offline access and better performance when internet connection is poor. However, the database we experimented with required that the near (or current) location be specified using a different class than the from (or origin) location so that their instance fields could have different names and not conflict in the database. This is why *NearLocation* and *FromLocation* exist, as they are otherwise essentially the same. Whenever they can be treated identically, *DatabaseLocation* can be used. DatabaseLocation also stores functionality that is common to both subclasses.

Networks, Languages, Events, and Posts

A *Network* is defined in one of two ways:

- Location-based: The network is defined by a *NearLocation* and a *FromLocation*.
- Language-based: The network is defined by a *NearLocation* and a *Language*.

When the network is initially received from the server as a JSON, it is parsed to create a *DatabaseNetwork*, which represents the above properties by their IDs. Then, that DatabaseNetwork is expanded into a *Network*, which includes full *Place* and/or *Language* objects for the above properties.

While not stored in the Network object, there are also lists of *Event* s and *Post* s associated with each network. These are fetched separately from the server each time they are needed. Instead of separate classes for their ID-only representations coming from the server and the fuller ones used within the app, they are instantiated in stages within the *API* class. First, their JSON representations are parsed to partially instantiate them. Then, missing parts (e.g. full Network objects) are fetched from the server and parsed to fully instantiate the objects.

Both Event and Post are subclasses of *FeedItem*, which requires them to have a public instance field containing a list of comments. This allows them to both be displayed via polymorphism within a feed like *TimelineActivity*. These comments are represented by *PostReply* objects.

Interfaces for Sending Objects

To reduce code redundancy, the *API* class uses a series of `model` methods that can send `PUT` and `POST` requests (separate `model` methods) with any object so long as that object can generate a JSON representation of itself for the request using `getPutJSON` or `getPostJSON`. The presence of these methods is enforced by the interfaces *Postable* and *Putable*, which allows for the `model` methods to be polymorphic.

Other

A *Point* describes a particular spot on the globe in terms of its latitude and longitude. It is really just a holder for the two values.

A *User* object represents any of CultureMesh's users. It only stores parts of their public profiles, so methods that work with private information like passwords or email addresses take those values as parameters.

1.2.3 Connections to CultureMesh's Servers

Networking operations are performed by making calls to methods in the *API* class. Since networking operations suffer from any inherent latency in the user's internet connection, they are performed in a separate thread using Volley. Generically then, these methods generally take the following arguments: `(RequestQueue, args ... , Response.Listener<responseType>)`

- `RequestQueue`: A queue that holds the asynchronous tasks to execute. A queue is generally created once for each activity and then used for all API calls in that activity.
- `args`: All the arguments the method needs to create the network request. This often includes IDs of resources to fetch.
- `Response.Listener<...>`: A listener whose `onResponse` method is called with the result of the operation. This occurs whether or not the operation completed successfully.
- `responseType`: The type of the object that is returned by the operation. This is generally some kind of *NetworkResponse* object.

API Authentication

API Key

The API key must be passed as a parameter with key `key` in the URL of all authenticated API endpoints. The key is stored in *Credentials*, which is not stored in version control or published publicly. The API method *API.getCredentials* method is used to access the key from within the *API* class.

User Credentials

When the user logs in to the app the first time, their email and password are used to authenticate a request for a login token using *API.Get.loginWithCred*. This token is stored in the app's SharedPreferences for future authentication. The user's password is not stored. If the token expires due to inactivity, the user is directed to login again.

All tokens older than *API.TOKEN_REFRESH* milliseconds are refreshed with the next authenticated request (this is handled automatically by *API.Get.loginToken*, which produces the tokens used by all API methods that access secured endpoints). Tokens are refreshed much faster than they expire because the difference between the refresh time and the expiration time is the maximum allowable inactivity period before users have to sign in again, and we want this to be long enough to avoid too much inconvenience.

Conveying Network Responses

This object simplifies error reporting by storing whether or not the operation failed using *NetworkResponse.fail*. It also stores the results of successful operations, which are available through *NetworkResponse.getPayload*. It can store messages describing errors and create ready-to-display error dialogs to communicate those messages to users using *NetworkResponse.showErrorDialog*.

Authentication Failures

In the special case of authentication errors, the *NetworkResponse.setAuthFailed* method can be used to specify that the failure was authentication-related. When the resulting error dialog is displayed and dismissed, the user is automatically redirected to the sign-in screen.

Recommended Usage

- Specify the network operation to be performed in a method in the *API* class. The method should take a `RequestQueue` and a `Response.Listener`.
  - Create the request, such as `JsonObjectRequest`, providing the method of the request (e.g. `GET`, `POST`, etc.), endpoint URL, listener, and error listener.
  - In the listener, specify an `onResponse` method that handles interpreting the response into a *NetworkResponse* and passing that to a call to the `Response.Listener` provided as a parameter to the API method.
  - In the error listener, interpret the error and select an appropriate error message. Create a *NetworkResponse* object to communicate the error. If appropriate, use *NetworkResponse.setAuthFailed*.
  - Example method:

```
static void user(RequestQueue queue, long id,
                 final Response.Listener<NetworkResponse<User>> listener) {
    JsonObjectRequest authReq = new JsonObjectRequest(Request.Method.GET,
            API_URL_BASE + "user/" + id + "?" + getCredentials(),
            null, new Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject res) {
            try {
                //make User object out of user JSON.
                User user = new User(res);
                listener.onResponse(new NetworkResponse<>(false, user));
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            listener.onResponse(new NetworkResponse<User>(true,
                    processNetworkError("API.Get.user", "ErrorListener",␣
↪error)));
        }
    });
    queue.add(authReq);
}
```

  Note that *API.API_URL_BASE* is a constant in the API class that specifies the base of the API URLs, that `API.processNetworkError` returns a reference to a message describing the error, and that *API.getCredentials* returns the API key.

- In any API methods that rely on another API method, call the used method as usual, but do anything that relies on the used method's results in the listener you provide to it. In addition, when passing along *NetworkResponse* errors from the used method, you may need to change the type of response when passing it along. Use the

constructor that takes another response object, as this discards any payload (which is not needed for errors) and preserves the authentication failure status.

- When using an API method in Activities or non-API classes, create a `RequestQueue` for the entire activity and pass it to all calls to API methods. In each call, pass along a listener that describes what to do with the response.

1.2.4 Other

First Activity

When the application starts from scratch (i.e. is not being launched by restoring a previous state), the *ApplicationStart* activity is loaded. This performs initialization for the app (e.g. Crashlytics). Then *StartActivity* loads, which is the parent activity at the root of the app. It doesn't display anything; all it does is redirect the user to either *TimelineActivity*, *OnboardActivity*, or *ExploreBubblesOpenGLActivity* based on whether they have logged in and whether they have a selected network. When tapping the `Back` button, the user eventually ends up at *StartActivity*, which then redirects them. This prevents a user from returning to *OnboardActivity* using the `Back` button.

Managing Formatted Text

In cases where the user can create formatted text using inline markup (i.e. bold, italics, and hyperlinks), *FormatManager* handles the markup.

Handling Redirections

In a few cases, a parent activity needs to launch a child activity while also directing the child to launch a particular grand-child activity. For example, when *SettingsActivity* launches *OnboardActivity*, the user should be sent back to SettingsActivity at the end. If *ApplicationStart* is instead launching *OnboardActivity*, the user should next be sent on to *LoginActivity*. This is handled by *Redirection*.

1.3 Code Reference

The Android code is documented using Javadoc, which can be viewed in three forms:

- Traditional Javadoc (includes `Private` methods)
- javasphinx *Javadoc* (excludes `Private` methods)
- The comments in the code itself

Specific sections of the javasphinx Javadoc are referenced throughout the documentation like so: *API.Get*.

1.3.1 Javadoc Table of Contents

Javadoc

org.codethechange.culturemesh

API

class **API**
This API serves as the interface between the rest of the app and the CultureMesh servers. When another part of the app needs to request information, it calls API methods to obtain it. Similarly, API methods should be used to store, send, and update information. The API then handles requesting it from the CultureMesh servers.

Fields

API_URL_BASE

static final String **API_URL_BASE**
Base of the URL all API endpoints use. For example, the `/token` endpoint has the URL `API_URL_BASE + "/token"`.

CURRENT_USER

static final String **CURRENT_USER**
Identifier for the currently-signed-in user's ID. If no user is signed-in, this key should be removed from the preferences Example: `settings.getLong(API.CURRENT_USER, -1)`.

FEED_ITEM_COUNT_SIZE

static final String **FEED_ITEM_COUNT_SIZE**
The number of items (e.g. *org.codethechange.culturemesh.models.Post*s or *Event*s to fetch with each paginated request

HOSTING

static final String **HOSTING**

LOGIN_TOKEN

static final String **LOGIN_TOKEN**
Settings identifier for the currently cached login token for the user. May be missing or expired. Expiration is tracked using *API.TOKEN_REFRESH*.

NO_MAX_PAGINATION

static final String **NO_MAX_PAGINATION**

SELECTED_NETWORK

static final String **SELECTED_NETWORK**
Identifier for the user's currently selected *Network*. This is used to save the network the user was last viewing so that network can be re-opened when the user navigates back. Example: `settings.getLong(API.SELECTED_NETWORK, -1)`.

SELECTED_USER

static final String **SELECTED_USER**
The SharedPreferences key for communicating to ViewProfileActivity which user we are viewing.

SETTINGS_IDENTIFIER

static final String **SETTINGS_IDENTIFIER**
Identifier for the app's shared preferences. Example: `SharedPreferences settings = getSharedPreferences(API.SETTINGS_IDENTIFIER, MODE_PRIVATE)`

TOKEN_REFRESH

static final int **TOKEN_REFRESH**
Number of milliseconds to use a login token before refreshing it. Note that this is not how long the token is valid, just how often to refresh it. Refresh time must be shorter than the validity time.

See also: *API.LOGIN_TOKEN*

TOKEN_RETRIEVED

static final String **TOKEN_RETRIEVED**
Settings identifier for when the current login token was retrieved. Stored as the number of milliseconds since the epoch.

See also: *API.LOGIN_TOKEN*

USER_EMAIL

static final String **USER_EMAIL**
Identifier for the currently-signed-in user's email. If no user is signed-in, this key should be removed from the preferences Example: `settings.getLong(API.USER_EMAIL, -1)`.

Methods

genBasicAuth

static String **genBasicAuth**(String *email*, String *password*)
Generate from a username/email and password the string to put in the header of a request as the value of the `Authorization` token in order to perform Basic Authentication. For example: `headers.put("Authorization", genBasicAuth(email, password))`. A login token can be used if it is passed as the `email`, in which case the `password` is ignored by the server.

Parameters

- **email** – Email or username of account to login as; can also be a login token
- **password** – Password to login with

Returns Value that should be passed in the header as the value of `Authorization`

genBasicAuth

static String **genBasicAuth**(String *token*)

Generate from a login token the string to put in the header of a request as the value of the `Authorization` token in order to perform Basic Authentication. For example: `headers.put("Authorization", genBasicAuth(token))`.

Parameters

- **token** – Login token to authenticate to server

Returns Value that should be passed in the header as the value of `Authorization`

getCredentials

static String **getCredentials**()

Use this method to append our credentials to our server requests. For now, we are using a static API key. In the future, we are going to want to pass session tokens.

Returns credentials string to be appended to request url as a param.

API.Get

static class **Get**

The protocol for GET requests is as follows... 1. Check if cache has relevant data. If so, return it. 2. Send network request to update data.

Methods

autocompleteLanguage

static void **autocompleteLanguage**(RequestQueue *queue*, String *text*, Response.Listener<*NetworkResponse*<List<*Language*>>> *listener*)

Get potential *Language*s that match a user's query text

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **text** – User's query text to get autocomplete results for
- **listener** – Listener whose `com.android.volley.Response.Listener.onResponse(Object)` is called with the *NetworkResponse* created by the query.

autocompletePlace

static void **autocompletePlace**(RequestQueue *queue*, String *text*, Response.Listener<*NetworkResponse*<List<*Location*>>> *listener*)

Get potential `Location`s that match a user's query text

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **text** – User's query text to get autocomplete results for
- **listener** – Listener whose `com.android.volley.Response.Listener.onResponse(Object)` is called with the `NetworkResponse` created by the query.

instantiatePostReplyUser

static void **instantiatePostReplyUser**(RequestQueue *queue*, *PostReply* *comment*, Response.Listener<*PostReply*> *listener*)

The API will return Post JSON Objects with id's for the user. Often, we will want to get the user information associated with a post, such as the name and profile picture. This method allows us to instantiate this user information for each post.

Parameters

- **queue** – The Volley RequestQueue object that handles all the request queueing.
- **comment** – An already instantiated PostReply object that has a null author field but a defined userId field.
- **listener** – the UI listener that will be called when we complete the task at hand.

instantiatePostUser

static void **instantiatePostUser**(RequestQueue *queue*, org.codethechange.*culturemesh.models.Post* *post*, Response.Listener<org.codethechange.*culturemesh.models.Post*> *listener*)

The API will return Post JSON Objects with id's for the user. Often, we will want to get the user information associated with a post, such as the name and profile picture. This method allows us to instantiate this user information for each post.

Parameters

- **queue** – The Volley RequestQueue object that handles all the request queueing.
- **post** – An already instantiated Post object that has a null author field but a defined userId field.
- **listener** – the UI listener that will be called when we complete the task at hand.

language

static void **language**(RequestQueue *queue*, long *id*, Response.Listener<*NetworkResponse*<*Language*>> *listener*)

Get the `Language` that has the provided ID

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **id** – ID of the *Language* to find. Must be unique, and the same ID must be used throughout.
- **listener** – Listener whose `com.android.volley.Response.Listener.onResponse(Object)` is called with the *NetworkResponse* created by the query.

loginToken

static void **loginToken**(RequestQueue *queue*, SharedPreferences *settings*, Response.Listener<*NetworkResponse*<String>> *listener*)

Generically get a login token. If the token is fresh (less than *API.TOKEN_REFRESH* seconds have passed since the last token was retrieved the current token is simply supplied. Otherwise, an attempt is made to login with the token to get a new one. If this fails, the token has expired, and the user is directed to sign in again by the error dialog. If it succeeds, the new token is stored in place of the old one.

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **listener** – Listener whose onResponse method will be called when task completes

See also: *NetworkResponse.genErrorDialog(Context,int,boolean,NetworkResponse.DialogTapListener)*, *API.LOGIN_TOKEN*, *API.TOKEN_RETRIEVED*

loginWithCred

static void **loginWithCred**(RequestQueue *queue*, String *email*, String *password*, SharedPreferences *settings*, Response.Listener<*NetworkResponse*<*LoginResponse*>> *listener*)

Use a user's login credentials to login to the server. A user's credentials consist of the email address associated with their account and their password for the CultureMesh website. If the credentials are accepted by the server, the resulting LoginResponse will be stored in the *NetworkResponse*, which will not be in a failed state, and passed to the listener. If the credentials are rejected, the *NetworkResponse* will be in a failed state with an error message communicating the occurrence of an authentication failure and instructing the user to sign in again. After dismissing the error dialog, the *LoginActivity* will be launched.

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **email** – Email address that will serve as the username in the attempted login
- **password** – Password to use in the login attempt
- **listener** – Will be called with the *NetworkResponse* when the operation completes

See also: *NetworkResponse.genErrorDialog(Context,int,boolean,NetworkResponse.DialogTapListener)*

loginWithToken

static void **loginWithToken**(RequestQueue *queue*, String *token*, SharedPreferences *settings*, Response.Listener<*NetworkResponse*<*LoginResponse*>> *listener*)

Same as *API.Get.loginWithCred(RequestQueue,String,String,SharedPreferences,Response.Listener)*, but a login token is used in place of the user's credentials.

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **token** – Login token to use to get another token
- **listener** – Will be called with the *NetworkResponse* when the operation completes

netFromFromAndNear

static void **netFromFromAndNear**(RequestQueue *queue*, *FromLocation* *from*, *NearLocation* *near*, Response.Listener<*NetworkResponse*<*Network*>> *listener*)

Get the *Network* that has the provided *FromLocation* and *NearLocation*

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **from** – *FromLocation* of the *Network* to find
- **near** – *NearLocation* of the *Network* to find
- **listener** – Listener whose `com.android.volley.Response.Listener.onResponse(Object)` is called with the *NetworkResponse* created by the query.

netFromLangAndNear

static void **netFromLangAndNear**(RequestQueue *queue*, *Language* *lang*, *NearLocation* *near*, Response.Listener<*NetworkResponse*<*Network*>> *listener*)

Get the *Network* that has the provided *Language* and *NearLocation*

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **lang** – *Language* of the *Network* to find
- **near** – *NearLocation* of the *Network* to find
- **listener** – Listener whose `com.android.volley.Response.Listener.onResponse(Object)` is called with the *NetworkResponse* created by the query.

network

static void **network**(RequestQueue *queue*, long *id*, Response.Listener<*NetworkResponse*<*Network*>> *callback*)

Get the *Network* corresponding to the provided ID

Parameters

- **queue** – Queue to which the asynchronous task to get the *Network* will be added
- **id** – ID of the *Network* to get
- **callback** – Listener whose `com.android.volley.Response.Listener.onResponse(Object)` is called with the *NetworkResponse* created by the query.

networkEvents

static void **networkEvents**(RequestQueue *queue*, long *id*, String *maxId*, Response.Listener<*NetworkResponse*<List<*Event*>>> *listener*)

Get the *Event*s corresponding to a *Network*

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **id** – ID of the *Network* whose *Event*s will be fetched
- **listener** – Listener whose `com.android.volley.Response.Listener.onResponse(Object)` is called with the *NetworkResponse* created by the query.

networkPostCount

static void **networkPostCount**(RequestQueue *queue*, long *id*, Response.Listener<*NetworkResponse*<Long>> *listener*)

Get the number of *org.codethechange.culturemesh.models.Post*s that are currently on a *Network*

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **id** – ID of the *Network* whose *org.codethechange.culturemesh.models.Post* count will be retrieved
- **listener** – Listener whose `Response.Listener.onResponse(Object)` is called with a *NetworkResponse* that stores the result of the network request

networkPosts

static void **networkPosts**(RequestQueue *queue*, long *id*, String *maxId*, Response.Listener<*NetworkResponse*<List<org.codethechange.*culturemesh.models.Post*>>> *listener*)

Get the *org.codethechange.culturemesh.models.Post*s of a *Network*

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **id** – ID of the *Network* whose *org.codethechange.culturemesh.models.Post*s will be returned
- **listener** – Listener whose `com.android.volley.Response.Listener.onResponse(Object)` is called with the *NetworkResponse* created by the query.

networkUserCount

static void **networkUserCount**(RequestQueue *queue*, long *id*, Response.Listener<*NetworkResponse*<Long>> *listener*)

Get the number of *User*s who are currently members of a *Network*

Parameters

- **queue** – Queue to which the asynchronous task will be added

- **id** – ID of the *Network* whose *User* count will be retrieved
- **listener** – Listener whose `Response.Listener.onResponse(Object)` is called with a *NetworkResponse* that stores the result of the network request

networkUsers

static void **networkUsers**(RequestQueue *queue*, long *id*, Response.Listener<*NetworkResponse*<ArrayList<*User*>>> *listener*)

Get all the *User*s who are members of a *Network*

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **id** – ID of the *Network* whose users will be fetched
- **listener** – Listener whose `com.android.volley.Response.Listener.onResponse(Object)` is called with the *NetworkResponse* created by the query.

post

static void **post**(RequestQueue *queue*, long *id*, Response.Listener<*NetworkResponse*<org.codethechange.*culturemesh.models.Post*>> *callback*)

Get a *org.codethechange.culturemesh.models.Post* from it's ID

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **id** – ID of the *org.codethechange.culturemesh.models.Post* to retrieve
- **callback** – Listener whose `com.android.volley.Response.Listener.onResponse(Object)` is called with the *NetworkResponse* created by the query.

postReplies

static void **postReplies**(RequestQueue *queue*, long *id*, Response.Listener<*NetworkResponse*<ArrayList<*PostReply*>>> *listener*)

Fetch the comments of a post.

