Tidbits

# Loading static JSON file & Globalization & Alert Dialog

# Resources:

<https://developer.android.com/guide/topics/resources/localization>

<https://developer.android.com/training/basics/supporting-devices/languages.html>

<http://docwiki.embarcadero.com/RADStudio/Rio/en/Language_Culture_Names,_Codes,_and_ISO_Values>

<https://developer.android.com/guide/topics/ui/dialogs>

## Loading static JSON file

In my Android Development journey Google, GSON is favorite arsenal for converting JSON to Java objects and vice versa. Benefits to using GSON are:  
  
1. It’s very handy to use.  
2. Minimal code.  
3. Once you converted Java objects to JSON, you can pass it through intent bundle as a String.  
  
Here’s a step by step example of how GSON can help you parse JSON objects to a Java POJO.

Using an Empty Activity, create a new Android application named “Global Institute of Technology” with minimum SDK of 26.

01 build.gradle.txt

Add the following to *build.gradle (Module: app)*:

implementation 'com.google.code.gson:gson:2.8.2'

Make sure to click on “*Sync Now*” link on top-right corner face you edit *build.gradle (Module: app)* file.

## GSON PART:

Copy the image from snippets drawable folder into application’s drawable folder.

Create a directory under *main* folder named *assets.*

src > main > assets

Create a file named *students.json.*

*src* > *main* > *assets* > *students.json*

Add the following contents to *students.json*:

02 students.json.txt

{

"students": [

{ "id" : "1", "firstName": "Tom", "lastName": "Day", "school": "Physics" },

{ "id" : "2", "firstName": "Zoe", "lastName": "Cox", "school": "Ecology" },

{ "id" : "3", "firstName": "Nia", "lastName": "Coe", "school": "Finance" },

{ "id" : "4", "firstName": "Ann", "lastName": "Fox", "school": "Geology" },

{ "id" : "5", "firstName": "Bob", "lastName": "Doe", "school": "History" },

{ "id" : "6", "firstName": "Sue", "lastName": "Roy", "school": "Nursing" },

{ "id" : "7", "firstName": "Sam", "lastName": "Ray", "school": "Science" },

{ "id" : "8", "firstName": "Dan", "lastName": "Fay", "school": "Nursing" },

{ "id" : "9", "firstName": "Jan", "lastName": "Day", "school": "Russian" },

{ "id" : "10", "firstName": "Rob", "lastName": "Ash", "school": "Trading" },

{ "id" : "11", "firstName": "Cat", "lastName": "Mac", "school": "Science" },

{ "id" : "12", "firstName": "Pat", "lastName": "May", "school": "Sailing" },

{ "id" : "13", "firstName": "Kim", "lastName": "Lee", "school": "Nursing" },

{ "id" : "14", "firstName": "Jim", "lastName": "Rex", "school": "Farming" },

{ "id" : "15", "firstName": "Tim", "lastName": "Kam", "school": "Nursing" },

{ "id" : "16", "firstName": "Meg", "lastName": "Shy", "school": "Singing" },

{ "id" : "17", "firstName": "Mia", "lastName": "Hay", "school": "Physics" },

{ "id" : "18", "firstName": "Joe", "lastName": "Coe", "school": "Science" },

{ "id" : "19", "firstName": "Kam", "lastName": "Gil", "school": "History" },

{ "id" : "20", "firstName": "Liz", "lastName": "May", "school": "Physics" },

{ "id" : "21", "firstName": "Zak", "lastName": "Gad", "school": "Russian" },

{ "id" : "22", "firstName": "Art", "lastName": "Fay", "school": "Weather" },

{ "id" : "23", "firstName": "Mat", "lastName": "Day", "school": "Trading" },

{ "id" : "24", "firstName": "Deb", "lastName": "Ray", "school": "Finance" },

{ "id" : "25", "firstName": "Art", "lastName": "Ash", "school": "Ecology" }

]

}

Create a class named *Student.java* that serves as a JSON parser and maps data items in *students.json*. Add this code to *Student.java*:

03 Student.java.txt

public class Student {

@SerializedName("id")

@Expose

private int id;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

@SerializedName("firstName")

@Expose

private String firstName;

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

@SerializedName("lastName")

@Expose

private String lastName;

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

@SerializedName("school")

@Expose

private String school;

public String getSchool() {

return school;

}

public void setSchool(String school) {

this.school = school;

}

}

Create class name *BaseStudent.java* that serves as a JSON parser and uses the Google GSON library. Add this code to *BaseStudent.java*:

04 BaseStudent.java.txt

public class BaseStudent {

@SerializedName("students")

