Mobile Computing

Practice # 2d

Android Applications – Local DB

In this installment we will add persistent storage to the restaurants' application.

For that, we will create a database with a table for holding our restaurant data and switch from our ArrayAdapter to a CursorAdapter, to make use of that database. This will allow our restaurants to persist in every execution of LunchList.

1. Create a class to make the interface to a SQLite database. We need to be able to define what our database name is, what is the schema for the table storing our restaurants, etc. These definitions should be wrapped up in a SQLiteOpenHelper object implementation that can open or create a database (or upgrade its version if already exists in an older form).

a. Create a new class in a file RestaurantsHelper.java extending SQLiteOpenHelper, defining some constants and overriding onCreate() and onUpgrade(), as follows:

```
class RestaurantsHelper extends SQLiteOpenHelper {
   private static final String DATABASE_NAME="lunchlist.db";
   private static final int SCHEMA_VERSION=1;

   public RestaurantsHelper(Context context) {
      super(context, DATABASE_NAME, null, SCHEMA_VERSION);
   }

   @Override
   public void onCreate(SQLiteDatabase db) {
   }

   @Override
   public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
   }
}
```

This says that our database is in file lunchlist.db and this is the first version of the schema. This version of the schema should be created in the onCreate() method. In onUpgrade() we should put code needed to convert a database and schema from an oldVersion to a newVersion. Such an upgrade, when needed, of course implies to copy all the data to some intermediate tables, fix up the tables to the new schema and copy back the data, before deleting the intermediate tables. But for now we stick with the 1st version and the SQL 'create table' statement.

```
@Override
public void onCreate(SQLiteDatabase db) {
   db.execSQL("CREATE TABLE Restaurants (_id INTEGER PRIMARY KEY AUTOINCREMENT, " +
                    "name TEXT, address TEXT, type TEXT, notes TEXT);");
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
   // no-op, since it will not be called until we need to define a 2nd schema version
}
```

b. We will be using RestaurantsHelper as our bridge to the database, and we will not use the Application object anymore for sharing data.
So, delete the LunchApp class and all references to it in Main and Details activities.
Also let's get some access, in both those classes, to the RestaurantsHelper, in the following way:
- Declare a RestaurantsHelper (helper) variable in both classes.
- In the onCreate() method of both classes, and after the call to setContentLayout(), create the helper object like: helper = new RestaurantsHelper(this);
- Also add an override to onDestroy() in both classes with:

```
super.onDestroy();
helper.close();
```

c. We are going to be replacing our restaurant object model restaurants (and its associated ArrayList) with the database and a Cursor representing the list of restaurants. This will involve adding some more logic to RestaurantsHelper to aid in this process.

First add an insert() and an update() methods:

```
public long insert(String name, String address, String type, String notes) {
   ContentValues cv=new ContentValues();
   cv.put("name", name);
   cv.put("address", address);
   cv.put("type", type);
   cv.put("notes", notes);
   return getWritableDatabase().insert("Restaurants", "name", cv);
}

public void update(String id, String name, String address, String type, String notes) {
   ContentValues cv=new ContentValues();
   String[] args={id};
   cv.put("name", name);
   cv.put("address", address);
   cv.put("type", type);
   cv.put("notes", notes);
   getWritableDatabase().update("Restaurants", cv, "_ID=?", args);
}
```

These methods should be called from the save button listener when a new restaurant is created or modified. Replace the listener in the Details class, after the switch statement, to contain:

```
if (restaurantId==null) {
   MainActivity.currentId = helper.insert(name.getText().toString(), address.getText().toString(),
                                          type, notes.getText().toString());
}
else {
   helper.update(restaurantId, name.getText().toString(), address.getText().toString(), type,
                        notes.getText().toString());
}
finish();
```

