Uddhav Prasad Gautam T00596539 <upgautam@ualr.edu) Supervisor: Prof. Dr. Kenji Yoshigoe



ENT Tool (Source code gist)



|  |  |
| --- | --- |
| **ENTTool\_aboutActivity.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | package com.Uddhav.ENTTool;  import android.os.Bundle;  import android.support.v7.app.AppCompatActivity; public class aboutActivity extends AppCompatActivity {  @Override  protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_about);  }  } |

|  |  |
| --- | --- |
| **ENTTool\_adapters\_ListviewAdapter.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44 | package com.Uddhav.ENTTool.adapters;  import android.app.Activity; import android.content.Context; import android.view.LayoutInflater; import android.view.View;  import android.view.ViewGroup; import android.widget.ArrayAdapter; import android.widget.TextView;  import com.Uddhav.ENTTool.R;  import com.Uddhav.ENTTool.database.EarthQuakes;  import java.util.Date; import java.util.List;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class ListviewAdapter extends ArrayAdapter<EarthQuakes> { // just to GUI present for earthquake records in ListView  private static LayoutInflater inflater = null; private List<EarthQuakes> list;  private Activity act;  public ListviewAdapter(Activity activity, List<EarthQuakes> datas) { super(activity, R.layout.list\_row, datas);  list = datas; act = activity;  inflater = (LayoutInflater) activity.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);  }  public View getView(int position, View convertView, ViewGroup parent) { View vi = convertView;  if (convertView == null)  vi = inflater.inflate(R.layout.list\_row, null);  TextView tvLocation = (TextView) vi.findViewById(R.id.tv\_location); TextView tvDate = (TextView) vi.findViewById(R.id.tv\_date); TextView tvDepth = (TextView) vi.findViewById(R.id.tv\_depth); TextView tvMag = (TextView) vi.findViewById(R.id.tv\_mag);  EarthQuakes earthQuake = list.get(position); |



|  |  |  |
| --- | --- | --- |
| 45 | } | tvLocation.setText(earthQuake.getLocationName());  tvDate.setText(": " + new Date(earthQuake.getDateMilis()).toLocaleString()); tvDepth.setText(": " + Float.toString(earthQuake.getDepth()) + " KM"); tvMag.setText(Float.toString(earthQuake.getMagnitude()));  float magnitude = earthQuake.getMagnitude(); if (magnitude < 3) {  tvMag.setBackgroundColor(act.getResources().getColor(R.color.COLOR\_GREEN));  } else if (magnitude >= 3 && magnitude < 5) { tvMag.setBackgroundColor(act.getResources().getColor(R.color.COLOR\_YELLOW));  } else if (magnitude >= 5) { tvMag.setBackgroundColor(act.getResources().getColor(R.color.COLOR\_RED));  }  return vi;  }  @Override  public EarthQuakes getItem(int position) { return super.getItem(position);  }  @Override  public int getCount() { return super.getCount();  }  @Override  public long getItemId(int position) { return super.getItemId(position);  }  @Override  public int getPosition(EarthQuakes item) { return super.getPosition(item);  }  @Override  public void notifyDataSetChanged() { super.notifyDataSetChanged();  } |
| 46 |
| 47 |
| 48 |
| 49 |
| 50 |
| 51 |
| 52 |
| 53 |
| 54 |
| 55 |
| 56 |
| 57 |
| 58 |
| 59 |
| 60 |
| 61 |
| 62 |
| 63 |
| 64 |
| 65 |
| 66 |
| 67 |
| 68 |
| 69 |
| 70 |
| 71 |
| 72 |
| 73 |
| 74 |
| 75 |
| 76 |
| 77 |
| 78 |
| 79 |
| 80 |
| 81 |
| 82 |
| 83 |
| 84 |
| 85 |
| 86 |
| 87 |
| 88 |
| 89 |

|  |  |
| --- | --- |
| **ENTTool\_adapters\_MarkerInfoAdapter.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | package com.Uddhav.ENTTool.adapters;  import android.view.LayoutInflater; import android.view.View;  import android.widget.TextView;  import com.Uddhav.ENTTool.R;  import com.Uddhav.ENTTool.database.EarthQuakes;  import com.google.android.gms.maps.GoogleMap.InfoWindowAdapter; import com.google.android.gms.maps.model.Marker;  import java.util.Date;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class MarkerInfoAdapter implements InfoWindowAdapter { // this is to mark location via google maps when user clicks earthquake recor LayoutInflater inflater = null;  TextView tvLoc, tvMag, tvDepth, tvDate, tvLat, tvLng;  public MarkerInfoAdapter(LayoutInflater inflater) { this.inflater = inflater;  } |





|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 27 | | } | @Override  public View getInfoContents(Marker marker) {  View info = inflater.inflate(R.layout.marker\_info, null); tvLoc = (TextView) info.findViewById(R.id.tv1);  tvMag = (TextView) info.findViewById(R.id.tv2); tvDate = (TextView) info.findViewById(R.id.tv3); tvDepth = (TextView) info.findViewById(R.id.tv4); tvLat = (TextView) info.findViewById(R.id.tv5); tvLng = (TextView) info.findViewById(R.id.tv6);  String snippet = marker.getSnippet();  EarthQuakes earthQuakes = new EarthQuakes().getEarthquakesById(Long.parseLong(snippet)); tvLoc.setText(" : " + earthQuakes.getLocationName());  tvMag.setText(" : " + earthQuakes.getMagnitude()); tvDepth.setText(" : " + earthQuakes.getDepth() + " KM");  tvDate.setText(" : " + new Date(earthQuakes.getDateMilis()).toLocaleString()); tvLat.setText(" : " + earthQuakes.getLatitude());  tvLng.setText(" : " + earthQuakes.getLongitude());  return info;  }  @Override  public View getInfoWindow(Marker marker) {  // TODO Auto‐generated method stub return null;  } | |  |  |
| 28 | |
| 29 | |
| 30 | |
| 31 | |
| 32 | |
| 33 | |
| 34 | |
| 35 | |
| 36 | |
| 37 | |
| 38 | |
| 39 | |
| 40 | |
| 41 | |
| 42 | |
| 43 | |
| 44 | |
| 45 | |
| 46 | |
| 47 | |
| 48 | |
| 49 | |
| 50 | |
| 51 | |
| 52 | |
| 53 | |
| 54 | |
| 55 | |
| 56 | |
| 57 | |
| 58 | |
| 59 | |
|  |  | | |  |  |  |

|  |  |
| --- | --- |
| **ENTTool\_ChartActivity.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | package com.Uddhav.ENTTool;  import android.graphics.Color;  import android.graphics.drawable.ColorDrawable; import android.os.Bundle;  import android.support.v7.app.ActionBar;  import android.support.v7.app.AppCompatActivity; import android.view.Gravity;  import android.view.MotionEvent; import android.view.View;  import com.Uddhav.ENTTool.database.EarthQuakes; import java.util.ArrayList;  import java.util.Calendar; import java.util.Date; import java.util.List;  import de.keyboardsurfer.android.widget.crouton.Crouton; import de.keyboardsurfer.android.widget.crouton.Style; import lecho.lib.hellocharts.gesture.ZoomType;  import lecho.lib.hellocharts.listener.ColumnChartOnValueSelectListener; import lecho.lib.hellocharts.listener.LineChartOnValueSelectListener; import lecho.lib.hellocharts.listener.ViewportChangeListener;  import lecho.lib.hellocharts.model.Axis; import lecho.lib.hellocharts.model.Column;  import lecho.lib.hellocharts.model.ColumnChartData; import lecho.lib.hellocharts.model.PointValue; import lecho.lib.hellocharts.model.SubcolumnValue; import lecho.lib.hellocharts.model.Viewport; import lecho.lib.hellocharts.view.ColumnChartView;  import lecho.lib.hellocharts.view.PreviewColumnChartView;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/ |

38 public class ChartActivity extends AppCompatActivity { 39

1. Crouton crouton = null;
2. private ColumnChartView chart;
3. private PreviewColumnChartView previewChart;
4. private ColumnChartData data;
5. private ColumnChartData previewData; 45
6. @Override
7. protected void onCreate(Bundle savedInstanceState) {
8. super.onCreate(savedInstanceState); 49
9. ActionBar ab = getSupportActionBar();
10. ab.setDisplayShowHomeEnabled(true);
11. ab.setIcon(R.mipmap.ic\_launcher); // R means R.java, which is for dynamic (run‐time) setting of icon.
12. // Syntax: ActionBar\_instance.setIcon(R.<folder\_name>.icon\_name)
13. // By default, android stores mipmap/mipmap‐\*\*\* folder/same\_icon\_name for all resolution (quality)
14. ab.setTitle(" " + getResources().getString(R.string.app\_name)); 56
15. ColorDrawable cd = new ColorDrawable(getResources().getColor(R.color.statusbar));
16. ab.setBackgroundDrawable(cd); 59
17. // ab.setDisplayShowTitleEnabled(false);
18. ab.setDisplayShowTitleEnabled(true); 62

63 setContentView(R.layout.activity\_chart);

64

1. chart = (ColumnChartView) findViewById(R.id.chart);
2. chart.setOnValueTouchListener(new ValueTouchListener());
3. chart.setOnClickListener(new ValueTouchListener());
4. chart.setOnLongClickListener(new ValueTouchListener());
5. chart.setOnTouchListener(new ValueTouchListener()); 70

71 generateDefaultData();

72

1. chart.setColumnChartData(data);
2. chart.setZoomEnabled(true);
3. chart.setScrollEnabled(true); 76
4. previewChart = (PreviewColumnChartView) findViewById(R.id.chart\_preview);
5. previewChart.setColumnChartData(previewData);
6. previewChart.setViewportChangeListener(new ViewportListener()); 80

81 previewX(true);

82

83 }

84

85 private void generateDefaultData() { 86

87 List<Column> columns = getDatasForChart();

88

89 data = new ColumnChartData(columns); 90

1. Axis axisX = new Axis();
2. Axis axisY = new Axis().setHasLines(true); 93
3. axisX.setName(getString(R.string.ChartAxisX));
4. axisY.setName(getString(R.string.ChartAxisY)); 96
5. data.setAxisXBottom(axisX);
6. data.setAxisYLeft(axisY); 99
7. previewData = new ColumnChartData(data);
8. for (Column column : previewData.getColumns()) {
9. for (SubcolumnValue value : column.getValues()) {
10. value.setColor(Color.BLACK);

104 }

105 }

106 }

107

1. private void previewX(boolean animate) {
2. Viewport tempViewport = new Viewport(chart.getMaximumViewport());
3. float dx = tempViewport.width() / 3f;
4. tempViewport.inset(dx, 0);
5. if (animate) {

118 }

119

previewChart.setCurrentViewportWithAnimation(tempViewport);

} else {

previewChart.setCurrentViewport(tempViewport);

}

previewChart.setZoomType(ZoomType.HORIZONTAL);

120 public List<Column> getDatasForChart() { 121

1. Calendar cal = Calendar.getInstance();
2. // cal.setTimeInMillis(earthQuakes.getDateMilis());
3. int month = cal.get(Calendar.MONTH) + 1;
4. int day = cal.get(Calendar.DAY\_OF\_MONTH); 126

127 List<Column> columns = new ArrayList<>();

128

129 for (int i = 0; i <= 31; ++i) { 130

1. List<EarthQuakes> EarthQuakeList = new EarthQuakes().getEarthquakesByDay(i, month);
2. List<SubcolumnValue> subcolumn = new ArrayList<>(); 133
3. if (EarthQuakeList.size() < 0) {
4. continue;

136 }

137 for (EarthQuakes earthQuakes : EarthQuakeList) { 138

139 SubcolumnValue subcolumnValue = new SubcolumnValue();

140

141 float magnitude = earthQuakes.getMagnitude(); 142

1. if (magnitude < 3) {
2. subcolumnValue.setColor(getResources().getColor(R.color.COLOR\_GREEN));
3. } else if (magnitude >= 3 && magnitude < 5) {
4. subcolumnValue.setColor(getResources().getColor(R.color.COLOR\_YELLOW));
5. } else if (magnitude >= 5) {
6. subcolumnValue.setColor(getResources().getColor(R.color.COLOR\_RED));

