Video.java

*/\*\**

*\* Video represents a YouTube video loaded from a Json query*

*\* Used to retrieve information needed to play and display the YouTube video*

*\*/*

**public class** Video **extends** DatabaseObject {

**private** JsonObject **fields**;

**public** Video (JsonObject json) {

**super**(json);

**fields** = json;

}

**private** JsonObject getSnippet() {

**return fields**.get(Youtube.***JSON\_SNIPPET***).getAsJsonObject();

}

*// The title of the video*

**public** String getTitle() {

**return** getSnippet().get(Youtube.***JSON\_TITLE***).getAsString();

}

*// The description of the video*

**public** String getDescription() {

**return** getSnippet().get(Youtube.***JSON\_DESCRIPTION***).getAsString();

}

*// The video id of the video*

**public** String getId() {

JsonObject resourceId;

String id = **""**;

**if** (getSnippet().get(Youtube.***JSON\_RESOURCE***) != **null**) {

resourceId = getSnippet().get(Youtube.***JSON\_RESOURCE***).getAsJsonObject();

id = resourceId.get(Youtube.***JSON\_VIDEO\_ID***).getAsString();

} **else if** (**fields**.get(Youtube.***JSON\_ID***) != **null**) {

resourceId = **fields**.get(Youtube.***JSON\_ID***).getAsJsonObject();

id = resourceId.get(Youtube.***JSON\_VIDEO\_ID***).getAsString();

}

**return** id;

}

*// The date the video was uploaded*

**public** String getPublishDate() {

**return** getSnippet().get(Youtube.***JSON\_DATE***).getAsString();

}

*// How long ago the video was created (ex: 1 day ago)*

**public** String getAge() {

**int** videoAgeInDays;

**try** {

DateFormat dateFormat = **new** SimpleDateFormat(Database.***ISO\_FORMAT***);

Date published = dateFormat.parse(getPublishDate());

Date current = **new** Date();

videoAgeInDays = (**int**)((current.getTime() - published.getTime()) / (1000\*60\*60\*24l));

} **catch** (Exception e) {

videoAgeInDays = 0;

}

**return** videoAgeInDays + **" days ago"**;

}

*// The image url for the default thumbnail*

**public** String getThumbnailUrl() {

JsonObject imageData = getSnippet().getAsJsonObject(Youtube.***JSON\_THUMBNAIL***);

JsonObject defaultImage = imageData.getAsJsonObject(Youtube.***JSON\_THUMBNAIL\_DEFAULT***);

**return** defaultImage.get(Youtube.***JSON\_THUMBNAIL\_URL***).getAsString();

}

}

Playlist.java

*/\*\**

*\* Created by Kayla on 2/15/2016.*

*\*/*

**public class** Playlist **extends** DatabaseObject {

**private** String **baseUrl**; *// base url to load videos from*

**private** JsonObject **fields**; *// holds the fields*

**private** String **nextPage**; *// holds the next page token*

**private** List<Video> **videoList**;

**private** Content<Video> **videoContent**;

**public** Playlist(JsonObject jsonObject, String url) {

**super**(jsonObject);

**baseUrl** = url;

**fields** = jsonObject;

**nextPage** = getNextPage();

JsonArray videos = **fields**.get(Youtube.***JSON\_LIST***).getAsJsonArray();

**videoList** = **new** ArrayList<>();

**for** (**int** i = 0; i < videos.size(); i++) {

**videoList**.add(**new** Video(videos.get(i).getAsJsonObject()));

}

**videoContent** = **new** Content<>(**videoList**, ContentType.***LIVE***);

}

**public** List<Video> getVideoList() {

**return videoList**;

}

**public** Content<Video> getVideoContent() {

**return videoContent**;

}

**public** String getChannelId() {

JsonArray videos = **fields**.get(Youtube.***JSON\_LIST***).getAsJsonArray();

JsonObject video = videos.get(0).getAsJsonObject();

JsonObject snippet = video.get(Youtube.***JSON\_SNIPPET***).getAsJsonObject();

