

Mobile Application Development

Higher Diploma in Science in Computer Science

Produced
by

Eamonn de Leastar (edelestar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

<http://www.wit.ie>

<http://elearning.wit.ie>



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE



File IO in MyRent



MyRent



52.253456, -7.187162

Registered: Sep 6, 2014 11:48:29 AM



52.253456, -7.187162

Registered: Sep 6, 2014 11:48:31 AM



12,12

Registered: Sep 14, 2014 12:00:00 AM



- Json Representation

```
[
  {
    "date"      : 1410000509060,
    "geolocation": "52.253456, -7.187162",
    "id"        : "6a8d44df-7534-43c5-8130-444678fcc187",
    "rented"    : false
  },
  {
    "date"      : 1410000511943,
    "geolocation": "52.253456, -7.187162",
    "id"        : "483eb7da-0dd2-41d3-b53e-ba3d51f9c55d",
    "rented"    : false
  },
  {
    "date"      : 1410649200000,
    "geolocation": "12,12",
    "id"        : "9cf116a4-2c75-40a8-b28b-f3bc9a3939a6",
    "rented"    : true
  }
]
```

Residence

- 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_GEOLOCATION  = "geolocation"  ;  
private static final String JSON_DATE        = "date"         ;  
private static final String JSON_RENTED      = "rented"       ;
```

Residence

- Write a json Residence Object

```
public JSONObject toJSON() throws JSONException
{
    JSONObject json = new JSONObject();
    json.put(JSON_ID, id.toString());
    json.put(JSON_GEOLOCATION, geolocation);
    json.put(JSON_DATE, date.getTime());
    json.put(JSON_RENTED, rented);
    return json;
}
```

- Read a json Residence Object

```
public Residence(JSONObject json) throws JSONException
{
    id = UUID.fromString(json.getString(JSON_ID));
    geolocation = json.getString(JSON_GEOLOCATION);
    date = new Date(json.getLong(JSON_DATE));
    rented = json.getBoolean(JSON_RENTED);
}
```

Residence

- Complete json support

```
public class Residence
{
    public UUID id;

    public String geolocation;
    public Date date;
    public boolean rented;

    private static final String JSON_ID = "id" ;
    private static final String JSON_GEOLOCATION = "geolocation" ;
    private static final String JSON_DATE = "date" ;
    private static final String JSON_RENTED = "rented" ;

    public Residence()
    {
        this.id = UUID.randomUUID();
        this.date = new Date();
        this.geolocation = "";
    }

    public Residence(JSONObject json) throws JSONException
    {
        id = UUID.fromString(json.getString(JSON_ID));
        geolocation = json.getString(JSON_GEOLOCATION);
        date = new Date(json.getLong(JSON_DATE));
        rented = json.getBoolean(JSON_RENTED);
    }

    public JSONObject toJSON() throws JSONException
    {
        JSONObject json = new JSONObject();
        json.put(JSON_ID, id.toString());
        json.put(JSON_GEOLOCATION, geolocation);
        json.put(JSON_DATE, date.getTime());
        json.put(JSON_RENTED, rented);
        return json;
    }

    public String getDateString()
    {
        return "Registered: " + DateFormat.getDateTimeInstance().format(date);
    }
}
```

PortfolioSerializer

- A class to orchestrate the Serialization of a collection of Residence Object to/from Disk
- Constructor + 2 Methods
- Fully Encapsulates serialisation mechanisms

```
public class PortfolioSerializer
{
    private Context mContext;
    private String mFilename;

    public PortfolioSerializer(Context c, String f)
    {
        mContext = c;
        mFilename = f;
    }

    public void saveResidences(ArrayList<Residence> residences) throws ...
    {
        //...
    }

    public ArrayList<Residence> loadResidences() throws ...
    {
        //...
    }
}
```

Portfolio

- Uses the serialiser to read and write the residence list
- Read - in constructor when Portfolio create
- Write - in 'saveResidences()' method called when data may have changed

```
public class Portfolio
{
    public ArrayList<Residence> residences;
    private PortfolioSerializer serializer;

    public Portfolio(PortfolioSerializer serializer)
    {
        this.serializer = serializer;
        try
        {
            residences = serializer.loadResidences();
        }
        catch (Exception e)
        {
            info(this, "Error loading residences: " + e.getMessage());
            residences = new ArrayList<Residence>();
        }
    }

    public boolean saveResidences()
    {
        try
        {
            serializer.saveResidences(residences);
            info(this, "Residences saved to file");
            return true;
        }
        catch (Exception e)
        {
            info(this, "Error saving residences: " + e.getMessage());
            return false;
        }
    }

    //...
}
```


MyRentApp

- Creates the serializer, giving it a file name to use
- Passes the serialiser to the Portfolio object, which will be responsible for loading / saving to the file using the serializer

```
public class MyRentApp extends Application
{
    private static final String FILENAME = "portfolio.json";

    public Portfolio portfolio;