149 }

150

1. subcolumnValue.setValue(earthQuakes.getMagnitude());
2. subcolumnValue.setLabel("" + earthQuakes.getDateMilis()); 153

154 subcolumn.add(subcolumnValue);

155 }

156

157 columns.add(new Column(subcolumn));

158 }

159

160 return columns;

161 }

162

1. private void previewY() {
2. Viewport tempViewport = new Viewport(chart.getMaximumViewport());
3. float dy = tempViewport.height() / 4;
4. tempViewport.inset(0, dy);
5. previewChart.setCurrentViewportWithAnimation(tempViewport);
6. previewChart.setZoomType(ZoomType.VERTICAL);

169 }

170

1. private void previewXY() {
2. Viewport tempViewport = new Viewport(chart.getMaximumViewport());
3. float dx = tempViewport.width() / 4;
4. float dy = tempViewport.height() / 4;
5. tempViewport.inset(dx, dy);
6. previewChart.setCurrentViewportWithAnimation(tempViewport);

177 }

178

1. @Override
2. public void onBackPressed() {
3. finish();

182 }

183

1. @Override
2. protected void onDestroy() {
3. Crouton.cancelAllCroutons();

|  |  |  |
| --- | --- | --- |
| 187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239 | super.onDestroy();  }  private class ViewportListener implements ViewportChangeListener {  @Override  public void onViewportChanged(Viewport newViewport) { chart.setCurrentViewport(newViewport);  }  }  private class ValueTouchListener implements LineChartOnValueSelectListener, ColumnChar  @Override  public void onValueSelected(int lineIndex, int pointIndex, PointValue value) {  }  @Override  public void onValueDeselected() {  // TODO Auto‐generated method stub  }  @Override  public void onValueSelected(int i, int i1, SubcolumnValue subcolumnValue) { String str = String.valueOf(subcolumnValue.getLabel());  EarthQuakes earthQuakes = new EarthQuakes().getEarthquakesById(Long.parseLong(  Style.Builder sBuilder = new Style.Builder(); sBuilder.setGravity(Gravity.LEFT);  float magnitude = earthQuakes.getMagnitude(); if (magnitude < 3) {  sBuilder.setBackgroundColorValue(getResources().getColor(R.color.COLOR\_GRE  } else if (magnitude >= 3 && magnitude < 5) { sBuilder.setBackgroundColorValue(getResources().getColor(R.color.COLOR\_YEL  } else if (magnitude >= 5) { sBuilder.setBackgroundColorValue(getResources().getColor(R.color.COLOR\_RED  }  String message = "";  if (getString(R.string.Location).equalsIgnoreCase("Konum")) {  message = " " + getString(R.string.Magnitude) + " : " + earthQuakes.get  + " " + getString(R.string.Location) + " : " + earthQuakes.getLo  + " " + getString(R.string.Date) + " : " + new Date(earthQuak  } else {  message = " " + getString(R.string.Magnitude) + " : " + earthQuakes.getM  + " " + getString(R.string.Location) + " : " + earthQuakes.ge  + " " + getString(R.string.Date) + " : " + new Date(ea | |
| 240 |  | } |
| 241 |  |  |
| 242 |  | if (crouton == null) { |
| 243 |  | crouton = Crouton.makeText(ChartActivity.this, message, sBuilder.build()); |
| 244 |  | crouton.show(); |
| 245 |  | } else { |
| 246 |  | crouton.hide(); |
| 247 |  | crouton = Crouton.makeText(ChartActivity.this, message, sBuilder.build()); |
| 248 |  | crouton.show(); |
| 249 |  | } |
| 250 |  |  |
| 251  252  253  254  255  256  257  258  259  260 | }  @Override  public void onClick(View v) {  }  @Override  public boolean onLongClick(View v) { return false; | |

tOnValueSelectListener, View.OnClickListener, Vie

str));

EN));

LOW));

));

Magnitude() + "\n" // cationName() + "\n" //

es.getDateMilis()).toLocaleString();

agnitude() + "\n" // tLocationName() + "\n" //

rthQuakes.getDateMilis()).toLocaleString();



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 261 | | } | } | }  @Override  public boolean onTouch(View v, MotionEvent event) { return false;  } | |  |  |
| 262 | |
| 263 | |
| 264 | |
| 265 | |
| 266 | |
| 267 | |
| 268 | |
|  |  | | | |  |  |  |

|  |  |
| --- | --- |
| **ENTTool\_database\_DatabaseHelper.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62 | package com.Uddhav.ENTTool.database;  import android.content.Context;  import android.database.sqlite.SQLiteDatabase;  import com.Uddhav.ENTTool.utils.App; import com.Uddhav.ENTTool.utils.Tools;  import com.j256.ormlite.android.apptools.OrmLiteSqliteOpenHelper; import com.j256.ormlite.dao.Dao;  import com.j256.ormlite.support.ConnectionSource; import com.j256.ormlite.table.TableUtils;  import java.sql.SQLException;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class DatabaseHelper extends OrmLiteSqliteOpenHelper {  private static final String DATABASE\_NAME = "ENTTool.db"; private static final int DATABASE\_VERSION = 1;  private static DatabaseHelper dbHelper;  private static Object syncObject = new Object(); private final Context myContext;  private Dao<EarthQuakes, Long> EarthQuakesDataHelper = null;  private Dao<LastEarthquakeDate, Integer> LastEarthquakeDateDataHelper = null;  public DatabaseHelper(Context context) {  super(context, DATABASE\_NAME, null, DATABASE\_VERSION); this.myContext = context;  }  public static DatabaseHelper getDbHelper() { synchronized (syncObject) {  if (dbHelper == null) {  dbHelper = new DatabaseHelper(App.AppContext);  }  }  return dbHelper;  }  @Override  public void onCreate(SQLiteDatabase db, ConnectionSource connectionSource) { try {  TableUtils.createTable(connectionSource, EarthQuakes.class); TableUtils.createTable(connectionSource, LastEarthquakeDate.class);  } catch (java.sql.SQLException e) { Tools.catchException(e);  }  }  @Override  public void onUpgrade(SQLiteDatabase db, ConnectionSource connectionSource, int oldVersion, int newVersion) { try {  TableUtils.dropTable(connectionSource, EarthQuakes.class, true); TableUtils.dropTable(connectionSource, LastEarthquakeDate.class, true); onCreate(db, connectionSource);  } catch (java.sql.SQLException e) { Tools.catchException(e);  }  } |



|  |  |  |
| --- | --- | --- |
| 63 | } | public void clearDatabase() {  ConnectionSource connectionSource = getConnectionSource(); try {  TableUtils.clearTable(connectionSource, EarthQuakes.class);  } catch (SQLException e) { Tools.catchException(e);  }  }  public Dao<EarthQuakes, Long> getEarthQuakesDataHelper() throws SQLException { if (EarthQuakesDataHelper == null) {  EarthQuakesDataHelper = getDao(EarthQuakes.class);  }  return EarthQuakesDataHelper;  }  public Dao<LastEarthquakeDate, Integer> getLastEarthquakeDateDataHelper() throws SQLException { if (LastEarthquakeDateDataHelper == null) {  LastEarthquakeDateDataHelper = getDao(LastEarthquakeDate.class);  }  return LastEarthquakeDateDataHelper;  } |
| 64 |
| 65 |
| 66 |
| 67 |
| 68 |
| 69 |
| 70 |
| 71 |
| 72 |
| 73 |
| 74 |
| 75 |
| 76 |
| 77 |
| 78 |
| 79 |
| 80 |
| 81 |
| 82 |
| 83 |
| 84 |
| 85 |
| 86 |

|  |  |
| --- | --- |
| **ENTTool\_database\_EarthQuakes.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47 | package com.Uddhav.ENTTool.database;  import android.os.Parcel; import android.os.Parcelable;  import com.Uddhav.ENTTool.utils.AppSettings; import com.Uddhav.ENTTool.utils.Tools; import com.j256.ormlite.dao.Dao;  import com.j256.ormlite.field.DatabaseField; import com.j256.ormlite.stmt.DeleteBuilder; import com.j256.ormlite.stmt.PreparedQuery; import com.j256.ormlite.stmt.QueryBuilder; import com.j256.ormlite.table.DatabaseTable;  import java.sql.SQLException; import java.util.ArrayList; import java.util.Calendar; import java.util.Comparator; import java.util.List;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  @DatabaseTable(tableName = "EarthQuakes")  public class EarthQuakes implements Parcelable, Comparator<EarthQuakes> {  public static final Creator<EarthQuakes> CREATOR = new Creator<EarthQuakes>() {  @Override  public EarthQuakes createFromParcel(Parcel in) { return new EarthQuakes(in);  }  @Override  public EarthQuakes[] newArray(int size) { return new EarthQuakes[size];  }  };  @DatabaseField(id = true) private Long DateMilis;  @DatabaseField  private String LocationName;  @DatabaseField  private double Latitude;  @DatabaseField  private double Longitude;  @DatabaseField  private float Magnitude; |

* 1. @DatabaseField
  2. private float Depth;
  3. @DatabaseField
  4. private int Source;
  5. @DatabaseField
  6. private int Day;
  7. @DatabaseField
  8. private int Month; 56

57 public EarthQuakes() {

58

59 }

60

1. protected EarthQuakes(Parcel in) {
2. LocationName = in.readString();
3. Latitude = in.readDouble();
4. Longitude = in.readDouble();
5. Magnitude = in.readFloat();
6. Depth = in.readFloat();
7. Source = in.readInt();
8. Day = in.readInt();
9. Month = in.readInt(); 70 }

71

72 public static Long backDate() { 73

74 int value = AppSettings.getInstance().getTimeInterval();

75

76 int goBack = 0; 77

1. if (value == 0) {
2. goBack = 1;
3. } else if (value == 1) {
4. goBack = 7;
5. } else if (value == 2) {
6. goBack = 30;

84 }

85

1. Calendar cal = Calendar.getInstance();
2. cal.add(Calendar.DAY\_OF\_MONTH, ‐goBack);
3. return cal.getTimeInMillis(); 89 }

90

91 public void Insert() { 92

1. Calendar cal = Calendar.getInstance();
2. cal.setTimeInMillis(DateMilis);
3. Day = cal.get(Calendar.DAY\_OF\_MONTH);
4. Month = cal.get(Calendar.MONTH) + 1; 97
5. try {
6. Dao<EarthQuakes, Long> Missionsinsert = (DatabaseHelper.getDbHelper()).getEarthQuakesDataHelper();
7. EarthQuakes existenceCheck = Missionsinsert.queryForId(this.DateMilis); 101
8. if (existenceCheck != null) {
9. Missionsinsert.update(this);
10. } else {
11. Missionsinsert.create(this);

106 }

107

1. } catch (SQLException e) {
2. Tools.catchException(e);

110 }

111 }

112

113 public List<EarthQuakes> GetAllData() { 114

115 List<EarthQuakes> data = new ArrayList<>();

116

117 try {

118

1. Dao<EarthQuakes, Long> dao = DatabaseHelper.getDbHelper().getEarthQuakesDataHelper();
2. QueryBuilder<EarthQuakes, Long> qBuilder = dao.queryBuilder(); 121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157 }

158

int sortingType = AppSettings.getInstance().getSorting(); Long backdate = backDate();

if (AppSettings.getInstance().getSource() == 0) { qBuilder.where()//

.gt("Magnitude", AppSettings.getInstance().getMagnitude()) //

.and()//

.gt("DateMilis", backdate);

} else {

qBuilder.where()//

.eq("Source", AppSettings.getInstance().getSource()) //

.and()//

.gt("Magnitude", AppSettings.getInstance().getMagnitude()) //

.and()//

.gt("DateMilis", backdate);

}

if (sortingType == 0) { qBuilder.orderBy("DateMilis", true);