**return** snippet.get(Youtube.***JSON\_CHANNEL\_ID***).getAsString();

}

**public void** loadMore() {

**new** UpdatePlaylist().execute();

}

**private void** loadNextPage() {

**if** (**nextPage** != **""**) {

String query = **baseUrl**;

String page = Youtube.***QUERY\_PAGE*** + **nextPage**;

Playlist newPlaylist = RestUtil.*getPlaylist*(query + page);

**if** (newPlaylist != **null**) {

**fields** = newPlaylist.**fields**;

**nextPage** = newPlaylist.getNextPage();

**videoList**.addAll(newPlaylist.getVideoList());

**videoContent** = **new** Content<>(**videoList**, ContentType.***LIVE***);

}

}

}

**private** String getNextPage() {

**if** (**fields**.get(Youtube.***JSON\_NEXTPAGE***) != **null**)

**return fields**.get(Youtube.***JSON\_NEXTPAGE***).getAsString();

**return ""**;

}

*/\*\**

*\* Update the videos in the existing playlist by loading more*

*\*/*

**private class** UpdatePlaylist **extends** AsyncTask<Void, Void, Void> {

@Override

**protected** Void doInBackground(Void... voids) {

loadNextPage();

**return null**;

}

}

}

Youtube.java

**public class** Youtube {

*// YOUTUBE*

**public static final** String ***MINRESULTS*** = **"10"**;

**public static final** String ***MAXRESULTS*** = **"50"**;

**public static final** String ***VIDEO\_URL*** = **"https://www.youtube.com/watch?v="**;

*// JSON KEYS*

**public static final** String ***JSON\_LIST*** = **"items"**;

**public static final** String ***JSON\_CONTENT\_DETAILS*** = **"contentDetails"**;

**public static final** String ***JSON\_RELATED\_PLAYLISTS*** = **"relatedPlaylists"**;

**public static final** String ***JSON\_UPLOADS*** = **"uploads"**;

**public static final** String ***JSON\_SNIPPET*** = **"snippet"**;

**public static final** String ***JSON\_CHANNEL\_ID*** = **"channelId"**;

**public static final** String ***JSON\_DATE*** = **"publishedAt"**;

**public static final** String ***JSON\_TITLE*** = **"title"**;

**public static final** String ***JSON\_DESCRIPTION*** = **"description"**;

**public static final** String ***JSON\_RESOURCE*** = **"resourceId"**;

**public static final** String ***JSON\_ID*** = **"id"**;

**public static final** String ***JSON\_VIDEO\_ID*** = **"videoId"**;

**public static final** String ***JSON\_THUMBNAIL*** = **"thumbnails"**;

**public static final** String ***JSON\_THUMBNAIL\_DEFAULT*** = **"default"**;

**public static final** String ***JSON\_THUMBNAIL\_URL*** = **"url"**;

**public static final** String ***JSON\_NEXTPAGE*** = **"nextPageToken"**;

**public static final** String ***JSON\_PREVPAGE*** = **"prevPageToken"**;

*// JSON QUERIES*

**public static final** String ***QUERY*** = **"https://www.googleapis.com/youtube/v3/"**;

**public static final** String ***QUERY\_CHANNEL*** = **"channels?part=contentDetails"**;

**public static final** String ***QUERY\_PLAYLIST*** = **"playlistItems?part=snippet"**;

**public static final** String ***QUERY\_PLAYLIST\_ID*** = **"&playlistId="**;

**public static final** String ***QUERY\_USERNAME*** = **"&forUsername="**;

**public static final** String ***QUERY\_MAXRESULTS*** = **"&maxResults="**;

**public static final** String ***QUERY\_KEY*** = **"&key=AIzaSyDx-YHJnv8FcLKlzgpD99ZpfLDjICRRFNI"**;

**public static final** String ***QUERY\_PAGE*** = **"&pageToken="**;

*// Completed example queries*

