Mobile Application Development

Produced by

David Drohan (ddrohan@wit.ie)

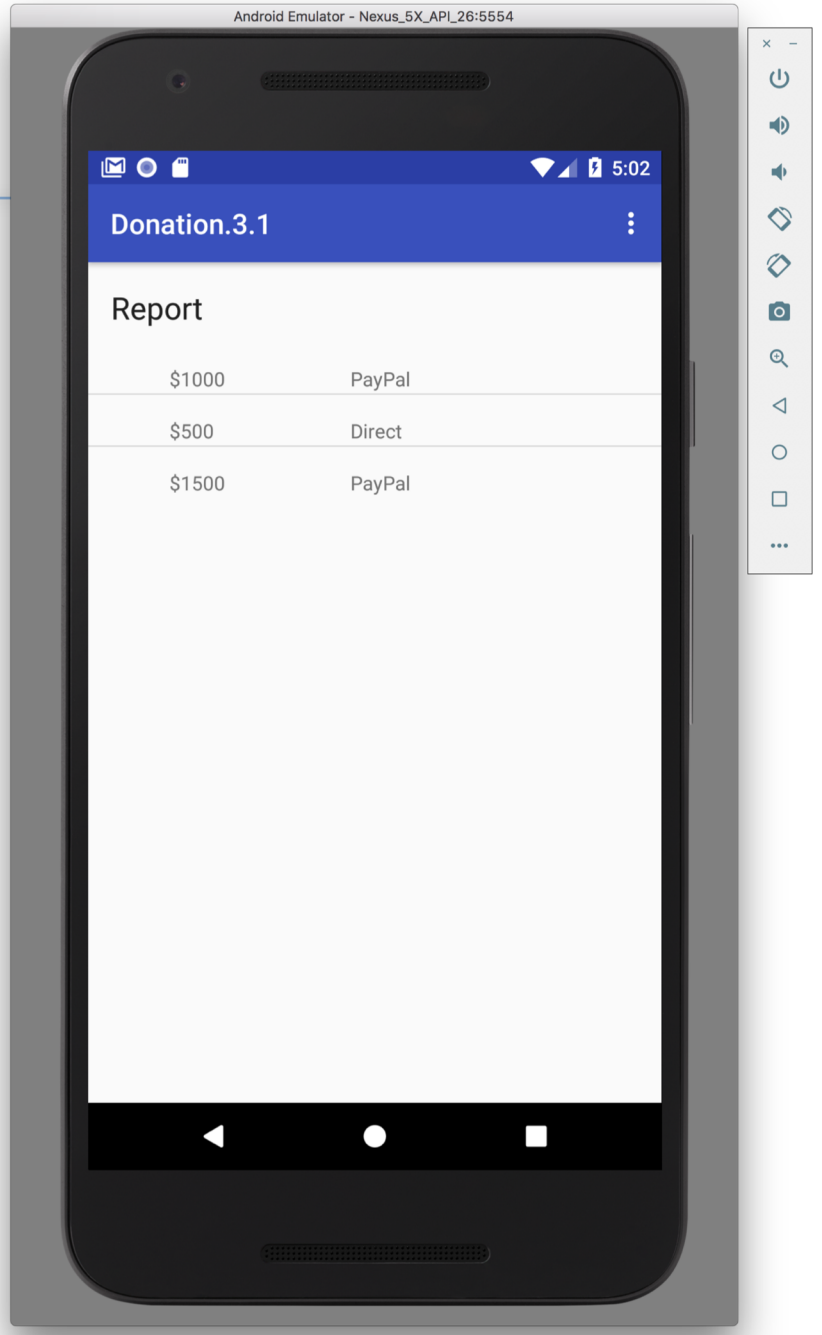
Department of Computing & Mathematics
Waterford Institute of Technology
http://www.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

JSON File IO in Donation

Json Representation

```
[
    {
        "id":3833090328024346907,
        "amount":1000,
        "method":"PayPal”
    },
    {
        "id":1730211668061230 83,
        "amount":500,
        "method":"Direct”
    },
    {
        "id":1322434707975718824,
        "amount":1500,
        "method":"PayPal”
    }
]
```



Donation Model

- Introduce JSON Support libraries

```
import org.json.JSONException;
import org.json.JSONObject;
```

- Define string IDs for each field

```
private static final String JSON_ID          = "id";
private static final String JSON_AMOUNT      = "amount";
private static final String JSON_METHOD      = "method";
```

Donation Model

- Write/Convert a json Donation Object (`JSONObject`)

```
public JSONObject toJSON() throws JSONException
{
    JSONObject json = new JSONObject();
    json.put(JSON_ID        , id);
    json.put(JSON_AMOUNT    , amount);
    json.put(JSON_METHOD    , method);
    return json;
}
```