} else if (sortingType == 1) { qBuilder.orderBy("DateMilis", false);

} else if (sortingType == 2) { qBuilder.orderBy("Magnitude", true);

} else if (sortingType == 3) { qBuilder.orderBy("Magnitude", false);

}

PreparedQuery<EarthQuakes> pQuery = qBuilder.prepare(); data = dao.query(pQuery);

} catch (SQLException e) { Tools.catchException(e);

}

return data;

159 public Long GetLastEarthQuakeDate() { 160

161 List<EarthQuakes> data = new ArrayList<>();

162

163 try {

164

1. Dao<EarthQuakes, Long> dao = DatabaseHelper.getDbHelper().getEarthQuakesDataHelper();
2. QueryBuilder<EarthQuakes, Long> qBuilder = dao.queryBuilder(); 167

168 qBuilder.orderBy("DateMilis", false);

169

1. PreparedQuery<EarthQuakes> pQuery = qBuilder.prepare();
2. data = dao.query(pQuery); 172
3. } catch (SQLException e) {
4. Tools.catchException(e);

175 }

176

177 return data.get(0).getDateMilis();

178 }

179

180 public List<EarthQuakes> newEarthquakes() { 181

182 List<EarthQuakes> data = new ArrayList<>();

183

184 try {

185

1. Dao<EarthQuakes, Long> dao = DatabaseHelper.getDbHelper().getEarthQuakesDataHelper();
2. QueryBuilder<EarthQuakes, Long> qBuilder = dao.queryBuilder(); 188
3. if (AppSettings.getInstance().getSource() == 0) {
4. qBuilder.distinct().where()//
5. .gt("Magnitude", AppSettings.getInstance().getMagnitude()) //

192 .and()//

1. .gt("DateMilis", new LastEarthquakeDate().GetLastEarthquakeMilisDate());
2. } else {
3. qBuilder.distinct().where()//

201 }

202

.eq("Source", AppSettings.getInstance().getSource()) //

.and()//

.gt("Magnitude", AppSettings.getInstance().getMagnitude()) //

.and()//

.gt("DateMilis", new LastEarthquakeDate().GetLastEarthquakeMilisDate());

203

204

205

206

207

208

209

210

211

212

213 }

214

qBuilder.orderBy("DateMilis", false);

PreparedQuery<EarthQuakes> pQuery = qBuilder.prepare(); data = dao.query(pQuery);

} catch (SQLException e) { Tools.catchException(e);

}

return data;

1. public int GetRowCount() {
2. int count = 0; 217
3. try {
4. Dao<EarthQuakes, Long> dao = DatabaseHelper.getDbHelper().getEarthQuakesDataHelper();
5. count = (int) dao.countOf();
6. } catch (Exception e) {
7. Tools.catchException(e);

223 }

224

225 return count;

226 }

227

1. public void DeleteRow(int deleteId) {
2. try {
3. Dao<EarthQuakes, Long> dao = DatabaseHelper.getDbHelper().getEarthQuakesDataHelper();
4. DeleteBuilder<EarthQuakes, Long> deleteBuilder = dao.deleteBuilder();
5. deleteBuilder.where().eq("DateMilis", deleteId);
6. deleteBuilder.delete();
7. } catch (Exception e) {
8. Tools.catchException(e);

237 }

238 }

239

240 public EarthQuakes getEarthquakesById(Long DateMilis) { 241

242 List<EarthQuakes> eqList = new ArrayList<>();

243

1. try {
2. Dao<EarthQuakes, Long> dao = DatabaseHelper.getDbHelper().getEarthQuakesDataHelper();
3. QueryBuilder<EarthQuakes, Long> qBuilder = dao.queryBuilder();
4. qBuilder.distinct().where().eq("DateMilis", DateMilis);
5. PreparedQuery<EarthQuakes> pQuery = qBuilder.prepare();
6. eqList = dao.query(pQuery); 250
7. } catch (Exception e) {
8. Tools.catchException(e);

253 }

254

255 return eqList.get(0);

256 }

257

258 public List<EarthQuakes> getEarthquakesByDay(int day, int month) { 259

260 List<EarthQuakes> data = new ArrayList<>();

261

262 try {

263

1. Dao<EarthQuakes, Long> dao = DatabaseHelper.getDbHelper().getEarthQuakesDataHelper();
2. QueryBuilder<EarthQuakes, Long> qBuilder = dao.queryBuilder(); 266

267 int sortingType = AppSettings.getInstance().getSorting();

268

269 if (AppSettings.getInstance().getSource() == 0) {

|  |  |  |
| --- | --- | --- |
| 270 |  | qBuilder.where()// |
| 271 |  | .eq("Day", day).and().eq("Month", month) // |
| 272 |  | .and()// |
| 273 |  | .gt("Magnitude", AppSettings.getInstance().getMagnitude()); |
| 274 |  | } else { |
| 275 |  | qBuilder.where()// |
| 276 |  | .eq("Day", day).and().eq("Month", month) // |
| 277 |  | .and()// |
| 278 |  | .eq("Source", AppSettings.getInstance().getSource()) // |
| 279 |  | .and()// |
| 280 |  | .gt("Magnitude", AppSettings.getInstance().getMagnitude()); |
| 281 |  | } |
| 282 |  |  |
| 283 |  | if (sortingType == 0) { |
| 284 |  | qBuilder.orderBy("DateMilis", true); |
| 285 |  | } else if (sortingType == 1) { |
| 286 |  | qBuilder.orderBy("DateMilis", false); |
| 287 |  | } else if (sortingType == 2) { |
| 288 |  | qBuilder.orderBy("Magnitude", true); |
| 289 |  | } else if (sortingType == 3) { |
| 290 |  | qBuilder.orderBy("Magnitude", false); |
| 291 |  | } |
| 292 |  |  |
| 293 |  | PreparedQuery<EarthQuakes> pQuery = qBuilder.prepare(); |
| 294 |  | data = dao.query(pQuery); |
| 295 |  |  |
| 296 |  | } catch (SQLException e) { |
| 297 |  | Tools.catchException(e); |
| 298 |  | } |
| 299 |  |  |
| 300 |  | // try { |
| 301 |  | // Dao<EarthQuakes, Long> dao = DatabaseHelper.getDbHelper().getEarthQuakesDataHelper(); |
| 302 |  | // QueryBuilder<EarthQuakes, Long> qBuilder = dao.queryBuilder(); |
| 303 |  | // qBuilder.where().eq("Day", day).and().eq("Month", month); |
| 304 |  | // PreparedQuery<EarthQuakes> pQuery = qBuilder.prepare(); |
| 305 |  | // data = dao.query(pQuery); |
| 306 |  | // |
| 307 |  | // } |
| 308 |  | // catch (Exception e) { |
| 309 |  | // Tools.catchException(e); |
| 310 |  | // } |
| 311 |  |  |
| 312 |  | return data; |
| 313 | } |  |
| 314 |  |  |
| 315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342  343 | public List<EarthQuakes> getColumn(String ColumnName) throws SQLException { Dao<EarthQuakes, Long> dao = DatabaseHelper.getDbHelper().getEarthQuakesDataHelper();  List<EarthQuakes> results = dao.queryBuilder().distinct().selectColumns(ColumnName).quer return results;  }  public Long getDateMilis() { return DateMilis;  }  public void setDateMilis(Long dateMilis) { DateMilis = dateMilis;  }  public String getLocationName() { return LocationName;  }  public void setLocationName(String locationName) { LocationName = locationName;  }  public double getLatitude() { return Latitude;  }  public void setLatitude(double latitude) { Latitude = latitude;  } | |

y();

344

1. public double getLongitude() {
2. return Longitude;

347 }

348

1. public void setLongitude(double longitude) {
2. Longitude = longitude;

351 }

352

1. public float getMagnitude() {
2. return Magnitude;

355 }

356

1. public void setMagnitude(float magnitude) {
2. Magnitude = magnitude;

359 }

360

1. public float getDepth() {
2. return Depth;

363 }

364

1. public void setDepth(float depth) {
2. Depth = depth;

367 }

368

1. public int getSource() {
2. return Source;

371 }

372

1. public void setSource(int source) {
2. Source = source;

375 }

376

1. public int getDay() {
2. return Day;

379 }

380

1. public void setDay(int day) {
2. Day = day;

383 }

384

1. public int getMonth() {
2. return Month;

387 }

388

1. public void setMonth(int month) {
2. Month = month;

391 }

392

1. @Override
2. public int describeContents() {
3. // TODO Auto‐generated method stub
4. return 0;

397 }

398

1. @Override
2. public void writeToParcel(Parcel dest, int flags) {
3. // TODO Auto‐generated method stub 402
4. dest.writeString(LocationName);
5. dest.writeDouble(Latitude);
6. dest.writeDouble(Longitude);
7. dest.writeFloat(Magnitude);
8. dest.writeFloat(Depth);
9. dest.writeInt(Source);
10. dest.writeInt(Day);
11. dest.writeInt(Month);

411 }

412

413 @Override

414 public int compare(EarthQuakes lhs, EarthQuakes rhs) {

415 return (int) (lhs.getDateMilis() ‐ rhs.getDateMilis());

416 }

417

418 }

|  |  |
| --- | --- |
| **ENTTool\_database\_LastEarthquakeDate.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70 | package com.Uddhav.ENTTool.database;  import android.os.Parcel; import android.os.Parcelable;  import com.Uddhav.ENTTool.utils.Tools; import com.j256.ormlite.dao.Dao;  import com.j256.ormlite.field.DatabaseField; import com.j256.ormlite.table.DatabaseTable;  import java.sql.SQLException; import java.util.Comparator;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  @DatabaseTable(tableName = "LastEarthquakeDate")  // This is the annotation (@). It is just telling metadata information  // annotation line does not do anything in the class  public class LastEarthquakeDate implements Parcelable, Comparator<LastEarthquakeDate> {  public static final Creator<LastEarthquakeDate> CREATOR = new Creator<LastEarthquakeDate>() {  //Parceable interface includes the Creator interface. Parceable is contained in android.os.  //Creator<T>. T is any abstract parceable instance.  @Override  public LastEarthquakeDate createFromParcel(Parcel in) { return new LastEarthquakeDate(in);  }  // earthquake is included as a parcel. So, you just return an parcel object. Parcel means, like a bag.  @Override  public LastEarthquakeDate[] newArray(int size) { return new LastEarthquakeDate[size];  }  };  @DatabaseField(id = true) private int id;  @DatabaseField  private Long DateMilis;  public LastEarthquakeDate() { this.id = 1;  }  protected LastEarthquakeDate(Parcel in) { id = in.readInt();  }  public void Insert() { try {  Dao<LastEarthquakeDate, Integer> Missionsinsert = (DatabaseHelper.getDbHelper()).getLastEarthquakeDateDataHelper(); LastEarthquakeDate existenceCheck = Missionsinsert.queryForId(this.id); //retrieves an object associated with specific ID. exis  if (existenceCheck != null) { Missionsinsert.update(this);  } else {  Missionsinsert.create(this);  }  } catch (SQLException e) { Tools.catchException(e);  }  }  public Long GetLastEarthquakeMilisDate() { LastEarthquakeDate lastDate = null; try {  Dao<LastEarthquakeDate, Integer> dao = DatabaseHelper.getDbHelper().getLastEarthquakeDateDataHelper(); |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125 | | lastDate = dao.queryForId(1);  } catch (Exception e) { Tools.catchException(e);  }  return lastDate.getDateMilis();  }  public int GetRowCount() { int count = 0;  try {  Dao<LastEarthquakeDate, Integer> dao = DatabaseHelper.getDbHelper().getLastEarthquakeDateDataHelper(); //dao is to count the La count = (int) dao.countOf();  } catch (Exception e) { Tools.catchException(e);  }  return count;  }  public int getId() { return id;  }  public void setId(int id) { this.id = id;  }  public Long getDateMilis() { return DateMilis;  }  public void setDateMilis(Long dateMilis) { DateMilis = dateMilis;  }  @Override  public int describeContents() {  // TODO Auto‐generated method stub return 0;  }  @Override  public void writeToParcel(Parcel dest, int flags) {  // TODO Auto‐generated method stub  dest.writeInt(id);  }  @Override  public int compare(LastEarthquakeDate lhs, LastEarthquakeDate rhs) { return (int) (lhs.DateMilis ‐ rhs.DateMilis);  }  } | | | |
|  |  | |  |  |  |

|  |  |
| --- | --- |
| **ENTTool\_MainActivity.java** | |
| 1 | package com.Uddhav.ENTTool; |
| 2 |  |
| 3 | import android.app.ProgressDialog; |
| 4 | import android.content.Intent; |
| 5 | import android.graphics.drawable.ColorDrawable; |
| 6 | import android.os.Bundle; |
| 7 | import android.support.v7.app.ActionBar; |
| 8 | import android.support.v7.app.AppCompatActivity; |
| 9 | import android.util.Log; |
| 10 | import android.view.Menu; |
| 11 | import android.view.MenuItem; |
| 12 | import android.view.View; |
| 13 | import android.widget.AbsListView; |
| 14 | import android.widget.AbsListView.OnScrollListener; |
| 15 | import android.widget.AdapterView; |