Parameters

- **queue** – The `RequestQueue` to house the network requests.
- **id** – the id of the post that we want comments for.
- **listener** – the listener that we will call when the request is finished.

topTen

static void **topTen**(RequestQueue *queue*, Response.Listener<*NetworkResponse*<ArrayList<*Network*>>> *listener*)

Fetches the ten *Network*s with the most subscribers.

Parameters

- **queue** – Queue to which the asynchronous task will be added

- **listener** – Will be called with the *NetworkResponse* when the operation completes

user

static void **user**(RequestQueue *queue*, long *id*, Response.Listener<*NetworkResponse*<*User*>> *listener*)

Get a *User* object from it's ID

Parameters

- **id** – ID of user to find

Returns If such a user was found, it will be the payload. Otherwise, the request will be marked as failed.

userEvents

static void **userEvents**(RequestQueue *queue*, long *id*, String *role*, Response.Listener<*NetworkResponse*<ArrayList<org.codethechange.*culturemesh.models.Event*>>> *listener*)

Get the *Event*s a *User* is subscribed to.

Parameters

- **queue** – Queue to which the asynchronous task is added.
- **id** – ID of the *User* whose events are being searched for
- **role** – Either `hosting` or `attending`
- **listener** – Listener whose `onResponse` method is called with the results of the task

userEventsForNetwork

static void **userEventsForNetwork**(RequestQueue *queue*, SharedPreferences *settings*, long *networkId*, Response.Listener<*NetworkResponse*<ArrayList<*Event*>>> *listener*)

Get the *Event*s a *User* is subscribed to for a given *Network*.

Parameters

- **queue** – Queue to which the asynchronous task is added.
- **settings** – SharedPreferences instance storing the token.
- **networkId** – the id of the *Network* of interest.
- **listener** – The response listener to be called when the request completes.

userID

static void **userID**(RequestQueue *queue*, String *email*, Response.Listener<*NetworkResponse*<Long>> *listener*)

Get the ID of a *User* from an email address. Errors are communicated via a failed *NetworkResponse*.

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **email** – Email of user whose ID to look up

- **listener** – Listener whose onResponse method is called when the task has completed

userNetworks

static void **userNetworks**(RequestQueue *queue*, long *id*, Response.Listener<*NetworkResponse*<ArrayList<*Network*>>> *listener*)

Get the networks a user belongs to

Parameters

- **queue** – RequestQueue to which the asynchronous job will be added
- **id** – ID of the user whose networks will be fetched
- **listener** – Listener whose `com.android.volley.Response.Listener.onResponse(Object)` is called with a *NetworkResponse* of an `ArrayList` of *Network*s

userPosts

static void **userPosts**(RequestQueue *queue*, long *id*, Response.Listener<*NetworkResponse*<ArrayList<org.codethechange.*cultureme* *listener*)

Get the *org.codethechange.culturemesh.models.Post*s a *User* has made.

Parameters

- **queue** – The `RequestQueue` that will house the network requests.
- **id** – The id of the *User*.
- **listener** – The listener that the UI will call when the request is finished.

API.Get.LoginResponse

public static class **LoginResponse**

Bundle object to store responses from getting tokens, which yield *User*s, tokens, and emails.

Fields

email

public String **email**

token

public String **token**

user

public *User* **user**

Constructors

LoginResponse

public **LoginResponse**(*User user*, String *token*, String *email*)
Store the provided parameters in the bundle object

Parameters

- **user** – User object described by returned JSON
- **token** – Login token
- **email** – User's email address

API.Post

static class **Post**

Methods

event

static void **event**(RequestQueue *queue*, *Event event*, SharedPreferences *settings*, Response.Listener<*NetworkResponse*<String>> *listener*)
POST to the server a request, via `/event/new`, to create a new *Event*. Success or failure status will be passed via a *NetworkResponse* to the listener.

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **event** – *Event* to create.
- **listener** – Listener whose onResponse method will be called when task completes

joinEvent

static void **joinEvent**(RequestQueue *queue*, long *eventId*, SharedPreferences *settings*, Response.Listener<*NetworkResponse*<String>> *listener*)
Add a user to an existing event. This operation requires authentication, so the user must be logged in.

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **settings** – `SharedPreferences` instance so we can get the token.
- **eventId** – ID of the event to add the user to
- **listener** – Listener whose onResponse method will be called when the operation completes

joinNetwork

static void **joinNetwork**(RequestQueue *queue*, long *networkId*, SharedPreferences *settings*, Response.Listener<*NetworkResponse*<String>> *listener*)

Add the current user to an existing network. This operation requires authentication, so the user must be logged in.

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **networkId** – ID of the network to add the user to
- **listener** – Listener whose onResponse method will be called when the operation completes

leaveEvent

static void **leaveEvent**(RequestQueue *queue*, long *eventId*, SharedPreferences *settings*, Response.Listener<*NetworkResponse*<String>> *listener*)

Removes user from event subscription listing.

Parameters

- **queue** – Queue to which network request will be added.
- **eventId** – id of event to remove user from.
- **settings** – `SharedPreferences` instance that stores token.
- **listener** – Listener whose onResponse will be called when the operation completes.

leaveNetwork

static void **leaveNetwork**(RequestQueue *queue*, long *networkId*, SharedPreferences *settings*, Response.Listener<*NetworkResponse*<String>> *listener*)

Remove the current user from a network. This operation requires authentication, so the user must be logged in. If the user is not in the specified network, no error is thrown.

Parameters

- **queue** – Asynchronous task to which the request will be added
- **networkId** – ID of the network to remove the user from
- **settings** – Reference to the `SharedPreferences` storing the user's login token
- **listener** – Listener whose `onResponse` method will be called when the operation completes

post

static void **post**(RequestQueue *queue*, org.codethechange.*culturemesh.models.Post* *post*, SharedPreferences *settings*, Response.Listener<*NetworkResponse*<String>> *listener*)

POST to the server a request, via `/post/new`, to create a new *`org.codethechange.culturemesh.models.Post`*. Success or failure status will be passed via a *`NetworkResponse`* to the listener.

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **post** – *org.codethechange.culturemesh.models.Post* to create.
- **listener** – Listener whose onResponse method will be called when task completes

reply

static void **reply**(RequestQueue *queue*, *PostReply* *comment*, SharedPreferences *settings*, Response.Listener<*NetworkResponse*<String>> *listener*)

POST to the server a request, via `/post/{postId}/reply`, to create a new *PostReply*. Success or failure status will be passed via a *NetworkResponse* to the listener.

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **comment** – *PostReply* to create.
- **listener** – Listener whose onResponse method will be called when task completes

user

static void **user**(RequestQueue *queue*, *User* *user*, String *email*, String *password*, Response.Listener<*NetworkResponse*<String>> *listener*)

POST to the server a request, via `/user/users`, to create a new user. Note that Success or failure status will be passed via a *NetworkResponse* to the listener.

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **user** – User to create. **Must have password set.**
- **email** – User's email address
- **listener** – Listener whose onResponse method will be called when task completes

API.Put

static class **Put**

Methods

event

static void **event**(RequestQueue *queue*, *Event* *event*, SharedPreferences *settings*, Response.Listener<*NetworkResponse*<String>> *listener*)

PUT to the server a request, via `/event/new`, to update an *Event*. Success or failure status will be passed via a *NetworkResponse* to the listener.

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **event** – Updated version of the *Event* to change
- **listener** – Listener whose onResponse method will be called when task completes

post

static void **post**(RequestQueue *queue*, org.codethechange.*culturemesh.models.Post* *post*, SharedPreferences *settings*, Response.Listener<*NetworkResponse*<String>> *listener*)

PUT to the server, via `/user/users`, a request to make changes a *org.codethechange.culturemesh.models.Post*. Success or failure status will be passed via a *NetworkResponse* to the listener.

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **post** – Updated version of the post to change
- **listener** – Listener whose onResponse method will be called when task completes

reply

static void **reply**(RequestQueue *queue*, *PostReply* *comment*, SharedPreferences *settings*, Response.Listener<*NetworkResponse*<String>> *listener*)

PUT to the server a request, via `/post/{postId}/reply`, to update a *PostReply*. Success or failure status will be passed via a *NetworkResponse* to the listener.

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **comment** – Updated version of the *PostReply* to make changes to
- **listener** – Listener whose onResponse method will be called when task completes

user

static void **user**(RequestQueue *queue*, *User* *user*, String *email*, SharedPreferences *settings*, Response.Listener<*NetworkResponse*<String>> *listener*)

PUT to the server, via `/user/users`, a request to make changes a *User*. Success or failure status will be passed via a *NetworkResponse* to the listener.

Parameters

- **queue** – Queue to which the asynchronous task will be added
- **user** – Updated version of the user to change
- **email** – User's email address
- **listener** – Listener whose onResponse method will be called when task completes

AboutActivity

public class **AboutActivity** extends *DrawerActivity*

Activity for displaying author attributions, copyright notices, and version information on an `About` page

Methods

onCreate

protected void **onCreate** (Bundle *savedInstanceState*)

When the activity is created, it pulls what to display from `R.layout.activity_about`. It does not have a `setSupportActionBar(toolbar)` call because that is handled by *DrawerActivity*. The toolbar MUST have an ID of `action_bar`.

Parameters

- **savedInstanceState** – Passed to superclass onCreate method

openLegal

public void **openLegal** (View *v*)

Open *Acknowledgements* activity to display legally required attributions for the open-source libraries we use

Parameters

- **v** – The `View` of the button clicked on to run this method. Not used.

Acknowledgements

public class **Acknowledgements** extends *DrawerActivity*

A *DrawerActivity* that displays legally required attributions for the open-source code we use.

Methods

onCreate

protected void **onCreate** (Bundle *savedInstanceState*)

Link the activity to its layout specified in `R.layout.activity_acknowledgements`

Parameters

- **savedInstanceState** – {@inheritDoc}

AnimationUtils

public class **AnimationUtils**

This is a utility class to show the loading overlay for activities that require network requests to display their data.

Methods

animateLoadingOverlay

public static void **animateLoadingOverlay** (View *view*, int *toVisibility*, float *toAlpha*, int *duration*)

Shows or hides loading overlay with smooth alpha transition. Sourced from https://stackoverflow.com/questions/18021148/display-a-loading-overlay-on-android-screen

Parameters

- **view** – View to animate
- **toVisibility** – Visibility at the end of animation
- **toAlpha** – Alpha at the end of animation
- **duration** – Animation duration in ms

ApplicationStart

public class **ApplicationStart** extends Application
This serves as a landing page for when the app is started from scratch. It does some initialization.

Methods

onCreate

public void **onCreate**()
Initialize Crashyltics.

ChooseNearLocationActivity

public class **ChooseNearLocationActivity** extends AppCompatActivity implements SearchView.OnQueryTextListener
This screen let's the user choose where they live now. This is used by *FindNetworkActivity* to restrict displayed networks to those with a `near` that matches where the user lives.

Fields

CHOSEN_PLACE

public static final String **CHOSEN_PLACE**
Identifier for the `Intent` whose value is the *Location* the user chose

RESULT_OK

public static final int **RESULT_OK**
Result code to signal via the `Intent` that the user successfully chose a *Location*

Methods

onCreate

protected void **onCreate**(Bundle *savedInstanceState*)
Setup the activity. Also initializes the `com.android.volley.RequestQueue`, the adapter that populates the list of results, and the listener that handles clicks on items in the results list

Parameters

- **savedInstanceState** – Previous state that is passed through to superclass

onQueryTextChange

public boolean **onQueryTextChange** (String *newText*)
Whenever the query text changes, do nothing because sending network requests every time is unnecessary.

Parameters

- **newText** – The updated query text

Returns Always returns `true`

onQueryTextSubmit

public boolean **onQueryTextSubmit** (String *query*)
When the user submits their query, *ChooseNearLocationActivity.search()* is run to populated the results with matching *Location*s

Parameters

- **query** – User's query. Not used.

Returns Always returns `true`

search

public void **search** ()
Get the query present in the `ChooseNearLocationActivity.searchView` and pass it to the server via *API.Get.autocompletePlace(RequestQueue,String,Response.Listener)* to get a list of matching *Location*s. These are used to populate the `ChooseNearLocationActivity.adapter`.

CommentsFrag

public class **CommentsFrag** extends Fragment
Fragment for displaying comments to posts

Fields

settings

SharedPreferences **settings**
The app's shared settings that store user info and preferences

Methods

onAttach

public void **onAttach** (Context *context*)
{@inheritDoc}

Parameters

- **context** – {@inheritDoc}

onCreate

public void **onCreate**(Bundle *savedInstanceState*)
Initialize references to `CommentsFrag.queue` and *`CommentsFrag.settings`*.

Parameters

- **savedInstanceState** –

onCreateView

public View **onCreateView**(LayoutInflater *inflater*, ViewGroup *container*, Bundle *savedInstanceState*)
Populate the activity with UI elements

Parameters

- **inflater** – Inflates the xml `R.layout.fragment_comments` into the displayed UI
- **container** – TODO: What is this?
- **savedInstanceState** – Saved state that can be restored. Not used.

Returns The inflated view produced by `inflater`

onDetach

public void **onDetach**()
{@inheritDoc}

onStop

public void **onStop**()
This ensures that we are canceling all network requests if the user is leaving this activity. We use a RequestFilter that accepts all requests (meaning it cancels all requests)

CreateEventActivity

public class **CreateEventActivity** extends AppCompatActivity
Screen through which users can create an event in their currently selected network

Methods

createEvent

public void **createEvent**(View *v*)
Create an event based on the entered data after validating it

Parameters

- **v** – The button that was clicked to create the event

isValid

public boolean **isValid**()
Check whether the data entered by the user (if any) is valid and complete

Returns true if the entered data is valid and complete, false otherwise

onCreate

protected void **onCreate**(Bundle *savedInstanceState*)
Initialize activity with saved state

Parameters

- **savedInstanceState** – State to use for initialization

showDatePickerDialog

public void **showDatePickerDialog**(View *v*)
Show the calendar dialog to let the user select the event date Intended to be called when the user presses button to set date

Parameters

- **v** – The button that was clicked to show the date picker

showTimePickerDialog

public void **showTimePickerDialog**(View *v*)
Show the clock dialog to let the user select the event start time Intended to be called when user presses button to set time

Parameters

- **v** – The button that was clicked to show the time picker

CreateEventActivity.DatePickerFragment

public static class **DatePickerFragment** extends DialogFragment implements DatePickerDialog.OnDateSetListener
DatePicker static class that handles operations of the time selection fragment

Methods

getDatePicker

public DatePicker **getDatePicker**()
Get the DatePicker

Returns The DatePicker

getDay

public int **getDay**()
Get the selected day

Returns The selected day of the month with the first day represented by 1

getMonth

public int **getMonth**()
Get the selected month

Returns The selected month as an integer with January as 0 and December as 11

getYear

public int **getYear**()
Get the selected year

Returns The selected year (e.g. 2004 returns the integer 2004)

isSet

public boolean **isSet**()
Check whether the user has set a date

Returns true if the user has set a date, false otherwise

onCreateDialog

public Dialog **onCreateDialog**(Bundle *savedInstanceState*)
Called when the fragment is created Sets the initial state of the calendar to the current date and returns the resulting DatePickerDialog to display

Parameters

- **savedInstanceState** – Last saved state of fragment

Returns DatePickerDialog to display to the user

onDateSet

public void **onDateSet**(DatePicker *view*, int *year*, int *month*, int *day*)
When user sets the date, show their choice in the eventDate textView

Parameters

- **view** – The date picker shown via the fragment
- **year** – Year the user chose
- **month** – Month the user chose
- **day** – Day the user chose

CreateEventActivity.TimePickerFragment

public static class **TimePickerFragment** extends DialogFragment implements TimePickerDialog.OnTimeSetListener
TimePicker static class that handles operations of the time selection fragment

Methods

getHour

public int **getHour**()
Return the selected hour

Returns The selected hour

getMinute

public int **getMinute**()
Return the selected minute

Returns The selected minute

getTimePicker

public TimePicker **getTimePicker**()
Return the TimePicker

Returns the TimePicker

isSet

public boolean **isSet**()
Check whether the user has set a time yet

Returns true if the user has set the time, false otherwise

onCreateDialog

public Dialog **onCreateDialog**(Bundle *savedInstanceState*)
Called when the fragment is created Sets the initial state of the clock to the current time and returns the resulting TimePickerDialog to display

Parameters

- **savedInstanceState** – Last saved state of fragment

Returns TimePickerDialog to display

onTimeSet

public void **onTimeSet** (TimePicker *view*, int *inHour*, int *inMin*)
When user sets the time, show their choice in the eventTime textView

Parameters

- **view** – The time picker shown via the fragment
- **inHour** – Hour the user set
- **inMin** – Minute the user set

CreatePostActivity

public class **CreatePostActivity** extends AppCompatActivity implements *FormatManager.IconUpdateListener*
Creates screen the user can use to create a new `Post`

Fields

content

ListenableEditText **content**
Field the user uses to type the body of their `Post`

formatManager

FormatManager **formatManager**
Handles markup of the body text

menuItems

SparseArray<MenuItem> **menuItems**
All the items in the formatting menu

networkLabel

TextView **networkLabel**
Displays the `Network` the user's `Post` will be added to

progressBar

ProgressBar **progressBar**
Displays progress as the post is being sent over the network

Methods

onCreate

protected void **onCreate**(Bundle *savedInstanceState*)

Create the screen from `R.layout.activity_create_post`, fill *CreatePostActivity.networkLabel* with a description of the *Network* from *API.Get.network(RequestQueue,long,Response.Listener)*, setup *CreatePostActivity.formatManager*, and link a listener to the submission button that sends the *Post* using *API.Post.post(RequestQueue,Post,SharedPreferences,Response.Listener)*

Parameters

- **savedInstanceState** – {@inheritDoc}

onCreateOptionsMenu

public boolean **onCreateOptionsMenu**(Menu *menu*)

Populate the options menu with controls to make text bold, italic, or a link

Parameters

- **menu** – Menu to populate with options

Returns Always returns `true`

onOptionsItemSelected

public boolean **onOptionsItemSelected**(MenuItem *item*)

This function handles what happens when our format toggle buttons are clicked. We want to update the content formatting when this happens as well with Spannables. Check out https://stackoverflow.com/questions/10828182/spannablestringbuilder-to-create-string-with-multiple-fonts-text-sizes-etc-examp for more info.

Parameters

- **item** – the MenuItem that was tapped.

onStop

public void **onStop**()

This ensures that we are canceling all network requests if the user is leaving this activity. We use a RequestFilter that accepts all requests (meaning it cancels all requests)

updateIconToggles

public void **updateIconToggles**(SparseBooleanArray *formTogState*, SparseArray<int[]> *toggleIcons*)

This fancy function uses our SparseArray's to concisely iterate over our toggle icons and update their colors - white if untoggled, black if toggled.

