# Data Storage

Presentation is based on Android Storage Options documentation

# Data Storage Options

- Shared Preferences
- Internal Storage
- External Storage
- APK static files
- SQLite Databases
- Network Connection

# Shared Preferences

# Shared Preferences

- Lightweight mechanism for key/value pairs
  - Key is a String
  - Value is Integer, Long, Double, Boolean, or String
- Great option storing primitive data

# Shared Preferences

`getPreferences()` - Use this if you need only one preferences file for your Activity.

`getSharedPreferences()` - Use this if you need multiple preferences files identified by name.

# Access Modes

`MODE_PRIVATE` - private to app (recommended – default mode)
`MODE_WORLD_READABLE` - any app can read (deprecated)
`MODE_WORLD_WRITEABLE` - any app can write (deprecated)

# Shared Preferences - Example Writing

```
SharedPreferences settings =
    getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
SharedPreferences.Editor editor = settings.edit();
editor.putBoolean("silentMode", mSilentMode);
editor.commit();
```

# Shared Preferences - Example Reading

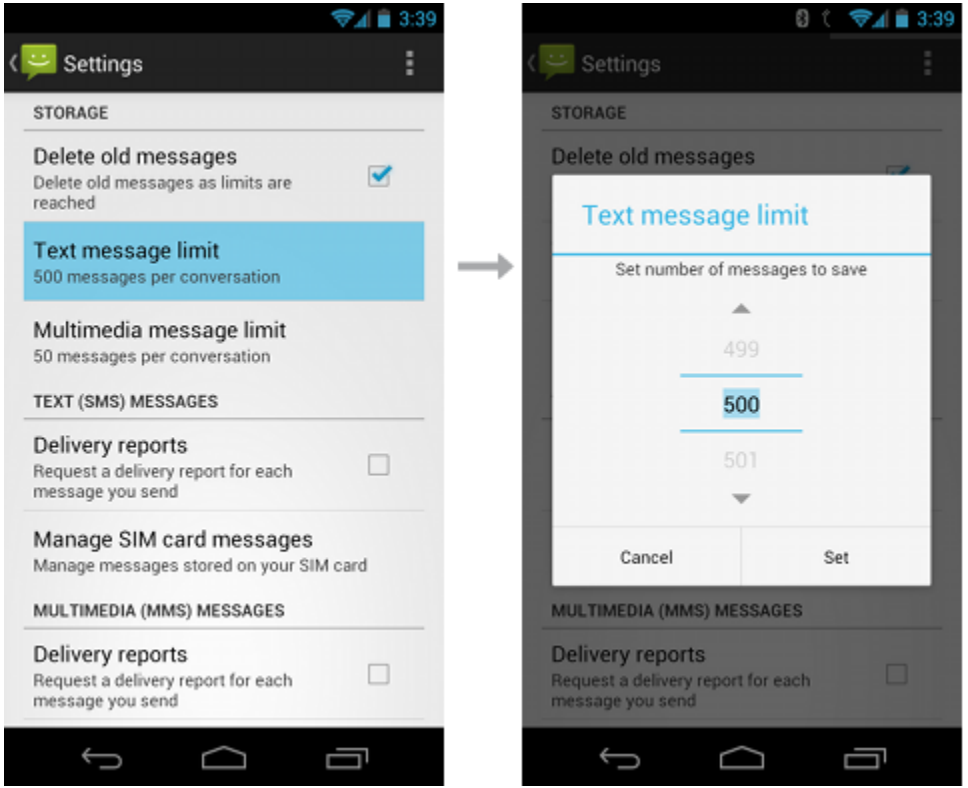
```
SharedPreferences settings =
    getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
mSilent = settings.getBoolean("silentMode", false);
```

# Shared Preferences - Location

Data is stored under
`/data/data/$PACKAGE_NAME/shared_prefs`

# PreferenceFragment

- Preexisting Fragment class to create user preferences.
- Automatically persisted using shared preferences
- Documentation



# Internal Storage

# Internal Storage

- Save files directly to device's internal storage.
- Private data by default
  - Other applications cannot access it (nor can the user)
- Removed when app is uninstalled.

# Using the Internal Storage APIs

## Writing to Internal Storage

1. Call openFileOutput() with the name of the file and the operating mode.
2. Write to the file with write().
3. Close the stream with close().

