ITIS/ITCS 4180/5180 Mobile Application Development
Homework 5
Date Posted: 03/16/2012 at 4:00pm
Due Date: 03/30/2012 at 5:00pm

Basic Instructions:

1. In every file submitted you MUST place the following comments:
   a) Assignment #
   b) File Name.
   c) Full name of all students in your group.
2. Each group should submit only one assignment. Only the group leader is supposed to submit the assignment on behalf of all the other group members.
3. Please download the support files provided with this assignment and use them when implementing your project.
4. **Export your Android project as follows**:
   a) From eclipse, choose "*Export…*" from the *File* menu.
   b) From the Export window, choose *General* then *File System.* Click *Next.*
   c) Make sure that your Android project for this assignment is selected. Make sure that all of its subfolders are also selected.
   d) Choose the location you want to save the exported project directory to. For example, your *Desktop* or *Documents* folder.
   e) The first time you export, make sure you select *Create directory structure for files*.
   f) Click Finish, and then go to the directory you exported the project to. Make sure the exported directory contains all necessary files, such as the `.java` and resource files.
5. Submission details:
   a) When you submit the assignment, compress your exported Android project into a single zip file. The format of compressed file name is:
      i. HW#.zip
   b) You should submit the assignment through Moodle:
      i. Submit the zip file.
6. **Failure to follow the above instructions will result in point deductions.**

Homework 5 (200 Points)

In this assignment you will develop a simple Digg client application (see **http://digg.com**). The client will be able to load Digg topics and the latest diggs for each topic. Users of the client will also be able to add/remove diggs to their local Favorites. Favorites will be stored into a SQLite database. For this homework you will also be using a JSON parser as data loaded from Digg is in the JSON format.

Notes: All API calls and parsing should happen through threads and should not block the main thread. All string values should be used via strings.xml. For further dig API documentation visit **http://developers.digg.com/documentation**.

Part A: Loading and Parsing Digg Topics and Items (50 Points)

Topics: To load the list of Digg topics, you should call the **topic.getAll** API. The end point for this API is **http://services.digg.com/2.0/topic.getAll**. An excerpt from an example JSON return value looks as follows:

```
"topics": [
      {
       "name": "Business",
       "short_name": "business"
      },
      {
       "name": "Entertainment",
       "short_name": "entertainment"
      },
.
.
.
    ],
```

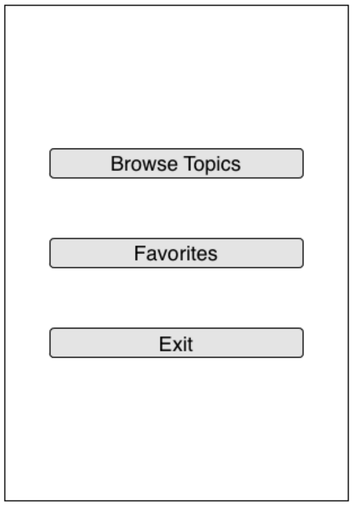
When parsing the JSON, we are interested in the topics array, as it contains the list of topic names. You should define a **DiggTopic** class that has a topic **name** and **short_name**. As you parse each topic, you will create a corresponding **DiggTopic** object. To keep track of the list of topics, you should store **DiggTopic** objects into a List. This list will be used to populate a ListView later on.

Digg Items: For each topic, you will be able to load a list of digged items. To do so, use the digg.getAll API at http://services.digg.com/2.0/digg.getAll. To get the items for a specific topic you must pass a **topic** argument using GET. The value for this argument must be the **short_name** of a topic (previously returned). The return value of this API will include a list of diggs in JSON format. An excerpt from a return value for this API looks as follows:

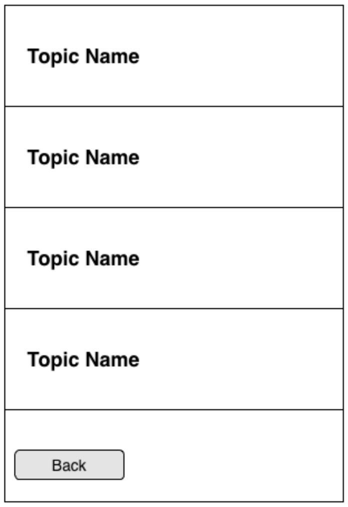
```
"diggs": [
        {
            "date": 1331871378,
            "item": {
                "status": "upcoming",
                "container": {
                    "name": "Business",
                    "short_name": "business"
                },
                "description": "Check out this post BMI Calculator for ..",
                "title": "BMI Calculator for Women, Accurate Measure of Healthy Body Weight",
                "submit_date": 1331871378,
                "media": 0,
                "diggs": 1,
                "comments": 0,
                "topic": {
                    "name": "Business",
                    "short_name": "business"
                },
                "shorturl": {
                    "short_url": "http://digg.com/news/business/bmi_calculator",
                    "view_count": 0
                },
                "promote_date": null,
                "link": "http://onlywire.com/x/74335708",
                "href": "http://digg.com/news/business/bmi_calculator",
                "id": "20120316041618:36d05dd3-e6ba-45eb-b7d6-9912d8d03812"
            },
            "user": {
                ...
            },
            "digg_id": "20120316041619:6fafc971-76b2-41b4-9e54-d41d975238ff"
        },
.
.
.
```

Notice the **diggs** array which contains all retuned dig items. The excerpt also shows the first dig item JSON. For this homework we are interested in a digg's **digg_id**, **title**, **description**, **link**, **diggs**, and **date**. You should define a **DiggItem** class to hold these values. You should also have a boolean within the class called **isFavorite** to indicate whether an item has been favorited or not, in addition to a Date to indicate when an item was marked as favorite (**dateFavorited**). When parsing the JSON, you will create a **DiggItem** object associated with each parsed item. The **DiggItem** objects should then be stored within a List.