@Expose

private List<Student> students = new ArrayList<>();

public List<Student> getStudents() {

return students;

}

public void setStudents(List<Student> students) {

this.students = students;

}

}

05 activity\_main.xml.txt

## activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical" >

<ImageView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:scaleType="centerCrop"

android:src="@drawable/country\_flag" />

<ListView

android:id="@+id/listView"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" />

</LinearLayout>

## list\_item\_student.xml

This layout is for each list item in a *ListView*. Create a file named *list\_item\_student.xml* under *layout* with following content:

06 list\_item\_student.xml.txt

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="horizontal"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

<TextView

android:id="@+id/tvId"

android:layout\_width="0dp"

android:layout\_height="match\_parent"

android:layout\_weight=".1"

android:gravity="center"

android:textSize="15dp"

android:textColor="@color/colorAccent"/>

<TextView

android:id="@+id/tvFullName"

android:layout\_width="0dp"

android:layout\_height="match\_parent"

android:layout\_weight=".1"

android:gravity="center"

android:textSize="15dp"

android:textStyle="bold"/>

<TextView

android:id="@+id/tvSchool"

android:layout\_width="0dp"

android:layout\_height="match\_parent"

android:layout\_weight=".1"

android:gravity="center"

android:textSize="15dp"

android:textStyle="bold"/>

</LinearLayout>

Create a class named *StudentAdapter.java*. This class is responsible for inflating and binding data to list items in a *ListView*.

07 StudentAdapter.java.txt

public class StudentAdapter extends BaseAdapter {

private Context context;

private List<Student> dataList;

private TextView tvId;

private TextView tvFullName;

private TextView tvSchool;

private View vi;

private static LayoutInflater inflater = null;

public StudentAdapter(Context context, List<Student> dataList){

this.context = context;

this.dataList = dataList;

inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

}

@Override

public int getCount() { return dataList.size(); }

@Override

public Object getItem(int i) { return i; }

@Override

public long getItemId(int i) { return i; }

@Override

public View getView(int position, View view, ViewGroup viewGroup) {

//Populate the Listview

final int pos = position;

Student stu = dataList.get(pos);

View vi = inflater.inflate(R.layout.list\_item\_student, viewGroup, false);

tvId = vi.findViewById(R.id.tvId);

tvFullName = vi.findViewById(R.id.tvFullName);

tvSchool = vi.findViewById(R.id.tvSchool);

String full = String.format("%s %s", stu.getFirstName(), stu.getLastName());

tvId.setText(String.valueOf(stu.getId()));

tvFullName.setText(full);

tvSchool.setText(stu.getSchool());

return vi;

}

}

Add these instance variables to *MainActivity.java*:

private final static String JSON\_FILE = "students.json";

private StudentAdapter studentAdapter;

private ListView listView;

09 MainActivity methods.txt

Add these methods to *MainActivity.java*:

public void init() {

listView = findViewById(R.id.listView);

studentAdapter = new StudentAdapter(MainActivity.this, getStudentsData());

listView.setAdapter(studentAdapter);

}

/\* Convert JSON String to BaseStudent Model via GSON \*/

public List<Student> getStudentsData() {

String jsonString = getAssetsJSON(JSON\_FILE);

Log.d(TAG, "Json: " + jsonString);

Gson gson = new Gson();

BaseStudent baseStudent = gson.fromJson(jsonString, BaseStudent.class);

return baseStudent.getStudents();

}

/\* Get File in Assets Folder \*/

public String getAssetsJSON(String fileName) {

String json = null;

try {

InputStream inputStream = this.getAssets().open(fileName);

int size = inputStream.available();

byte[] buffer = new byte[size];

inputStream.read(buffer);

inputStream.close();

json = new String(buffer, "UTF-8");

} catch (IOException e) {

e.printStackTrace();

}

return json;

}

Append the following method call to the *onCreate()* method in *MainActivity.java:*

init();

If all goes well, this is what you will see when you run the application.

[](https://www.bing.com/images/search?view=detailV2&ccid=g6nYHgjm&id=9DCD800E35FF6162FE4C44ABC6C70A5783B46357&thid=OIP.g6nYHgjml5I7LyjheiVEGwElEs&q=run+logo&simid=608050058236200411&selectedIndex=67)

## Globalization

*Globalization* is the process of designing apps that support different languages.

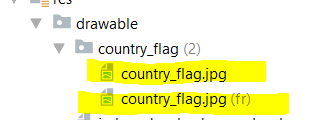
*Localization* is the process of adapting a globalized app, to a particular culture/locale.