1. import android.widget.AdapterView.OnItemClickListener;
2. import android.widget.ListView;
3. import android.widget.TextView; 19
4. import com.Uddhav.ENTTool.adapters.ListviewAdapter;
5. import com.Uddhav.ENTTool.database.EarthQuakes;
6. import com.Uddhav.ENTTool.database.LastEarthquakeDate;
7. import com.Uddhav.ENTTool.utils.App;
8. import com.Uddhav.ENTTool.utils.AppSettings;
9. import com.Uddhav.ENTTool.utils.EBus;
10. import com.Uddhav.ENTTool.utils.SyncService;
11. import com.Uddhav.ENTTool.utils.Tools;
12. import com.squareup.otto.Subscribe;
13. import com.startapp.android.publish.StartAppAd;
14. import com.startapp.android.publish.StartAppSDK; 31

32 import java.util.List;

33

34 /\*\*

35 \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu) 36 \*/

37 public class MainActivity extends AppCompatActivity implements OnItemClickListener, OnScrollListener { 38

1. private ProgressDialog pd;
2. private ListView list;
3. private int currentScrollState, currentFirstVisibleItem, currentVisibleItemCount, currentTotalItemCount;
4. private ListviewAdapter adapter;
5. private StartAppAd startAppAd;
6. private TextView tvEmptyMessage; 45

46 private boolean isConnectToInternet = true;

47

1. @Override
2. public void onCreate(Bundle savedInstanceState) {
3. super.onCreate(savedInstanceState); 51
4. StartAppSDK.init(this, getString(R.string.StartApp\_AccountId), getString(R.string.StartApp\_AppId), true);
5. // StartAppSDK is decompiled class file. Two blank string means, it would take default AccountID and AppID.
6. // Since, this file is already decompiled, I can't see the Java Docs. 55

56 ActionBar ab = getSupportActionBar(); // for icon, title etc. Retrieves a references for this activity actionbar

57

58 ab.setDisplayShowHomeEnabled(true); // for icon. Caz, icon, here, is used as CustomView 59

1. ab.setIcon(R.mipmap.ic\_launcher);
2. ab.setTitle(" " + getResources().getString(R.string.app\_name));// "string", here, is another inner class of R, auto generated clas 62
3. ColorDrawable cd = new ColorDrawable(getResources().getColor(R.color.statusbar));
4. //Drawable means anything that can be drawn. ColorDrawable means like an ink to fill the colors. ColorDrawable is
5. // defined in color.xml file in the resource directory in Android.
6. // Syntax: ColorDrawable(Object\_to\_draw\_color().getColor(COLOR)) 67

68

69 ab.setBackgroundDrawable(cd); //It makes ActionBar is drawable with cd ink. Top title bar is the ActionBar 70

1. ab.setDisplayShowTitleEnabled(false);
2. ab.setDisplayShowTitleEnabled(true); //title enabled display is set. Means, title/subtitle are enabled 73
3. setContentView(R.layout.activity\_main);
4. //AppCompatActivity.setContentView();. This sets the activity content from the layout resource
5. // Activity.setContentView(activity\_main.xml's ID) is used to set the layout for the MainActivity.
6. // All layout resources are defined in activity\_main.xml file, by default. 78
7. AppSettings.setDefaultSettings();
8. // I can see the classes residing in child folders
9. // AppSettings is Uddhav created class. So, no Java Doc 82
10. if (new LastEarthquakeDate().GetRowCount() == 0) {
11. LastEarthquakeDate led = new LastEarthquakeDate();
12. led.setDateMilis(606175200000l);
13. led.Insert();// either creates or updates the ID for this led object.
14. // Note: when new things are created, Android Studio associates that thing with an his ID. 88 }

89

90 startAppAd = new StartAppAd(this); // for the Publish

|  |  |  |
| --- | --- | --- |
| 91  92  93  94  95  96  97  98  99  100  101  102  103 | // Banner banner = (com.startapp.android.publish.banner.Banner) findViewB  // banner.showBanner();  startAppAd.showAd(); //shows Ad, Means, AdddDisplayListener, and returns startAppAd.loadAd(); //loads the displayListener and returns the boolean  tvEmptyMessage = (TextView) findViewById(R.id.tv\_empty\_message);  list = (ListView) findViewById(R.id.list2); //adds ListView in this, Main  // earthquakes record in GUI  // The below two lines give error. I couldn't complete this in time, beca | |
| 104 | // | list.setOnItemClickListener(this); // adds listeners on that ListView |
| 105 | // | list.setOnScrollListener(this); // |
| 106 |  |  |
| 107 |  | if (!SyncService.isServiceRunning) { |
| 108 |  | Log.i("MainActivity", "Servis Started"); |
| 109 |  | Intent intent = new Intent(getBaseContext(), SyncService.class); |
| 110 |  | startService(intent); |
| 112 |  | pd = new ProgressDialog(MainActivity.this); |
| 113 |  | pd.setProgressStyle(ProgressDialog.STYLE\_SPINNER); |
| 114 |  | pd.setTitle(getString(R.string.PleaseWait)); |
| 115 |  | pd.setMessage(getString(R.string.DatasLoading)); |
| 116 |  | pd.setCancelable(false); |
| 117 |  | pd.setIndeterminate(false); |
| 118 |  | pd.show(); |
| 119 |  | } |
| 120 |  |  |
| 121 |  | } |
| 122 |  |  |
| 123 |  | @Override |
| 124 |  | protected void onResume() { |
| 125 |  | super.onResume(); |
| 126 |  |  |
| 127 |  | if (isConnectToInternet) { |
| 128 |  | List<EarthQuakes> EarthQuakeList = new EarthQuakes().GetAllData(); |
| 129 |  |  |
| 130 |  | if (EarthQuakeList.size() > 0) { |
| 131 |  | adapter = new ListviewAdapter(MainActivity.this, EarthQuakeList); |
| 132 |  | adapter.notifyDataSetChanged(); |
| 133 |  | list.setAdapter(adapter); |
| 134 |  | list.setSelectionFromTop(currentFirstVisibleItem, 0); |
| 135 |  | } |
| 136 |  | } |
| 137 |  |  |
| 138 |  | } |
| 139 |  |  |
| 140 |  | @Subscribe |
| 141 |  | public void messageReceived(EBus event) { |
| 142 |  | Log.i("MainActivity", "Triggered! " + event.getStatus()); |
| 143 |  |  |
| 144 |  | if (event.getStatus() == 999) { |
| 145 |  | isConnectToInternet = false; |
| 146 |  | list.setEmptyView(tvEmptyMessage); |
| 147 |  | list.setAdapter(null); |
| 148 |  | } else { |
| 149 |  | isConnectToInternet = true; |
| 150 |  | List<EarthQuakes> EarthQuakeList = new EarthQuakes().GetAllData(); |
| 152 |  | if (EarthQuakeList.size() > 0) { |
| 153 |  | adapter = new ListviewAdapter(MainActivity.this, EarthQuakeList); |
| 154 |  | adapter.notifyDataSetChanged(); |
| 155 |  | list.setAdapter(adapter); |
| 156 |  | list.setSelectionFromTop(currentFirstVisibleItem, 0); |
| 157 |  | } |
| 158 |  |  |
| 159 |  | } |
| 160 |  |  |
| 161 |  | if (pd != null && pd.isShowing()) { |
| 162 |  | pd.dismiss(); |
| 163 |  | pd = null; |
| 164 |  | } |

yId(R.id.startAppBanner);

boolean

Activity. This list is for storing

use I have presentation in 3 days.

111

151

165 }

166

1. @Override
2. public boolean onCreateOptionsMenu(Menu menu) {
3. getMenuInflater().inflate(R.menu.activity\_main, menu);
4. return true;

171 }

172

1. @Override
2. public boolean onOptionsItemSelected(MenuItem item) { 175

176 if (item.getItemId() == R.id.action\_main) {

177

1. Intent i1 = new Intent(MainActivity.this, SettingsActivity.class);
2. startActivity(i1); 180
3. return true;
4. } else if (item.getItemId() == R.id.activity\_about) {
5. Intent i2 = new Intent(MainActivity.this, aboutActivity.class);
6. startActivity(i2); 185

186 return true;

187 }

188 return super.onOptionsItemSelected(item);

189 }

190

1. @Override
2. public void onScrollStateChanged(AbsListView view, int scrollState) {
3. this.currentScrollState = scrollState;
4. this.isScrollCompleted();

195 }

196

1. @Override
2. public void onScroll(AbsListView view, int firstVisibleItem, int visibleItemCount, int totalItemCount) {
3. this.currentFirstVisibleItem = firstVisibleItem;
4. this.currentVisibleItemCount = visibleItemCount;
5. this.currentTotalItemCount = totalItemCount;

202 }

203

204 private void isScrollCompleted() { 205

1. if (currentFirstVisibleItem + currentVisibleItemCount >= currentTotalItemCount) {
2. if (this.currentVisibleItemCount > 0 && this.currentScrollState == OnScrollListener.SCROLL\_STATE\_IDLE) { 208

209 }

210 }

211 }

212

1. @Override
2. public void onItemClick(AdapterView<?> parent1, View view, int position, long id) {
3. try {
4. EarthQuakes eq = (EarthQuakes) parent1.getAdapter().getItem(position); 217
5. Intent i = new Intent(MainActivity.this, com.Uddhav.ENTTool.MapsActivity.class);
6. i.putExtra("selectedItem", eq.getDateMilis());
7. startActivity(i);
8. } catch (Exception e) {
9. Tools.catchException(e);

223 }

224 }

225

1. @Override
2. protected void onStart() {
3. super.onStart();
4. App.bus.register(this);

230 }

231

1. @Override
2. protected void onStop() {
3. super.onStop();
4. App.bus.unregister(this);