d. We need also to query the database and put the result in a cursor for all the restaurants and also for a single restaurant, given its _id. For that we need two more methods in our RestaurantsHelper class, as well as some other methods to retrieve the individual pieces of data out of a cursor. These new methods should be as follows:

```
public Cursor getAll() {
   return(getReadableDatabase()
      .rawQuery("SELECT _id, name, address, type, notes FROM restaurants ORDER BY name",
                     null));
}

public Cursor getById(String id) {
   String[] args={id};

   return(getReadableDatabase()
       .rawQuery("SELECT _id, name, address, type, notes FROM restaurants WHERE _ID=?",
                     args));
}

public String getName(Cursor c) {
   return(c.getString(1));
}

public String getAddress(Cursor c) {
   return(c.getString(2));
}

public String getType(Cursor c) {
   return(c.getString(3));
}

public String getNotes(Cursor c) {
   return(c.getString(4));
}
```

e. Declare a new String field rId, eliminate the rPos field, and replace the data from the intent, and the last if (…) in onCreate(), and also the load() method of the Details class, by the following:

```
String rId;
   . . .
   rId=getIntent().getStringExtra(LunchList.ID_EXTRA);

   if (rId!=null) {
      load();
   }
   . . .

private void load() {
   Cursor c=helper.getById(rId);
   c.moveToFirst();
   ((EditText)findViewById(R.id.ed_name)).setText(helper.getName(c));
   ((EditText)findViewById(R.id.ed_address)).setText(helper.getAddress(c));
   ((EditText)findViewById(R.id.ed_notes)).setText(helper.getNotes(c));
   RadioGroup rgTypes = findViewById(R.id.rg_types);
   if (helper.getType(c).equals("sit"))
     rgTypes.check(R.id.sit);
   else if (helper.getType(c).equals("take"))
```

```
          rgTypes.check(R.id.take);
        else
          rgTypes.check(R.id.delivery);
        c.close();
      }
```

Now the Details activity expects to be invoked from the Main with an intent transporting the _id of the selected restaurant, or nothing if we want to add a new restaurant (from the Main menu).

f. On the Main activity replace now the onItemClick() listener to send the current restaurant id and start the Details activity:

```
@Override
public void onItemClick(ListView list, View view, int pos, long id) {
   Intent i=new Intent(this, DetailsActivity.class);
   currentId = id;
   i.putExtra(ID_EXTRA, String.valueOf(id));
   startActivity(i);
}
```

We need a new static field in the MainActivity class to store the current id, like:

```
static long currentId = -1;
```

and correct the toast listener in onOptionsItemSelected() using:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
  . . .
   if (currentId != -1) {
     Cursor c = helper.getById(String.valueOf(currentId));
     c.moveToNext();
     message = String.format("%s:\n%s", helper.getName(c), helper.getNotes(c));
     c.close();
   }
  . . .
}
```

g. Next, we need to replace, in the Main activity, our model containing the restaurants by a Cursor, and make our RestaurantAdapter a cursor adapter:

Let's start by changing our needed fields in the class:

```
Cursor model=null;
RestaurantAdapter adapter=null;
```

and replace their initialization in the onCreate() method (see that the model is initialized with all restaurants available in the database). The call to startManagingCursor() allows the activity to retrieve automatically again the model cursor, in the case it has to be recreated.

```
  model=helper.getAll();
  startManagingCursor(model);
  adapter=new RestaurantAdapter(model);
  list.setAdapter(adapter);
```

Finally we have to adapt the RestaurantAdapter to be a CursorAdapter:

```
class RestaurantAdapter extends CursorAdapter {
  RestaurantAdapter(Cursor c) {
    super(MainActivity.this, c);
  }

  public void bindView(View row, Context ctxt, Cursor c) {
  }

  public View newView(Context ctxt, Cursor c, ViewGroup parent) {
    View row=getLayoutInflater().inflate(R.layout.row, parent, false);
    ((TextView)row.findViewById(R.id.title)).setText(helper.getName(c));
    ((TextView)row.findViewById(R.id.address)).setText(helper.getAddress(c));
    ImageView symbol = row.findViewById(R.id.symbol);
    if (helper.getType(c).equals("sit"))
      symbol.setImageResource(R.drawable.ball_red);
    else if (helper.getType(c).equals("take"))
      symbol.setImageResource(R.drawable.ball_yellow);
    else
      symbol.setImageResource(R.drawable.ball_green);

    return(row);
  }
```

h. Remove the Restaurant class and file, also the manifest reference to LunchApp, compile, install and use.