Part B: Activities (100 Points)



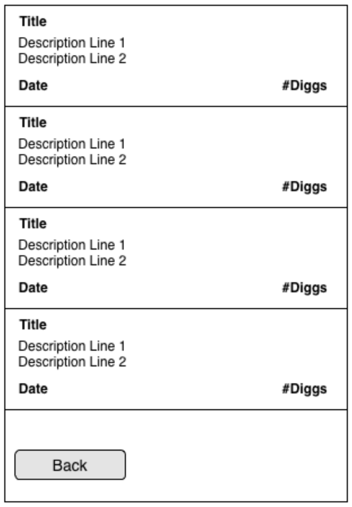
StartActivity: In this activity you will include 3 buttons. A "Browse Topics", "Favorites", and an "Exit button". The "Browse Topics" should bring up the **TopicsActivity**. The "Favorites" button will load the **ItemsActivity** in *favorites* mode.



TopicsActivity: This activity should perform the following tasks:

1- Initiate the call to the **topic.getAll** API. It is recommended to have the logic for calling the API and parsing the return values within a separate controller (e.g. TopicController class). While loading and parsing the topics, you should show a ProgressDialog to the user. This gives the user the notion of progress rather than showing an idle and empty screen.
2- Once the list of topics (**DiggTopic** objects) is loaded, the activity should populate a ListView with the topic **name** for each **DiggTopic** object within the list.
3- When selecting a certain item within the ListView, you should bring up the **ItemsActivity** and pass along the topic for the selected item (**short_name**). You should also set the mode of the **ItemsActivity** to *normal*.

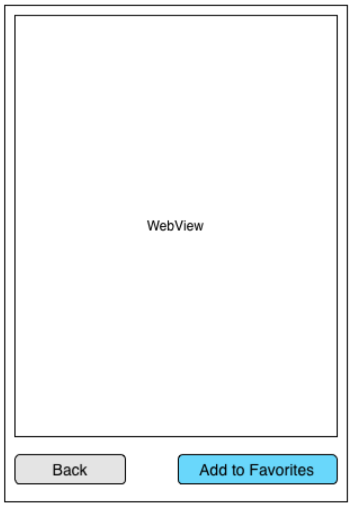
The activity should also include a "Back" button to take the user back to the **StartActivity**.



ItemsActivity: There are 2 modes for this activity: *normal* mode and *favorites* mode.

1- In *normal* mode, the activity should initiate the loading and parsing of a topic's digg items using the API **dig.getAll**. It should call this API with the passed topic **short_name**. While loading and parsing show a ProgressDialog to the user. Once the list of digg items is loaded, you should populate a ListView with the contents of this list.
2- In *favorites* mode, you should load the list of digg items from a SQLite database that has the user's favorite diggs. Once loaded, you will populate the ListView within the activity with these diggs. You should use the **dateFavorited** attribute of a **DiggItem** to sort the list by most recently favorited.

In both modes, you will be using a custom view for the ListView items. To do so, you will need to extend the **ArrayAdapter** class and override the **getView** method. The **getView** method should return a view that incorporates the following elements: TextViews for a **DiggItem**'s **title**, **description**, **date**, and **diggs** (This is the number of times the item was digged by other users). See the figure for the appropriate layout. You need only show 2 lines of an item's description. The date should be properly formatted to a human readable format (e.g. mm/dd/yyyy).
Clicking on a ListView item should bring up the **WebViewActivity** and pass along the appropriate **DiggItem** object (Remember **Parcelable**). Note that you can get the associated **DiggItem** using the position from the **onItemClick** method.
Finally the activity should have a "Back" button.



__WebViewActivity:__ This activity will be passed a **DiggItem** when created. The **DiggItem** will either be a favorite or not. The activity will incorporate a **WebView**, and a "Add to Favorites" (If **DiggItem** is not already a favorite) or "Remove from Favorites" button (If item is a favorite). When the activity first loads, it should load the **DiggItem**'s **link** into the WebView, hence showing the webpage associated with the link.
Clicking on the "Add to Favorites" button will add the **DiggItem** to the database. If the **DiggItem** is already a favorite then clicking on the "Remove from Favorites" will remove it from the database. In both cases you should appropriately update the **isFavorite** attribute of the item. Notice that the "Add to Favorites" and "Remove from Favorites" buttons are actually one button where you change its text according to the **isFavorite** value of a **DiggItem**.

Note: It is recommended to load the **WebViewActivity** via the **startActivityForResult** method. This allows you to report back to the initiating activity. You would need to do so if you remove a **DiggItem** from the favorites, so you can inform the **ItemsActivity** that it should reload its ListView (this only applies in *favorites* mode).

__Part C: SQLite (50 Points)__

A user's favorites should be stored within a SQLite database. Follow the example provided in the lecture (Instead of using a Note object you would use **DiggItem**), and design the required schema. Note that only favorited items are stored, hence you do not need a table column for an item's **isFavorite** attribute.

Bonus:

1- (25 Points) Some returned diggs will have a thumbnail value within the JSON. Using the thumbnail src, add an ImageView to the View items within the ListView of the **ItemsActivity**. The ImageView should show up to the far right within a ListView item. To get credit for this bonus, you must load the Image using threads so you don't block the main thread.

2- (25 Points) Within the **ItemsActivity**, add a "remove" button to the View items of the ListView so users can easily remove favorites without going to the **WebViewActivity**. When removing an item, make sure to reload the ListView to reflect the latest favorites. Note this only applies to the **ItemsActivity** in *favorites* mode.