236 }

237

238 @Override



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 239 | | } | public void onBackPressed() { startAppAd.onBackPressed(); super.onBackPressed();  } |  |  |  |
| 240 | |
| 241 | |
| 242 | |
| 243 | |
| 244 | |
|  |  | | |  |  |  |

|  |  |
| --- | --- |
| **ENTTool\_MapsActivity.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64 | package com.Uddhav.ENTTool;  import android.app.Dialog; import android.os.Bundle;  import android.support.v4.app.FragmentActivity; import android.widget.Toast;  import com.Uddhav.ENTTool.adapters.MarkerInfoAdapter; import com.Uddhav.ENTTool.database.EarthQuakes; import com.Uddhav.ENTTool.utils.AppSettings;  import com.Uddhav.ENTTool.utils.Tools;  import com.google.android.gms.common.ConnectionResult; import com.google.android.gms.common.GooglePlayServicesUtil; import com.google.android.gms.maps.CameraUpdateFactory; import com.google.android.gms.maps.GoogleMap;  import com.google.android.gms.maps.SupportMapFragment;  import com.google.android.gms.maps.model.BitmapDescriptorFactory; import com.google.android.gms.maps.model.LatLng;  import com.google.android.gms.maps.model.Marker; import com.google.android.gms.maps.model.MarkerOptions;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class MapsActivity extends FragmentActivity implements GoogleMap.OnMapClickListener, GoogleMap.OnMapLongClickListener, GoogleMap.OnM  private GoogleMap myMap;  private String Depth, Lat, Lng, Loc, Mag, Date; private MarkerInfoAdapter infoAdapter;  private long itemId;  @Override  protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_maps);  itemId = getIntent().getLongExtra("selectedItem", 0); if (itemId == 0) {  finish();  }  setUpMapIfNeeded();  }  @Override  protected void onResume() { super.onResume(); setUpMapIfNeeded();  }  private void setUpMapIfNeeded() { if (myMap == null) {  myMap = ((SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map)).getMap(); if (myMap != null) {  int resultCode = GooglePlayServicesUtil.isGooglePlayServicesAvailable(getApplicationContext()); if (resultCode != ConnectionResult.SUCCESS) {  Dialog dialog = GooglePlayServicesUtil.getErrorDialog(resultCode, this, 1); dialog.show();  return;  } else {  setUpMap(); |

65 }

66 }

67 }

68 }

69

70 private void setUpMap() { 71

1. try {
2. int mapType = AppSettings.getInstance().getMapType(); 74
3. if (mapType == 0) {
4. myMap.setMapType(GoogleMap.MAP\_TYPE\_NORMAL);
5. } else if (mapType == 1) {
6. myMap.setMapType(GoogleMap.MAP\_TYPE\_SATELLITE);
7. } else if (mapType == 2) {
8. myMap.setMapType(GoogleMap.MAP\_TYPE\_HYBRID);
9. } else if (mapType == 3) {
10. myMap.setMapType(GoogleMap.MAP\_TYPE\_TERRAIN); 83 }

84

1. infoAdapter = new MarkerInfoAdapter(MapsActivity.this.getLayoutInflater());
2. myMap.clear();
3. myMap.setInfoWindowAdapter(infoAdapter);
4. myMap.setMyLocationEnabled(true);
5. myMap.setOnMapClickListener(this);
6. myMap.setOnMapLongClickListener(this);
7. myMap.setOnMarkerClickListener(this);
8. myMap.setOnMarkerDragListener(this); 93

94 EarthQuakes currentEarthquakes = new EarthQuakes().getEarthquakesById(itemId);

95

96 for (EarthQuakes earthQuakes : new EarthQuakes().GetAllData()) { 97

1. LatLng position = new LatLng(earthQuakes.getLatitude(), earthQuakes.getLongitude());
2. MarkerOptions mOptions = new MarkerOptions(); 100
3. mOptions.position(position);
4. mOptions.snippet(Long.toString(earthQuakes.getDateMilis()));
5. mOptions.visible(true); 104

105 float magnitude = earthQuakes.getMagnitude();

106

1. if (magnitude < 3) {
2. mOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE\_GREEN));
3. } else if (magnitude >= 3 && magnitude < 5) {
4. mOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE\_YELLOW));
5. } else if (magnitude >= 5) {
6. mOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE\_RED));

113 }

114

1. if (earthQuakes.getDateMilis().equals(currentEarthquakes.getDateMilis())) {
2. myMap.moveCamera(CameraUpdateFactory.newLatLngZoom(position, 6));
3. Marker marker = myMap.addMarker(mOptions);
4. marker.showInfoWindow();
5. } else {
6. myMap.addMarker(mOptions);

121 }

122

123 }

124

1. } catch (Exception e) {
2. Tools.catchException(e);
3. Toast.makeText(this, getString(R.string.MapCanNotBeDisplayed), Toast.LENGTH\_LONG).show();
4. finish();

129 }

130

131 }

132

1. @Override
2. public void onMapClick(LatLng latLng) { 135

136 }

137

138 @Override

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 139 | | } | public void onMapLongClick(LatLng latLng) {  }  @Override  public boolean onMarkerClick(Marker marker) { return false;  }  @Override  public void onMarkerDragStart(Marker marker) {  }  @Override  public void onMarkerDrag(Marker marker) {  }  @Override  public void onMarkerDragEnd(Marker marker) {  }  @Override  public void onBackPressed() { finish();  } | |  |  |
| 140 | |
| 141 | |
| 142 | |
| 143 | |
| 144 | |
| 145 | |
| 146 | |
| 147 | |
| 148 | |
| 149 | |
| 150 | |
| 151 | |
| 152 | |
| 153 | |
| 154 | |
| 155 | |
| 156 | |
| 157 | |
| 158 | |
| 159 | |
| 160 | |
| 161 | |
| 162 | |
| 163 | |
| 164 | |
| 165 | |
| 166 | |
| 167 | |
| 168 | |
|  |  | | |  |  |  |

|  |  |
| --- | --- |
| **ENTTool\_SettingsActivity.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | package com.Uddhav.ENTTool;  import android.content.SharedPreferences; import android.os.Bundle;  import android.preference.CheckBoxPreference; import android.preference.EditTextPreference; import android.preference.ListPreference;  import android.preference.MultiSelectListPreference; import android.preference.Preference;  import android.preference.Preference.OnPreferenceChangeListener; import android.preference.PreferenceActivity;  import android.preference.PreferenceGroup;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class SettingsActivity extends PreferenceActivity implements OnPreferenceChangeListener, SharedPreferences.OnSharedPreferenceChangeL  private String Key\_TimeInterval, Key\_Source, Key\_Magnitude;  private String Key\_UpdatePeriod, Key\_Sorting, Key\_Notifications, Key\_Vibration, Key\_Sound;  private ListPreference lpTimeInterval, lpSource, lpMagnitude, lpUpdatePeriod, lpSorting; private CheckBoxPreference cbNotifications, cbVibration, cbSound;  @SuppressWarnings("deprecation")  @Override  protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); addPreferencesFromResource(R.xml.pref);  Key\_TimeInterval = getResources().getString(R.string.listPref\_Key\_TimeInterval); Key\_Source = getResources().getString(R.string.listPref\_Key\_Source); Key\_Magnitude = getResources().getString(R.string.listPref\_Key\_Magnitude); Key\_UpdatePeriod = getResources().getString(R.string.listPref\_Key\_UpdatePeriod); Key\_Sorting = getResources().getString(R.string.listPref\_Key\_Sorting);  Key\_Notifications = getResources().getString(R.string.CheckBoxPref\_Key\_Notifications); Key\_Vibration = getResources().getString(R.string.CheckBoxPref\_Key\_Vibration); Key\_Sound = getResources().getString(R.string.CheckBoxPref\_Key\_Sound);  lpTimeInterval = (ListPreference) findPreference(Key\_TimeInterval); |

1. lpSource = (ListPreference) findPreference(Key\_Source);
2. lpMagnitude = (ListPreference) findPreference(Key\_Magnitude);
3. lpUpdatePeriod = (ListPreference) findPreference(Key\_UpdatePeriod);
4. lpSorting = (ListPreference) findPreference(Key\_Sorting);
5. cbNotifications = (CheckBoxPreference) findPreference(Key\_Notifications);
6. cbVibration = (CheckBoxPreference) findPreference(Key\_Vibration);
7. cbSound = (CheckBoxPreference) findPreference(Key\_Sound); 48
8. lpTimeInterval.setOnPreferenceChangeListener(this);
9. lpSource.setOnPreferenceChangeListener(this);
10. lpMagnitude.setOnPreferenceChangeListener(this);
11. lpUpdatePeriod.setOnPreferenceChangeListener(this);
12. // lpSorting.setOnPreferenceChangeListener(this);
13. cbNotifications.setOnPreferenceChangeListener(this);
14. cbVibration.setOnPreferenceChangeListener(this);
15. cbSound.setOnPreferenceChangeListener(this); 57

58 initSummary(getPreferenceScreen());

59

1. if (cbNotifications.isChecked()) {
2. cbVibration.setEnabled(true);
3. cbSound.setEnabled(true);
4. } else {
5. cbVibration.setEnabled(false);
6. cbSound.setEnabled(false); 66 }

67

68 }

69

1. @Override
2. protected void onResume() {
3. super.onResume();
4. getPreferenceScreen().getSharedPreferences().registerOnSharedPreferenceChangeListener(this); 74 }

75

1. @Override
2. protected void onPause() {
3. super.onPause();
4. getPreferenceScreen().getSharedPreferences().unregisterOnSharedPreferenceChangeListener(this); 80 }

81

1. @Override
2. public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String key) {
3. updatePrefSummary(findPreference(key)); 85 }

86

1. private void initSummary(Preference p) {
2. if (p instanceof PreferenceGroup) {
3. PreferenceGroup pGrp = (PreferenceGroup) p;
4. for (int i = 0; i < pGrp.getPreferenceCount(); i++) {
5. initSummary(pGrp.getPreference(i)); 92 }
6. } else {
7. updatePrefSummary(p); 95 }

96 }

97

1. private void updatePrefSummary(Preference p) {
2. if (p instanceof ListPreference) {
3. ListPreference listPref = (ListPreference) p;
4. p.setSummary(listPref.getEntry());

102 }

1. if (p instanceof EditTextPreference) {
2. EditTextPreference editTextPref = (EditTextPreference) p;
3. if (p.getTitle().toString().contains("password")) {

106 p.setSummary("\*\*\*\*\*\*");

1. } else {
2. p.setSummary(editTextPref.getText());

109 }

110 }

1. if (p instanceof MultiSelectListPreference) {
2. EditTextPreference editTextPref = (EditTextPreference) p;
3. p.setSummary(editTextPref.getText());

114 }

115 if (p instanceof CheckBoxPreference) {



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172 | | CheckBoxPreference listPref = (CheckBoxPreference) p;  p.setSummary(listPref.isChecked() ? getResources().getString(R.string.statu\_on) : getResources().getString(R.string.statu\_off))  }  }  @Override  public boolean onPreferenceChange(Preference preference, Object newValue) {  String key = preference.getKey(); String value = newValue.toString();  if (key.equalsIgnoreCase(Key\_TimeInterval)) { lpTimeInterval.setSummary(lpTimeInterval.getEntries()[Integer.parseInt(value)]);  } else if (key.equalsIgnoreCase(Key\_Source)) { lpSource.setSummary(lpSource.getEntries()[Integer.parseInt(value)]);  } else if (key.equalsIgnoreCase(Key\_Magnitude)) { lpMagnitude.setSummary(lpMagnitude.getEntries()[Integer.parseInt(value)]);  } else if (key.equalsIgnoreCase(Key\_UpdatePeriod)) { lpUpdatePeriod.setSummary(lpUpdatePeriod.getEntries()[Integer.parseInt(value)]);  } else if (key.equalsIgnoreCase(Key\_Sorting)) { lpSorting.setSummary(lpSorting.getEntries()[Integer.parseInt(value)]);  } else if (key.equalsIgnoreCase(Key\_Notifications)) {  if (value.equals("true")) { cbNotifications.setSummary(getResources().getString(R.string.statu\_on)); cbVibration.setEnabled(true);  cbSound.setEnabled(true);  } else {  cbNotifications.setSummary(getResources().getString(R.string.statu\_off)); cbVibration.setEnabled(false);  cbVibration.setChecked(false); cbVibration.setSummary(getResources().getString(R.string.statu\_off)); cbSound.setEnabled(false);  cbSound.setChecked(false); cbSound.setSummary(getResources().getString(R.string.statu\_off));  }  } else if (key.equalsIgnoreCase(Key\_Vibration)) { if (value.equals("true")) {  cbVibration.setSummary(getResources().getString(R.string.statu\_on));  } else {  cbVibration.setSummary(getResources().getString(R.string.statu\_off));  }  } else if (key.equalsIgnoreCase(Key\_Sound)) {  if (value.equals("true")) { cbSound.setSummary(getResources().getString(R.string.statu\_on));  } else {  cbSound.setSummary(getResources().getString(R.string.statu\_off));  }  }  return true;  }  } | | | |
|  |  | |  |  |  |