Credentials

public class **Credentials**
Just a file out of source control that you can use to hide our API Key. In reality, we won't be using a static API Key (in theory, someone could reverse engineer this somehow), but we'll be using it for beta testing. Thus, make sure you don't add it to source control (Git) and push it onto the master branch.

Fields

APIKey

public static String **APIKey**

DrawerActivity

public class **DrawerActivity** extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener
Superclass for all Activities that have a navigation drawer

Fields

currentUser

protected long **currentUser**
ID of the current *User*

frameLayout

protected FrameLayout **frameLayout**
Parent for the drawer activity

fullLayout

protected DrawerLayout **fullLayout**
The inflated user interface for the activity with the drawer

mDrawerLayout

protected DrawerLayout **mDrawerLayout**
User interface for the drawer itself

mDrawerToggle

protected ActionBarDrawerToggle **mDrawerToggle**
Toggles whether the drawer is visible

mToolbar

protected Toolbar **mToolbar**

navView

NavigationView **navView**
The navigation view

queue

RequestQueue **queue**
Queue for asynchronous tasks

subscribedNetworkIds

protected Set<Long> **subscribedNetworkIds**
IDs of the *Network*s the current *User* is subscribed to

subscribedNetworks

protected SparseArray<*Network*> **subscribedNetworks**
The *User*'s current *Network*s

thisActivity

Activity **thisActivity**
Reference to the current activity

Methods

fetchNetworks

public void **fetchNetworks**()
This fetches the users subscribed networks and displays them in the navigation drawer.

onConfigurationChanged

public void **onConfigurationChanged**(Configuration *newConfig*)
{@inheritDoc} Also updates the configuration of the drawer toggle by calling `DrawerActivity.mDrawerToggle.onConfigurationChanged(Configuration)` with the provided parameter.

Parameters

- **newConfig** – {@inheritDoc}

onNavigationItemSelected

public boolean **onNavigationItemSelected**(MenuItem *item*)

Handle navigation items the user selects. If they select a *Network*, they are sent to *TimelineActivity* after the selected network is set as their chosen one. Otherwise, the appropriate activity is launched based on the option they select.

Parameters

- **item** – Item the user selected.

Returns Always returns `true`

onPostCreate

protected void **onPostCreate**(Bundle *savedInstanceState*)

{@inheritDoc} Also syncs the state of *DrawerActivity.mDrawerToggle*

Parameters

- **savedInstanceState** – {@inheritDoc}

onStop

public void **onStop**()

This ensures that we are canceling all network requests if the user is leaving this activity. We use a RequestFilter that accepts all requests (meaning it cancels all requests)

setContentView

public void **setContentView**(int *layoutResID*)

Create the drawer from `R.layout.activity_drawer`, which has parent with ID `R.id.drawer_frame`. Populate the drawer with data from the current *User* and their *Network*s.

Parameters

- **layoutResID** – ID for the layout file to inflate

DrawerActivity.WaitForSubscribedList

public interface **WaitForSubscribedList**

Interface for classes that have actions that must wait until after the list of subscribed *Network*s has been populated. Subclasses can use this list instead of making another API call.

Methods

onSubscribeListFinish

void **onSubscribeListFinish**()

ExploreBubblesOpenGLActivity

public class **`ExploreBubblesOpenGLActivity`** extends *DrawerActivity*
Display moving bubbles which show suggested networks for the user to join

Fields

hintText

TextView **`hintText`**
The even smaller view that will explain to the user which hint to do.

languages

HashMap<String, *Language*> **`languages`**
A mapping from the title of the bubble (Location#getShortName()) to the language object.

locations

HashMap<String, *Location*> **`locations`**
A mapping from the title of the bubble (Location#getShortName()) to the location object.

picker

BubblePicker **`picker`**
The custom view that displays locations/languages as bubbles.

selectedNearLocation

NearLocation **`selectedNearLocation`**

subTitle

TextView **`subTitle`**
The smaller text view responsible for clarifying the title text.

title

TextView **`title`**
The text view responsible for guiding the user with the interface

Methods

onCreate

protected void **onCreate** (Bundle *savedInstanceState*)

onPause

protected void **onPause** ()

onResume

protected void **onResume** ()

visitNetwork

void **visitNetwork** (long *id*)
Navigates to TimelineActivity to view the selected network.

Parameters

- **id** – id of network.

FindNetworkActivity

public class **FindNetworkActivity** extends *DrawerActivity*

Fields

REQUEST_NEW_NEAR_LOCATION

public final int **REQUEST_NEW_NEAR_LOCATION**

near

static *Location* **near**
The user's chosen `Location` they are near

queue

static RequestQueue **queue**
Queue to hold asynchronous tasks

Methods

onActivityResult

protected void **onActivityResult** (int *requestCode*, int *resultCode*, Intent *data*)

When the user has chosen a near location using *ChooseNearLocationActivity*, this method is called by the Intent that launched the near location chooser with the result of the user's selection. If they did indeed choose a location, that location is saved and the button text is updated to reflect the location's name.

Parameters

- **requestCode** – Status code that indicates a location was chosen if it equals *ChooseNearLocationActivity.RESULT_OK*
- **resultCode** – {@inheritDoc}
- **data** – Passed to superclass, but the value associated with *ChooseNearLocationActivity.CHOSEN_PLACE*, which should be the location the user chose, is extracted if requestCode indicates they made a choice

onCreate

protected void **onCreate** (Bundle *savedInstanceState*)

Setup the activity based on content specified in R.layout.activity_find_network. See code comments for details on implementation.

Parameters

- **savedInstanceState** – Previous state that is passed to superclass.

onCreateOptionsMenu

public boolean **onCreateOptionsMenu** (Menu *menu*)

Inflate the menu; this adds items to the action bar if it is present.

Parameters

- **menu** – Menu to create

Returns Always returns true

onOptionsItemSelected

public boolean **onOptionsItemSelected** (MenuItem *item*)

Handles clicks to the action bar.

Parameters

- **item** – {@inheritDoc}

Returns true if the item ID is that of R.id.action_settings. Otherwise, superclass onOptionsItemSelected is called and the resulting value is returned.

onResume

protected void **onResume**()
{@inheritDoc}

onStop

public void **onStop**()
This ensures that we are canceling all network requests if the user is leaving this activity. We use a RequestFilter that accepts all requests (meaning it cancels all requests)

FindNetworkActivity.FindLanguageFragment

public static class **FindLanguageFragment** extends Fragment implements SearchView.OnQueryTextListener
The fragment for finding language networks.

Constructors

FindLanguageFragment

public **FindLanguageFragment**()
Empty constructor that does nothing.

Methods

newInstance

public static *FindLanguageFragment* **newInstance**(int *sectionNumber*)
Returns a new instance of this fragment for the given section number.

onCreateView

public View **onCreateView**(LayoutInflater *inflater*, ViewGroup *container*, Bundle *savedInstanceState*)
Create the displayed fragment.

Parameters

- **inflater** – Creates the user interface from `R.layout.fragment_find_language`
- **container** – Parent container to attach inflated `View` to
- **savedInstanceState** – Previous state that is not used.

Returns The inflated view to display.

onQueryTextChange

public boolean **onQueryTextChange** (String *newText*)
When the query text changes, do nothing to avoid expensive API calls.

Parameters

- **newText** – The updated query text.

Returns Always returns `true`.

onQueryTextSubmit

public boolean **onQueryTextSubmit** (String *query*)
When the user submits a query, call *FindLanguageFragment.search()*

Parameters

- **query** – Query text that is discarded.

Returns Always returns `true`

search

public void **search** ()
Use *API.Get.autocompleteLanguage(RequestQueue,String,Response.Listener)* to get autocomplete results for the user's query. Pass those results to `FindLanguageFragment.adapter`, which will then populate `FindLanguageFragment.searchList`

FindNetworkActivity.FindLocationFragment

public static class **FindLocationFragment** extends Fragment implements SearchView.OnQueryTextListener
The fragment for finding the from location.

Constructors

FindLocationFragment

public **FindLocationFragment** ()
Empty constructor that does nothing.

Methods

newInstance

public static *FindLocationFragment* **newInstance** (int *sectionNumber*)
Returns a new instance of this fragment for the given section number.

onCreateView

public View **onCreateView**(LayoutInflater *inflater*, ViewGroup *container*, Bundle *savedInstanceState*)
Create the displayed fragment.

Parameters

- **inflater** – Creates the user interface from `R.layout.fragment_find_location`
- **container** – Parent container to attach inflated `View` to
- **savedInstanceState** – Previous state that is not used.

Returns The inflated view to display.

onQueryTextChange

public boolean **onQueryTextChange**(String *newText*)
When the query text changes, do nothing to avoid expensive API calls.

Parameters

- **newText** – The updated query text.

Returns Always returns `true`.

onQueryTextSubmit

public boolean **onQueryTextSubmit**(String *query*)
When the user submits a query, call *`FindLocationFragment.search()`*

Parameters

- **query** – Query text that is discarded.

Returns Always returns `true`

search

public void **search**()
Use *`API.Get.autocompletePlace(RequestQueue,String,Response.Listener)`* to get autocomplete results for the user's query. Pass those results to `FindLocationFragment.adapter`, which will then populate `FindLocationFragment.searchList`

FindNetworkActivity.SectionsPagerAdapter

public class **SectionsPagerAdapter** extends FragmentPagerAdapter
A `FragmentPagerAdapter` that returns a fragment corresponding to one of the two available tabs: `From`, for location-based networks, and `Speaks`, for language-based networks.

Constructors

SectionsPagerAdapter

SectionsPagerAdapter(FragmentManager *fm*)
{@inheritDoc}

Parameters

- **fm** – {@inheritDoc}

Methods

getCount

public int **getCount**()
Always returns 2 because there are 2 tabs

Returns Always 2

getItem

public Fragment **getItem**(int *position*)
Get the appropriate fragment depending on which tab is selected

Parameters

- **position** – Either 0, for near or 1, for speaks

Returns *FindLocationFragment* for position=1, *FindLanguageFragment* otherwise.

getPageTitle

public CharSequence **getPageTitle**(int *position*)
Get the titles for each tab

Parameters

- **position** – Position of tab to get name of (0 or 1)

Returns Reference to name of tab

FormatManager

public class **FormatManager** implements *ListenableEditText.onSelectionChangedListener*
Created by Drew Gregory on 3/26/18. This class provides a little decomposition from CreatePost/SpeciticPostActivity in that it handles all the formatting involved in writing posts/post replies. The supported formatting is: - bold - italic - links This formatting is embedded in the SpannableStrings that EditTexts can produce and maintain. This manager will also handle the tedious tasks of updating the toggle icons and maintaining state. When the user is done formatting and wants to publish their post/post reply, call the toString(), which will convert the spannable to a string with the proper tags as specified by Ian Nottage: **Bold text** *Italic text* Link text

Fields

START

final int **START**

toggleIcons

SparseArray<int[]> **toggleIcons**

Constructors

FormatManager

FormatManager(*ListenableEditText content*, *IconUpdateListener listener*, int *boldIcon*, int *italicIcon*, int *linkIcon*)

Methods

abbreviateNumber

public static String **abbreviateNumber**(long *number*)

In the interest of screen space and accessibility, we will format the number to have a magnitude suffix instead of the exact number.

Parameters

- **number** – exact number, in floating point if necessary.

Returns Formatted String representing number magnitude (e.x. 100K)

fromHtml

public static Spanned **fromHtml**(String *html*)

Different Android versions use different fromHtml method signatures. Sourced from https://stackoverflow.com/questions/37904739/html-fromhtml-deprecated-in-android-n

Parameters

- **html** –

onSelectionChanged

public void **onSelectionChanged**(int *selStart*, int *selEnd*)

parseText

public static Spanned **parseText**(String *formattedText*, String *colorString*)
This function converts the CultureMesh tags into a spannable string for textview.

Parameters

- **formattedText** – should only have `<b></b>, <link></link>, <i></i> or [b][/b][link][/link][i][/i]`
- **colorString** – the link color in RGB. Some text has different color backgrounds.

Returns Spannable to be passed to TextView.

setBold

void **setBold**()
This method will set the currently selected text to bold.

setItalic

void **setItalic**()
This method will set the currently selected text to italic

setLink

void **setLink**()
This method will set the currently selected text to a link.

toString

public String **toString**()
Gets the EditText content in the desired tag format. See comment above.

FormatManager.IconUpdateListener

public interface **IconUpdateListener**

Methods

updateIconToggles

void **updateIconToggles**(SparseBooleanArray *formTogState*, SparseArray<int[]> *toggleIcons*)
This method will require the parent activity to update the toggle button icons.

Parameters

- **formTogState** – a SparseBooleanArray (HashMap) with int as key and boolean as value key: int id of toggleButton View we are using. value: true if toggled, false if not toggled.

- **toggleIcons** – a SparseArray (HashMap) with int as key and int[] as value. key: int id of toggleButton View we are using. value: int[0] being untoggled icon, int[1] being toggled icon.

HelpActivity

public class **HelpActivity** extends *OnboardActivity*
Show user onboarding screens again as help

Methods

getFinishButtonTitle

public String **getFinishButtonTitle**()

onFinishButtonPressed

public void **onFinishButtonPressed**()
When finish button pressed return user to previous page

ListNetworksFragment

public class **ListNetworksFragment** extends Fragment implements *NetworkSummaryAdapter.OnNetworkTapListener*
Fragment for displaying lists of clickable networks

Fields

SELECTED_USER

static final String **SELECTED_USER**
Key stored in the fragment's arguments and whose value is the ID of the user whose networks are to be displayed.

emptyText

TextView **emptyText**
Displays `R.string.no_networks` if there are no networks to display

queue

RequestQueue **queue**
Queue for asynchronous tasks

root

View **root**
Inflated user interface created by `ListNetworksFragment.onCreate(Bundle)`

rv

RecyclerView **rv**
Scrollable list of networks

Methods

newInstance

public static *ListNetworksFragment* **newInstance** (long *selUser*)
Returns a new instance of this fragment for the given section number.

onCreateView

public View **onCreateView** (LayoutInflater *inflater*, ViewGroup *container*, Bundle *savedInstanceState*)
Setup the user interface to display the list of networks and populate that list with the result of calling *API.Get.userNetworks(RequestQueue,long,Response.Listener)*.

Parameters

- **inflater** – Inflates the user interface specified in `R.layout.rv_container`
- **container** – Parent of the generated hierarchy of user interface elements
- **savedInstanceState** – Saved state to restore

Returns Inflated user interface

onItemClick

public void **onItemClick** (View *v*, *Network network*)
This is the onClick() passed to NetworkSummaryAdapter. Thus, this is executed when the user taps on of the network card views. We want to view the tapped network in TimelineActivity.

Parameters

- **v** – the CardView.
- **network** – The Network

onStop

public void **onStop** ()
This ensures that we are canceling all network requests if the user is leaving this activity. We use a RequestFilter that accepts all requests (meaning it cancels all requests)

ListUserEventsFragment

public class **ListUserEventsFragment** extends Fragment implements *RVAdapter.OnItemClickListener*
This fragment lists the the events a user is subscribed to. It is used in ViewProfileActivity.

Fields

emptyText

TextView **emptyText**
Text field that displays `R.string.no_events` if there are no events to display

queue

RequestQueue **queue**
Queue for asynchronous tasks

rv

RecyclerView **rv**
Scrollable list of events.

Methods

newInstance

public static *ListUserEventsFragment* **newInstance** (long *selUser*)
Returns a new instance of this fragment for the given section number.

onCreateView

public View **onCreateView** (LayoutInflater *inflater*, ViewGroup *container*, Bundle *savedInstanceState*)
Setup the user interface to display the list of events and populate that list with the result of calling *API.Get.userEvents(RequestQueue,long,String,Response.Listener)*.

Parameters

- **inflater** – Inflates the user interface specified in `R.layout.rv_container`
- **container** – Parent of the generated hierarchy of user interface elements
- **savedInstanceState** – Saved state to restore

Returns Inflated user interface

onItemClick

public void **onItemClick** (*FeedItem item*)
When an item is clicked, if it is a *Post*, the user is sent to a screen to view the post in more detail, including comments. If the item is an *Event*, no action is taken.

Parameters

- **item** – The item that was clicked

onStop

public void **onStop**()
This ensures that we are canceling all network requests if the user is leaving this activity. We use a RequestFilter that accepts all requests (meaning it cancels all requests)

ListUserPostsFragment

public class **ListUserPostsFragment** extends Fragment implements *RVAdapter.OnItemClickListener*
Creates screen that displays the *Post*s a *org.codethechange.culturemesh.models.User* has made.

Fields

emptyText

TextView **emptyText**
Displays `R.string.no_posts` if there are no *Post*s to display

queue

RequestQueue **queue**
Queue for asynchronous tasks

root

View **root**
The inflated user interface

rv

RecyclerView **rv**
Scrollable list of *Post*s

Methods

newInstance

public static *ListUserPostsFragment* **newInstance**(long *selUser*)
Returns a new instance of this fragment for the given section number.

onCreateView

public View **onCreateView**(LayoutInflater *inflater*, ViewGroup *container*, Bundle *savedInstanceState*)
Create the user interface. Also populate the list of *Post*s with the result from *API.Get.userPosts(RequestQueue,long,Response.Listener)*

Parameters

- **inflater** – Inflates the user interface from `R.layout.rv_container` with the provided `container` as the parent.
- **container** – Parent used by `inflater`
- **savedInstanceState** – Not used

Returns The inflated user interface

onItemClick

public void **onItemClick**(*FeedItem item*)

When the user clicks on an item, redirect them to *SpecificPostActivity* where more details, including comments, are displayed.

Parameters

- **item** – The clicked item.

onStop

public void **onStop**()

This ensures that we are canceling all network requests if the user is leaving this activity. We use a RequestFilter that accepts all requests (meaning it cancels all requests)

Listable

public interface **Listable**

Interface for objects that need to be listed in the user interface.

Fields

MAX_CHARS

int **MAX_CHARS**

ellipses

String **ellipses**

Methods

getListableName

String **getListableName**()

Get a label (maximum of *Listable.MAX_CHARS* characters long) to display as an identifier for the object.

Returns Displayable name for the object, which must be less than or equal to *Listable.MAX_CHARS* characters long

ListenableEditText

public class **ListenableEditText** extends EditText

This is a custom EditText that allows us to listen for changes in cursor position. *CreatePostActivity* uses this view so that the format toggle buttons can update their settings when a new near_region in the edit text is selected.

Fields

mListener

onSelectionChangedListener **mListener**

Constructors

ListenableEditText

public **ListenableEditText**(Context *context*)

{@inheritDoc}

Parameters

- **context** – {@inheritDoc}

ListenableEditText

public **ListenableEditText**(Context *context*, AttributeSet *attrs*)

{@inheritDoc}

Parameters

- **context** – {@inheritDoc}
- **attrs** – {@inheritDoc}

ListenableEditText

public **ListenableEditText**(Context *context*, AttributeSet *attrs*, int *defStyleAttr*)

{@inheritDoc}

Parameters

- **context** – {@inheritDoc}
- **attrs** – {@inheritDoc}
- **defStyleAttr** – {@inheritDoc}

Methods

onSelectionChanged

protected void **onSelectionChanged**(int *selStart*, int *selEnd*)
When the selection changes, if it is due to the user typing a character, `ListenableEditText.mListener.onSelectionChanged(int,int)` is called with the provided parameters. Otherwise, the superclass method `EditText.onSelectionChanged(int,int)` is called with the parameters.