## Reading from Internal Storage

1. Call openFileInput() and pass it the name of the file to read.
2. Read bytes from the file with read().
3. Then close the stream with close().

# Internal Storage - Example Writing

```
String FILENAME = "diary.txt";
String string = "hello world!";

FileOutputStream fos = openFileOutput(FILENAME,
    Context.MODE_PRIVATE);
fos.write(string.getBytes());
fos.close();
```

# Internal Storage - Example Reading

```
BufferedReader reader = new BufferedReader(new
    InputStreamReader(openFileInput(FILENAME)));
Log.d("TAG", "From file: " + reader.readLine());
```

# Internal Storage - Location

Data is stored under
`/data/data/$PACKAGE_NAME/files`

# Cache Files

getCacheDir() - for storing temporary files

`/data/data/$PACKAGE_NAME/cache`

To create a cached file -- File.createTempFile()

```
File.createTempFile(filename, null, context.getCacheDir());
```

Access a file in this directory using cacheDir()

```
File cacheFile = new File(context.getCacheDir(), filename);
```

Removing Cache File

```
cacheFile.delete();
```

# External Storage

# External Storage

- May be removable storage media (e.g., SD card) or internal (non-removal) storage.
- Files saved to the external storage are world-readable!
  - Avoid leaking data, like WhatsApp
- Can be modified by the user when they enable USB mass storage to transfer files on a computer.

# External Storage

- New files acquired through your app should be saved to a "public" location
- Shared public directories: Music/, Pictures/, Ringtones/, etc.

# Public directories

The Environment class has constants for common public directories:

Environment.DIRECTORY_ALARMS - audio files that should be in the list of alarms that the user can select

Environment.DIRECTORY_DCIM - traditional location for pictures and videos when mounting the device as a camera

Environment.DIRECTORY_DOWNLOADS

Environment.DIRECTORY_MOVIES

Environment.DIRECTORY_MUSIC

Environment.DIRECTORY_NOTIFICATIONS - audio files that should be in the list of notifications that the user can select

Environment.DIRECTORY_PICTURES

Environment.DIRECTORY_PODCASTS

Environment.DIRECTORY_RINGTONES

# External Storage - Permission

Writing to external storage requires the android.permission.WRITE_EXTERNAL_STORAGE permission.

Add to AndroidManifest.xml file:

```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
```

# External Storage - Example Saving

```
public File getAlbumStorageDir(String albumName) {
    // Get the directory for the user's public pictures directory.
    File file = new File(Environment.getExternalStoragePublicDirectory(
            Environment.DIRECTORY_PICTURES), albumName);
    if (!file.mkdirs()) {
        Log.e(LOG_TAG, "Directory not created");
    }
    return file;
}
```

# External Storage - Example Saving

```
// Example from Taking Photos Simply
(http://developer.android.com/training/camera/photobasics.html)

private File createImageFile() throws IOException {
    // Create an image file name
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new
Date());
    String imageFileName = "JPEG_" + timeStamp + "_";
    File storageDir = getAlbumStorageDir("Zoo");
    File image = File.createTempFile(
        imageFileName,  /* prefix */
        ".jpg",         /* suffix */
        storageDir      /* directory */
    );
    return image;
}
```

# External Storage - Example Saving

```
private static final int RESULT_CODE = 0;
private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    // Ensure that there's a camera activity to handle the intent
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        // Create the File where the photo should go
        File photoFile = null;
        try {
            photoFile = createImageFile();
        } catch (IOException ex) {
            // Error occurred while creating the File
        }
        // Continue only if the File was successfully created
        if (photoFile != null) {
            takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT,
                    Uri.fromFile(photoFile));
            startActivityForResult(takePictureIntent, RESULT_CODE);
        }
    }
}
```

# APK Static Files

assets/ and res/raw

# Assets folder

- /assets directory in app's project folder
- Compiled into an .apk file as-is, and the original filename is preserved.
- Example use cases:
  - game data, game textures
  - sound files
  - static data files

# Assets Directory

```
InputStream inputStream =
    getAssets().open("diary.txt");
BufferedReader reader = new BufferedReader(new
    InputStreamReader(inputStream));
Log.d("TAG", reader.readLine());
```

# res/raw

- Files placed in the res/raw directory are resolvable via the R.java class.
  - Example: the ID of `res/raw/diary.txt` is `R.raw.diary`
- Read file using getResources().openRawResource()