|  |  |
| --- | --- |
| **ENTTool\_sources\_seismicportal\_features.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | package com.Uddhav.ENTTool.sources.seismicportal;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class features<S, T> {  private S geometry; private String type; private String id; private T properties;  public S getGeometry() { |

|  |  |  |
| --- | --- | --- |
| 14 | } | return geometry;  }  public void setGeometry(S geometry) { this.geometry = geometry;  }  public String getType() { return type;  }  public void setType(String type) { this.type = type;  }  public String getId() { return id;  }  public void setId(String id) { this.id = id;  }  public T getProperties() { return properties;  }  public void setProperties(T properties) { this.properties = properties;  } |
| 15 |
| 16 |
| 17 |
| 18 |
| 19 |
| 20 |
| 21 |
| 22 |
| 23 |
| 24 |
| 25 |
| 26 |
| 27 |
| 28 |
| 29 |
| 30 |
| 31 |
| 32 |
| 33 |
| 34 |
| 35 |
| 36 |
| 37 |
| 38 |
| 39 |
| 40 |
| 41 |
| 42 |
| 43 |
| 44 |
| 45 |

|  |  |
| --- | --- |
| **ENTTool\_sources\_seismicportal\_geometry.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | package com.Uddhav.ENTTool.sources.seismicportal; import java.util.List;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class geometry {  private String type;  private List<Float> coordinates;  public String getType() { return type;  }  public void setType(String type) { this.type = type;  }  public List<Float> getCoordinates() { return coordinates;  }  public void setCoordinates(List<Float> coordinates) { this.coordinates = coordinates;  }  } |

|  |  |
| --- | --- |
| **ENTTool\_sources\_seismicportal\_object.java** | |
| 1  2  3  4  5  6 | package com.Uddhav.ENTTool.sources.seismicportal;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class object<S, T> { |

|  |  |  |
| --- | --- | --- |
| 7 | } | private String type; private S metadata; private T features;  public String getType() { return type;  }  public void setType(String type) { this.type = type;  }  public S getMetadata() { return metadata;  }  public void setMetadata(S metadata) { this.metadata = metadata;  }  public T getFeatures() { return features;  }  public void setFeatures(T features) { this.features = features;  } |
| 8 |
| 9 |
| 10 |
| 11 |
| 12 |
| 13 |
| 14 |
| 15 |
| 16 |
| 17 |
| 18 |
| 19 |
| 20 |
| 21 |
| 22 |
| 23 |
| 24 |
| 25 |
| 26 |
| 27 |
| 28 |
| 29 |
| 30 |
| 31 |
| 32 |
| 33 |
| 34 |
| 35 |
| 36 |

|  |  |
| --- | --- |
| **ENTTool\_sources\_seismicportal\_properties.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | package com.Uddhav.ENTTool.sources.seismicportal;  /\*\*   * Created by Uddhav Gautam on 7.3.2016. * [upgautam@ualr.edu](mailto:upgautam@ualr.edu)   \*/  import java.util.Date; public class properties {  private Date lastupdate; private String magtype; private String evtype; private double lon; private String auth; private double lat; private float depth; private String unid; private float mag; private Date time; private String source\_id;  private String source\_catalog; private String flynn\_region;  public Date getLastupdate() { return lastupdate;  }  public void setLastupdate(Date lastupdate) { this.lastupdate = lastupdate;  }  public String getMagtype() { return magtype;  }  public void setMagtype(String magtype) { this.magtype = magtype;  } |

1. public String getEvtype() {
2. return evtype; 44 }

45

1. public void setEvtype(String evtype) {
2. this.evtype = evtype; 48 }

49

1. public double getLon() {
2. return lon; 52 }

53

1. public void setLon(double lon) {
2. this.lon = lon; 56 }

57

1. public String getAuth() {
2. return auth; 60 }

61

1. public void setAuth(String auth) {
2. this.auth = auth; 64 }

65

1. public double getLat() {
2. return lat; 68 }

69

1. public void setLat(double lat) {
2. this.lat = lat; 72 }

73

1. public float getDepth() {
2. return depth; 76 }

77

1. public void setDepth(float depth) {
2. this.depth = depth; 80 }

81

1. public String getUnid() {
2. return unid; 84 }

85

1. public void setUnid(String unid) {
2. this.unid = unid; 88 }

89

1. public float getMag() {
2. return mag; 92 }

93

1. public void setMag(float mag) {
2. this.mag = mag; 96 }

97

1. public Date getTime() {
2. return time;

100 }

101

1. public void setTime(Date time) {
2. this.time = time;

104 }

105

1. public String getSource\_id() {
2. return source\_id;

108 }

109

1. public void setSource\_id(String source\_id) {
2. this.source\_id = source\_id;

112 }

113

1. public String getSource\_catalog() {
2. return source\_catalog;

116 }

|  |  |  |
| --- | --- | --- |
| 117 | } | public void setSource\_catalog(String source\_catalog) { this.source\_catalog = source\_catalog;  }  public String getFlynn\_region() { return flynn\_region;  }  public void setFlynn\_region(String flynn\_region) { this.flynn\_region = flynn\_region;  } |
| 118 |
| 119 |
| 120 |
| 121 |
| 122 |
| 123 |
| 124 |
| 125 |
| 126 |
| 127 |
| 128 |
| 129 |
| 130 |

|  |  |
| --- | --- |
| **ENTTool\_sources\_seismicportal\_totalCount.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | package com.Uddhav.ENTTool.sources.seismicportal;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class totalCount {  private int totalCount; public int getTotalCount() {  return totalCount;  }  public void setTotalCount(int totalCount) { this.totalCount = totalCount;  }  } |

|  |  |
| --- | --- |
| **ENTTool\_sources\_usgs\_features1.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | package com.Uddhav.ENTTool.sources.usgs;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class features1<P, G> {  private String type; private P properties; private G geometry; private String id;  public String getType() { return type;  }  public void setType(String type) { this.type = type;  }  public P getProperties() { return properties;  }  public void setProperties(P properties) { this.properties = properties;  }  public G getGeometry() { return geometry;  }  public void setGeometry(G geometry) { this.geometry = geometry;  } |

|  |  |  |
| --- | --- | --- |
| 36 | } | public String getId() { return id;  }  public void setId(String id) { this.id = id;  } |
| 37 |
| 38 |
| 39 |
| 40 |
| 41 |
| 42 |
| 43 |
| 44 |
| 45 |

|  |  |
| --- | --- |
| **ENTTool\_sources\_usgs\_geometry1.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | package com.Uddhav.ENTTool.sources.usgs;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  import java.util.List; public class geometry1 {  private String type;  private List<Float> coordinates;  public String getType() { return type;  }  public void setType(String type) { this.type = type;  }  public List<Float> getCoordinates() { return coordinates;  }  public void setCoordinates(List<Float> coordinates) { this.coordinates = coordinates;  }  } |

|  |  |
| --- | --- |
| **ENTTool\_sources\_usgs\_metadata1.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | package com.Uddhav.ENTTool.sources.usgs;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class metadata1 {  private long generated; private String url; private String subTitle; private int cacheMaxAge;  public long getGenerated() { return generated;  }  public void setGenerated(long generated) { this.generated = generated;  }  public String getUrl() { return url;  }  public void setUrl(String url) { this.url = url;  } |

|  |  |  |
| --- | --- | --- |
| 28 | } | public String getSubTitle() { return subTitle;  }  public void setSubTitle(String subTitle) { this.subTitle = subTitle;  }  public int getCacheMaxAge() { return cacheMaxAge;  }  public void setCacheMaxAge(int cacheMaxAge) { this.cacheMaxAge = cacheMaxAge;  } |
| 29 |
| 30 |
| 31 |
| 32 |
| 33 |
| 34 |
| 35 |
| 36 |
| 37 |
| 38 |
| 39 |
| 40 |
| 41 |
| 42 |
| 43 |
| 44 |
| 45 |

|  |  |
| --- | --- |
| **ENTTool\_sources\_usgs\_properties1.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53 | package com.Uddhav.ENTTool.sources.usgs;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class properties1 {  private float mag; private String place; private String time; private String updated; private int tz;  private String url; private String status; private String sig; private String net; private String code; private String ids; private String sources; private String types; private String gap;  private String magnitudeType;  private Object nst; private Object dmin; private Object rms; private Object felt; private Object cdi; private Object mmi; private Object alert; private Object tsunami;  public float getMag() { return mag;  }  public void setMag(float mag) { this.mag = mag;  }  public String getPlace() { return place;  }  public void setPlace(String place) { this.place = place;  }  public String getTime() { return time;  }  public void setTime(String time) { |

54

55 }

56

this.time = time;

* 1. public String getUpdated() {
  2. return updated; 59 }

60

1. public void setUpdated(String updated) {
2. this.updated = updated; 63 }

64

1. public int getTz() {
2. return tz; 67 }

68

1. public void setTz(int tz) {
2. this.tz = tz; 71 }

72

1. public String getUrl() {
2. return url; 75 }

76

1. public void setUrl(String url) {
2. this.url = url; 79 }

80

1. public String getStatus() {
2. return status; 83 }

84

1. public void setStatus(String status) {
2. this.status = status; 87 }

88

1. public String getSig() {
2. return sig; 91 }

92

1. public void setSig(String sig) {
2. this.sig = sig; 95 }

96

1. public String getNet() {
2. return net; 99 }

100

1. public void setNet(String net) {
2. this.net = net;

103 }

104

1. public String getCode() {
2. return code;

107 }

108

1. public void setCode(String code) {
2. this.code = code;

111 }

112

1. public String getIds() {
2. return ids;

115 }

116

1. public void setIds(String ids) {
2. this.ids = ids;

119 }

120

1. public String getSources() {
2. return sources;

123 }

124

1. public void setSources(String sources) {
2. this.sources = sources;

127 }

128

1. public String getTypes() {
2. return types;

131 }

132

1. public void setTypes(String types) {
2. this.types = types;

135 }

136

1. public String getGap() {
2. return gap;

139 }

140

1. public void setGap(String gap) {
2. this.gap = gap;

143 }

144

1. public String getMagnitudeType() {
2. return magnitudeType;

147 }

148

1. public void setMagnitudeType(String magnitudeType) {
2. this.magnitudeType = magnitudeType;

151 }

152

1. public Object getNst() {
2. return nst;

155 }

156

1. public void setNst(Object nst) {
2. this.nst = nst;

159 }

160

1. public Object getDmin() {
2. return dmin;

163 }

164

1. public void setDmin(Object dmin) {
2. this.dmin = dmin;

167 }

168

1. public Object getRms() {
2. return rms;

171 }

172

1. public void setRms(Object rms) {
2. this.rms = rms;

175 }

176

1. public Object getFelt() {
2. return felt;

179 }

180

1. public void setFelt(Object felt) {
2. this.felt = felt;

183 }

184

1. public Object getCdi() {
2. return cdi;

187 }

188

1. public void setCdi(Object cdi) {
2. this.cdi = cdi;

191 }

192

1. public Object getMmi() {
2. return mmi;

195 }

196

1. public void setMmi(Object mmi) {
2. this.mmi = mmi;

