Data Management

Today's Topics

SharedPreferences

Internal Storage

External Storage

SQLite databases

Shared preferences

Small amounts of primitive data

Internal Storage

Small to medium amounts of private data

External Storage

Larger amounts of non-private data

Databases

Store small to large amounts of private, structured data

SharedPreferences

A persistent map

Holds key-value pairs of simple data types

Automatically persisted across application sessions

SharedPreferences

Often used for long-term storage of customizable application data

Account name

Favorite WiFi networks

User customizations

Activity SharedPreferences

To get a SharedPreference object associated with a given Activity

Activity.getPreferences (int mode)

MODE_PRIVATE

Named SharedPreferences

```
Context.getSharedPreferences (
                    String name, int mode)
```

- NAME – NAME OF SharedPreferences FILE
- MODE – MODE_PRIVATE

Writing SharedPreferences

Call SharedPreferences.edit()

Returns a SharedPreferences.Editor instance

Writing SharedPreferences

Add values to SharedPreferences using SharedPreferences.Editor instance

putInt(String key, int value)

putString(String key, String value)

remove(String key)

Writing SharedPreferences

Commit edited values with
SharedPreferences.Editor.commit()

Reading SharedPreferences

Use SharedPreferences methods to read values

getAll()

getBoolean(String key, ...)

getString(String key, ...)

DataManagementSharedPreferences

When the user presses the play button, the application displays a random number

The application keeps track of the highest number seen so far


10:01
DataManagementSharedPreference
HighScore: 412
0
Play
Reset

DataManagementSharedPreferences

```
final SharedPreferences prefs = getPreferences(MODE_PRIVATE);

setContentView(R.layout.main);

// High Score
final TextView highScore = (TextView) findViewById(R.id.high_score_text);
highScore.setText(String.valueOf(prefs.getInt(HIGH_SCORE, 0)));

//Game Score
final TextView gameScore = (TextView) findViewById(R.id.game_score_text);
gameScore.setText(String.valueOf("0"));
```

DataManagementSharedPreferences

```
// Play Button
final Button playButton = (Button) findViewById(R.id.play_button);
playButton.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {

        Random r = new Random();
        int val = r.nextInt(1000);
        gameScore.setText(String.valueOf(val));

        // Get Stored High Score
        if (val > prefs.getInt(HIGH_SCORE, 0)) {

            // Get and edit high score
            SharedPreferences.Editor editor = prefs.edit();
            editor.putInt(HIGH_SCORE, val);
            editor.commit();

            highScore.setText(String.valueOf(val));

        }
    }
});
```

PreferenceFragment

A Class that supports displaying & modifying user preferences

DataManagement PreferenceFragment

This application displays a PreferenceFragment, which allows the user to enter and change a persistent user name


10:21
DataManagementPreferenceFragment
User Name
Adam

DataManagementPreferenceFragment

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

    // Load the preferences from an XML resource
    addPreferencesFromResource(R.xml.user_prefs);

    // Get the username Preference
    mUserNamePreference = (Preference) getPreferenceManager()
            .findPreference(USERNAME);

    // Attach a listener to update summary when username changes
    mListener = new OnSharedPreferenceChangeListener() {
        @Override
        public void onSharedPreferenceChanged(
                SharedPreferences sharedPreferences, String key) {
            mUserNamePreference.setSummary(sharedPreferences.getString(
                    USERNAME, "None Set"));
        }
    };
```

DataManagementPreferenceFragment

```
        // Get SharedPreferences object managed by the PreferenceManager for
        // this Fragment
        SharedPreferences prefs = getPreferenceManager()
                .getSharedPreferences();

        // Register a listener on the SharedPreferences object
        prefs.registerOnSharedPreferenceChangeListener(mListener);

        // Invoke callback manually to display the current username
        mListener.onSharedPreferenceChanged(prefs, USERNAME);