**public static final** String ***QUERY\_SLOCRUSADE\_CHANNEL*** =

**"https://www.googleapis.com/youtube/v3/"** +

**"channels?part=contentDetails"** +

**"&maxResults=10"** +

**"&key=AIzaSyDx-YHJnv8FcLKlzgpD99ZpfLDjICRRFNI"** +

**"&forUsername=slocrusade"**;

**public static final** String ***QUERY\_SLOCRUSADE\_UPLOADS*** =

**"https://www.googleapis.com/youtube/v3/"** +

**"playlistItems?part=snippet"** +

**"&maxResults=10"** +

**"&key=AIzaSyDx-YHJnv8FcLKlzgpD99ZpfLDjICRRFNI"** +

**"&playlistId=UUe-RJ-3Q3tUqJciItiZmjdg"**;

**public static final** String ***QUERY\_SLOCRUSADE\_SEARCH*** =

**"https://www.googleapis.com/youtube/v3/"** +

**"search?part=snippet"** +

**"&key=AIzaSyDx-YHJnv8FcLKlzgpD99ZpfLDjICRRFNI"** +

**"&maxResults=10"** +

**"&channelId=UCe-RJ-3Q3tUqJciItiZmjdg"** +

**"&q="**;

RestUtil.java

*/\*\**

*\* Gets a playlist given its url query*

*\*/*

**public static** Playlist getPlaylist(String url) {

**try** {

JsonParser parser = **new** JsonParser();

HttpURLConnection conn = *createGetConnection*(url, **""**);

String toParse = *request*(conn);

**if** (toParse.equals(**"!error"**)) {

**return null**;

}

**else** {

JsonElement videoElement = parser.parse(toParse);

**return new** Playlist(videoElement.getAsJsonObject(), url);

}

} **catch** (Exception e) {

**return null**;

}

}

*/\*\**

*\* Gets the uploads playlist from the given user's Youtube account*

*\*/*

**public static** Playlist getPlaylistFromUser(String username) {

**try** {

String channelQuery = Youtube.***QUERY*** + Youtube.***QUERY\_CHANNEL***;

String userQuery = Youtube.***QUERY\_USERNAME*** + username;

String uploadsId = *getUploadsId*(channelQuery + userQuery + Youtube.***QUERY\_KEY***);

String playlistQuery = Youtube.***QUERY*** + Youtube.***QUERY\_PLAYLIST***;

String uploadsQuery = Youtube.***QUERY\_PLAYLIST\_ID*** + uploadsId;

**return** *getPlaylist*(playlistQuery + uploadsQuery + Youtube.***QUERY\_KEY***);

}

**catch** (Exception e) {

**return null**;

}

}

*/\*\**

*\* Gets the id of the uploads playlist given a channel query*

*\*/*

**private static** String getUploadsId(String url) **throws** Exception {

JsonParser parser = **new** JsonParser();

HttpURLConnection conn = *createGetConnection*(url, **""**);

String toParse = *request*(conn);

**if** (toParse.equals(**"!error"**)) {

**return ""**;

} **else** {

JsonObject channel = parser.parse(toParse).getAsJsonObject();

JsonArray items = channel.get(Youtube.***JSON\_LIST***).getAsJsonArray();

JsonObject content = items.get(0).getAsJsonObject();

JsonObject contentDetails = content.get(Youtube.***JSON\_CONTENT\_DETAILS***).getAsJsonObject();

JsonObject playlists = contentDetails.get(Youtube.***JSON\_RELATED\_PLAYLISTS***).getAsJsonObject();

String uploads = playlists.get(Youtube.***JSON\_UPLOADS***).getAsString();

**return** uploads;

}

}

ViewVideosFragment.java

*/\*\**

*\* Created by Kayla on 2/11/2016.*

*\*/*

**public class** ViewVideosFragment **extends** CruFragment **implements** TextView.OnEditorActionListener {

@Override

**public** View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

View fragmentView = **super**.onCreateView(inflater, container, savedInstanceState);

setHasOptionsMenu(**true**);

**new** VideoTask().execute();

**return** fragmentView;

}

@Override

**public void** onCreateOptionsMenu(Menu menu, MenuInflater inflater) {

menu.clear();

inflater.inflate(R.menu.***options\_menu***, menu);

menu.findItem(R.id.***search***).setActionView(R.layout.***action\_search***);