Parameters

- **selStart** – TODO: What is this?
- **selEnd** – TODO: What is this?

setOnSelectionChangedListener

public void **setOnSelectionChangedListener**(*onSelectionChangedListener* *listener*)
Set the listener to the provided parameter

Parameters

- **listener** – Listener to use when text selection changes

ListenableEditText.onSelectionChangedListener

public interface **onSelectionChangedListener**
Interface that all listeners for *`ListenableEditText.mListener`* must satisfy.

Methods

onSelectionChanged

void **onSelectionChanged**(int *selStart*, int *selEnd*)

LoginActivity

public class **LoginActivity** extends *RedirectableAppCompatActivity*
Login screen that lets a user either sign in with email and password or create a new account

Fields

confirmPassword

EditText **confirmPassword**
Reference to the text field for password confirmation

firstNameText

EditText **firstNameText**
Reference to the text field for the user's first name

lastNameText

EditText **lastNameText**
Reference to the text field for the user's last name

needAccountText

TextView **needAccountText**
Text field the user can click to toggle between creating an account and signing in

passwordText

EditText **passwordText**
Reference to the text field for the user's password

usernameText

EditText **usernameText**
Reference to the text field for the user's username

Methods

isLoggedIn

public static boolean **isLoggedIn** (SharedPreferences *settings*)
Check whether any user is currently signed in

Parameters

- **settings** – The app's shared settings, which store user preferences

Returns `true` if a user is signed in, `false` otherwise

onCreate

protected void **onCreate** (Bundle *savedInstanceState*)
Create the user interface from `R.layout.activity_login`. Also setup buttons to perform the associated actions, including log-ins with *API.Get.loginWithCred(RequestQueue,String,String,SharedPreferences,Response.Listener)* and account creation with *API.Post.user(RequestQueue,User,String,String,Response.Listener)*. Also sets up the animations to convert between signing in and creating an account.

Parameters

- **savedInstanceState** – {@inheritDoc}

setLoggedIn

public static void **setLoggedIn**(SharedPreferences *settings*, long *userID*, String *email*)

Largely for testing, this public method can be used to set which user is currently logged in This is useful for PickOnboardingStatusActivity because different login states correspond to different users. No logged-in user is signalled by a missing SharedPreferences entry.

Parameters

- **settings** – The SharedPreferences storing user login state
- **userID** – ID of the user to make logged-in

setLoggedOut

public static void **setLoggedOut**(SharedPreferences *settings*)

Logout the currently logged-out user. If no user is logged in, nothing happens

Parameters

- **settings** – The app's shared settings, which store user preferences

NetworkResponse

public class **NetworkResponse**<E>

Class to store responses after attempting networking tasks

Constructors

NetworkResponse

public **NetworkResponse**(*NetworkResponse*<?> *toConvert*)

Create a new NetworkResponse of the type designated in <> from another NetworkResponse of any other type. Any payload in the source object will not be transferred to the created one. All other fields are copied.

Parameters

- **toConvert** – Source to create new object from. All properties except payload will be copied.

NetworkResponse

public **NetworkResponse**(boolean *inFail*)

Constructor that creates a generic message based on "inFail"

Parameters

- **inFail** – Failure state provided by user (true if failed)

NetworkResponse

public **NetworkResponse**(boolean *inFail*, int *inMessageID*)
Constructor that sets message and failures state based on arguments

Parameters

- **inFail** – Failure state provided by user (true if failed)
- **inMessageID** – ID for string resource containing message

NetworkResponse

public **NetworkResponse**(E *inPayload*)
Constructor that stores a payload and sets the failure state to false

Parameters

- **inPayload** – Payload returned by networking request

NetworkResponse

public **NetworkResponse**(boolean *inFail*, E *inPayload*)
Constructor that both stores a payload and sets the failure state from parameters

Parameters

- **inFail** – Whether or not the network operation failed
- **inPayload** – Payload returned by networking request

NetworkResponse

public **NetworkResponse**(boolean *inFail*, E *inPayload*, int *messageID*)
Constructor that both stores a payload and sets the failure state from parameters

Parameters

- **inFail** – Whether or not the network operation failed
- **inPayload** – Payload returned by networking request

Methods

fail

public boolean **fail**()
Check whether the network request failed

Returns true if the request failed, false if it succeeded

genErrorDialog

public static AlertDialog **genErrorDialog**(Context *context*, int *messageID*)
Get an error dialog that can be displayed to the user

Parameters

- **context** – Context upon which to display error dialog (Should be `someClass.this`)
- **messageID** – String resource ID of message to display

Returns `AlertDialog` with specified alert message.

genErrorDialog

public static AlertDialog **genErrorDialog**(Context *context*, int *messageID*, *DialogTapListener listener*)
Get an error dialog that can be displayed to the user

Parameters

- **context** – Context upon which to display error dialog (Should be `someClass.this`)
- **messageID** – String resource ID of message to display
- **listener** – A *`DialogTapListener`* for when the user dismisses the dialog.

Returns `AlertDialog` with specified alert message.

genErrorDialog

public static AlertDialog **genErrorDialog**(Context *context*, int *messageID*, boolean *authFail*, *DialogTapListener mListener*)
Get an error dialog that can be displayed to the user

Parameters

- **context** – Context upon which to display error dialog (Should be `someClass.this`)
- **messageID** – String resource ID of message to display
- **authFail** – Whether or not the user should be directed to *`LoginActivity`* upon dismissing the dialog
- **mListener** – A *`DialogTapListener`* for when the user dismisses the dialog.

Returns `AlertDialog` with specified alert message and which directs the user to *`LoginActivity`* upon dismissal if `authFail` is true.

genSuccessDialog

public static AlertDialog **genSuccessDialog**(Context *context*, int *messageID*)
Get a confirmation dialog that can be displayed to the user to reflect a successful operation

Parameters

- **context** – Context upon which to display dialog (Should be `someClass.this`)
- **messageID** – String resource ID of message to display

Returns `AlertDialog` with specified alert message

getAuthFailed

public static *NetworkResponse*<*API.Get.LoginResponse*> **getAuthFailed**(int *messageID*)

Get a NetworkResponse object with *NetworkResponse.isAuthFailed* is `true`. This means that when the user dismisses the error dialog generated by *NetworkResponse.getErrorDialog(Context, DialogTapListener)* or *NetworkResponse.showErrorDialog(Context)*, *LoginActivity* will be launched.

Parameters

- **messageID** – String reference to the message describing the error. Will be shown to user

Returns NetworkResponse object to describe an authentication failure.

getErrorDialog

public AlertDialog **getErrorDialog**(Context *context*, *DialogTapListener listener*)

Get an error dialog that can be displayed to show message from messageID to user

Parameters

- **context** – Context upon which to display error dialog (Should be `someClass.this`)
- **listener** – A *DialogTapListener* to be called when they dismiss the dialog.

Returns Dialog that can be shown

getMessageID

public int **getMessageID**()

Get the resource ID of the message to display to the user

Returns Resource ID of message

getPayload

public E **getPayload**()

Get the payload returned by the network operation

Returns Payload returned by network operation

isAuthFailed

public boolean **isAuthFailed**()

Get whether the current object represents a failed authentication

Returns `true` if object represents an authentication failure, `false` otherwise

setAuthFailed

public void **setAuthFailed**(boolean *isAuthFailed*)

Set whether the current object represents a failed authentication

Parameters

- **isAuthFailed** – `true` if object represents an authentication failure, `false` otherwise

showErrorDialog

public void **showErrorDialog**(Context *context*, *DialogTapListener listener*)
Show an error dialog that can be displayed to show message from messageID to user

Parameters

- **context** – Context upon which to display error dialog
- **listener** – A *DialogTapListener* object which allows you control behavior after they dismiss the dialog.

showErrorDialog

public void **showErrorDialog**(Context *context*)
Show an error dialog that can be displayed to show message from messageID to user

Parameters

- **context** – Context upon which to display error dialog

toString

public String **toString**()
Get a String representation of the object that conveys the current state of all instance fields

Returns String representation of the form `NetworkResponse<?>[field1=value1, ...]`

NetworkResponse.DialogTapListener

public interface **DialogTapListener**

Methods

onDismiss

public void **onDismiss**()

NetworkSummaryAdapter

public class **NetworkSummaryAdapter** extends RecyclerView.Adapter<*NetworkSummaryAdapter.PostViewHolder*>
This functions as the recyclerview adapter for the listview in ViewProfileActivity, where the user can view other users' subscribed networks.

Constructors

NetworkSummaryAdapter

NetworkSummaryAdapter(ArrayList<*Network*> *networks*, HashMap<String, Integer> *postCounts*, HashMap<String, Integer> *userCounts*, *OnNetworkTapListener listener*)

Initialize instance fields with parameters

Parameters

- **networks** – List of `Network`s to display
- **postCounts** – Mapping from the ID of each `Network` to the number of `org.codethechange.culturemesh.models.Post`s it contains
- **userCounts** – Mapping from the ID of each `Network` to the number of `org.codethechange.culturemesh.models.User`s it contains
- **listener** – Listener to handle clicks on list items

Methods

getItemCount

public int **getItemCount**()

Get the number of `Network`s that are stored in the list

Returns Number of items in the list

getNetworks

public ArrayList<*Network*> **getNetworks**()

Get the list of `Network`s

Returns List of `Network`s being shown in the list

getPostCounts

public HashMap<String, Integer> **getPostCounts**()

Get the mappings between `Network.id` (as a `String`) and the number of `org.codethechange.culturemesh.models.Post`s in that network.

Returns Mappings that relate `Network` ID to the number of `org.codethechange.culturemesh.models.Post`s in the network

getUserCounts

public HashMap<String, Integer> **getUserCounts**()

Get the mappings between `Network.id` (as a `String`) and the number of `org.codethechange.culturemesh.models.User`s in that network.

Returns Mappings that relate `Network` ID to the number of `org.codethechange.culturemesh.models.User`s in the network

onBindViewHolder

public void **onBindViewHolder**(*PostViewHolder holder*, int *position*)

Fill the fields of `holder` with the information stored in the *`Network`* at index `position` in `NetworkSummaryAdapter.networks`

Parameters

- **holder** – ViewHolder whose fields to fill in
- **position** – Index of *`Network`* in `NetworkSummaryAdapter.networks` whose information will be used to fill in the fields of `holder`

onCreateViewHolder

public *PostViewHolder* **onCreateViewHolder**(ViewGroup *parent*, int *viewType*)

Create a new *`NetworkSummaryAdapter.PostViewHolder`* from the `View` created by inflating `R.layout.network_summary`

Parameters

- **parent** – Parent for created `View` used to create the new *`NetworkSummaryAdapter.PostViewHolder`*
- **viewType** – Not used

Returns ViewHolder that has been created using an inflated `View`

NetworkSummaryAdapter.OnNetworkTapListener

interface **OnNetworkTapListener**

Interface for all listeners for clicks on list items

Methods

onItemClick

void **onItemClick**(View *v*, *Network network*)

NetworkSummaryAdapter.PostViewHolder

class **PostViewHolder** extends RecyclerView.ViewHolder

This ViewHolder is for network_summary, a CardView for networks.

Fields

fromLocation

TextView **fromLocation**

nearLocation

TextView **nearLocation**

postCount

TextView **postCount**

subscribedUserCount

TextView **subscribedUserCount**

Constructors

PostViewHolder

PostViewHolder(View *itemView*)

OnboardActivity

public class **OnboardActivity** extends AhoyOnboarderActivity

Introduce user to the app through a series of informational screens that end with a button that redirects the user to a login page

Methods

getFinishButtonTitle

public String **getFinishButtonTitle**()

onActivityResult

protected void **onActivityResult**(int *requestCode*, int *response*, Intent *data*)

After the user has logged in, this function is called to redirect user to new activity

Parameters

- **requestCode** – Code that indicates what startActivityForResult call has finished
- **response** – Response from the completed call
- **data** – Data returned from the call

onCreate

protected void **onCreate**(Bundle *savedInstanceState*)

Generate onboarding pages and display them

Parameters

- **savedInstanceState** – Previous state to restore from

onFinishButtonPressed

public void **onFinishButtonPressed**()
When finish button pressed at end of onboarding, send user to login page

PostsFrag

public class **PostsFrag** extends Fragment
Created by Dylan Grosz (dgrosz@stanford.edu) on 11/10/17.

Fields

maxEventId

String **maxEventId**

maxPostId

String **maxPostId**

noPosts

TextView **noPosts**
The textview that is shown if no feed items have been created for this network.

queue

RequestQueue **queue**

selectedNetwork

long **selectedNetwork**

settings

SharedPreferences **settings**

Methods

fetchNewPage

public void **fetchNewPage**(Response.Listener<Void> *listener*)
If the user has exhausted the list of fetched posts/events, this will fetch another batch of posts.

Parameters

- **listener** – the listener that will be called when we finish fetching the stuffs.

onAttach

public void **onAttach**(Context *context*)
{@inheritDoc}

Parameters

- **context** – {@inheritDoc}

onCreate

public void **onCreate**(Bundle *savedInstanceState*)
{@inheritDoc} Also initialize *PostsFrag.settings* and *PostsFrag.queue*

Parameters

- **savedInstanceState** – {@inheritDoc}

onCreateView

public View **onCreateView**(LayoutInflater *inflater*, ViewGroup *container*, Bundle *savedInstanceState*)
Create user interface and handle clicks on posts by launching *SpecificPostActivity*, which displays more detailed information.

Parameters

- **inflater** – Inflates `R.layout.fragment_posts` into a full user interface that is a child of `container`
- **container** – Parent of created user interface
- **savedInstanceState** – Not used

Returns Inflated user interface

onDetach

public void **onDetach**()
{@inheritDoc}

onStop

public void **onStop**()
This ensures that we are canceling all network requests if the user is leaving this activity. We use a RequestFilter that accepts all requests (meaning it cancels all requests)

RVAdapter

public class **RVAdapter** extends RecyclerView.Adapter<*RVAdapter.PostViewHolder*>
Adapter that provides the *Post*s and/or *Event*s of a *org.codethechange.culturemesh.models.Network* to displayed, scrollable lists

Constructors

RVAdapter

public **RVAdapter** (List<*FeedItem*> *netPosts*, *OnItemClickListener* *listener*, Context *context*)
Initialize instance fields with provided parameters

Parameters

- **netPosts** – List of objects to represent in the displayed list
- **listener** – Listener to handle clicks on list tiems
- **context** – `Context` in which the list will be displayed

Methods

getItemCount

public int **getItemCount** ()
Get the number of items to display

Returns Number of items in the list of items to display (`RVAdapter.netPosts`)

getNetPosts

public List<*FeedItem*> **getNetPosts** ()
Get the items being represented as elements of the displayed list (not just the ones currently visible).

Returns Items represented as elements in the displayed list

getUserAttendingEvents

public Set<Long> **getUserAttendingEvents** ()
Get the events in this network that the user is attending, which affects some aspects of the event UI.

Returns a set of the ids of the events.

onBindViewHolder

public void **onBindViewHolder** (*PostViewHolder* *pvh*, int *i*)
Link the provided *PostViewHolder* to an object in the list `RVAdapter.netPosts`, which is used to fill the fields in the *PostViewHolder*

Parameters

- **pvh** – Item in the displayed list whose fields to fill with information
- **i** – Index of object in `RVAdapter.netPosts` that will serve as the source of information to fill into the displayed list item

onCreateViewHolder

public *PostViewHolder* **onCreateViewHolder** (ViewGroup *parent*, int *viewType*)

Create a new *PostViewHolder* from a View created by inflating the layout described by `R.layout.post_view`.

Parameters

- **parent** – Parent for created View used to create *PostViewHolder*
- **viewType** – Not used

Returns A new *PostViewHolder* for inclusion in the displayed list

RVAdapter.OnItemClickListener

public interface **OnItemClickListener**

Interface listeners for clicks on items must implement

Methods

onItemClick

void **onItemClick** (*FeedItem item*)

Handle a click on the provided item

Parameters

- **item** – Item that was clicked on

RVAdapter.PostViewHolder

static class **PostViewHolder** extends RecyclerView.ViewHolder

Stores the View elements of each item in the displayed list. Instances of this class are linked to objects in `RVAdapter.netPosts` by *RVAdapter.onBindViewHolder(PostViewHolder,int)*, which fills the fields with content from the object.

Fields

comment1Layout

RelativeLayout **comment1Layout**

Layout within which the two displayed comments are defined

cv

CardView **cv**
The `View` for the displayed list item

eventDescription

TextView **eventDescription**
Description of the *Event*

eventDetailsLL

LinearLayout **eventDetailsLL**
Layout within which the details section of the displayed list item is defined

eventLocation

TextView **eventLocation**
Where the *Event* will take place

eventTime

TextView **eventTime**
Time of the *Event*

images

ImageView[] **images**
Array of all image displays

layout

ConstraintLayout **layout**
Layout within which the displayed list item is defined

personName

TextView **personName**
Text fields for both *Post* and *Event* information

personPhoto

ImageView **personPhoto**
Display images with the displayed list item

post

boolean **post**
Whether this instance is configured to display the information for a *Post* or for a *Event*. `true` if it is for a *Post*

Constructors

PostViewHolder

PostViewHolder(View *itemView*)
Initialize instance fields by retrieving UI elements by their IDs in the provided `View`

Parameters

- **itemView** – Canvas upon which the displayed list item is built. Should already have the needed fields and other elements.