Example:

```
InputStream myFile = getResources().openRawResource(R.raw.diary);
BufferedReader reader = new BufferedReader(new
    InputStreamReader(inputStream));
Log.d("TAG", reader.readLine());
```

# SQLite

# SQLite Overview

- Open-source
- Standards-compliant
- Lightweight (less than 400kb)
- Zero-configuration

# SQLite Overview

- Most widely deployed SQL database engine in the world!
- Implements most of the SQL-92 standard
- Great option for storing structured data

# Quick SQL Review

http://en.wikipedia.org/wiki/SQL

# Content Values and Cursors

ContentValue - represents a single table row as a key/value map.

Cursor - pointer to result set

# SQLiteOpenHelper

- Helper class for creating, opening, and upgrading databases.
- Create a subclass SQLiteOpenHelper for your database instance

# SQLiteDatabase - APIs for CRUD

| **Operation** | **API Method** |
| --- | --- |
| **C**reation | SQLiteDatabase.insert() |
| **R**ead | SQLiteDatabase.query() |
| **U**pdate | SQLiteDatabase.update() |
| **D**eletion | SQLiteDatabase.delete() |

# Example: Zoo Database

Example: create a Zoo table. Each row represents an animal and has 4 fields:

- ID - a unique identifier for each row
- Name
- Description
- File Path - for the image of the animal

# Example Content in Zoo Table

| _id | name | description | file_path |
|---|---|---|---|
| 1 | alpaca | Furry, four-legged animal | /sdcard/alpaca.jpg |
| 2 | monkey | Funny, likes to jump. | /sdcard/monkey.jpg |
| 3 | whale | Big sea creature | /sdcard/whale.jpg |
| ... | ... | ... | ... |

# Database Schema

A database schema defines the structure of a database.

You can create a table with the CREATE TABLE query:

```
CREATE TABLE Zoo (
    _id integer primary key autoincrement,
    name text,
    description text,
    file_path text);
```

# ZooDbHelper.java

```
public class ZooDbHelper extends SQLiteOpenHelper {
        public static final String ID_COLUMN = "_id";
        public static final String NAME_COLUMN = "name";
        public static final String DESCRIPTION_COLUMN = "description";
        public static final String FILE_PATH_COLUMN = "filepath";

        public static final String DATABASE_TABLE = "Zoo";
        public static final int DATABASE_VERSION = 1;
        private static final String DATABASE_CREATE = String.format(
                        "CREATE TABLE %s (" +
                        "  %s integer primary key autoincrement, " +
                        "  %s text," +
                        "  %s text," +
                        "  %s text)",
                        DATABASE_TABLE, ID_COLUMN, NAME_COLUMN,
DESCRIPTION_COLUMN, FILE_PATH_COLUMN);
```

# ZooDbHelper.java (Continued)

```
public ZooDbHelper(Context context) {
        super(context, DATABASE_TABLE, null, DATABASE_VERSION);
}

@Override
public void onCreate(SQLiteDatabase db) {
        db.execSQL(DATABASE_CREATE);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + DATABASE_TABLE);
        onCreate(db);
}
```

# Inserting

```
SQLiteDatabase db = new ZooDbHelper(this).getWritableDatabase();
ContentValues newValues = new ContentValues();
newValues.put(ZooDbHelper.NAME_COLUMN, "alpaca");
newValues.put(ZooDbHelper.DESCRIPTION_COLUMN, "An alpaca looks like a
llama.");
newValues.put(ZooDbHelper.FILE_PATH_COLUMN, "/storage/alpaca.png");
db.insert(ZooDbHelper.DATABASE_TABLE, null, newValues);
```

# File location

Internal storage (by default) path:
/data/data/$PACKAGE_NAME/databases

# SQL Queries

SQL query to select field with a specific field value:

```
SELECT field1, field2
FROM table
WHERE field1 = value
```