EditText search = (EditText)menu.findItem(R.id.***search***).getActionView().findViewById(R.id.***text***);

search.setOnEditorActionListener(**this**);

search.setImeActionLabel(**"Search"**, KeyEvent.***KEYCODE\_ENTER***);

}

@Override

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

**if** (item.getItemId() == R.id.***search***) {

**return true**;

}

**return super**.onOptionsItemSelected(item);

}

@Override

**public boolean** onEditorAction(TextView v, **int** actionId, KeyEvent event) {

**if** (event == **null** || event.getAction() == KeyEvent.***KEYCODE\_ENTER***) {

*//****TODO: make this work again***

*//Playlist filtered = RestUtil.getPlaylist(Youtube.QUERY\_SLOCRUSADE\_SEARCH + v.getText());*

*//new DisplayVideosTask().execute(filtered);*

}

**return true**;

}

*/\*\**

*\* Fetches and displays all of the ministries' videos*

*\*/*

**private class** VideoTask **extends** AsyncTask<Void, Void, Void> {

**private** List<Playlist> **playlists**;

**public** VideoTask() {

**super**();

**playlists** = **new** ArrayList<>();

}

@Override

**protected** Void doInBackground(Void... params) {

*//****TODO: once ministry DB has YT accounts in it, get from there***

**playlists**.add(RestUtil.*getPlaylistFromUser*(**"slocrusade"**));

**return null**;

}

@Override

**protected void** onPostExecute(Void aVoid) {

**super**.onPostExecute(aVoid);

**new** DisplayVideosTask().execute(**playlists**.toArray(**new** Playlist[**playlists**.size()]));

}

}

*/\*\**

*\* Loads a list of videos from the Cru YT channel and displays them in cards*

*\*/*

**private class** DisplayVideosTask **extends** AsyncTask<Playlist, Void, Void> {

**private** Playlist **videoPlaylist**;

**private** List<Video> **videos**;

**private** MainActivity **currentActivity**;

**private** CardFragmentFactory **cardFactory**;

**private** ListView **list**;

*// used for scrolling pagination*

**private int firstItem**, **visibleItems**, **totalItems**;

**public** DisplayVideosTask() {

**super**();

**currentActivity** = (MainActivity) MainActivity.*context*;

**cardFactory** = **new** VideoCardFactory();

}

@Override

**protected** Void doInBackground(Playlist... playlists) {

*//****TODO: make this work with multiple playlists***

**videoPlaylist** = playlists[0];

**videos** = **videoPlaylist**.getVideoList();

**return null**;

}

@Override

**protected void** onPostExecute(Void aVoid) {

**super**.onPostExecute(aVoid);

**list** = (ListView) **currentActivity**.findViewById(R.id.***list\_cards***);

**if** ((**videos** != **null**) && (!**videos**.isEmpty())) {

**final** Content<Video> videoContent = **videoPlaylist**.getVideoContent();

**list**.setAdapter(**cardFactory**.createAdapter(videoContent));

**list**.setOnItemClickListener(**cardFactory**.createCardListener(**currentActivity**, videoContent));

**list**.setOnScrollListener(**new** AbsListView.OnScrollListener() {

@Override

**public void** onScroll(AbsListView view,

**int** first, **int** visible, **int** total) {

**firstItem** = first;

**visibleItems** = visible;

**totalItems** = total;

}

@Override

**public void** onScrollStateChanged(AbsListView view, **int** scrollState) {

**final int** lastItem = **firstItem** + **visibleItems**;

**if** (lastItem == **totalItems** && scrollState == ***SCROLL\_STATE\_IDLE***) {

**videoPlaylist**.loadMore();

**final** Content<Video> videoContent = **videoPlaylist**.getVideoContent();

**list**.setAdapter(**cardFactory**.createAdapter(videoContent));

**list**.setOnItemClickListener(**cardFactory**.createCardListener(**currentActivity**, videoContent));

**list**.setSelection(**firstItem**);

}

}

});

} **else** {

String errorMessage = Util.*getString*(R.string.***toast\_no\_videos***);

Toast.*makeText*(**currentActivity**.getApplicationContext(), errorMessage, Toast.***LENGTH\_LONG***).show();

}

}

}

}