    @Override
    public void onCreate()
    {
        super.onCreate();
        PortfolioSerializer serializer = new PortfolioSerializer(this, FILENAME);
        portfolio = new Portfolio(serializer);

        info(this, "RentControl app launched");
    }
}
```

ResidenceActivity

- Call to save triggered by 'onPause' event detected by ResidenceActivity
- This will occur when the user leaves the activity

```
public class ResidenceActivity extends Activity implements ...
{
    //...
    private Portfolio portfolio;

    public void onPause()
    {
        super.onPause();
        portfolio.saveResidences();
    }
    //...
}
```

The Serialization Mechanism

```
public class PortfolioSerializer
{
    private Context mContext;
    private String mFilename;

    public PortfolioSerializer(Context c, String f)
    {
        mContext = c;
        mFilename = f;
    }

    public void saveResidences(ArrayList<Residence> r)
    {
        //...
    }

    public ArrayList<Residence> loadResidences() thro
    {
        //...
    }
}
```

```
public class Residence
{
    public UUID id;

    public String geolocation;
    public Date date;
    public boolean rented;

    private static final String JSON_ID = "id" ;
    private static final String JSON_GEOLOCATION = "geolocation" ;
    private static final String JSON_DATE = "date" ;
    private static final String JSON_RENTED = "rented" ;
}
```

```
[
  {
    "date" : 1410000509060,
    "geolocation": "52.253456, -7.187162",
    "id" : "6a8d44df-7534-43c5-8130-444678fcc187",
    "rented" : false
  },
  {
    "date" : 1410000511943,
    "geolocation": "52.253456, -7.187162",
    "id" : "483eb7da-0dd2-41d3-b53e-ba3d51f9c55d",
    "rented" : false
  },
  {
    "date" : 1410649200000,
    "geolocation": "12,12",
    "id" : "9cf116a4-2c75-40a8-b28b-f3bc9a3939a6",
    "rented" : true
  }
]
```

```
    }

    public
    {
        return "Registered: " + DateFormat.getDateInstance().format(date);
    }
}
```

saveResidences

- Create a JSONArray object
- Place each residence 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)

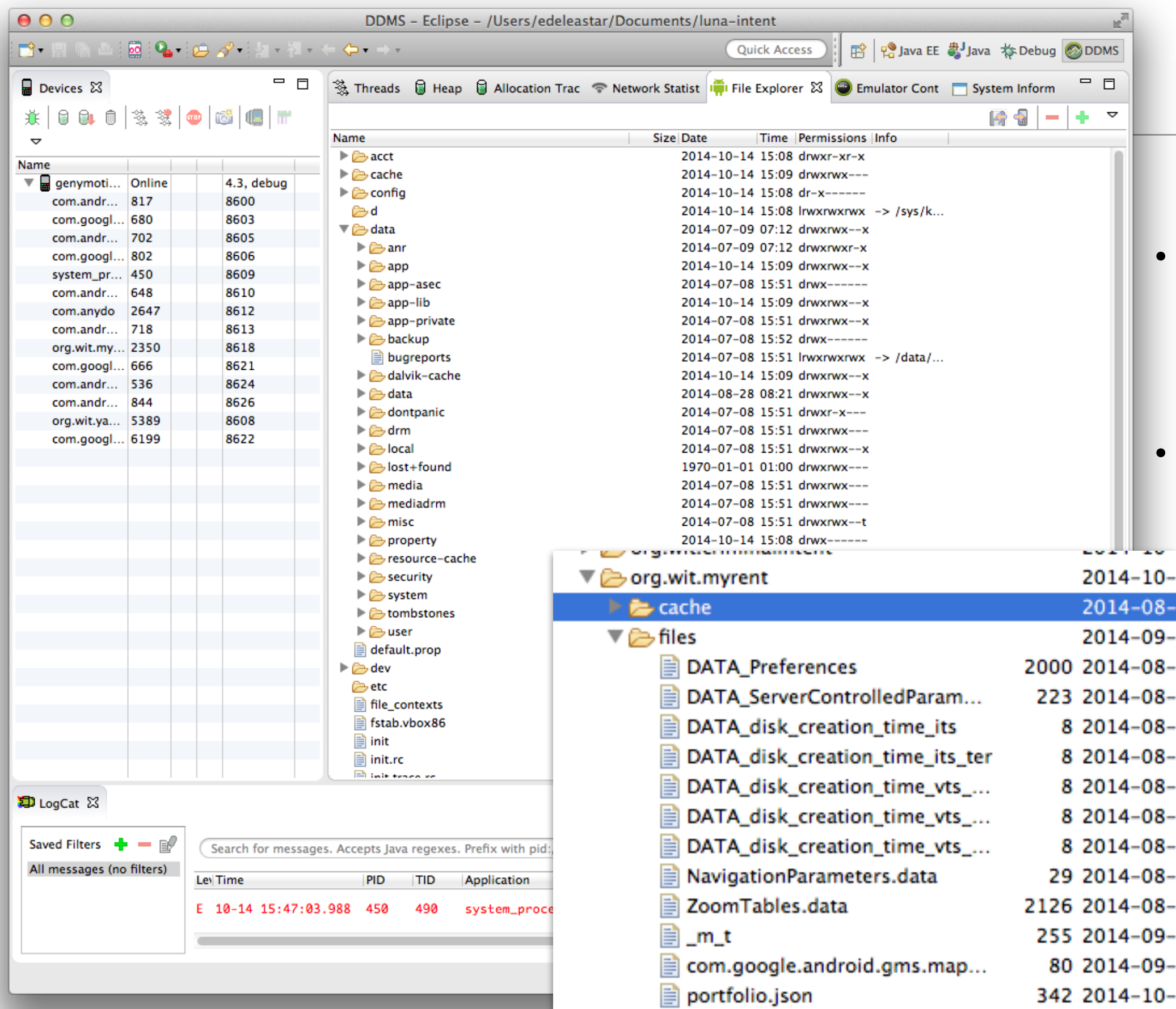
```
public void saveResidences(ArrayList<Residence> residences)
    throws JSONException, IOException
{
    // build an array in JSON
    JSONArray array = new JSONArray();
    for (Residence c : residences)
        array.put(c.toJSON());