- Read/Parse a json Donation Object (`Donation`)

```
public Donation(JSONObject json) throws JSONException
{
    id        = json.getLong(JSON_ID);
    amount    = json.getInt(JSON_AMOUNT);
    method    = json.getString(JSON_METHOD);
}
```

Donation Model

- Complete json support

```
public class Donation
{
    public long   id;
    public int    amount;
    public String method;

    private static final String JSON_ID          = "id";
    private static final String JSON_AMOUNT      = "amount";
    private static final String JSON_METHOD      = "method";

    public Donation (int amount, String method)
    {
        this.id = unsignedLong();
        this.amount = amount;
        this.method = method;
    }

    public Donation(JSONObject json) throws JSONException
    {
        id       = json.getLong(JSON_ID);
        amount   = json.getInt(JSON_AMOUNT);
        method   = json.getString(JSON_METHOD);
    }

    public JSONObject toJSON() throws JSONException
    {
        JSONObject json = new JSONObject();
        json.put(JSON_ID       , id);
        json.put(JSON_AMOUNT   , amount);
        json.put(JSON_METHOD   , method);
        return json;
    }

    @Override
    public String toString() {
        return "Donation{" + "id= " + id +
                "amount=$" + amount +
                ", method='" + method + '\'' +
                '}';
    }

    /**...*/
    private Long unsignedLong() {...}
}
```

DonationSerializer

- A class to orchestrate the Serialization of a collection of Donation Objects to/from Disk
- Constructor + 3 Methods
- Fully Encapsulates serialization mechanisms

```
public class DonationSerializer
{
    private Context mContext;
    private String mFilename;
    private int totalDonated;

    public DonationSerializer(Context c, String f)
    {...}

    public void saveDonations(List<Donation> donations)
            throws JSONException, IOException
    {...}

    public ArrayList<Donation> loadDonations()
            throws IOException, JSONException
    {...}

    public int getTotalDonated() { return totalDonated; }
}
```

Using the Application Object

Maintaining Global Application State

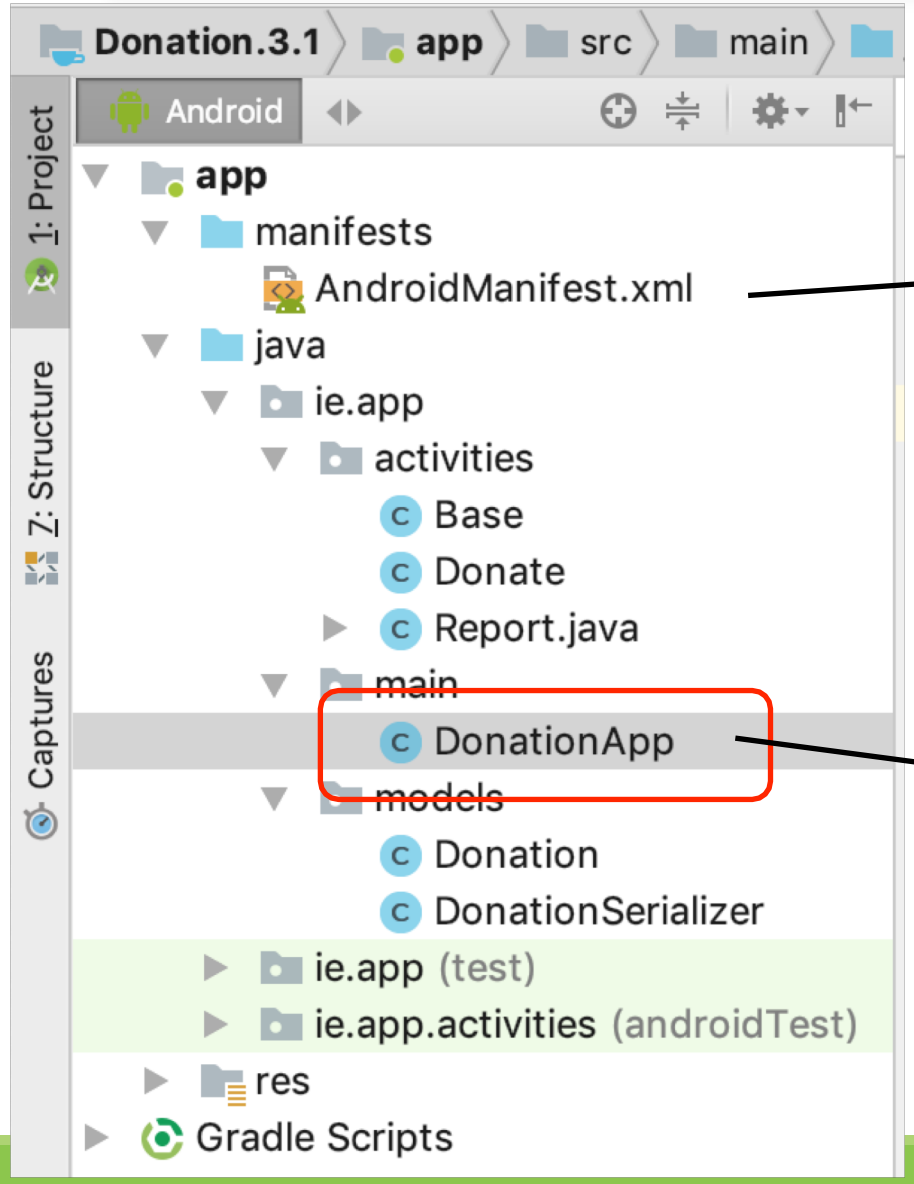
- Sometimes you want to store data, like global variables which need to be accessed from multiple Activities – sometimes everywhere within the application. In this case, the Application object will help you.
- Activities come and go based on user interaction
- Application objects can be a useful '*anchor*' for an android app
- You can use it to hold information shared by all activities