Differences in locales are characterized by:

* Date Formats
* Temperature measurements (Celsius vs Fahrenheit)
* Distance measurements (miles vs kilometers)
* Weight measurements (kilos vs lbs)
* Spelling differences (Example: color vs colour)

We will make our application work for the French audience.

Create a folder named *drawable-fr* under *src/main/res/* and copy into it the image of the French flag from the *images* folder in the snippets. Android Studio will show two flavors of the *country\_flag.jpg* file:



Create a folder named *values-fr* under *src/main/res/* and copy into it the strings.xml file in *src/main/res/values* folder. Android Studio will show two flavors of the *strings.xml* file:

Open the French version of the *strings.xml* file and translate the English text “Global Institute of Technology” to the French equivalent of:

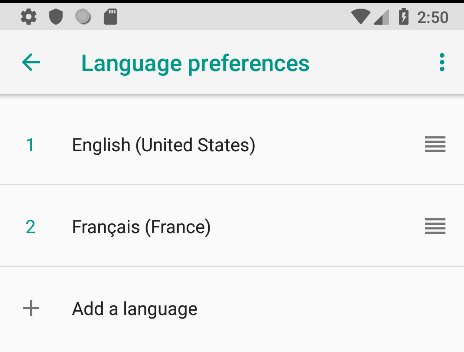
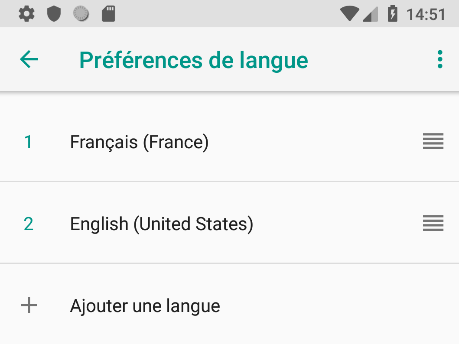
10 french translation.txt

Institut mondial de technologie

Run your application. The default state of your application is English and, therefore, it looks like this:

[](https://www.bing.com/images/search?view=detailV2&ccid=g6nYHgjm&id=9DCD800E35FF6162FE4C44ABC6C70A5783B46357&thid=OIP.g6nYHgjml5I7LyjheiVEGwElEs&q=run+logo&simid=608050058236200411&selectedIndex=67)

Next, inside the Android device, find the language settings and add French >> France.

Move French up so it is #1

Run your app again. You will see that the title and flag are changed to French.

[](https://www.bing.com/images/search?view=detailV2&ccid=g6nYHgjm&id=9DCD800E35FF6162FE4C44ABC6C70A5783B46357&thid=OIP.g6nYHgjml5I7LyjheiVEGwElEs&q=run+logo&simid=608050058236200411&selectedIndex=67)

How about if we need to provide different data depending on the language? For example, if we need to supply data from *students.json* for English speakers and *students-fr.json* for French speakers.

In the *assets* folder, create a file named *students-fr.json* and add to it the following data:

{

11 students-fr.json.txt

"students": [

{ "id" : "1", "firstName": "Ivette", "lastName": "Trdeau", "school": "Exploitation minière" },

{ "id" : "2", "firstName": "Antoine", "lastName": "Destin", "school": "Entreprise" },

{ "id" : "3", "firstName": "Luc", "lastName": "Richaud", "school": "L'informatique" },

{ "id" : "4", "firstName": "Marise", "lastName": "Buchard", "school": "Médicament" },

{ "id" : "5", "firstName": "Marie", "lastName": "Jose", "school": "Santé" },

{ "id" : "6", "firstName": "Adeline", "lastName": "Bordeaux", "school": "Allaitement" },

{ "id" : "7", "firstName": "Dorothée", "lastName": "Carbonneau", "school": "Médicament" },

{ "id" : "8", "firstName": "Emile", "lastName": "Dansereau", "school": "Allaitement" },

{ "id" : "9", "firstName": "Dominique", "lastName": "Etienne", "school": "Entreprise" },

{ "id" : "10", "firstName": "Elodie", "lastName": "Arsenault", "school": "L'informatique" }

]

}

We will next specify the names of these language dependent files in their respective *strings.xml* files.

For English, open *strings.xml* in the editor and add to it the following string:

12 strings.xml.txt

<string name="json\_file\_name">students.json</string>

Similarly, for French, open *strings-fr.xml* and add to it the following string:

13 strings-fr.xml.txt

<string name="json\_file\_name">students-fr.json</string>

Let us adjust the code in *MainActivity.java* so that it reads the file name from the correct language-based *strings.xml* file.