Example:

```
SELECT name, description
FROM Zoo
WHERE name = 'alpaca';
```

# Querying

```
String where = null;
String whereArgs[] = null;
String groupBy = null;
String having = null;
String order = null;
String[] resultColumns = {ZooDbHelper.ID_COLUMN, ZooDbHelper.NAME_COLUMN,
ZooDbHelper.DESCRIPTION_COLUMN, ZooDbHelper.FILE_PATH_COLUMN};
Cursor cursor = db.query(ZooDbHelper.DATABASE_TABLE, resultColumns, where,
whereArgs, groupBy, having, order);
while (cursor.moveToNext()) {
        int id = cursor.getInt(0);
        String name = cursor.getString(1);
        String description = cursor.getString(2);
        String filepath = cursor.getString(3);
        Log.d("ZOO", String.format("%s,%s,%s,%s", id, name, description,
filepath));
}
```

# Deleting

```
String whereClause = ZooDbHelper.ID_COLUMN + "=?";
String[] whereArgs = {"4"};
db.delete(ZooDbHelper.DATABASE_TABLE, whereClause, whereArgs);
```

# Updating

```
String whereClause = ZooDbHelper.ID_COLUMN + "= ?";
String[] whereArgs = {"1"};

ContentValues newValues = new ContentValues();
newValues.put(ZooDbHelper.NAME_COLUMN, "alpaca");
newValues.put(ZooDbHelper.DESCRIPTION_COLUMN, "An alpaca is ugly.");
newValues.put(ZooDbHelper.FILE_PATH_COLUMN, "/storage/alpaca.png");

db.update(ZooDbHelper.DATABASE_TABLE, newValues, whereClause, whereArgs);
```

# Network Storage

- Save and retrieve data over the network
- Advantage
  - Everything backed up in cloud
  - Easy syncing between devices
- Disadvantages
  - Requires network connection
  - Adds latency

# Mobile Backend-as-a-Service (MBaaS)

- Cloud services designed for mobile:
  - storage
  - user management
  - push notifications
  - social networking services
  - analytics
- Cross-platform APIs: Android, iOS, web
- MBaaS comparison spreadsheet

# Firebase - Real-time communication

- MBaaS acquired by Google in 2014
- Popular for real-time communication
- Real-time updates
  - Firebase Android Quick Start Guide
  - Web view of data: https://intense-torch-2798.firebaseio.com/

# Firebase - Example Writing

```
public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Firebase.setAndroidContext(this);
        Firebase myFirebaseRef = new Firebase("https://intense-
torch-2798.firebaseio.com/");

        // Setting a value
        myFirebaseRef.child("message").setValue("Do you have
data? You'll love Firebase.");
```

# Firebase - Example Reading

```
// Callback to get the message element
myFirebaseRef.child("message").addValueEventListener(new
ValueEventListener() {

@Override
public void onDataChange(DataSnapshot snapshot) {
        System.out.println("ValueEventListener: " + snapshot.getValue());
}
@Override public void onCancelled(FirebaseError error) { }
});
```

# Firebase - Example Reading Updates

```
        // Callback to only get updates
        myFirebaseRef.child("message").addChildEventListener(new ChildEventListener() {
            @Override
            public void onChildAdded(DataSnapshot dataSnapshot, String s) {
                System.out.println("ChildEventListener: " + dataSnapshot.getValue());
            }

            @Override
            public void onChildChanged(DataSnapshot dataSnapshot, String s) {}

            @Override
            public void onChildRemoved(DataSnapshot dataSnapshot) {}

            @Override
            public void onChildMoved(DataSnapshot dataSnapshot, String s) {}

            @Override
            public void onCancelled(FirebaseError firebaseError) {}
        });
    }
}
```

# Firebase – API Refernce

https://firebase.google.com/docs/reference

# Summary

- Use Shared Preferences for primitive data
- Use internal device storage for private data
- Use external storage for large data sets that are not private
- Use SQLite databases for structured storage

# Questions?

# Appendix A:
# Android Folder Hierarchy

# Android Folder Hierarchy

- Android uses multiple partitions
- Directory structures might differ between manufacturers

# /data partition

| Directory Name | Description |
|---|---|
| /data/anr | traces from app crashes (App Not Responding) |
| /data/app | .apk files of apps installed by the user |
| /data/backup | Google's Cloud Backup |
| /data/dalvik-cache | optimized versions of installed apps |
| /data/data | app data |
| /data/local | temporary files e.g., from Google Play |
| /data/misc | system configuration (WiFi, VPN, etc.) |
| /data/system | system data (certs, battery stats) |
| /data/tombstones | crash data (e.g., core dumps) |