199 }

200

201 public Object getAlert() {

|  |  |  |
| --- | --- | --- |
| 202 | } | return alert;  }  public void setAlert(Object alert) { this.alert = alert;  }  public Object getTsunami() { return tsunami;  }  public void setTsunami(Object tsunami) { this.tsunami = tsunami;  } |
| 203 |
| 204 |
| 205 |
| 206 |
| 207 |
| 208 |
| 209 |
| 210 |
| 211 |
| 212 |
| 213 |
| 214 |
| 215 |
| 216 |
| 217 |

|  |  |
| --- | --- |
| **ENTTool\_sources\_usgs\_usgs.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | package com.Uddhav.ENTTool.sources.usgs; import java.util.List;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class usgs<M, F> {  private String type; private M metadata; private List<F> features;  public String getType() { return type;  }  public void setType(String type) { this.type = type;  }  public M getMetadata() { return metadata;  }  public void setMetadata(M metadata) { this.metadata = metadata;  }  public List<F> getFeatures() { return features;  }  public void setFeatures(List<F> features) { this.features = features;  }  } |

|  |  |
| --- | --- |
| **ENTTool\_utils\_App.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | package com.Uddhav.ENTTool.utils;  import android.app.Application; import android.content.Context;  import com.splunk.mint.Mint;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class App extends Application { //Application is android's class like PreferenceManager is.  // Application in the android is the base process that runs inside the Dalvik VM. Since, it is a process,  // it has a lifecycle like Activity. But, Activity is What you see/do in the screen. So, one Application may contain |

|  |  |  |
| --- | --- | --- |
| 14 | //  } | // many Activities Application are the basic fundamental things. Application is defined in the AndroidManifest.xml  //file.  //In this project, there are 4 activities, defined in the AndroidManifest.xml file, inside the Application  //So, basically, activities are like the screen of the projects, related to the GUI. So, activity extends ContextThemeWrapper  //whereas, the Application extends ContextWrapper.  // Application's father is the activity's grandpa  public static Context AppContext = null; //Context is an abstract class which has Object as father. This tells overall  // system environment  public static MainThreadBus bus = null;  //bus is just a Otto bus. It uses handler for the communication between activity and fragments or activity and services.  @Override  public void onCreate() { super.onCreate();  AppContext = getApplicationContext(); //Now, AppContext is the context of the Application class (App class here)  Mint.initAndStartSession(App.this, getString(R.string.Mint\_apiKey)); Mint.initAndStartSession(App.this, "29463cb0"); //I am using Splunk Mint SDK to initialize and do start session  // of this App providing the Splunk Mint API key. 29463cb0 is the key. Splunk Mint, here, I am using for the Data collector  bus = new MainThreadBus();  // I have initialized the MainThreadBus. Mean, I have called the MainThreadBus() default construction. But, since, tehre  // is no constructor. So, "bus", here, has no use  } |
| 15 |
| 16 |
| 17 |
| 18 |
| 19 |
| 20 |
| 21 |
| 22 |
| 23 |
| 24 |
| 25 |
| 26 |
| 27 |
| 28 |
| 29 |
| 30 |
| 31 |
| 32 |
| 33 |
| 34 |
| 35 |
| 36 |
| 37 |
| 38 |
| 39 |
| 40 |
| 41 |
| 42 |

|  |  |
| --- | --- |
| **ENTTool\_utils\_AppSettings.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | package com.Uddhav.ENTTool.utils;  import android.content.Context;  import android.content.SharedPreferences; import android.preference.PreferenceManager;  import com.Uddhav.ENTTool.R; import java.util.Map;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class AppSettings {  public static AppSettings pojoPref = null;  private static Context ctx = App.AppContext; //App is a class. AppContext is a public static object  // ctx is the context (system environment) of the base Application (process) of my project.  private int TimeInterval, Source, Magnitude, MapType, UpdatePeriod, Sorting;  private boolean Notifications, Vibration, Sound; // for checking notification, vibration and sound whehter to do ON/OFF  private String Key\_TimeInterval, Key\_Source, Key\_Magnitude;  private String Key\_UpdatePeriod, Key\_Sorting, Key\_Notifications, Key\_Vibration, Key\_Sound; AppSettings() { // constructor  Key\_TimeInterval = ctx.getResources().getString(R.string.listPref\_Key\_TimeInterval); Key\_Source = ctx.getResources().getString(R.string.listPref\_Key\_Source); Key\_Magnitude = ctx.getResources().getString(R.string.listPref\_Key\_Magnitude); Key\_UpdatePeriod = ctx.getResources().getString(R.string.listPref\_Key\_UpdatePeriod); Key\_Sorting = ctx.getResources().getString(R.string.listPref\_Key\_Sorting);  Key\_Notifications = ctx.getResources().getString(R.string.CheckBoxPref\_Key\_Notifications); Key\_Vibration = ctx.getResources().getString(R.string.CheckBoxPref\_Key\_Vibration); Key\_Sound = ctx.getResources().getString(R.string.CheckBoxPref\_Key\_Sound);  SharedPreferences pref = PreferenceManager.getDefaultSharedPreferences(ctx); Map<String, ?> allEntries = pref.getAll();  String asd = (String) allEntries.get(Key\_TimeInterval); TimeInterval = Integer.parseInt(asd); |

43

44

45

46

47

48

49

50

51

52

53

54

55

56 }

57

Source = Integer.parseInt((String) allEntries.get(Key\_Source)); Magnitude = Integer.parseInt((String) allEntries.get(Key\_Magnitude));

UpdatePeriod = Integer.parseInt((String) allEntries.get(Key\_UpdatePeriod)); Sorting = Integer.parseInt((String) allEntries.get(Key\_Sorting));

Notifications = (Boolean) allEntries.get(Key\_Notifications); Vibration = (Boolean) allEntries.get(Key\_Vibration);

Sound = (Boolean) allEntries.get(Key\_Sound);

// for (Map.Entry<String, ?> entry : allEntries.entrySet()) {

// Log.d("map values", entry.getKey() + ": " + entry.getValue().toString());

// }

1. public static void setDefaultSettings() {
2. PreferenceManager.setDefaultValues(ctx, R.xml.pref, false);
3. //Note, all resources are got from the resource ID.
4. //PreferenceManager is a helper method to create a preference.
5. //Used to help create Preference hierarchies from activities or XML.
6. // In most cases, clients should use addPreferencesFromIntent(Intent) or addPreferencesFromResource(int). 64 }

65

1. public static AppSettings getInstance() {
2. return pojoPref == null ? new AppSettings() : pojoPref; 68 }

69

1. public int getTimeInterval() {
2. return TimeInterval; 72 }

73

1. public void setTimeInterval(int timeInterval) {
2. TimeInterval = timeInterval; 76 }

77

1. public int getSource() {
2. return Source; 80 }

81

1. public void setSource(int source) {
2. Source = source; 84 }

85

1. public int getMagnitude() {
2. return Magnitude; 88 }

89

1. public void setMagnitude(int magnitude) {
2. Magnitude = magnitude; 92 }

93

1. public int getMapType() {
2. return MapType; 96 }

97

1. public void setMapType(int mapType) {
2. MapType = mapType;

100 }

101

1. public int getUpdatePeriod() {
2. return UpdatePeriod;

104 }

105

1. public void setUpdatePeriod(int updatePeriod) {
2. UpdatePeriod = updatePeriod;

108 }

109

1. public int getSorting() {
2. return Sorting;

112 }

113

1. public void setSorting(int sorting) {
2. Sorting = sorting;

116 }

|  |  |  |
| --- | --- | --- |
| 117 | } | public boolean isSound() { return Sound;  }  public void setSound(boolean sound) { Sound = sound;  }  public boolean isVibration() { return Vibration;  }  public void setVibration(boolean vibration) { Vibration = vibration;  }  public boolean isNotifications() { return Notifications;  }  public void setNotifications(boolean notifications) { Notifications = notifications;  } |
| 118 |
| 119 |
| 120 |
| 121 |
| 122 |
| 123 |
| 124 |
| 125 |
| 126 |
| 127 |
| 128 |
| 129 |
| 130 |
| 131 |
| 132 |
| 133 |
| 134 |
| 135 |
| 136 |
| 137 |
| 138 |
| 139 |
| 140 |
| 141 |

|  |  |
| --- | --- |
| **ENTTool\_utils\_CreateRequestUrl.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46 | package com.Uddhav.ENTTool.utils;  import java.text.Format;  import java.text.SimpleDateFormat; import java.util.Calendar;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class CreateRequestUrl {  public static String URL\_SEISMICPORTAL(int day) { // maksimum 10 gun geriye gidilebilir int dateRange = 1;  if (day == 0) { dateRange = 1;  } else if (day == 1) { dateRange = 7;  } else if (day == 2) { dateRange = 10;  }  Calendar cal = Calendar.getInstance(); cal.add(Calendar.DAY\_OF\_MONTH, ‐dateRange);  Format formatter = new SimpleDateFormat("yyyy‐MM‐dd"); String s = formatter.format(cal.getTime());  return "[http://www.seismicportal.eu/fdsnws/event/1/query?start=](http://www.seismicportal.eu/fdsnws/event/1/query?start)" + s + "&format=json";  }  public static String URL\_KOERI(int day) { String str = "son24saat";  if (day == 0) {  str = "son24saat";  } else if (day == 1) { str = "sonHafta";  } else if (day == 2) { str = "sonay";  }  return "<http://www.koeri.boun.edu.tr/sismo/zeqmap/xmlt/>" + str + ".xml";  }  public static String URL\_USGS(int day) { |

|  |  |  |  |
| --- | --- | --- | --- |
| 47 | } | } | String str = "all\_day"; if (day == 0) {  str = "all\_day";  } else if (day == 1) { str = "1.0\_week";  } else if (day == 2) { str = "2.5\_month";  }  return "<http://earthquake.usgs.gov/earthquakes/feed/v0.1/summary/>" + str + ".geojson"; |
| 48 |
| 49 |
| 50 |
| 51 |
| 52 |
| 53 |
| 54 |
| 55 |
| 56 |
| 57 |
| 58 |
| 59 |
| 60 |
| 61 |

|  |  |
| --- | --- |
| **ENTTool\_utils\_EBus.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | package com.Uddhav.ENTTool.utils;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class EBus { private int status;  public EBus(int s) { status = s;  }  public int getStatus() { return status;  }  public void setStatus(int status) { this.status = status;  }  } |

|  |  |
| --- | --- |
| **ENTTool\_utils\_MainThreadBus.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | package com.Uddhav.ENTTool.utils;  import android.os.Handler; import android.os.Looper;  import com.squareup.otto.Bus;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class MainThreadBus extends Bus {  // I am using the Otto bus. Otto is a great way to communicate between your activity and fragments or  // to communicate between an activity and a service.  //Bus does communication through handlers. Handler is the android OS's handler. Means, android.os.handler  private final Handler mHandler = new Handler(Looper.getMainLooper()); //This return the Application's main looper  // to the Handler. Handler is used to send messages and runnable objects to the.  //A Handler allows you to send and process Message and Runnable objects associated with a thread's MessageQueue. Each Handler instance  // There are two main uses for a Handler:  // (1) to schedule messages and runnables to be executed as some point in the future; and (  // 2) to enqueue an action to be performed on a different thread than your own.  @Override  public void post(final Object event) {  if (Looper.myLooper() == Looper.getMainLooper()) { // means, I am in the active Looper  super.post(event); // post and event. event, here, is an object. So, I am posting Object  //Super.post(event) does post the event to all registered handlers of the MainLooper  } else { |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 32 | | } | } | mHandler.post(new Runnable() {  @Override  public void run() { MainThreadBus.super.post(event);  }  });  } // hover mouse over arrow to know the above code | | |  |
| 33 | |
| 34 | |
| 35 | |
| 36 | |
| 37 | |
| 38 | |
| 39 | |
| 40 | |
|  |  | | | |  |  |  |

|  |  |
| --- | --- |
| **ENTTool\_utils\_SaveResponseToDB.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61 | package com.Uddhav.ENTTool.utils;  import com.Uddhav.ENTTool.database.EarthQuakes;  import com.Uddhav.ENTTool.sources.seismicportal.features; import com.Uddhav.ENTTool.sources.seismicportal.geometry; import com.Uddhav.ENTTool.sources.seismicportal.object; import com.Uddhav.ENTTool.sources.seismicportal.properties; import com.Uddhav.ENTTool.sources.seismicportal.totalCount; import com.Uddhav.ENTTool.sources.usgs.features1;  import com.Uddhav.ENTTool.sources.usgs.geometry1; import com.Uddhav.ENTTool.sources.usgs.metadata1; import com.Uddhav.ENTTool.sources.usgs.properties1; import com.Uddhav.ENTTool.sources.usgs.usgs; import com.google.gson.Gson;  import com.google.gson.GsonBuilder; import com.google.gson.reflect.TypeToken; import com.squareup.okhttp.OkHttpClient; import com.squareup.okhttp.Request; import com.squareup.okhttp.Response;  import java.lang.reflect.Type; import java.math.BigDecimal; import java.util.Calendar; import java.util.List;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class SaveResponseToDB {  private int decimalPlace = 1; public SaveResponseToDB() {  // DatabaseHelper.getDbHelper().clearDatabase();  }  public void saveDatabaseSeismicPortal(String url) { try {  Gson gson = new GsonBuilder().setPrettyPrinting().setDateFormat(Tools.DATEFORMAT\_SEISMICPORTAL).create();  Type listType = new TypeToken<object<totalCount, List<features<geometry, properties>>>>() {  }.getType(); //TypeToken(items).getType == listType  //Type is just used to abstractly define it is one type of object. So, type is like generic way of defining things. String json = getJson(url);  if (json == null || json.length() < 10) { return;  }  object<totalCount, List<features<geometry, properties>>> items = gson.fromJson(json, listType); if (items == null || items.getFeatures() == null || items.getFeatures().size() == 0) {  return;  }  for (features<geometry, properties> item : items.getFeatures()) { //every feature has geometry and properties |