Application Object Callbacks

- onConfigurationChanged( ) Called by the system when the device configuration changes while your component is running.
- onCreate( ) Called when the application is starting, before any other application objects have been created.
- onLowMemory( ) This is called when the overall system is running low on memory, and would like actively running processes to tighten their belts.
- onTerminate( ) This method is for use in emulated process environments. It will never be called on a production Android device, where processes are removed by simply killing them; no user code (including this callback) is executed when doing so.

The Application Object *

Androidmanifest.xml




```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="Donation.3.1"
    android:supportsRtl="true"
    android:theme="@style/AppTheme"
    android:name="ie.app.main.DonationApp">
```

```
public class DonationApp extends Application
{
    public final int       target       = 10000;
    public int             totalDonated = 0;
    public List <Donation> donations    = new ArrayList<~>();

    public DonationSerializer serializer;

    public boolean newDonation(Donation donation)
    {...}

    @Override
    public void onCreate()
    {...}
}
```

Donation 3.1 – **DonationApp** Application Object *

```
public class DonationApp extends Application
{
    public final int       target       = 10000;
    public int             totalDonated = 0;
    public List <Donation> donations    = new ArrayList<~>();

    public DonationSerializer serializer;

    public boolean newDonation(Donation donation)
    {...}

    @Override
    public void onCreate()
    {
        super.onCreate();
        Log.v("Donate", "Donation App Started");
        serializer = new DonationSerializer(this,"donations.json");
            try {
                donations = serializer.loadDonations();
                Log.v("Donate", "Donation JSON File Created/Loaded");
                if(!donations.isEmpty())
                        totalDonated = serializer.getTotalDonated();
                }
            catch (Exception e) // Catch everything!!
                {
                    Log.v("Donate", "Error loading Donations: " + e.getMessage());
                    donations = new ArrayList<Donation>();
                }
    }
}
```

Our List

Our Serializer

Reading from Json File

Refactor existing Activities/Classes

- In order to make full use of the Application object we need to refactor some of the classes in the project.
- This will form part of the Practical Lab but we'll have a quick look now at some of the refactoring that needs to be done.

Donation 3.1 – **Base** (Refactored, extract) *

```
public class Base extends AppCompatActivity {

    public DonationApp app;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        app = (DonationApp) getApplication();
    }
```

Our 'App' Object

```
@Override
protected void onDestroy() {
    super.onDestroy();

    try {
        app.serializer.saveDonations(app.donations);
        Log.v("Donate", "Donation JSON File Saved...");
    }
    catch (Exception e)
    {
        Log.v("Donate", "Error Saving Donations... " + e.getMessage());
    }
}
```

Donation 3.1 – **Donate** (Refactored) *

```
public void donateButtonPressed (View view)
{
    String method = paymentMethod.getCheckedRadioButtonId() == R.id.PayP
    int donatedAmount =  amountPicker.getValue();
    if (donatedAmount == 0)
    {
        String text = amountText.getText().toString();
        if (!text.equals(""))
            donatedAmount = Integer.parseInt(text);
    }
    if (donatedAmount > 0)
    {
        app.newDonation(new Donation(donatedAmount, method));
        progressBar.setProgress(app.totalDonated);
        String totalDonatedStr = "$" + app.totalDonated;
        amountTotal.setText(totalDonatedStr);
    }
}
```

