MIT AITI
Android Software Development

Lab 2: Navigating between Screens and Accessing the Web

In this lab, you will get hands-on experience with navigating between screens as well as with accessing the web. Your task is to create an app that displays information about the states of India. Follow the steps below to get started.

Setup:
You will need the following software installed.
1. Eclipse
2. Android SDK
3. ADT plug-in (to use the Android SDK in Eclipse)
4. EGit plug-in (to use Git through Eclipse)

Steps:
1. Get Lab 2 from bitbucket.org
    a. In Eclipse, go to File -> Import -> Git -> Projects from Git -> URI
    b. Type https://bitbucket.org/aiti_india/lab2.git into the URI field
    c. Select the master branch
    d. Select Finish. You should now see a new Android project called Lab 2 in your Eclipse workspace.
2. Display a list of Indian states from the web
    a. Run Lab 2 as it is before you have made any changes. Once the virtual device starts, you should see a screen with the title "Indian States". The rest of the screen is empty. You will be modifying this screen, so that it gets a list of Indian states from the web and displays them.
    b. Find the file ListStatesActivity.java in the src folder. This is the file you will make changes to.
    c. The list of states is located at the following URL: http://telegraphis.net/demoapps/aiti_india/india.json. The data at this URL is stored as a JSON array of JSON objects taking the form:
    [{"name":"name of State 1","url":"url of State 1"},{"name":"name of State 2","url":"url of State 2"},…]
    d. Inside the method "onCreate", write some code that will access this URL and store its data somewhere. See the notes on JSON (attached) to help you figure out how best to read the input stream data.
    e. Display a list of only the state names (not the urls). We have already setup the layout of ListStatesActivity.java to contain a ListView. Make an ArrayAdapter to display the list of state names in the ListView.
    f. Run Lab 2 again. You should be able to see a list of Indian states on the screen. *NOTE:* If you are on the IITB campus behind the proxy, you may need to set the proxy settings IN the Android emulator. See http://stackoverflow.com/questions/28380/proxy-which-requires-authentication-with-android-emulator for more details. If the options there do not work, let us know and we will see what we can do to make sure that the internet works in your emulator.
3. Get ready to display detailed information about a state.
    a. Find the file StateDetailActivity.java in the src folder. This is the file you will make changes to.

   b. Get the text fields (already added to the activity_state_detail.xml file) where you will put the state name, capital, languages, etc.
   c. Create variables that will store the data for name, capital, languages, … and make sure that the text fields will be set from the variables
   d. At this point, this activity should be ready to display detailed information about any state. It does not have any actual information, only placeholders for the information.

4. Allow users to click on a state on the main screen to learn more details about that state. This means that you will have to navigate from ListStatesActivity.java to StateDetailActivity.java.
   a. In ListStatesActivity.java, you should have a list of state names as well as a list of URLs for each state. These URLs link to the details about each state.
   b. Add an ItemClickListener to the ListView of states in ListStatesActivity.java in the "onCreate" method (see setItemClickListener() method of the ListView).
   c. Inside the onItemClick() method of the ItemClickListener, send an Intent to start the StateDetailActivity and pass it the correct URL based on the state that was clicked. (You can pass the URL as the data of the intent by using "intent.setData(Uri.parse(theURIString))")
   d. Now move to the StateDetailActivity.java file. Inside the onCreate method, write code to access the data from the URL that was passed in (you can get this by calling "getIntent()" to get the Intent which started the activity, and then using "getData()" to get the Uri you set as the data in ListStatesActivity.java). Load this URI as a URL (You can get the URL back by creating a new URL(uri.toString())).
   Parse this data (see notes on reading JSON input). Assign the appropriate information to the variables you created earlier for name, capital, languages, population, and area. Ignore the map information for now. The state detail data is stored as a JSON object that looks like this:
   {"name":"Goa","capital":"Panaji","languages":["Konkani"],"population":1457723,"area":3702,"map":"http://telegraphis.net/demoapps/aiti_india/ga.png"}
   NOTE: The "languages" key takes a JSON array as its value! You will need to find out a way to concatenate the languages in that array into a single string!
   e. Now StateDetailActivity should be able to display actual information based on the URL that is passed to it.
   f. Run Lab 2 again. You should be able to click on any state in the main screen, which will take you to a new screen that displays details about that state!

5. Finally, you will add one more option, which is that users can choose to see a map of the state. The steps are very similar to what you have already accomplished.
   a. Go to activity_state_detail.xml in the res/menu/ directory. Add a menu item that will switch to the map view.
   b. Add the onOptionsItemSelected() method to StateDetailActivity.java so that when the map item is selected, an Intent will be made to start the StateMapActivity. Pass it the URL of the map just like how you passed the URL of the state data to StateDetailActivity. (The URL of the map was in the details of each state).
   c. Setup StateMapActivity, so that it can load and display image data from the URL that is passed to it. See the notes on how to display images (an ImageView is already in the activity_state_map.xml layout. Its id should be imageMap)
   d. Now you should be able to select the menu option in StateDetailActivity, which will take you to a new screen that shows the map of the state.

6. Your finished app should be able to display the states of India, provide details about each state, and display a map of each state. You have created a very simple tourism app!