Methods

bind

public void **bind**(*FeedItem item*, *OnItemClickListener listener*)
Set the displayed list item's listener that handles clicks to that of the provided listener

Parameters

- **item** – The clicked-on item which will be passed to the listener's *OnItemClickListener.onItemClick(FeedItem)* method when the item is clicked
- **listener** – Listener to handle all clicks on items in the list

hideEventViews

void **hideEventViews**()
This instance will display the information from a *Post*, so hide all the fields that describe *Event*s

hidePostViews

void **hidePostViews**()
This instance will display the information from a *Event*, so hide all the fields that describe *Post*s

isPost

public boolean **isPost**()
Check whether the instance is displaying information for a *Post* or a *Event*

Returns `true` if displaying information for a *Post*. `false` if for an *Event*

RVCommentAdapter

public class **RVCommentAdapter** extends RecyclerView.Adapter<*RVCommentAdapter.PostReplyViewHolder*>
Adapter that populates a UI list with comments

Constructors

RVCommentAdapter

public **RVCommentAdapter** (List<*PostReply*> *comments*, *OnItemClickListener listener*, Context *context*)
Store parameters in instance fields

Parameters

- **comments** – List of comments to display in scrollable list to user
- **listener** – Will be called whenever an item is clicked
- **context** – `Context` within which the list will be displayed

Methods

getItemCount

public int **getItemCount** ()
Get the number of comments in the list

Returns Number of comments in list

onBindViewHolder

public void **onBindViewHolder** (*PostReplyViewHolder pvh*, int *i*)
Fill in the fields of `pvh` with the information stored in the *`PostReply`* at position `i` in the list of comments

Parameters

- **pvh** – `View` in the list whose fields will be filled-in
- **i** – Index of *`PostReply`* in `RVCommentAdapter.comments` to use as the source of information to fill with

onCreateViewHolder

public *PostReplyViewHolder* **onCreateViewHolder** (ViewGroup *parent*, int *viewType*)
Create a *`PostReplyViewHolder`* for `parent` with a `View` inflated from `R.layout.comment_view`

Parameters

- **parent** – `ViewGroup` within which to create the *`PostReplyViewHolder`*
- **viewType** – Not used

Returns The *`PostReplyViewHolder`* associated with the inflated `View`

RVCommentAdapter.OnItemClickListener

public interface **OnItemClickListener**
Interface implemented by any listener for item clicks

Methods

onCommentClick

void **onCommentClick**(*PostReply item*)
Handles clicks on a list item

Parameters

- **item** – Item in the list that was clicked

RVCommentAdapter.PostReplyViewHolder

static class **PostReplyViewHolder** extends RecyclerView.ViewHolder
Holder for the parts of each `View` in the list

Fields

cv

CardView **cv**
The `View` to display a single list item

images

ImageView[] **images**
Array of image components associated with a list item

layout

ConstraintLayout **layout**
Layout within which the list item components are arranged

personName

TextView **personName**
Textual components of the display for a single list item

personPhoto

ImageView **personPhoto**
Image components of the display for a single list item

reply

boolean **reply**

Constructors

PostReplyViewHolder

PostReplyViewHolder(View *itemView*)
Instantiate instance fields with `Views` using `View.findViewById(int)`

Parameters

- **itemView** – Item display whose fields are stored in instance fields

Methods

bind

public void **bind**(*PostReply item*, *OnItemClickListener listener*)
Attach a listener to an item in the displayed list

Parameters

- **item** – Item in the list to bind the listener to
- **listener** – Listener to bind to the list item

isPostReply

public boolean **isPostReply**()

RedirectableAppCompatActivity

public abstract class **RedirectableAppCompatActivity** extends AppCompatActivity
Superclass for all classes that support redirection instructions from the activity they are launched from. For instance, if `A` launches `B`, which is a subclass of *`RedirectableAppCompatActivity`*, `A` can give `B` instructions to launch `C` when it finishes. If instead `Z` launches `B`, it can give `B` instructions to next launch `X`.

Methods

onDestroy

protected void **onDestroy**()
{@inheritDoc} Also uses the extras in the launching `Intent` to decide which Activity to launch next

See also: *`Redirection`*

Redirection

public class **Redirection**
Classes that extend this one can be sent information when launched regarding where the user should be directed next.

Fields

LAUNCH_ON_FINISH_EXTRA

static final String **LAUNCH_ON_FINISH_EXTRA**
Key in `android.content.Intent`'s extras whose argument specifies the Class of the Activity to launch when finishing

See also: `Intent.getExtras()`

PASS_ON_FINISH_EXTRA

static final String **PASS_ON_FINISH_EXTRA**
Key in `android.content.Intent`'s extras whose argument specifies a `android.os.Bundle` whose contents will be passed as extras via the Intent called on finishing

SearchAdapter

public class **SearchAdapter**<T extends Listable> extends ArrayAdapter<T> implements Filterable
Populates a displayed list with items

Parameters

- **<T>** – Type of item to put in the list

Constructors

SearchAdapter

public **SearchAdapter**(Context *context*, int *resource*, int *listViewID*, List<T> *items*)
Initialize instance fields with provided parameters

Parameters

- **context** – {@inheritDoc}
- **resource** – {@inheritDoc}
- **listViewID** – Identifier for list the adapter will populate
- **items** – {@inheritDoc}

SearchAdapter

SearchAdapter (Context *context*, int *resource*, int *listViewID*)
Initialize context variables without a starting list

Parameters

- **context** – application context
- **resource** – int resource layout id

Methods

addAll

public void **addAll** (Collection<? extends T> *collection*)
Add all items in a `Collection` to the list of items the adapter displays in the list

Parameters

- **collection** – Items to add to the list

clear

public void **clear** ()
Clears the list of all items

getItem

public T **getItem** (int *position*)
Get the item associated with the list entry at a certain position

Parameters

- **position** – Position of list item

Returns The object represented at the specified position

getView

public View **getView** (int *position*, View *convertView*, ViewGroup *parent*)
Get a `View` for the list

Parameters

- **position** – Position of list element to get the `View` for
- **convertView** – `View` inflated from `R.layout.network_list_item` that will represent the list entry
- **parent** – Parent of the created `View`

Returns Inflated `View` for an element of the list

SearchAdapter.HolderItem

class **HolderItem**
Keeping views accessible saves calls to findViewById, which is a performance bottleneck. This is exactly why we have RecyclerView!

Fields

itemName

TextView **itemName**

numUsers

TextView **numUsers**

peopleIcon

ImageView **peopleIcon**

SettingsActivity

public class **SettingsActivity** extends *DrawerActivity* implements *NetworkSummaryAdapter*.*OnNetworkTapListener*
Screen that displays the current user's profile and let's them update it

Fields

MAX_PIXELS

final long **MAX_PIXELS**
The max number of pixels for an image given the image. Each pixel is 8 bytes large (according to RGBA_F16), and a MB is 2^20 bytes

MAX_QUALITY

final int **MAX_QUALITY**
Constant that clarifies that quality 100 means no compression.

MAX_SIDE

final double **MAX_SIDE**
The maximum number of pixels allowed on a single side of an image

bio

EditText **bio**
: Editable text fields that make up parts of the *User*'s profile

emptyText

TextView **emptyText**
: Text field that displays `R.string.no_networks` if the user has not joined any *Network*s

profilePicture

ImageView **profilePicture**
: The field for the *User*'s profile picture

queue

RequestQueue **queue**
: Queue for asynchronous tasks

rv

RecyclerView **rv**

scrollView

ScrollView **scrollView**
: The user whose profile is displayed and being edited

updateProfile

Button **updateProfile**
: Button for updating the *User*'s profile on the server with the one currently displayed

user

User **user**

Methods

onActivityResult

protected void **onActivityResult** (int *requestCode*, int *resultCode*, Intent *data*)
: This function is overridden to handle image selection. Inspiration from http://www.tauntaunwonton.com/blog/2015/1/21/simple-posting-of-multipartform-data-from-android

Parameters

- **requestCode** – PICK_IMAGE if we asked them to choose an image from the gallery.
- **resultCode** – {@inheritDoc}
- **data** – Hopefully, the URI.

onCreate

protected void **onCreate** (Bundle *savedInstanceState*)

Setup the user interface with the layout defined in `R.layout.activity_settings`. Also initialize instance fields for UI fields with the elements defined in the layout file. Fill the fields with the current profile (fetched using *API.Get.user(RequestQueue,long,Response.Listener)*). Link listeners to buttons and the displays of *Network*s to handle interactions.

Parameters

- **savedInstanceState** – {@inheritDoc}

onItemClick

public void **onItemClick** (View *v*, *Network network*)

Handle what happens when a user clicks on a *Network*. Right now, nothing is done.

Parameters

- **v** – {@inheritDoc}
- **network** – {@inheritDoc}

onStop

public void **onStop** ()

This ensures that we are canceling all network requests if the user is leaving this activity. We use a RequestFilter that accepts all requests (meaning it cancels all requests)

resetAdapter

void **resetAdapter** ()

Reset the adapter by clearing it and then populating it with new information from *API.Get.userNetworks(RequestQueue,long,Response.Listener)*, *API.Get.networkPostCount(RequestQueue,long,Response.Listener)*, and *API.Get.networkUserCount(RequestQueue,long,Response.Listener)*.

updateUser

public void **updateUser** (SharedPreferences *settings*)

Updates user info via PUT request to server.

Parameters

- **settings** – SharedPreferences instance to save email.

SpecificPostActivity

public class **SpecificPostActivity** extends AppCompatActivity implements *FormatManager.IconUpdateListener*
Displays a particular `Post` along with its comments (`PostReply`). Also allows the user to add comments.

Fields

boldButton

ImageButton **boldButton**
Buttons for inline markup of the text of the reply

commentField

ListenableEditText **commentField**
Field for the user to enter a comment

content

TextView **content**
Body of the `Post`

cv

CardView **cv**
The `View` that holds the UI elements that make up the displayed `Post`

editTextOpened

boolean **editTextOpened**
Whether the "window" to write a reply is open. Starts off `false`

formatManager

FormatManager **formatManager**
Manages markup of the text of the reply

images

ImageView[] **images**
Array of images associated with the `Post`

loadingOverlay

FrameLayout **loadingOverlay**

personName

TextView **personName**
Name of the creator of the *Post*

personPhoto

ImageView **personPhoto**
Profile photo of the author of the *Post*

postButton

Button **postButton**
Button to submit a comment on the *Post*

postTypePhoto

ImageView **postTypePhoto**
Other photo associated with the *Post*

progressBar

ProgressBar **progressBar**
Progress bar for displaying the progress of network operations

queue

RequestQueue **queue**
Queue for asynchronous tasks

timestamp

TextView **timestamp**
When the *Post* was created

toggleButtons

SparseArray<ImageButton> **toggleButtons**
Tracks whether the inline markup buttons have been toggled to "on"

username

TextView **username**
Unique display name of the creator of the *Post*

writeReplyView

ConstraintLayout **writeReplyView**
Layout within which the compose reply UI elements are arranged

Methods

closeEditTextView

void **closeEditTextView**()
When the user selects out of the text field, the view will shrink back to its original position.

genResizeAnimation

void **genResizeAnimation**(int *oldSize*, int *newSize*, ConstraintLayout *layout*)
This little helper handles the animation involved in changing the size of the write reply view.

Parameters

- **oldSize** – start height, in pixels.
- **newSize** – final height, in pixels.
- **layout** – writeReplyView

onCreate

protected void **onCreate**(Bundle *savedInstanceState*)
Create the user interface from the layout defined by `R.layout.activity_specific_post`. Initialize instance fields with the UI elements defined in the layout. Setup listeners to handle loading more comments, clicks to post replies, and load the *Post* to display.

Parameters

- **savedInstanceState** – {@inheritDoc}

onStop

protected void **onStop**()
This ensures that we are canceling all network requests if the user is leaving this activity. We use a RequestFilter that accepts all requests (meaning it cancels all requests)

openEditTextView

void **openEditTextView**()
This function animates the bottom view to expand up, allowing for a greater text field as well as toggle buttons.

updateIconToggles

public void **updateIconToggles**(SparseBooleanArray *formTogState*, SparseArray<int[]> *toggleIcons*)
Update whether an icon has been "toggled", or selected

Parameters

- **formTogState** – a SparseBooleanArray (HashMap) with int as key and boolean as value key: int id of toggleButton View we are using. value: true if toggled, false if not toggled.
- **toggleIcons** – a SparseArray (HashMap) with int as key and int[] as value. key: int id of toggleButton View we are using.

StartActivity

public class **StartActivity** extends AppCompatActivity
Transparent `android.app.Activity` that is the default Activity. It is the one launched when the app first starts, and it is the farthest back the "back" button (on the phone, not in the app) can go before leaving the app. It redirects the user based on their onboarding and login status.

Methods

onResume

protected void **onResume**()
Whenever this screen becomes "visible", immediately redirect the user to *TimelineActivity* if they have a selected network and are logged in. If they are logged-in without a selected network, redirect them to *ExploreBubblesOpenGLActivity*. If they are logged-out, redirect them to *OnboardActivity*.

TimelineActivity

public class **TimelineActivity** extends *DrawerActivity* implements *DrawerActivity.WaitForSubscribedList*
Show a feed of *org.codethechange.culturemesh.models.Post*s and *org.codethechange.culturemesh.models.Event*s for the currently selected *Network*

Fields

BUNDLE_NETWORK

static final String **BUNDLE_NETWORK**
The tag for showing that we're passing in the network to a new activity.

FILTER_CHOICE_EVENTS

static final String **FILTER_CHOICE_EVENTS**
The key in SharedPreferences for determining whether to display events in the feed.

FILTER_CHOICE_NATIVE

static final String **FILTER_CHOICE_NATIVE**
The key in SharedPreferences for determining whether to display posts in the feed.

FILTER_LABEL

final String **FILTER_LABEL**
The tag for FragmentManager to know we're opening the filter dialog.

joinNetwork

Button **joinNetwork**
The button that is shown if the user isn't subscribed to thus network. If they tap it, they join the network!

settings

static SharedPreferences **settings**
The app's preferences

Methods

animateFAB

void **animateFAB**()
This function controls the animation for the FloatingActionButtons. When the user taps the pencil icon, two other floating action buttons rise into view - create post and create event. The

createDefaultNetwork

protected void **createDefaultNetwork**()
Use API methods to fetch details of the user's selected network. Then setup activity to display that network's feed.

createNoNetwork

protected void **createNoNetwork**()
If the user has no selected network, direct them to *ExploreBubblesOpenGLActivity*

onBackPressed

public void **onBackPressed**()
Handle the back button being pressed. If the drawer is open, close it. If the user has scrolled down the feed, return it to the start. Otherwise, go back to the previous activity.

onCreate

protected void **onCreate**(Bundle *savedInstanceState*)

Setup user interface using layout defined in `R.layout.activity_timeline` and initialize instance fields with that layout's fields (elements)

Parameters

- **savedInstanceState** – {@inheritDoc}

onCreateOptionsMenu

public boolean **onCreateOptionsMenu**(Menu *menu*)

Inflate `menu` from `R.menu.timeline`

Parameters

- **menu** – `Menu` to inflate

Returns Always returns `true`

onOptionsItemSelected

public boolean **onOptionsItemSelected**(MenuItem *item*)

{@inheritDoc}

Parameters

- **item** – {@inheritDoc}

Returns If `item` is selected or if it has the same ID as `R.id.action_settings`, return `true`. Otherwise, return the result of `DrawerActivity.onOptionsItemSelected(MenuItem)` with parameter `item`

onStart

protected void **onStart**()

Check if user has selected a network to view, regardless of whether the user is subscribed to any networks yet. Previously, we checked if the user joined a network, and instead navigate the user to ExploreBubbles. This is not ideal because if a user wants to check out a network before joining one, then they will be unable to view the network. Also calls `DrawerActivity.onStart()`

onSubscribeListFinish

public void **onSubscribeListFinish**()

If the user is subscribed to the network, they are able to write posts and events. If the user is not subscribed to the network, there should be a pretty button for them that encourages the user to join the network. This control flow relies on checking if the user is subscribed to a network or not, which requires an instantiated subscribedNetworkIds set in DrawerActivity. This set is instantiated off the UI thread, so we need to wait until that thread completes. Thus, this function is called by DrawerActivity after the network thread completes.

onSwipeRefresh

public void **onSwipeRefresh**()
Restart activity to refresh the feed

TimelineActivity.FilterDialogFragment

public static class **FilterDialogFragment** extends DialogFragment
This dialog allows us to filter out native/twitter posts from the feed

Fields

filterSettings

boolean[] **filterSettings**

Methods

onCreateDialog

public Dialog **onCreateDialog**(Bundle *savedInstanceState*)

UsersListAdapter

public class **UsersListAdapter** extends RecyclerView.Adapter<*UsersListAdapter.ViewHolder*>
This Adapter is used for viewing the subscribed users of a network.

Fields

context

Context **context**
`Context` in which the list is being displayed

Constructors

UsersListAdapter

public **UsersListAdapter**(Context *context*, ArrayList<*User*> *users*)
Create a new object by instantiating instance fields with parameters

Parameters

- **context** – `Context` in which the list is displayed
- **users** – List of *`User`*s to display in the list

Methods

getItemCount

public int **`getItemCount`**`()`
Get the number of items in the list of objects to display

Returns Number of items in list to display

getUsers

public ArrayList<*User*> **`getUsers`**`()`
Get the list of objects to display

Returns List of objects represented in list

onBindViewHolder

public void **`onBindViewHolder`**`(`*ViewHolder holder*, int *position*`)`
Fill the name and profile picture fields of `holder` with the contents of an item in `UsersListAdapter.users`.

Parameters

- **`holder`** – *`ViewHolder`* whose fields to fill with information
- **`position`** – Index of item in list of users to use as source of information for filling

onCreateViewHolder

public *ViewHolder* **`onCreateViewHolder`**`(`ViewGroup *parent*, int *viewType*`)`
Create a new *`UsersListAdapter.ViewHolder`* from a `View` inflated from `R.layout.user_list_item` and with parent `parent`

Parameters

- **`parent`** – Parent for the `View` used to create the new *`UsersListAdapter`*
- **`viewType`** – Not used.

Returns The created *`UsersListAdapter.ViewHolder`*

UsersListAdapter.ViewHolder

class **`ViewHolder`** extends RecyclerView.ViewHolder
Holder of UI elements that compose each element of the displayed list

Fields

fullName

TextView **`fullName`**
`User`'s name

profilePicture

ImageView **profilePicture**
User's profile picture

Constructors

ViewHolder

ViewHolder(View *v*)
Initialize instance fields with fields in v and set the listener for clicks to open a more detailed view of the profile in *ViewProfileActivity*

Parameters

- **v** – View to use to display the list item

ViewProfileActivity

public class **ViewProfileActivity** extends AppCompatActivity
Displays the profile of a user other than the currently-logged-in one

Fields

SELECTED_USER

public static final String **SELECTED_USER**
Key for extra in android.content.Intents that specifies the user whose profile is to be displayed. This should be included in the intent that launches this activity.

loadingOverlay

FrameLayout **loadingOverlay**

mTabLayout

TabLayout **mTabLayout**
Handles the tabs available in the interface and serves as the framework on which the rest of the UI elements are arranged.

mViewPager

ViewPager **mViewPager**
Manages the variety of lists that could be displayed: networks, posts, and events

profilePic

ImageView **profilePic**
Field for the displayed profile's photo

queue

RequestQueue **queue**
Queue for asynchronous tasks

selUser

long **selUser**
ID of the *User* whose profile to display

userName

TextView **userName**
Text fields for the displayed profile's display name, bio, and name

Methods

onCreate

protected void **onCreate**(Bundle *savedInstanceState*)
Setup the user interface using the layout defined in `R.layout.activity_view_profile` and configure the various tabs. Initialize instance fields with the elements of the `android.view.View` created from the layout and fill the UI fields with the content of the profile using *API.Get.user(RequestQueue,long, Response.Listener)*

Parameters

- **savedInstanceState** – {@inheritDoc}

onStop

public void **onStop**()
This ensures that we are canceling all network requests if the user is leaving this activity. We use a RequestFilter that accepts all requests (meaning it cancels all requests)

onSupportNavigateUp

public boolean **onSupportNavigateUp**()
This allows the user to hit the back button on the toolbar to go to the previous activity.

Returns Always `true`

ViewProfileActivity.ContributionsPager

class **ContributionsPager** extends FragmentStatePagerAdapter

This PagerAdapter returns the correct fragment based on which list the user wishes to see. This could be seeing the list of networks the user is subscribed to, the list of posts the user has written, or the list of events the user has attended.