Our 'App' Object

Donation 3.1 – **Report**

```
public class Report extends Base implements OnItemClickListener
{
    ListView listView;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_report);

        listView = (ListView) findViewById(R.id.reportList);
        DonationAdapter adapter = new DonationAdapter(this, app.donations);
        listView.setAdapter(adapter);
        listView.setOnItemClickListener(this);
    }

    @Override
    public void onItemClick(AdapterView<?> arg0, View arg1, int pos, long id) {

        Toast.makeText(this, "You Selected Row [ " + pos + "]\n" +
                "For Donation Data [ " + app.donations.get(pos) + "]\n " +
                "With ID of [" + id + "]", Toast.LENGTH_LONG).show();

    }
}
```

Our 'App' Object

DonationApp

- Uses the serializer to read the donations list and current total amount donated

```
public class DonationApp extends Application
{
    public final int       target       = 10000;
    public int             totalDonated = 0;
    public List <Donation> donations    = new ArrayList<~>();

    public DonationSerializer serializer;

    public boolean newDonation(Donation donation)
    {...}

    @Override
    public void onCreate()
    {
        super.onCreate();
        Log.v( tag: "Donate",  msg: "Donation App Started");
        serializer = new DonationSerializer( c: this, f: "donations.json");
            try {
                donations = serializer.loadDonations();
                Log.v( tag: "Donate",  msg: "Donation JSON File Created/Loaded");
                if(!donations.isEmpty())
                        totalDonated = serializer.getTotalDonated();
                }
            catch (Exception e) // Catch everything!!
                {
                    Log.v( tag: "Donate",  msg: "Error loading Donations: "
                            + e.getMessage());
                    donations = new ArrayList<Donation>();
                }
    }
}
```

Base (Activity)

- Call to save triggered by ‘onPause’ event detected by Base Activity
- This will occur when the user leaves either the Donate or Report activities

```
@Override
protected void onPause() {
    super.onPause();

    try {
        app.serializer.saveDonations(app.donations);
        Log.v( tag: "Donate",  msg: "Donation JSON File Saved...");
    }
    catch (Exception e)
    {
        Log.v( tag: "Donate",  msg: "Error Saving Donations... "
                + e.getMessage());
    }
}
```

The Serialization Mechanism *

```
public class DonationApp extends Application
{
    public final int       target       = 10000;
    public int             totalDonated = 0;
    public List <Donation> donations    = new ArrayList<~>();

    public DonationSerializer serializer;

    public boolean newDonation(Donation donation)
    {...}

    @Override
    public void onCreate()
    {
        super.onCreate();
        Log.v( tag: "Donate",  msg: "Donation App Started");
        serializer = new DonationSerializer( c: this, f: "donations.json");
            try
            {
                donations = serializer.loadDonations();
                Log.v( tag: "Donate",  msg: "Donation JSON File Created/Loaded");
                if(!donations.isEmpty())
                    totalDonated = serializer.getTotalDonated();
                }
            catch (Exception e) // Catch everything!!
                {
                    Log.v( tag: "Donate",  msg: "Error loading Donations: "
                            + e.getMessage());
                    donations = new ArrayList<Donation>();
                }
    }
}
```

```
public class DonationSerializer
{
    private Context mContext;
    private String mFilename;
    private int totalDonated;

    public DonationSerializer(Context c, String f)
    {...}

    public void saveDonations(List<Donation> donations)
            throws JSONException, IOException
    {...}

    public ArrayList<Donation> loadDonations()
            throws IOException, JSONException
    {...}

    public int getTotalDonated() { return totalDonated; }
}
```

```
@Override
protected void onPause() {
    super.onPause();

    try {
        app.serializer.saveDonations(app.donations);
        Log.v( tag: "Donate",  msg: "Donation JSON File Saved...");
    }
    catch (Exception e)
    {
        Log.v( tag: "Donate",  msg: "Error Saving Donations... "
                + e.getMessage());
    }
}
```

```
public class Donation
{
    public long   id;
    public int    amount;
    public String method;

    private static final String JSON_ID         = "id";
    private static final String JSON_AMOUNT     = "amount";
    private static final String JSON_METHOD     = "method";

    public Donation (int amount, String method)
    {
        this.id = unsignedLong();
        this.amount = amount;
        this.method = method;
    }

    public Donation(JSONObject json) throws JSONException
    {
        id       = json.getLong(JSON_ID);
        amount   = json.getInt(JSON_AMOUNT);
        method   = json.getString(JSON_METHOD);
    }

    public JSONObject toJSON() throws JSONException
    {
        JSONObject json = new JSONObject();
        json.put(JSON_ID       , id);
        json.put(JSON_AMOUNT   , amount);
        json.put(JSON_METHOD   , method);
        return json;
    }

    @Override
    public String toString() {
        return "Donation{" + "id= "
                "amount=$" + amount
                ", method='" + meth
                '}';
    }

    /**...*/
    private Long unsignedLong() {..
```

```
[
    {
        "id":3833090328024346907,
        "amount":1000,
        "method":"PayPal"
    },
    {
        "id":173021166806123083,
        "amount":500,
        "method":"Direct"
    },
    {
        "id":1322434707975718824,
        "amount":1500,
        "method":"PayPal"
    }
]
```