    // 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();
    }
}
```

LoadResidences

- Create Residence 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 Residence Object from it
- Add to our Residence list

```
public ArrayList<Residence> loadResidences() throws IOException, JSONException
{
    ArrayList<Residence> residences = new ArrayList<Residence>();
    BufferedReader reader = null;
    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);
        }
        // parse the JSON using JSNTokener
        JSONArray array = (JSONArray) new JSNTokener(jsonString.toString()).nextValue();
        // build the array of residences from JSONObjects
        for (int i = 0; i < array.length(); i++)
        {
            residences.add(new Residence(array.getJSONObject(i)));
        }
    }
    catch (FileNotFoundException e)
    {
        // we will ignore this one, since it happens when we start fresh
    }
    finally
    {
        if (reader != null)
            reader.close();
    }
    return residences;
}
```



- open DDMS perspective

- Select File Explorer

Browse /
data/data/..
application id

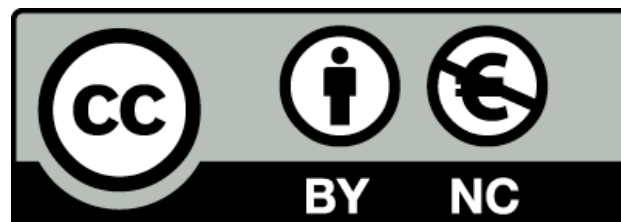
Alternatively ... log in to emulator

- “abd shell”

```
2. adb
Last login: Tue Oct 14 15:50:12 on ttys001
localhost:~ edeleastar$ adb shell
root@vbox86p:/ # cd data/data
root@vbox86p:/data/data # cd org.wit.myrent
root@vbox86p:/data/data/org.wit.myrent # cd files
root@vbox86p:/data/data/org.wit.myrent/files # cat portfolio.json
[{"date":1410000509060,"geolocation":"52.253456, -7.187162","id":"6a8d44df-7534-43c5-8130-444678fcc187","rented":false},{"date":1410000511943,"geolocation":"52.253456, -7.187162","id":"483eb7da-0dd2-41d3-b53e-ba3d51f9c55d","rented":false},{"date":1410649200000,"geolocation":"12,12","id":"9cf116a4-2c75-40a8-b28b-f3bc9a3939a6","rented":true}]root@vbox86p:/data/data/org.wit.myrent/files #
```

```
[
  {
    "date"      : 1410000509060,
    "geolocation": "52.253456, -7.187162",
    "id"        : "6a8d44df-7534-43c5-8130-444678fcc187",
    "rented"    : false
  },
  {
    "date"      : 1410000511943,
    "geolocation": "52.253456, -7.187162",
    "id"        : "483eb7da-0dd2-41d3-b53e-ba3d51f9c55d",
    "rented"    : false
  },
  {
    "date"      : 1410649200000,
    "geolocation": "12,12",
    "id"        : "9cf116a4-2c75-40a8-b28b-f3bc9a3939a6",
    "rented"    : true
  }
]
```

```
adb shell
root@vbox86p:/ # cd data/data
root@vbox86p:/data/data # cd org.wit.myrent
root@vbox86p:/data/data/org.wit.myrent # cd files
root@vbox86p:/data/data/org.wit.myrent/files # cat portfolio.json
[{"date":1410000509060,"geolocation":"52.253456, -7.187162","id":"6a8d44df-7534-43c5-8130-444678fcc187","rented":false},{"date":1410000511943,"geolocation":"52.253456, -7.187162","id":"483eb7da-0dd2-41d3-b53e-ba3d51f9c55d","rented":false},{"date":1410649200000,"geolocation":"12,12","id":"9cf116a4-2c75-40a8-b28b-f3bc9a3939a6","rented":true}]root@vbox86p:/data/data/org.wit.myrent/files #
```



Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