Constructors

ContributionsPager

ContributionsPager(FragmentManager *fm*)

Methods

getCount

public int **getCount**()

getItem

public Fragment **getItem**(int *position*)

getPageTitle

public CharSequence **getPageTitle**(int *position*)

ViewUsersModalSheetFragment

public class **ViewUsersModalSheetFragment** extends BottomSheetDialogFragment

Created By Drew Gregory on 03/30/18. This shows the subscribed users in the network using a modal bottom sheet https://material.io/guidelines/components/bottom-sheets.html#bottom-sheets-modal-bottom-sheets Also, inspiration from the following blog posts: - https://android-developers.googleblog.com/2016/02/android-support-library-232.html - https://code.tutsplus.com/articles/how-to-use-bottom-sheets-with-the-design-support-library–cms-26031

Fields

USER_NAMES

public static final String **USER_NAMES**

Keys for values passed as arguments to the fragment

queue

RequestQueue **queue**
Queue for asynchronous tasks

Methods

setupDialog

public void **setupDialog**(Dialog *dialog*, int *style*)
Create and configure `View` from `R.layout.rv_container`. Populate the fields in that `View` with the result of *`API.Get.networkUsers(RequestQueue,long,Response.Listener)`*

Parameters

- **dialog** – `Dialog` whose contents will be set using the `View` inflated from `R.layout.rv_container`
- **style** – Not used

org.codethechange.culturemesh.models

City

public class **City** extends *Place*
A *`City`* is a specific kind of *`Place`* that stores the ID and name of a city. It can also store the names and IDs of the city's country and region, but this is not mandatory. If any geographical descriptor (e.g. city, region, or country) is not specified, its name will be stored as *`Place.NOWHERE`*, but this constant should not be used by clients. Note that the `city` descriptor is mandatory.

Fields

cityName

public String **cityName**
Name of city

countryName

public String **countryName**
Name of country.

regionName

public String **regionName**
Name of region

Constructors

City

public **City**(long *cityId*, long *regionId*, long *countryId*, String *cityName*, String *regionName*, String *countryName*, *Point* *latLng*, long *population*, String *featureCode*)

Initialize instance fields and instance fields of superclasses based on provided arguments For creating cities that have city, region, and country all specified.

Parameters

- **cityId** – ID of city
- **regionId** – ID of city's region
- **countryId** – ID of country's region
- **cityName** – Name of city
- **regionName** – Name of region city lies within
- **countryName** – Name of country city lies within
- **latLng** – Latitude and longitude coordinates of city
- **population** – Population of the city
- **featureCode** – Feature code of the city

City

public **City**(long *cityId*, long *regionId*, String *cityName*, String *regionName*, *Point* *latLng*, long *population*, String *featureCode*)

Initialize instance fields and instance fields of superclasses based on provided arguments. For creating cities that have no country descriptor, but do have specified regions.

Parameters

- **cityId** – ID of city
- **regionId** – ID of city's region
- **cityName** – Name of city
- **regionName** – Name of region city lies within
- **latLng** – Latitude and longitude coordinates of city
- **population** – Population of the city
- **featureCode** – Feature code of the city

City

public **City**(long *cityId*, String *cityName*, *Point* *latLng*, long *population*, String *featureCode*)

Initialize instance fields and instance fields of superclasses based on provided arguments For creating cities that have no region nor country descriptor

Parameters

- **cityId** – ID of city

- **cityName** – Name of city
- **latLng** – Latitude and longitude coordinates of city
- **population** – Population of the city
- **featureCode** – Feature code of the city

City

public **City**(JSONObject *json*)

Initialize instance fields and those of superclass based on provided JSON This class extracts the following fields, if they are present: `country_name` and `region_name`. It requires that the key `name` exist, as its value will be used as the City's name

Parameters

- **json** – JSON object describing the city to create

Throws

- **JSONException** – May be thrown in response to an invalidly formatted JSON object

City

public **City**()

Empty constructor for database use only. This should never be called by our code.

Methods

getFullName

public String **getFullName**()

Get a name for the city that lists all available geographic descriptor names. For example, `Washington, D.C.` would be expressed as `Washington, D.C., United States`, while `San Francisco` would be expressed as `San Francisco, California, United States`.

Returns Name of city that includes all available geographic descriptors

getName

public String **getName**()

Get the name of the city

Returns City name

getShortName

public String **getShortName**()

Now display just city name.

newOnlyMissingRegion

public static *City* **newOnlyMissingRegion**(long *cityId*, long *countryId*, String *cityName*, String *countryName*, *Point* *latLng*, long *population*, String *featureCode*)

Return `City` object with fields initialized with provided parameters For creating cities that are only missing the region descriptor This unusual pseudo-constructor is required to avoid ambiguity between constructors

Parameters

- **cityId** – ID of city
- **countryId** – ID of country's region
- **cityName** – Name of city
- **countryName** – Name of country city lies within
- **latLng** – Latitude and longitude coordinates of city
- **population** – Population of the city
- **featureCode** – Feature code of the city

Returns City object that has been initialized

toString

public String **toString**()

Represent the object as a string suitable for debugging, but not for display to user.

Returns String representation of the form `Class[var=value, var=value, var=value, ...]`

Country

public class **Country** extends *Place*

A `Country` is a specific kind of `Place` that stores the ID and name of a country. No instance field should ever be set to `Place.NOWHERE`.

Fields

isoA2

public String **isoA2**

2-Letter ISO country code. This is not currently used.

name

public String **name**

Name of country

Constructors

Country

public **Country**(long *id*, String *name*, *Point* *latLng*, long *population*, String *featureCode*, String *isoA2*)
Initialize instance fields and those of superclass with provided parameters

Parameters

- **id** – ID of country
- **name** – Name of country
- **latLng** – Latitude and longitude coordinates of the region
- **population** – Population of the region
- **featureCode** – Region's feature code
- **isoA2** – 2-Letter ISO country code

Country

public **Country**(JSONObject *json*)
Initialize instance fields and those of superclass based on provided JSON It requires that the key `name` exist, as its value will be used as the country's name

Parameters

- **json** – JSON object describing the country to create

Throws

- **JSONException** – May be thrown in response to invalid JSON object

Country

public **Country**()
Empty constructor for database use only. This should never be called by our code.

Methods

getFullName

public String **getFullName**()
Get name of country, which is suitable for display in UI.

Returns Name of country, abbreviated if necessary to have a maximum length of *org.codethechange.culturemesh.Listable.MAX_CHARS*.

See also: *org.codethechange.culturemesh.Listable*

getName

public String **getName**()
Get name of country

Returns Name of country

getShortName

public String **getShortName**()
Now display just country name.

toString

public String **toString**()
Represent the object as a string suitable for debugging, but not for display to user.

Returns String representation of the form `Class[var=value, var=value, var=value, ...]`

DatabaseLocation

public abstract class **DatabaseLocation** extends *Location*
Superclass for Locations that will be stored in the database. Since the instance field names are used directly as column names in the database, a single class cannot be used for both From and Near locations (the column names would conflict). Therefore, two separate classes, `FromLocation` and `NearLocation` are used. They are nearly identical, however, so this superclass holds methods common to both. It also imposes requirements on them to ensure that those methods can function. The database will store the IDs of the city, region, and country.

Constructors

DatabaseLocation

public **DatabaseLocation**(long *countryId*, long *regionId*, long *cityId*)
Constructor that passes all parameters to superclass constructor

Parameters

- **countryId** – ID of country
- **regionId** – ID of region
- **cityId** – ID of city

DatabaseLocation

public **DatabaseLocation**(JSONObject *json*)
Constructor that passes all parameters to superclass constructor

Parameters

- **json** – JSON object that defines the location. See superclass constructor documentation.

Throws

- **JSONException** – May be thrown for improperly formatted JSON

DatabaseLocation

public **DatabaseLocation**(JSONObject *json*, String *cityIdKey*, String *regionIdKey*, String *countryIdKey*)

Passes all parameters, maintaining order, to *Location.Location(JSONObject,String,String,String)*

Parameters

- **json** –
- **cityIdKey** –
- **regionIdKey** –
- **countryIdKey** –

Throws

- **JSONException** –

DatabaseLocation

public **DatabaseLocation**()

Empty constructor for database use only. This should never be called by our code.

DatabaseNetwork

public class **DatabaseNetwork**

This class is solely for storing the bare, ID-only form of a network in the database. After being retrieved from the database or received from a network request, it should immediately be used to create a *Network* object, with the additional information that comes with. Storing only IDs in the database makes the *DatabaseNetwork.nearLocation*, *DatabaseNetwork.fromLocation* and *DatabaseNetwork.languageId* references pointers to database entries with more information. This reduces the risk of conflicting information and reduces the overhead of updating data in more than one spot in the database.

Fields

fromLocation

public *FromLocation* **fromLocation**

The location where the users of this network are from. It may be `null` to indicate that no location is specified only if *DatabaseNetwork.isLanguageBased* is `false`

id

public long **id**

The network's ID. This is used as its unique identifier in the database.

isLanguageBased

public boolean **isLanguageBased**
Denotes whether this network's *from* attribute is based on where an individual is from or on what language they speak. `true`: Based on what language they speak `false`: Based on what location they are from

languageId

public long **languageId**
The ID of the language the users of this network speak. It may be set to `-1` to indicate no language being specified only if *DatabaseNetwork.isLanguageBased* is `false`

nearLocation

public *NearLocation* **nearLocation**
The location where the users of this network currently reside. It must not be null.

Constructors

DatabaseNetwork

public **DatabaseNetwork**()
Empty constructor for database use only. This should never be called by our code.

DatabaseNetwork

public **DatabaseNetwork**(*NearLocation nearLocation*, *FromLocation fromLocation*, long *id*)
Create a new *DatabaseNetwork* for a network of people who come from the same area

Parameters

- **nearLocation** – Where the network's members currently reside
- **fromLocation** – Where the network's members are from
- **id** – ID for this network

DatabaseNetwork

public **DatabaseNetwork**(*NearLocation nearLocation*, long *langId*, long *id*)
Create a new *DatabaseNetwork* for a network of people who speak the same language

Parameters

- **nearLocation** – Where the network's members currently reside
- **langId** – ID for the language the network's members speak
- **id** – ID for this network

DatabaseNetwork

public **DatabaseNetwork**(JSONObject *json*)

If the key location_cur is present (old JSON version): Initialize instance fields with the data in the provided JSON. The following keys are mandatory and used: `location_cur`, whose value is expected to be a JSON describing a *NearLocation* object and can be passed to *NearLocation.NearLocation(JSONObject)*, and `network_class`, whose value is expected to be either `0`, indicating a location-based network, or `1`, indicating a language-based network. If the network is language-based, they key `language_origin` must exist with a value of a JSON object containing a key `id` whose value is the ID of a *Language*. If the network is location-based, the key `location_origin` must exist and have a value of a JSON object representing a *FromLocation* that can be passed to *FromLocation.FromLocation(JSONObject)*. **NOTE: This JSON format is deprecated and should not be used if possible. If the key location_cur is not present (new JSON version):** Initialize instance fields with the data in the provided JSON. The following keys are mandatory and used: All keys required by *NearLocation.NearLocation(JSONObject)* and the key `network_class`, whose value is expected to be either `_l`, indicating a language-based network, or one of `cc`, `rc`, and `co`, indicating a location-based network. If the network is language-based, the key `id_language_origin` must exist with a value of the ID of a *Language*. If the network is location-based, all keys required by *FromLocation.FromLocation(JSONObject)* must be present.

Parameters

- **json** – JSON object describing the network in terms of IDs

Throws

- **JSONException** – May be thrown in response to improperly formatted JSON

Methods

isLanguageBased

public boolean **isLanguageBased**()

Check whether this network is of people who speak the same language

Returns `true` if the network is defined in terms of language, `false` otherwise

isLocationBased

public boolean **isLocationBased**()

Check whether this network is of people who come from the same place

Returns `true` if the network is defined by where members are from, `false` otherwise

toString

public String **toString**()

Represent the object as a string suitable for debugging, but not for display to user.

Returns String representation of the form `Class[var=value, var=value, var=value, ...]`

Event

public class **Event** extends *FeedItem* implements Serializable, *Putable*, *Postable*

Describes an event like those shared in *Network*s

Fields

NOWHERE

public static final String **NOWHERE**

Value other classes should pass to this class and should expect to receive from this class to represent the portions of addresses that are not a part of the address. Note that *Event.getAddress()* uses this constant only when the entire address is missing.

addressLine1

public String **addressLine1**

First line of the address where the event is to take place. Some addresses may not have this value, in which case its value will be `Event.NOWHERE_INTERNAL`.

addressLine2

public String **addressLine2**

Second line of the address where the event is to take place. Some addresses may not have this value, in which case its value will be `Event.NOWHERE_INTERNAL`.

authorId

public long **authorId**

Unique identifier of the *User* who created the event

city

public String **city**

City portion of the address where the event is to take place. Some addresses may not have this value, in which case its value will be `Event.NOWHERE_INTERNAL`.

country

public String **country**

Country portion of the address where the event is to take place. Some addresses may not have this value, in which case its value will be `Event.NOWHERE_INTERNAL`.

description

public String **description**
User-generated description of the event. May contain formatting from *org.codethechange.culturemesh.FormatManager*.

See also: *org.codethechange.culturemesh.CreateEventActivity*

id

public long **id**
A unique identifier for the event. This should be generated server-side.

networkId

public long **networkId**
Unique identifier corresponding to the *Network* the *Event* is shared within

region

public String **region**
Region portion of the address where the event is to take place. Some addresses may not have this value, in which case its value will be `Event.NOWHERE_INTERNAL`.

timeOfEvent

public String **timeOfEvent**
Date and time of the event which must strictly conform to `yyyy-MM-ddTHH:mm:ss.SSSZ`. For example, `2015-01-01T15:00:00.000Z` is an acceptable value.

title

public String **title**
User-generated title for the event. Generally short (one line).

Constructors

Event

public **Event**(long *id*, long *networkId*, String *title*, String *description*, String *timeOfEvent*, long *author*, String *addressLine1*, String *addressLine2*, String *city*, String *region*, String *country*)
Construct an Event object from the provided parameters.

Parameters

- **id** – Unique identifier for the event
- **networkId** – Unique identifier for the *Network* the event is a part of
- **title** – User-generated title for the event

- **description** – User-generated description of the event
- **timeOfEvent** – Date and time of the event. Must strictly conform to the format `yyyy-MM-ddTHH:mm:ss.SSSZ`.
- **author** – Unique identifier for the *User* creating the *Event*
- **addressLine1** – Optional first line of the address. *Event.NOWHERE* if absent.
- **addressLine2** – Optional second line of the address. *Event.NOWHERE* if absent.
- **city** – Optional city portion of the address. *Event.NOWHERE* if absent.
- **region** – Optional region portion of the address. *Event.NOWHERE* if absent.
- **country** – Optional country portion of the address. *Event.NOWHERE* if absent.

Event

public **Event**()
Empty constructor that does nothing to initialize any instance fields. For database use only.

Event

public **Event**(JSONObject *json*)
Create a new Event object from a JSON representation that conforms to the following format:

```
{
          "id": 0,
          "id_network": 0,
          "id_host": 0,
          "date_created": "string",
          "event_date": "2018-06-23T04:39:42.600Z",
          "title": "string",
          "address_1": "string",
          "address_2": "string",
          "country": "string",
          "city": "string",
          "region": "string",
          "description": "string"
         }
```

Note that `date_created` is not used and may be omitted. Empty address fields should be `null`.

Parameters

- **json** – JSON representation of the *Event* to be created

Throws

- **JSONException** – May be thrown if an improperly formatted JSON is provided

Methods

getAddress

public String **getAddress**()
Generate a formatted form of the address for the event that is suitable for display to user.

Returns UI-suitable form of the address where the event will take place. Address portions (line1, line2, city, region, and country) are separated by commas, and missing portions are excluded. Example: `123 Any Street, New York, New York`. The address portions are user-generated, so this String may not describe a valid address. If no address is specified (i.e. if all address portions are missing), the *`Event.NOWHERE`* constant is returned.

getAuthor

public long **`getAuthor`**()
Get the unique identifier of the *`User`* who created the event

Returns Unique identifier of event author

getDescription

public String **`getDescription`**()
Get the author-generated description of the *`Event`*

Returns Text the *`User`* wrote to describe the event

getPostJson

public JSONObject **`getPostJson`**()
Create a JSON representation of the object that conforms to the following format:

```
{
            "id_network": 0,
            "id_host": 0,
            "event_date": "2018-07-21T15:10:30.838Z",
            "title": "string",
            "address_1": "string",
            "address_2": "string",
            "country": "string",
            "city": "string",
            "region": "string",
            "description": "string"
        }
```

This is intended to be the format used by the `/event/new` POST endpoint.

Throws

- **`JSONException`** – Unclear when this would be thrown

Returns JSON representation of the object

getPutJson

public JSONObject **`getPutJson`**()
Create a JSON representation of the object that conforms to the following format:

```
{
            "id": 0,
            "id_network": 0,
            "id_host": 0,
            "event_date": "2018-07-21T15:10:30.838Z",
            "title": "string",
            "address_1": "string",
            "address_2": "string",
            "country": "string",
            "city": "string",
            "region": "string",
            "description": "string"
        }
```

This is intended to be the format used by the `/event/new` PUT endpoint.

Throws

- **`JSONException`** – Unclear when this would be thrown

Returns JSON representation of the object

getTimeOfEvent

public String **`getTimeOfEvent`**()
Get the date and time of the event

Returns Timestamp for the event, which will be formatted as `yyyy-MM-ddTHH:mm:ss.SSSZ`

getTitle

public String **`getTitle`**()
Get the author-generated title for the *`Event`*

Returns Title the *`User`* chose to describe the event

setAuthor

public void **`setAuthor`**(*User author*)
Set the ID of the event's author. WARNING: The same ID must be used for a given *`User`* across CultureMesh.

Parameters

- **`author`** – Unique identifier of the *`User`* who created the event.

setDescription

public void **`setDescription`**(String *description*)
Set the author-generated description of the *`Event`*

Parameters

- **`description`** – Text the *`User`* wrote to describe the event

setTimeOfEvent

public void **setTimeOfEvent**(String *timeOfEvent*)
Set the date and time of the event

Parameters

- **timeOfEvent** – Timestamp for when the event will occur. Must strictly conform to `yyyy-MM-ddTHH:mm:ss.SSSZ`.

setTitle

public void **setTitle**(String *title*)
Set the author-generated title for the *Event*

Parameters

- **title** – Title the *User* chose to describe the event

FeedItem

public class **FeedItem**
Superclass for Posts and Events that mandates they both have a list of PostReply objects that can be displayed in a feed.

Fields

comments

public List<*PostReply*> **comments**
This list of PostReplies will be where we store the comments for each post.

FromLocation

public class **FromLocation** extends *DatabaseLocation*
Wrapper for *DatabaseLocation* that is for From locations. See the documentation for *DatabaseLocation* for information as to why this redundancy is necessary. All of these instance fields will be stored in the local cached database.

Fields

CITY_ID_KEY

public static final String **CITY_ID_KEY**
Constant that holds the JSON key whose value will be the ID of the city (`City.cityId`) in communications with the server.

See also: *Location.Location(JSONObject,String,String,String)*

COUNTRY_ID_KEY

public static final String **COUNTRY_ID_KEY**
Constant that holds the JSON key whose value will be the ID of the country (`Country.countryId`) in communications with the server.

See also: *Location.Location(JSONObject,String,String,String)*

REGION_ID_KEY

public static final String **REGION_ID_KEY**
Constant that holds the JSON key whose value will be the ID of the region (`Region.regionId`) in communications with the server.