Comment out only the following statement in method *getStudentsData()* in file *MainActivity.java*:

// String jsonString = getAssetsJSON(JSON\_FILE);

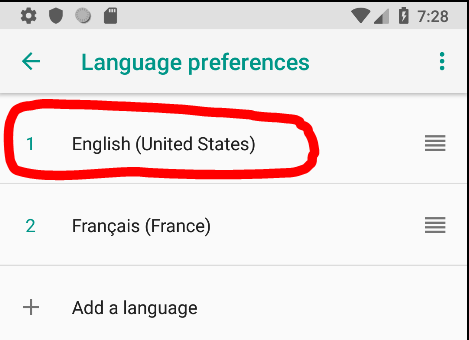
Replace the above commented-out code with the following that reads the value of json\_file\_name from the correct language-based *strings.xml* file:

14 getstudentsData.txt

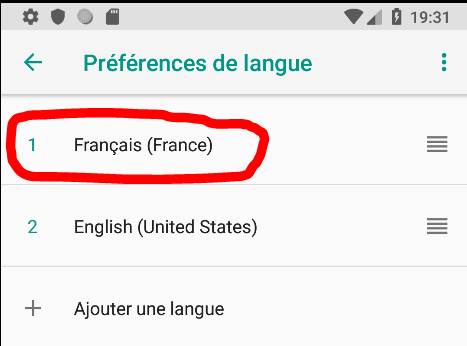
String filename = getResources().getString(R.string.json\_file\_name);

String jsonString = getAssetsJSON(filename);

Run your application. When in default language mode you will see students with English names.

[](https://www.bing.com/images/search?view=detailV2&ccid=g6nYHgjm&id=9DCD800E35FF6162FE4C44ABC6C70A5783B46357&thid=OIP.g6nYHgjml5I7LyjheiVEGwElEs&q=run+logo&simid=608050058236200411&selectedIndex=67) 

When in French language mode, you will see students with French names.

[](https://www.bing.com/images/search?view=detailV2&ccid=g6nYHgjm&id=9DCD800E35FF6162FE4C44ABC6C70A5783B46357&thid=OIP.g6nYHgjml5I7LyjheiVEGwElEs&q=run+logo&simid=608050058236200411&selectedIndex=67) 

## Alert Dialog

The *AlertDialog* is a subclass of *Dialog* that can display one, two or three buttons. If you only want to display a String in this dialog box, use the *setMessage()* method.

You can accomplish a wide variety of dialog designs—including custom layouts and those described in the Dialogs design guide—by extending *DialogFragment* and creating a *AlertDialog* in the *onCreateDialog()* callback method.

15 strings.xml.txt

Add the following to *strings.xml*:

<string name="vote\_btn\_msg">Vote for your favorite language.</string>

<string name="dialog\_favorite\_language">My favorite language is</string>

<string name="dialog\_vote\_btn">Vote</string>

<string name="dialog\_cancel\_btn">Cancel</string>

16 strings-fr.xml.txt

Add the following to *strings-fr.xml:*

<string name="vote\_btn\_msg">Votez pour votre langue préférée.</string>

<string name="dialog\_favorite\_language">Ma langue préférée est</string>

<string name="dialog\_vote\_btn">Votez</string>

<string name="dialog\_cancel\_btn">Annuler</string>

17 VoteDialogFragment.txt

Create a class named *VoteDialogFragment* and replace the class with the following code:

public class VoteDialogFragment extends DialogFragment {

@Override

public Dialog onCreateDialog(Bundle savedInstanceState) {

// Use the Builder class for convenient dialog construction

AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

String msg = "";

msg += getResources().getString(R.string.dialog\_favorite\_language);

msg += " ";

msg += getResources().getConfiguration().locale.getDisplayLanguage();

msg += ".";

builder.setMessage(msg)

.setPositiveButton(R.string.dialog\_vote\_btn, new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

// user cast your vote

}

})

.setNegativeButton(R.string.dialog\_cancel\_btn, new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

// user cancelled the dialog

}

});

// Create the AlertDialog object and return it

return builder.create();

}

}

Add a button to *activity\_main.xml* just below the *ImageView* with the following code:

18 button.txt

<Button

android:id="@+id/btnVote"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="@string/vote\_btn\_msg"

android:onClick="onVoteClick"/>

19 onVoteClick.txt

Add the following event handler code in *MainActivity.java*:

public void onVoteClick(View v) {

VoteDialogFragment dialog = new VoteDialogFragment();

dialog.show(getSupportFragmentManager(), "NoticeDialogFragment");

}

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedRun the application. You will notice a button in the center of the screen. Depending on which language (English or French) you select you will see the appropriate translation.