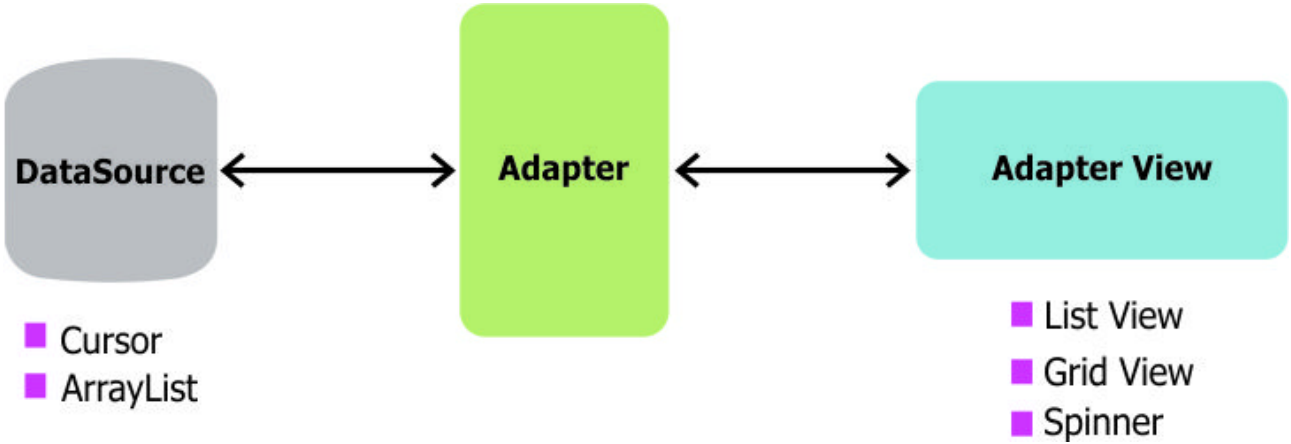
# /data/data partition

| Directory Name | Description |
| --- | --- |
| /data/data/<package_name>/databases | SQLite database files |
| /data/data/<package_name>/lib | Libraries and helpers for the app |
| /data/data/<package_name>/files | Data from Internal Storage API |
| /data/data/<package_name>/shared_prefs | Data from SharedPreferences API |
| /data/data/<package_name>/cache | Caches |

# Appendix B: SQL and List-based Views

# CursorAdapter

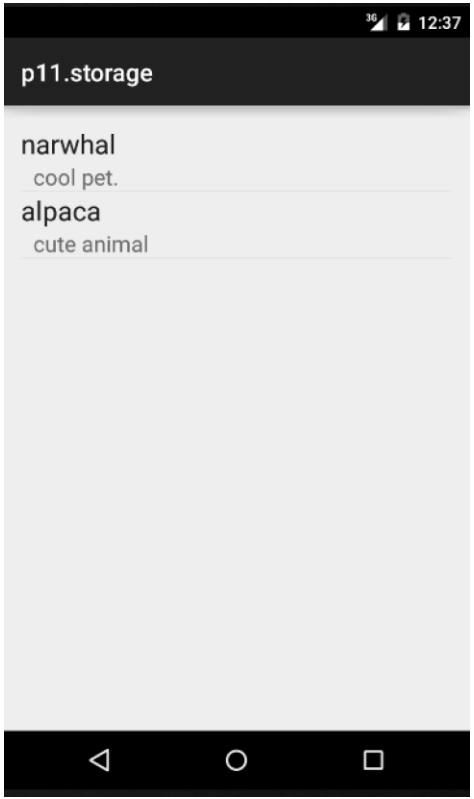
Mechanism to populate a list-based view from a SQLite database



# Steps

1. Create the View Template
2. Define the CursorAdapter
3. Retrieve the cursor for query results
4. Attach the Adapter to the ListView

# Example - Zoo Database



# Define the View Template

```
<!-- res/layout/item.xml -->
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:layout_marginLeft="16dp"
        android:text="Human"
        android:textAppearance="?android:attr/textAppearanceLarge" />
    <TextView
        android:id="@+id/description"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="16dp"
        android:text="Four-limbed mammal"
        android:textAppearance="?android:attr/textAppearanceMedium" />
</LinearLayout>
```