See also: *Location.Location(JSONObject,String,String,String)*

from_city_id

public long **from_city_id**
Mirrors the *Location.cityId* in *Location* to avoid collisions in the database

See also: *DatabaseLocation*

from_country_id

public long **from_country_id**
Mirrors the *Location.countryId* in *Location* to avoid collisions in the database

See also: *DatabaseLocation*

from_region_id

public long **from_region_id**
Mirrors the *Location.regionId* in *Location* to avoid collisions in the database

See also: *DatabaseLocation*

Constructors

FromLocation

public **FromLocation**(long *cityId*, long *regionId*, long *countryId*)
Initialize instance fields with provided parameters

Parameters

- **cityId** – ID of city
- **regionId** – ID of region
- **countryId** – ID of country

FromLocation

public **FromLocation**(JSONObject *json*)
Initializes instance fields by passing JSON to *Location.Location(JSONObject,String,String,String)* and then initializing instance fields using `FromLocation.initialize()`

Parameters

- **json** – JSON object describing the location

Throws

- **JSONException** – May be thrown in response to improperly formatted JSON

FromLocation

public **FromLocation**(JSONObject *json*, boolean *distinguisher*)
Initializes instance fields by passing JSON to *Location.Location(JSONObject)*)} and then initializing instance fields using `FromLocation.initialize()`

Parameters

- **json** – JSON object describing the location

Throws

- **JSONException** – May be thrown in response to improperly formatted JSON

FromLocation

public **FromLocation**()
Empty constructor for database use only. This should never be called by our code.

Language

public class **Language** implements Serializable, *Listable*
Represents a language that may be spoken by users. It may be included as part of the definition of a *Network* or as an attribute of a *User*, for example.

Fields

language_id

public long **language_id**
Unique identifier for the language and the `PrimaryKey` for databases

name

public String **name**
Name of the language, as used by the API.

numSpeakers

public int **numSpeakers**
The number of Culturemesh users who speak the language

Constructors

Language

public **Language**(long *id*, String *name*, int *numSpeakers*)
Create a new *Language* object with the provided properties

Parameters

- **id** – Unique identifier for the language. The same ID must be used everywhere
- **name** – Human-readable name of the language. This will be displayed to users. It must also be unique, as it is passed in API calls.
- **numSpeakers** – The number of Culturemesh users who speak the language

Language

public **Language**(JSONObject *json*)
Create a new *Language* from the JSON produced by an API call. The JSON must conform to the following format:

```
{
            "lang_id": 0,
            "name": "string",
            "num_speakers": 0,
            "added": 0
        }
```

Note that the `added` key is not used and therefore optional.

Parameters

- **json** – JSON representation of the language to create.

Throws

- **JSONException** – May be thrown for an improperly formatted JSON

Language

public **Language**()
Empty constructor solely for storing Language objects in a database. **Never use this!**

Methods

getListableName

public String **getListableName**()

Get a descriptive representation of the language suitable for display to user

Returns Name of the language, abbreviated to be at most *Listable.MAX_CHARS* characters long.

getNumUsers

public long **getNumUsers**()

Get the number of users who speak the language

Returns Number of users who speak the language

toString

public String **toString**()

Convert the language to a unique string, its name

Returns The name of the language

urlParam

public String **urlParam**()

Get a representation of the language suitable for passage in a URL for API calls

Returns Name of the language encoded for inclusion in a URL

Location

public class **Location** implements Serializable, *Listable*

This object stores only the city, region, and country ID values, so it acts as a pointer to the more detailed information for the location in each City, Region, and Country's database entries or network information. No instance of this class should have `countryId`, `regionId`, and `cityId` all equal to `NOWHERE`. This should only be possible by mis-using the JSON constructor or by supplying `-1` as an ID. Neither should ever be done.

```
Location
                     (IDs only)
                     /                            /                               ␣
↪         /                               /                        Place␣
↪(Abstract)    DatabaseLocation (Abstract)
          (Full Info)                   (IDs)
            /  |                     /                 /    |                 ␣
↪NearLocation  FromLocation
      City  Region  Country  (Wrappers for DatabaseLocation)
   (Specific cases of Place)
```

Fields

CITY

public static final int **CITY**
Represents a type of `Location` that has a city defined.

See also: *`Location.getType()`*

COUNTRY

public static final int **COUNTRY**
Represents a type of `Location` that has only a country defined.

See also: *`Location.getType()`*

NOWHERE

protected static final int **NOWHERE**
These constants are used to identify the type of location being stored. See the documentation for `getType` for more. `NOWHERE` is `protected` because it should never be used by clients. It is only for subclasses to denote empty IDs. Creating locations with empty IDs should be handled by subclass constructors or methods.

REGION

public static final int **REGION**
Represents a type of `Location` that has a region defined but not a city.

See also: *`Location.getType()`*

URL_NULL_ID

public static final int **URL_NULL_ID**
The value to be transmitted to the API in place of a missing country, region, or city ID

cityId

public long **cityId**

countryId

public long **countryId**
These instance fields store the IDs of the city, region, and country defining the location They can be `private` because a plain `Location` object should not need to be stored in the database.

locationName

public String **locationName**
This is is only used for other searching in *org.codethechange.culturemesh.FindNetworkActivity*. Do not use this field anywhere else.

regionId

public long **regionId**

Constructors

Location

public **Location**(long *countryId*, long *regionId*, long *cityId*)
Initializes ID instance fields using the provided IDs

Parameters

- **countryId** – ID of country
- **regionId** – ID of region
- **cityId** – ID of city

Location

public **Location**(JSONObject *json*)
Initializes ID instance fields using the provided JSON object If present, the values of the keys `city_id`, `region_id`, and `country_id` will be used automatically. Depending on the presence of those keys, the value of the key `id` will be used to fill the instance field for the JSON type. See `getJsonType` for more. This constructor is designed to be used when creating *Place*s. Precondition: The JSON must be validly formatted, with examples in `API.java`

Parameters

- **json** – JSON object containing the country, region, and city IDs

Throws

- **JSONException** – May be thrown if the JSON is improperly formatted

Location

public **Location**(JSONObject *json*, String *cityIdKey*, String *regionIdKey*, String *countryIdKey*)
Initializes ID instance fields using the provided JSON object. The keys extracted are provided as parameters, but those keys need not exist in the JSON. Any missing keys will be treated as if the location does not have such a geographic identifier. This may produce an invalid location, and the JSON is followed blindly. Precondition: JSON must describe a valid location

Parameters

- **json** – JSON that describes the location to create
- **cityIdKey** – The key that, if present in the JSON, has a value of the ID of the city

- **regionIdKey** – The key that, if present in the JSON, has a value of the ID of the region
- **countryIdKey** – The key that, if present in the JSON, has a value of the ID of the country

Throws

- **JSONException** – May be thrown in the case of an invalid JSON

Location

public **Location**()
Empty constructor for database use only. This should never be called by our code.

Methods

getCityId

public long **getCityId**()
Getter for the city ID, which may return `NOWHERE`, so `hasCityId` should be used to check first

Returns The city ID

getCountryId

public long **getCountryId**()
Getter for the country ID, which may return `NOWHERE`, so `hasCountryId` should be used to check first

Returns The country ID

getDatabaseId

protected long **getDatabaseId**()
Find the ID that should be used as the `PrimaryKey` for a database. It is the ID of the most specific geographical descriptor with an ID that is not `NOWHERE`. **WARNING: The returned ID is NOT guaranteed to be unique**

Returns ID for use as `PrimaryKey` in a database

getFromLocation

public *FromLocation* **getFromLocation**()
Transform a *Location* into a *FromLocation*

Returns A *FromLocation* with the same IDs as the *Location* object whose method was called

getListableName

public String **getListableName**()
Get a UI-ready name for the Location

Returns Name for the Location that is suitable for display to the user. Abbreviated to be a maximum of *Listable.MAX_CHARS* characters long.

getNearLocation

public *NearLocation* **getNearLocation**()
Transform a `Location` into a `NearLocation`

Returns A `NearLocation` with the same IDs as the `Location` object whose method was called

getRegionId

public long **getRegionId**()
Getter for the region ID, which may return `NOWHERE`, so `hasRegionId` should be used to check first

Returns The region ID

getType

public int **getType**()
The most specific ID that is not `NOWHERE` determines the location's type, even if more general IDs are `NOWHERE`. For example, if `regionId = 0` and `countryId = cityId = NOWHERE`, the type would be `REGION`

Returns Location's type as `CITY`, `REGION`, or `COUNTRY`

hasCityId

public boolean **hasCityId**()
Check if the city ID is specified (i.e. not `NOWHERE`)

Returns `true` if the city ID is specified, `false` otherwise

hasCountryId

public boolean **hasCountryId**()
Check if the country ID is specified (i.e. not `NOWHERE`)

Returns `true` if the country ID is specified, `false` otherwise

hasRegionId

public boolean **hasRegionId**()
Check if the region ID is specified (i.e. not `NOWHERE`)

Returns `true` if the region ID is specified, `false` otherwise

toString

public String **toString**()
Represent the object as a string suitable for debugging, but not for display to user.

Returns String representation of the form `Class[var=value, var=value, var=value, ...]`

urlParam

public String **urlParam**()

Represent the *Location* in a form suitable for use as the value of a key passed in a URL parameter to the API. Specifically, it returns the country, region, and city IDs separated by commas and in that order. The commas are escaped with the UTF-8 scheme and any missing IDs are replaced with the *Location.URL_NULL_ID* constant, which is understood by the API as signifying `null`.

Returns An API-compatible representation suitable for use as the value in a URL parameter

NearLocation

public class **NearLocation** extends *DatabaseLocation*

Wrapper for `DatabaseLocation` that is for Near locations. See the documentation for `DatabaseLocation` for information as to why this redundancy is necessary. All of these instance fields will be stored in the local cached database.

Fields

CITY_ID_KEY

public static final String **CITY_ID_KEY**

Constant that holds the JSON key whose value will be the ID of the city (`City.cityId`) in communications with the server.

See also: *Location.Location(JSONObject,String,String,String)*

COUNTRY_ID_KEY

public static final String **COUNTRY_ID_KEY**

Constant that holds the JSON key whose value will be the ID of the country (`Country.countryId`) in communications with the server.

See also: *Location.Location(JSONObject,String,String,String)*

REGION_ID_KEY

public static final String **REGION_ID_KEY**

Constant that holds the JSON key whose value will be the ID of the region (`Region.regionId`) in communications with the server.

See also: *Location.Location(JSONObject,String,String,String)*

near_city_id

public long **near_city_id**

Mirrors the *Location.cityId* in *Location* to avoid collisions in the database

See also: *DatabaseLocation*

near_country_id

public long **near_country_id**
Mirrors the *Location.countryId* in *Location* to avoid collisions in the database

See also: *DatabaseLocation*

near_region_id

public long **near_region_id**
Mirrors the *Location.regionId* in *Location* to avoid collisions in the database

See also: *DatabaseLocation*

Constructors

NearLocation

public **NearLocation** (long *cityId*, long *regionId*, long *countryId*)
Initialize instance fields with provided parameters

Parameters

- **cityId** – ID of city
- **regionId** – ID of region
- **countryId** – ID of country

NearLocation

public **NearLocation** (JSONObject *json*)
Initializes instance fields by passing JSON to *Location.Location(JSONObject,String,String,String)* and then initializing instance fields using `NearLocation.initialize()`

Parameters

- **json** – JSON object describing the location

Throws

- **JSONException** – May be thrown in response to improperly formatted JSON

NearLocation

public **NearLocation** (JSONObject *json*, boolean *distinguisher*)
Initializes instance fields by passing JSON to *Location.Location(JSONObject)* and then initializing instance fields using `NearLocation.initialize()`

Parameters

- **json** – JSON object describing the location
- **distinguisher** – Useless value used to distinguish from *NearLocation.NearLocation(JSONObject)*

Throws

- **`JSONException`** – May be thrown in response to improperly formatted JSON

NearLocation

public **`NearLocation`**()
Empty constructor for database use only. This should never be called by our code.

Network

public class **`Network`** implements Serializable, *Postable*
This class stores all the information related to a network. It is fully expanded, meaning that its instance fields like `Network.nearLocation` store expanded objects (i.e. `Place`, not the stripped-down forms for database storage.

Fields

fromLocation

public *Place* **`fromLocation`**
Where users of the network are from. Must be specified if the network is location-based.

id

public long **`id`**
ID of network. Must always be specified.

language

public *Language* **`language`**
What language the users of the network speak. Must be specified if the network is language- based.

nearLocation

public *Place* **`nearLocation`**
The current location of users in the network. Must always be specified.

Constructors

Network

public **`Network`**(*Place nearLocation*, *Place fromLocation*, long *id*)
Create a location-based network from the provided objects

Parameters

- **`nearLocation`** – Where the network's users currently reside

- **fromLocation** – Where the network's users are all from
- **id** – ID of the network

Network

public **Network**(*Place nearLocation*, *Language lang*, long *id*)
Create a language-based network from the provided objects

Parameters

- **nearLocation** – Where the network's users currently reside
- **lang** – What language the network's users all speak
- **id** – ID of the network

Methods

getDatabaseNetwork

public *DatabaseNetwork* **getDatabaseNetwork**()
Get a *DatabaseNetwork* with the IDs stored by the *Network* from which the method is called.

Returns The *DatabaseNetwork* associated with this *Network*

getPostJson

public JSONObject **getPostJson**()
Generate a JSON representation of the object suitable for use in POST requests. Wrapper for *Network.toJSON()*.

Throws

- **JSONException** – May be thrown if something that should be a value in the JSON is not a valid value in the JSON format.

Returns JSON that can be passed to the server in the body of a POST request

See also: *Network.toJSON();*

isLanguageBased

public boolean **isLanguageBased**()
Check whether this network is of people who speak the same language

Returns `true` if the network is defined in terms of language, `false` otherwise

isLocationBased

public boolean **isLocationBased**()
Check whether this network is of people who come from the same place

Returns `true` if the network is defined by where members are from, `false` otherwise

toJSON

public JSONObject **toJSON**()

Generate a JSON describing the object. The JSON will conform to the following format:

```
{
            "id_city_cur": 0,
            "city_cur": "string",
            "id_region_cur": 0,
            "region_cur": "string",
            "id_country_cur": 0,
            "country_cur": "string",
            "id_city_origin": 0,
            "city_origin": "string",
            "id_region_origin": 0,
            "region_origin": "string",
            "id_country_origin": 0,
            "country_origin": "string",
            "id_language_origin": 0,
            "language_origin": "string",
            "network_class": "string"
        }
```

where missing IDs are passed as *Location.NOWHERE*. This format is suitable for submission to the server using the `/network/new` POST endpoint.

Throws

- **JSONException** – Unclear when this would be thrown

Returns JSON representation of the object

toString

public String **toString**()

Represent the object as a string suitable for debugging, but not for display to user.

Returns String representation of the form `Class[var=value, var=value, var=value, ...]`

Place

public abstract class **Place** extends *Location* implements *Listable*, Serializable

A `Place` is a *Location* with more information. While a *Location* stores only city, region, and country IDs, `Place` also stores the areas position (latitude and longitude), population, and feature code. `Place` is abstract, and some examples of its subclasses are: *City*, *Region*, and *Country*. Created by Drew Gregory on 2/23/18. This is the superclass for cities, regions, and countries.

Fields

NOWHERE

protected static final String **NOWHERE**

The `NOWHERE` constant is used internally by this hierarchy as the name of a location's city, region, or country

when that geographic identifier is not specified. For example, Washington D.C. has no state (i.e. region), so its region might be stored as `NOWHERE`. **This should never be used by clients.** Instead, creating such places should be done through provided constructors or methods.

featureCode

public String **`featureCode`**
Feature code, which is a string describing the type of place represented (e.g. a capital, a religiously important area, an abandoned populated area). See http://www.geonames.org/export/codes.html for more examples.

id

public long **`id`**
The ID to be used by a database to identify this object. It is set using `Place.getDatabaseId()`. See that method's documentation for more information. Crucially **it is NOT guaranteed to be unique.**

latLng

public *Point* **`latLng`**
Latitude and longitude

population

public long **`population`**
The population of the described area. This is for display under the "people" icon when areas are listed.

Constructors

Place

public **`Place`**(long *countryId*, long *regionId*, long *cityId*, *Point* *latLng*, long *population*, String *featureCode*)
Initialize instance fields with provided parameters. Also calls *`Location.Location(long,long,long)`* with the provided IDs Postcondition: *`Place.id`* is initialized using `Place.getDatabaseId()`

Parameters

- **`countryId`** – ID of country
- **`regionId`** – ID of region
- **`cityId`** – ID of city
- **`latLng`** – Coordinates (latitude and longitude) of location
- **`population`** – Population of location
- **`featureCode`** – Feature code of location

Place

public **Place**(JSONObject *json*)

Initializes ID instance fields using the provided JSON object The following keys must be present and are used to fill the relevant instance fields: `latitude`, `longitude`, `population`, `feature_code`. In addition, the JSON object is passed to *Location.Location(JSONObject)*. See its documentation for details on its requirements. *Place.id* is initialized using `Place.getDatabaseId()`. Precondition: The JSON must be validly formatted, with examples in *org.codethechange.culturemesh.API*

Parameters

- **json** – JSON object to extract initializing information from

Throws

- **JSONException** – May be thrown for invalidly formatted JSON object

Place

public **Place**()

Empty constructor for database use only. This should never be called by our code.

Methods

abbreviateForListing

public static String **abbreviateForListing**(String *toAbbreviate*)

Abbreviate the provided string by truncating it enough so that, after adding *Listable.ellipses*, the string is *Listable.MAX_CHARS* characters long. If the string is already shorter than *Listable.MAX_CHARS*, it is returned unchanged.

Parameters

- **toAbbreviate** – String whose abbreviated form will be returned

Returns Abbreviated form of the string. Has a maximum length of *Listable.MAX_CHARS*

getCityName

public String **getCityName**()

Attempt to get the name of the *City* for this *Place*. May return *Place.NOWHERE*.

Returns Name of the *City* if one is available, or *Place.NOWHERE* otherwise.

getCountryName

public String **getCountryName**()

Attempt to get the name of the *Country* for this *Place*. May return *Place.NOWHERE*.

Returns Name of the *Country* if one is available, or *Place.NOWHERE* otherwise.

getFeatureCode

public String **getFeatureCode**()
Get the feature code describing the location. See http://www.geonames.org/export/codes.html for examples.

Returns Location's feature code

getFullName

public abstract String **getFullName**()
Subclasses are required to provide a method to generate their full, unambiguous name. For example, `New York, New York, United States of America`.

Returns Full, unambiguous name of place

getLatLng

public *Point* **getLatLng**()
Get the coordinates of the location

Returns Latitude and longitude of the location

getListableName

public String **getListableName**()
Get a name suitable for display in listings of places, as required to implement *`Listable`*. This name is created by abbreviating the output of *`Place.getFullName()`* and adding *`Listable.ellipses`* such that the total length is a no longer than *`Listable.MAX_CHARS`*

Returns Name of Location suitable for display in UI lists. Has a maximum length of *`Listable.MAX_CHARS`*.

getNumUsers

public long **getNumUsers**()
Get the number of users (population) to display in conjunction with the location

Returns Population of the location

getPopulation

public long **getPopulation**()
Get the population of the location

Returns Location's population

getRegionName

public String **getRegionName**()
Attempt to get the name of the *`Region`* for this *`Place`*. May return *`Place.NOWHERE`*.

Returns Name of the *Region* if one is available, or *Place.NOWHERE* otherwise.

getShortName

public abstract String **getShortName**()
In the interest of space, we also want the abbreviated version of the location (just the city name for example)

Returns Name of location suitable for header bar.

toString

public String **toString**()
Represent the object as a string suitable for debugging, but not for display to user.