saveDonations( ) *

- Create a JSONArray object
- Place each donation in turn into the object
- Write the object to the file
- Close the file
- If any exceptions occur, 'propagate' to the caller (whoever that is)

```
public void saveDonations(List<Donation> donations)
        throws JSONException, IOException
{
    // build an array in JSON
    JSONArray array = new JSONArray();
    for (Donation d : donations)
        array.put(d.toJSON());

    // write the file to disk
    Writer writer = null;
    try
    {
        OutputStream out = mContext.openFileOutput(mFilename, Context.MODE_PRIVATE);
        writer = new OutputStreamWriter(out);
        writer.write(array.toString());
    }
    finally
    {
        if (writer != null)
            writer.close();
    }
}
```

loadDonations( ) *

- Create Donation Array
- Attach a reader to the file (via a buffered reader)
- Read the file into a string
- Tokenize the string into individual json objects
- Extract each object in turn and create a new Donation Object from it
- Add to our Donations list

```
public ArrayList<Donation> loadDonations()
        throws IOException, JSONException
{
    ArrayList<Donation> donations = new ArrayList<~>();
    BufferedReader reader = null;
    totalDonated = 0;

    try
    {
        // open and read the file into a StringBuilder
        InputStream in = mContext.openFileInput(mFilename);
        reader = new BufferedReader(new InputStreamReader(in));
        StringBuilder jsonString = new StringBuilder();
        String line = null;
        while ((line = reader.readLine()) != null)
        {
            // line breaks are omitted and irrelevant
            jsonString.append(line);
        }

        Log.v( tag: "Donate",jsonString.toString());
        // parse the JSON using JSONTokener
        JSONArray array = (JSONArray) new JSONTokener(jsonString.toString()).nextValue();
        // build the array of donations from JSONObjects
        for (int i = 0; i < array.length(); i++)
        {
            donations.add(new Donation(array.getJSONObject(i)));
            totalDonated += donations.get(i).amount;
        }
    }
    catch (FileNotFoundException e)
    {
        // we will ignore this one, since it happens when we start fresh
    }
    finally
    {
        if (reader != null)
            reader.close();
    }
    return donations;
}
```

Gson Approach - Donation Model

- Introduce Gson dependencies

```
compile 'com.google.code.gson:gson:2.8.2' // for Googles Gson JSON Parser
```

- That's it!
- No change to existing original Model

Gson Approach - saveDonations( ) *

- Declare Object Type
- Convert donations list to json string
- Write the String to the file
- Close the file
- If any exceptions occur, 'propagate' to the caller

```
Type objType = new TypeToken<List<Donation>>(){}.getType();
```

```
public void saveDonations(List<Donation> donations)
        throws JSONException, IOException
{
    String result = new Gson().toJson(donations, objType);
    Writer writer = null;
    try
    {
        OutputStream out = mContext.openFileOutput(mFilename, Context.MODE_PRIVATE);
        writer = new OutputStreamWriter(out);
        writer.write(result);
    }
    finally
    {
        if (writer != null)
            writer.close();
    }
}
```

Gson Approach - loadDonations( ) *

- Create Donation Array
- Attach a reader to the file (via a buffered reader)
- Read the json string from the file
- Convert the string to list of donations
- Use donations list to calculate total donated
- Close the reader 'stream'

```
public ArrayList<Donation> loadDonations()
        throws IOException, JSONException
{
    ArrayList<Donation> donations = new ArrayList<Donation>();
    BufferedReader reader = null;
    totalDonated = 0;

    try
    {
        InputStream in = mContext.openFileInput(mFilename);
        reader = new BufferedReader(new InputStreamReader(in));
        donations = new Gson().fromJson(reader.readLine(), objType);

        for(Donation d : donations)
            totalDonated += d.amount;
    }
    catch (FileNotFoundException e)
    {
        // we will ignore this one, since it happens when we start fresh
    }
    finally
    {
        if (reader != null)
            reader.close();
    }
    return donations;
}
```

Questions?