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76 }

77

Calendar cal = Calendar.getInstance(); cal.setTime(item.getProperties().getTime()); cal.add(Calendar.HOUR\_OF\_DAY, 2);

EarthQuakes eq = new EarthQuakes(); eq.setDateMilis(cal.getTime().getTime()); eq.setDepth(round(item.getProperties().getDepth(), decimalPlace)); eq.setLatitude(item.getProperties().getLat()); eq.setLongitude(item.getProperties().getLon());

eq.setLocationName(item.getProperties().getFlynn\_region().trim().toUpperCase()); eq.setMagnitude(round(item.getProperties().getMag(), decimalPlace)); eq.setSource(2);

eq.Insert();

78

79

80

81

82 }

83

} catch (Exception e) { Tools.catchException(e);

}

84 public void saveDatabaseUsgs(String url) { //save every earthquake fields like magnitude, latitude etc. 85

86 try {

87

1. Gson gson = new GsonBuilder().setPrettyPrinting().setDateFormat(Tools.DATEFORMAT).create();
2. //Gson is used to serialize/deserialize Json.
3. Type listType = new TypeToken<usgs<metadata1, features1<properties1, geometry1>>>() {
4. }.getType();
5. //below I can use the getFeatures() because item is a type of object defined from Type interface 93

94

95 String json = getJson(url); //getJson is using OkHttpClient 96

1. if (json == null || json.length() < 10) { // JSON is null or empty
2. return; //note: any code below return does not get executed. So, I don't need to put in If and else block. If if is true th 99 }

100

101 usgs<metadata1, features1<properties1, geometry1>> items = gson.fromJson(json, listType); 102

1. if (items == null || items.getFeatures() == null || items.getFeatures().size() == 0) { //check if item null or items' features
2. return;

105 }

106

107 for (features1<properties1, geometry1> item : items.getFeatures()) { 108

1. String str1 = item.getProperties().getPlace().trim().toUpperCase();
2. String str2 = str1.substring(str1.indexOf("OF") + 3); 111
3. EarthQuakes eq = new EarthQuakes();
4. eq.setDateMilis(Long.parseLong(item.getProperties().getTime()));
5. eq.setDepth(round(item.getGeometry().getCoordinates().get(2), decimalPlace));
6. eq.setLatitude(item.getGeometry().getCoordinates().get(1));
7. eq.setLongitude(item.getGeometry().getCoordinates().get(0));
8. eq.setLocationName(str2);
9. eq.setMagnitude(round(item.getProperties().getMag(), decimalPlace));
10. eq.setSource(3);
11. eq.Insert();

122 }

123

1. } catch (Exception e) {
2. Tools.catchException(e);

126 }

127

128 }

129

1. private String getJson(String reqUrl) throws Exception {
2. Request request = new Request.Builder().url(reqUrl).build(); //Request builder is used to get JSON url
3. Response response = new OkHttpClient().newCall(request).execute(); //OkHttpClient is HTTP client to request
4. return response.isSuccessful() ? response.body().string() : "";

134 }

135

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 136 | | } | public float round(float d, int decimalPlace) { BigDecimal bd = new BigDecimal(Float.toString(d));  bd = bd.setScale(decimalPlace, BigDecimal.ROUND\_HALF\_UP); return bd.floatValue();  } | |  |  |
| 137 | |
| 138 | |
| 139 | |
| 140 | |
| 141 | |
| 142 | |
|  |  | | |  |  |  |

|  |  |
| --- | --- |
| **ENTTool\_utils\_SyncService.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63 | package com.Uddhav.ENTTool.utils;  import android.app.NotificationManager; import android.app.PendingIntent; import android.app.Service;  import android.content.Context; import android.content.Intent; import android.graphics.Color;  import android.media.RingtoneManager; import android.net.Uri;  import android.os.Handler; import android.os.IBinder; import android.os.Looper;  import android.support.v4.app.NotificationCompat; import android.support.v4.app.TaskStackBuilder; import android.util.Log;  import com.Uddhav.ENTTool.MainActivity; import com.Uddhav.ENTTool.R;  import com.Uddhav.ENTTool.database.EarthQuakes;  import com.Uddhav.ENTTool.database.LastEarthquakeDate; import java.util.List;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class SyncService extends Service {  public static boolean isServiceRunning = false; public static Context AppContextService; private int mStartMode;  private IBinder mBinder; private boolean mAllowRebind; private Handler handler;  @Override  public void onCreate() {  AppContextService = getApplicationContext(); App.bus.register(this);  handler = new Handler(Looper.getMainLooper());  }  @Override  public int onStartCommand(Intent intent, int flags, int startId) { isServiceRunning = true;  new Thread(new Runnable() {  @Override  public void run() {  while (isServiceRunning) {  if (Tools.isOnline(AppContextService)) { Log.i("SyncService", "Service Running"); updateSyncPeriod();  int day = AppSettings.getInstance().getTimeInterval(); int source = AppSettings.getInstance().getSource();  SaveResponseToDB clientHelper = new SaveResponseToDB(); |

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89 }

if (source == 0) { clientHelper.saveDatabaseUsgs(CreateRequestUrl.URL\_USGS(day));

clientHelper.saveDatabaseSeismicPortal(CreateRequestUrl.URL\_SEISMICPORTAL(day));

} else if (source == 1) { clientHelper.saveDatabaseSeismicPortal(CreateRequestUrl.URL\_SEISMICPORTAL(day));

} else if (source == 2) { clientHelper.saveDatabaseUsgs(CreateRequestUrl.URL\_USGS(day));

}

notificationHandler(); App.bus.post(new EBus(123));

} else {

App.bus.post(new EBus(999));

}

try {

Thread.sleep(1000 \* 60 \* Tools.syncPeriod);

} catch (InterruptedException e) { Tools.catchException(e);

}

}

90

91

92

93 }

94

}).start();

return Service.START\_STICKY;

95 private void updateSyncPeriod() { 96

97 int refreshtime = AppSettings.getInstance().getUpdatePeriod();

98

99 if (refreshtime == 0) {

1. Tools.syncPeriod = 2;
2. } else if (refreshtime == 1) {
3. Tools.syncPeriod = 15;
4. } else if (refreshtime == 2) {
5. Tools.syncPeriod = 30;
6. } else if (refreshtime == 3) {
7. Tools.syncPeriod = 60;
8. } else {
9. Tools.syncPeriod = 2;

109 }

110 }

111

112 private void notificationHandler() { 113

1. handler.post(new Runnable() {
2. @Override
3. public void run() {
4. showNotification();

118 }

119 });

120 }

121

1. private void showNotification() {
2. List<EarthQuakes> newEarthquakes = new EarthQuakes().newEarthquakes(); 124

125 if (newEarthquakes.size() > 0) {

126

1. if (AppSettings.getInstance().isNotifications()) {
2. createNotification(getString(R.string.EarthquakesDetect), //
3. "" + newEarthquakes.get(0).getMagnitude() + " | " + newEarthquakes.get(0).getLocationName());

130 }

131

1. LastEarthquakeDate led = new LastEarthquakeDate();
2. led.setDateMilis(new EarthQuakes().GetLastEarthQuakeDate());
3. led.Insert();

135 }

136 }

137

|  |  |  |
| --- | --- | --- |
| 138 | } | private void createNotification(String strContentTitle, String strContentText) {  NotificationCompat.Builder builder = new NotificationCompat.Builder(SyncService.this) //  .setSmallIcon(R.drawable.icon1) //  .setContentTitle(strContentTitle) //  .setContentText(strContentText);  Intent resultIntent = new Intent(this, MainActivity.class); TaskStackBuilder stackBuilder = TaskStackBuilder.create(this); stackBuilder.addParentStack(MainActivity.class); stackBuilder.addNextIntent(resultIntent);  PendingIntent resultPendingIntent = stackBuilder.getPendingIntent(0, PendingIntent.FLAG\_UPDATE\_CURRENT);  builder.setContentIntent(resultPendingIntent); builder.setAutoCancel(true); builder.setLights(Color.BLUE, 500, 500);  if (AppSettings.getInstance().isVibration()) { long[] pattern = {500, 500}; builder.setVibrate(pattern);  }  if (AppSettings.getInstance().isSound()) {  Uri alarmSound = RingtoneManager.getDefaultUri(RingtoneManager.TYPE\_NOTIFICATION); builder.setSound(alarmSound);  }  NotificationManager manager = (NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE); manager.notify(0, builder.build());  }  @Override  public IBinder onBind(Intent intent) { return mBinder;  }  @Override  public boolean onUnbind(Intent intent) { return mAllowRebind;  }  @Override  public void onRebind(Intent intent) {  }  @Override  public void onDestroy() { isServiceRunning = false; App.bus.unregister(this);  } |
| 139 |
| 140 |
| 141 |
| 142 |
| 143 |
| 144 |
| 145 |
| 146 |
| 147 |
| 148 |
| 149 |
| 150 |
| 151 |
| 152 |
| 153 |
| 154 |
| 155 |
| 156 |
| 157 |
| 158 |
| 159 |
| 160 |
| 161 |
| 162 |
| 163 |
| 164 |
| 165 |
| 166 |
| 167 |
| 168 |
| 169 |
| 170 |
| 171 |
| 172 |
| 173 |
| 174 |
| 175 |
| 176 |
| 177 |
| 178 |
| 179 |
| 180 |
| 181 |
| 182 |
| 183 |
| 184 |
| 185 |
| 186 |
| 187 |
| 188 |
| 189 |

|  |  |
| --- | --- |
| **ENTTool\_utils\_Tools.java** | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | package com.Uddhav.ENTTool.utils;  import android.content.Context;  import android.net.ConnectivityManager; import android.net.NetworkInfo;  import com.splunk.mint.Mint;  /\*\*  \* Created by Uddhav Gautam on 7.3.2016. [upgautam@ualr.edu](mailto:upgautam@ualr.edu)  \*/  public class Tools {  public static int syncPeriod = 1000 \* 15;  public static String DATEFORMAT = "yyyy‐MM‐dd HH:mm:ss"; public static String DATEFORMAT\_KOERI = "yyyy.MM.dd HH:mm:ss";  public static String DATEFORMAT\_SEISMICPORTAL = "yyyy‐MM‐dd'T'HH:mm:ss.S'Z'";  public static boolean isOnline(Context act) { |

|  |  |  |
| --- | --- | --- |
| 20 | } | ConnectivityManager cm = (ConnectivityManager) act.getSystemService(Context.CONNECTIVITY\_SERVICE); NetworkInfo netInfo = cm.getActiveNetworkInfo();  return netInfo != null && netInfo.isConnectedOrConnecting();  }  public static void catchException(Exception ex) { ex.printStackTrace();  Mint.logException(ex);  } |
| 21 |
| 22 |
| 23 |
| 24 |
| 25 |
| 26 |
| 27 |
| 28 |
| 29 |