Returns String representation of the form `Class[var=value, var=value, var=value, ...]`

Point

public class **Point**
Represents a point on the globe by its coordinates

Fields

latitude

public long **latitude**
Latitude of the point

longitude

public long **longitude**
Longitude of the point

Post

public class **Post** extends *FeedItem* implements Serializable, *Postable*, *Putable*
Represents a post made by a user in a network. A post is arbitrary, formatted text of the user's choosing.

Fields

author

public *User* **author**
The *User* who created the post. This may not be present and have to be instantiated from *Post.userId*. Currently, this is handled by *org.codethechange.culturemesh.API*

content

public String **content**
The body of the post. May be formatted.

See also: *org.codethechange.culturemesh.FormatManager*

datePosted

public String **datePosted**
Timestamp for when the post was created. Should conform to `EEE, dd MMM yyyy kk:mm:ss z`

id

public long **id**
Uniquely identifies the post across all of CultureMesh

imgLink

public String **imgLink**
Link to an image, if available, that is associated with the post

network

public *Network* **network**
The *Network* who created the post. This may not be present and have to be instantiated from *Post.networkId*. Currently, this is handled by *org.codethechange.culturemesh.API*

networkId

public long **networkId**
Unique identifier for the network the post was made in. This is used when only a reference to the full *Network* object is needed, e.g. when getting a post from the API. The rest of the information associated with the network can be fetched later.

userId

public long **userId**
Unique identifier for the user who created the post. This is used when only a reference to the full *User* object is needed, e.g. when getting a post from the API. The rest of the information associated with the user can be fetched later.

vidLink

public String **vidLink**
Link to a video, if available, that is associated with the post TODO: Handle multiple links?

Constructors

Post

public **Post**(long *id*, long *author*, long *networkId*, String *content*, String *imgLink*, String *vidLink*, String *datePosted*)

Create a new post object from the provided parameters. The resulting object will not be fully instantiated (e.g. *Post.author* and *Post.network* will be `null`.

Parameters

- **id** – Uniquely identifies the post across all of CultureMesh
- **author** – ID of *User* who created the post
- **networkId** – ID of the *Network* in which the post was made
- **content** – Formatted text that composes the body of the post.
- **imgLink** – Link to an image associated with the post. `null` if none associated.
- **vidLink** – Link to a video associated with the post. `null` if none associated
- **datePosted** – When the post was created. Must conform to `EEE, dd MMM yyyy kk:mm:ss z`

See also: *org.codethechange.culturemesh.FormatManager*

Post

public **Post**()

Empty constructor for database

Post

public **Post**(JSONObject *json*)

Creates a bare (uninstantiated) *Post* from a JSON that conforms to the below format:

```
{
        "id": 0,
        "id_user": 0,
        "id_network": 0,
        "post_date": "string",
        "post_text": "string",
        "post_class": 0,
        "post_original": "string",
        "vid_link": "string",
        "img_link": "string"
       }
```

Parameters

- **json** – JSON representation of the *Post* to construct

Throws

- **JSONException** – May be thrown in response to an improperly formatted JSON

Methods

getAuthor

public *User* **getAuthor**()
Get the author of the post. Object must be fully instantiated, not just populated with IDs

Returns Author of the post

getContent

public String **getContent**()
Get the formatted text that makes up the body of the post.

Returns Body of the post, which may be formatted.

See also: *org.codethechange.culturemesh.FormatManager*

getDatePosted

public String **getDatePosted**()
Get when the post was created.

Returns Timestamp of when post was created. Conforms to `EEE, dd MMM yyyy kk:mm:ss z`

getImageLink

public String **getImageLink**()
Get the URL to the image associated with the post.

Returns URL to associated image. If no image is associated, `null`

getNetwork

public *Network* **getNetwork**()
Get the network of the post. Object must be fully instantiated, not just populated with IDs

Returns Network of the post

getPostJson

public JSONObject **getPostJson**()
Wrapper for *Post.toJSON()*

getPostedTime

public Date **getPostedTime**()
Sometimes, we will want to get the time not just as a string but as a Date object (i.e. for comparing time for sorting)

Returns Date object based on datePosted string.

getPutJson

public JSONObject **getPutJson**()
Wrapper for *Post.toJSON()*

getVideoLink

public String **getVideoLink**()
Get the URL to the video associated with the post.

Returns URL to associated video. If no video is associated, `null`

setContent

public void **setContent**(String *content*)
Set the body of the post to the parameter provided.

Parameters

- **content** – Formatted body of the post.

See also: *org.codethechange.culturemesh.FormatManager*

setDatePosted

public void **setDatePosted**(String *datePosted*)
Get the timestamp for when the post was created.

Parameters

- **datePosted** – When post was created. Conforms to `EEE, dd MMM yyyy kk:mm:ss z`

setImageLink

public void **setImageLink**(String *imgLink*)
Associate the image at the provided URL with the post. Replaces any existing image URL.

Parameters

- **imgLink** – URL to the image to add to the post

setVideoLink

public void **setVideoLink**(String *vidLink*)
Associate the video at the provided URL with the post. Replaces any existing video URL.

Parameters

- **vidLink** – URL to the video to add to the post

toJSON

public JSONObject **toJSON**()
Generate a JSON describing the object. The JSON will conform to the following format:

```
{
                "id_user": 0,
                "id_network": 0,
                "post_text": "string",
                "vid_link": "string",
                "img_link": "string"
            }
```

The resulting object is suitable for use with the `/post/new` endpoint (PUT and POST).

Throws

- **JSONException** – Unclear when this would be thrown

Returns JSON representation of the object

PostReply

public class **PostReply** implements *Postable*, *Putable*
Created by Drew Gregory on 3/4/18.

Fields

author

public *User* **author**

id

public long **id**

networkId

public long **networkId**

parentId

public long **parentId**

replyDate

public String **replyDate**

replyText

public String **replyText**

userId

public long **userId**

Constructors

PostReply

public **PostReply**(long *id*, long *parentId*, long *userId*, long *networkId*, String *replyDate*, String *replyText*)

PostReply

public **PostReply**(JSONObject *replyObj*)

PostReply

public **PostReply**()

Methods

getAuthor

public *User* **getAuthor**()

getPostJson

public JSONObject **getPostJson**()

getPutJson

public JSONObject **getPutJson**()

toJSON

public JSONObject **toJSON**()
Generate a JSON describing the object. The JSON will conform to the following format:

```
{
                "id_parent": 0,
                "id_user": 0,
                "id_network": 0,
                "reply_text": "string"
            }
```

The resulting object is suitable for use with the `/post/{postId}/reply` POST or PUT endpoints.

Throws

- **`JSONException`** – Unclear when this would be thrown

Returns JSON representation of the object

Postable

public interface **`Postable`**

Classes that implement this interface can be sent in the bodies of requests sent using `org.codethechange.culturemesh.API.Post.model(RequestQueue,Postable,String,String,Response.Listener)`.

Methods

getPostJson

JSONObject **`getPostJson`**`()`

Generates a JSON representation of the object that can be used in POST requests to the server. The exact format of the JSON depends upon the specifications of the server API. See the server's Swagger documentation for more.

Throws

- **`JSONException`** – May be thrown if any of the values to include in the JSON are incompatible with the JSON format

Returns JSON representation of the object suitable for inclusion in the bodies of POST requests

Putable

public interface **`Putable`**

Classes that implement this interface can be sent in the bodies of requests sent using `org.codethechange.culturemesh.API.Put.model(RequestQueue,Putable,String,String,Response.Listener)`.

Methods

getPutJson

JSONObject **`getPutJson`**`()`

Generates a JSON representation of the object that can be used in PUT requests to the server. The exact format

of the JSON depends upon the specifications of the server API. See the server's Swagger documentation for more.

Throws

- **`JSONException`** – May be thrown if any of the values to include in the JSON are incompatible with the JSON format

Returns JSON representation of the object suitable for inclusion in the bodies of PUT requests

Region

public class **`Region`** extends *Place*

A *`Region`* is a specific kind of *`Place`* that stores the ID and name of a region. It can also store the name and ID of the region's country, but this is not mandatory. If any geographical descriptor (e.g. city, region, or country) is not specified, its name will be stored as *`Place.NOWHERE`*, but this constant should not be used by clients. Note that the `region` descriptor is mandatory.

Fields

countryName

public String **`countryName`**

Name of the country (may store *`Place.NOWHERE`*

regionName

public String **`regionName`**

Name of the region (should always be specified and not as *`Place.NOWHERE`*

Constructors

Region

public **`Region`**(long *regionId*, long *countryId*, String *regionName*, String *countryName*, *Point* *latLng*, long *population*, String *featureCode*)

Initialize instance fields and those of superclass with provided parameters No parameters should be set to *`Place.NOWHERE`* or *`Location.NOWHERE`* For regions with explicitly specified countries

Parameters

- **`regionId`** – ID of region
- **`countryId`** – ID of country
- **`regionName`** – Name of region
- **`countryName`** – Name of country
- **`latLng`** – Latitude and longitude coordinates of the region
- **`population`** – Population of the region
- **`featureCode`** – Region's feature code

Region

public **Region** (long *regionId*, String *regionName*, *Point* *latLng*, long *population*, String *featureCode*)

Initialize instance fields and those of superclass with provided parameters No parameters should be set to `Place.NOWHERE` or `Location.NOWHERE` For regions that have no specified country

Parameters

- **regionId** – ID of region
- **regionName** – Name of region
- **latLng** – Latitude and longitude coordinates of the region
- **population** – Population of the region
- **featureCode** – Region's feature code

Region

public **Region** (JSONObject *json*)

Initialize instance fields and those of superclass based on provided JSON This class extracts the following fields, if they are present: `country_name`. It requires that the key `name` exist, as its value will be used as the region's name

Parameters

- **json** – JSON object describing the region to create

Throws

- **JSONException** – May be thrown in response to an invalidly formatted JSON object

Region

public **Region** ()

Empty constructor for database use only. This should never be called by our code.

Methods

getFullName

public String **getFullName** ()

Get a name for the region that lists all available geographic descriptor names. For example, `Washington, D.C.` would be expressed as `Washington, D.C., United States`, while `San Francisco` would be expressed as `San Francisco, California, United States`.

Returns Name of city that includes all available geographic descriptors

getName

public String **getName** ()

Get the name of the region

Returns Name of region

getShortName

public String **getShortName**()
Now display just region name.

toString

public String **toString**()
Represent the object as a string suitable for debugging, but not for display to user.

Returns String representation of the form `Class[var=value, var=value, var=value, ...]`

User

public class **User** implements Serializable
Represents a CultureMesh user's public profile. Methods that require non-public data (e.g. email or password) take that information in as parameters and do not store it after the method completes.

Fields

CM_LOGO_URL

public static final String **CM_LOGO_URL**

DEFAULT_BIO

public static final String **DEFAULT_BIO**

DEFAULT_GENDER

public static final String **DEFAULT_GENDER**

IMG_URL_PREFIX

public static final String **IMG_URL_PREFIX**

aboutMe

public String **aboutMe**
Bio user has written about themselves. Editable by user.

firstName

public String **firstName**
User's first name. Editable by user, and may be pseudonymous.

gender

public String **gender**
User's gender. Editable by user.

id

public long **id**
The user's unique identifier, which identifies them across all of CultureMesh and is constant. Not editable by user.

imgURL

public String **imgURL**
URL for the user's profile picture. Editable by user.

lastName

public String **lastName**
User's last name. Editable by user, and may be pseudonymous.

role

public int **role**
TODO: What does a user's role represent? This value seems to be `0` for all users. Editable by user.

username

public String **username**
User's display name that is publicly used to identify their posts, events, etc. Editable by user. Must be unique across all of CultureMesh's users.

Constructors

User

public **User** (long *id*, String *firstName*, String *lastName*, String *username*, String *imgURL*, String *aboutMe*, String *gender*)
Create a new object, storing the provided parameters into the related instance fields.

Parameters

- **id** – Uniquely identifies user across all of CultureMesh.
- **firstName** – User's first name (may be pseudonymous)
- **lastName** – User's last name (may be pseudonymous)
- **username** – The user's "display name" that will serve as their main public identifier. Must be unique across all of CultureMesh's users.

- **imgURL** – URL suffix (after *User.IMG_URL_PREFIX* to the user's profile picture
- **aboutMe** – Short bio describing the user
- **gender** – User's self-identified gender

User

public **User**(long *id*, String *firstName*, String *lastName*, String *username*)

Create a new object, storing the provided parameters into the related instance fields. Intended to be used when creating accounts, as `img_url`, `about_me`, and `gender` are initialized to defaults as described in the constants for *User*.

Parameters

- **id** – Uniquely identifies user across all of CultureMesh.
- **firstName** – User's first name (may be pseudonymous)
- **lastName** – User's last name (may be pseudonymous)
- **username** – The user's "display name" that will serve as their main public identifier. Must be unique across all of CultureMesh's users.

User

public **User**(JSONObject *res*)

Create a new user from a JSON that conforms to the following format:

```
{
           "id": 0,
           "username": "string",
           "first_name": "string",
           "last_name": "string",
           "role": 0,
           "gender": "string",
           "about_me": "string",
           "img_link": "string",
          }
```

Other key-value pairs are acceptable, but will be ignored. Note that `img_link` does not include the base *User.IMG_URL_PREFIX*. A missing, null, or empty `img_link` is interpreted as an unset link, which *User.CM_LOGO_URL* is used for.

Parameters

- **res** – JSON describing the user to create

Throws

- **JSONException** – May be thrown in the case of an improperly structured JSON

User

public **User**()

Empty constructor that does no initialization. For database use only.

Methods

getBio

public String **getBio**()
Get the user's self-written bio (i.e. "about me" text)

Returns User's description of themselves (i.e. their bio)

getFirstName

public String **getFirstName**()
Get the user's first name. May be pseudonymous.

Returns User's potentially pseudonymous first name.

getImgURL

public String **getImgURL**()
Get the URL to the user's profile photo

Returns URL that links to the user's profile photo

getLastName

public String **getLastName**()
Get the user's last name. May be pseudonymous.

Returns User's potentially pseudonymous last name.

getPostJson

public JSONObject **getPostJson**(String *email*, String *password*)
Create a JSON representation of the object that conforms to the following format:

```
{
            "username": "string",
            "password": "string",
            "first_name": "string",
            "last_name": "string",
            "email": "string",
            "role": 0,
            "img_link": "string",
            "about_me": "string",
            "gender": "string"
        }
```

This is intended to be the format used by the `/user/users` POST endpoint. Note that `img_link` does not include the base *`User.IMG_URL_PREFIX`*. A missing, null, or empty `img_link` is interpreted as an unset link, which *`User.CM_LOGO_URL`* is used for.

Throws

- **JSONException** – Unclear when this would be thrown

Returns JSON representation of the object

getPutJson

public JSONObject **getPutJson** (String *email*)

Create a JSON representation of the object that conforms to the following format:

```
{
          "id": 0,
          "username": "string",
          "first_name": "string",
          "last_name": "string",
          "email": "string",
          "role": 0,
          "gender": "string",
          "about_me": "string",
          "img_link": "string"
        }
```

This is intended to be the format used by the `/user/users` PUT endpoint. Note that `img_link` does not include the base *`User.IMG_URL_PREFIX`*. A missing, null, or empty `img_link` is interpreted as an unset link, which *`User.CM_LOGO_URL`* is used for.

Throws

- **JSONException** – Unclear when this would be thrown

Returns JSON representation of the object

getUsername

public String **getUsername** ()

Get the user's chosen display name, which should be used as their unique public identifier.

Returns User's display name, which must be unique across all of CultureMesh's users.

setBio

public void **setBio** (String *bio*)

Set the text of the user's bio

Parameters

- **bio** – New bio the user has chosen for themselves

setFirstName

public void **setFirstName** (String *firstName*)

Set the user's first name

Parameters

- **firstName** – New name to save as the user's first name

setImgURL

public void **setImgURL**(String *imgURL*)
Set the URL for the user's profile photo

Parameters

- **imgURL** – URL to the user's new profile photo

setLastName

public void **setLastName**(String *lastName*)
Set the user's last name

Parameters

- **lastName** – New name to save as the user's last name

setUsername

public void **setUsername**(String *username*)
Set the user's display name, which must be unique across CultureMesh

Parameters

- **username** – New display name to use for the user. Must be unique across all of CultureMesh's users.

1.4 Contributing

Thank you for your interest in contributing to CultureMesh Android! Here are a few steps to get you up and running:

1. Follow the instructions in *Documentation for CultureMesh Android* to get set up with the code base and Android Studio.
2. Open an issue on GitHub describing the changes you'd like to make. This is important beacuse your idea might already be in development or might not match the direction we are planning to take the app in. Reaching out to describe your proposal first will help avoid unnecessary work. You should also offer to work on it so people know not to do it themselves.
3. If your idea is accepted, start working on your idea! You might need to ask for suggestions or discuss implementation details in the issue first.
4. If you don't have commit access, you'll need to fork the repository and then clone your copy instead of the main fork.
5. Create a new branch for your changes:

```
$ git checkout -b your_branch_name
```

6. Make your changes. Please divide up your work into chunks, each of which could be un-done without breaking the app's functionality. Make each chunk a commit. Please include comments and documentation updates as needed in your changes, preferably in the commit which necessitated them. The commit message should follow the below style (inspired by Pro-Git, page 127):

```
Summary on one line and under 70 characters

After a blank line, you can have paragraphs as needed to more
fully detail your changes. Wrap them at ~72 lines (no more than
80) for people viewing it from a command line interface.

Separate paragraphs with a single line.
  - For bullet points, use hyphens or asterisks
  - You don't need a blank line between bullet points, but you
    should indent multiple lines to create a block of text.

In your message, describe both what you changed at a high level
and, more importantly, why you changed it. The rationale is
important to include because it might not be clear from your
code changes alone.
```

7. Push your changes:

```
$ git push --set-upstream origin your_branch_name
```

8. Create a pull request describing your changes and why they were made. Use the `develop` branch as the base for your pull request.
9. Before your pull request can be accepted, it must be reviewed. Your reviewer may suggest changes, which you should then make or explain why they aren't needed. This is a way to create dialogue about changes, which generally enhances code quality.
10. Once your pull request is accepted, you can delete your branch.

Warning: There are currently no automated tests for this project. Unfortunately, this means you will have to test manually to ensure your changes don't break anything.

CHAPTER 2

Getting Started

2.1 Getting the Latest Code

1. Clone the GitHub repository to your local machine:

```
$ git clone https://github.com/DrewGregory/CultureMeshAndroid
```

2. Switch to your desired branch. This will probably be either `master`, which holds the most recent release, or `develop`, which holds the current development version. For example:

```
$ git checkout master
```

2.2 Missing Information

For security reasons, some information is missing from the code repository:

- CultureMesh API Key: Stored in *`Credentials`*. The `Credentials.java` file must be created with the key in a public field `APIKey`.
- Fabric API Key and Secret: Stored in `app/fabric.properties`. See template below for the structure:

```
apiSecret=<API Secret>
apiKey=<API Key>
```

  Fill in `<API Secret>` and `<API Key>` with the appropriate values.

2.3 Running the App

Open the root of the repository in Android Studio. Let Android Studio index the repository, and let Gradle install dependencies. Then run the app by clicking the play button in the upper right. You may have to disable `Instant Run` in order to successfully use `Fabric` with the API key in `app/fabric.properties`.

CHAPTER 3

Indices and Tables

- genindex
- search

Index

H

I

J

L

M

N

O

P

Q

R

S

T