# Define the CursorAdapter (Part 1)

```
public class ZooAdapter extends CursorAdapter {
    public ZooAdapter(Context context, Cursor cursor) {
        super(context, cursor, 0);
    }

    // The newView method is used to inflate a new view and return it,
    // you don't bind any data to the view at this point.
    @Override
    public View newView(Context context, Cursor cursor, ViewGroup parent) {
        return LayoutInflater.from(context).inflate(R.layout.item, parent,
false);
    }

    // More on next slide
}
```

# Define the CursorAdapter (Part 2)

```
// The bindView method is used to bind all data to a given view
// such as setting the text on a TextView.
@Override
public void bindView(View view, Context context, Cursor cursor) {
    // Find fields to populate in inflated template
    TextView nameTextView = (TextView) view.findViewById(R.id.name);
    TextView descriptionTextView = (TextView)
view.findViewById(R.id.description);

    // Extract properties from cursor
    String name =
cursor.getString(cursor.getColumnIndexOrThrow(ZooDbHelper.NAME_COLUMN));
    String description =
cursor.getString(cursor.getColumnIndexOrThrow(ZooDbHelper.DESCRIPTION_COLUM
N));
    // Populate fields with extracted properties
    nameTextView.setText(name);
    descriptionTextView.setText(description);
}
```

# Activity Layout

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
    android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".MainActivity">

    <ListView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/listView"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />

</RelativeLayout>
```

# Activity

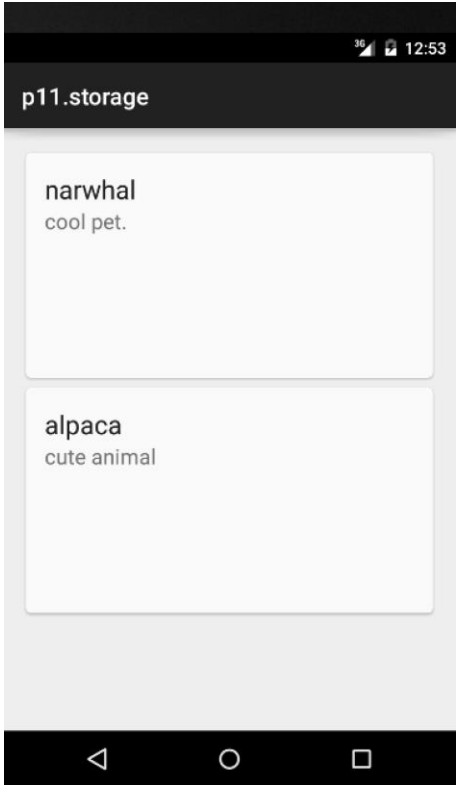
```
public class MainActivity extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // TodoDatabaseHandler is a SQLiteOpenHelper class connecting to SQLite
        ZooDbHelper handler = new ZooDbHelper(this);
        // Get access to the underlying writeable database
        SQLiteDatabase db = handler.getWritableDatabase();

        // Query for items from the database and get a cursor back
        String[] resultColumns = {ZooDbHelper.ID_COLUMN, ZooDbHelper.NAME_COLUMN,
ZooDbHelper.DESCRIPTION_COLUMN, ZooDbHelper.FILE_PATH_COLUMN};
        Cursor cursor = db.query(ZooDbHelper.DATABASE_TABLE, resultColumns, null, null,
null, null, null);

        // Setup cursor adapter using cursor from last step
        ZooAdapter adapter = new ZooAdapter(this, cursor);
        // Attach cursor adapter to the ListView
        ListView listView = (ListView) findViewById(R.id.listView);
        listView.setAdapter(adapter);
    }
```

# Material Design

- CardView
- RecyclerView

# CardView Example

- Wrap the item layout in a CardView element
- Remove the ListView divider by making it transparent:
  `android:divider="#00000000"`
- Add Gradle dependency:
  `compile 'com.android.support:cardview-v7:21.0.+'`



# CardView XML

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:id="@+id/card_view"
    android:layout_width="match_parent"
    android:layout_height="200dp"
    android:layout_gravity="center"
    card_view:cardUseCompatPadding="true"
    card_view:cardCornerRadius="4dp">
  <!-- Original Layout -->
</android.support.v7.widget.CardView>
```

# RecyclerView

There isn't an analogous CursorAdapter for the RecyclerView, but here's a third-party solution: CursorRecyclerAdapter.java