    }
```

File

Class represents a file system entity identified by a pathname

File

Storage areas are classified as internal or external

Internal memory usually used for smaller, application private data sets

External memory usually used for larger, non-private data sets

File API

FileOutputStream

openFileOutput (String name, int mode)

Open private file for writing. Creates the file if it doesn't already exist

FileInputStream

openFileInput (String name)

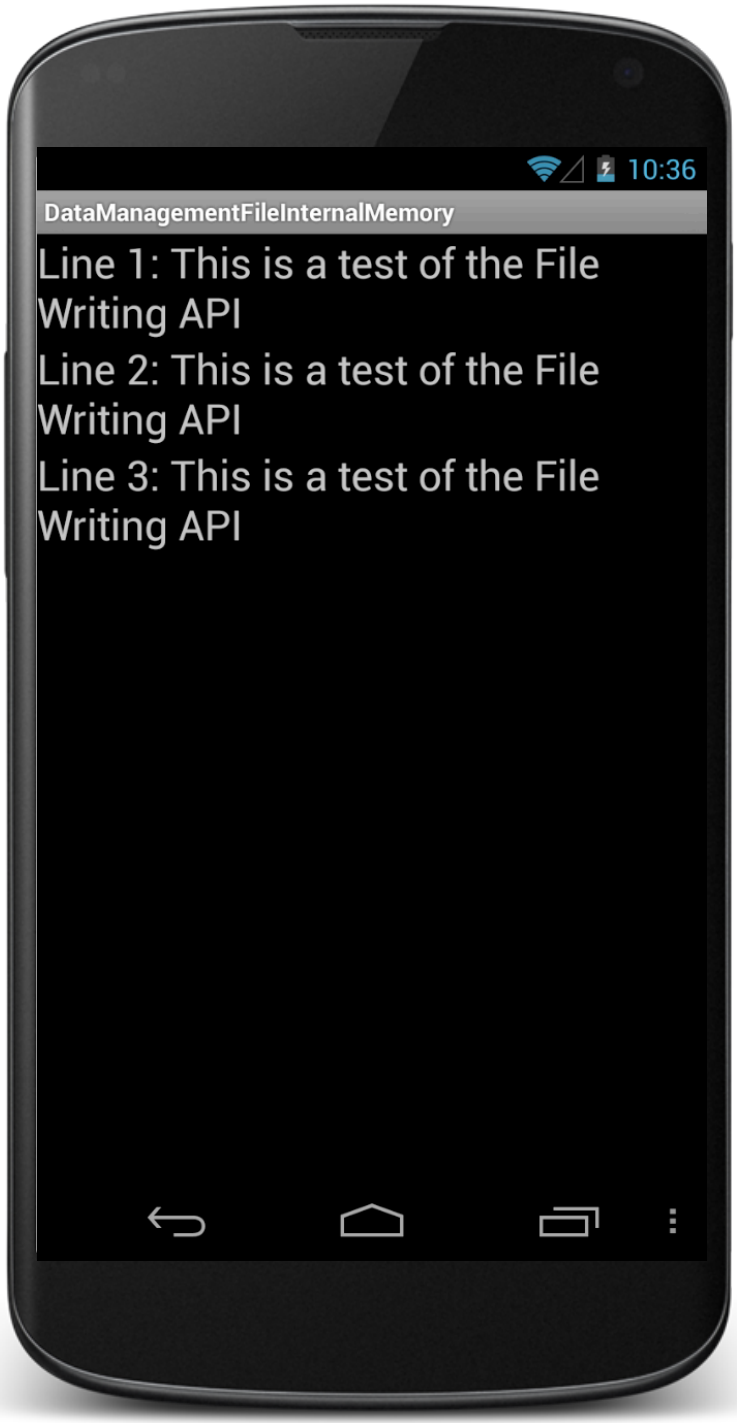
Open private file for reading

Many others. See documentation.

DataManagement
FileInternalMemory

If a text file does not already exist, application writes text to that text file

Application then reads data from the text file and displays it


10:36
DataManagementFileInternalMemory
Line 1: This is a test of the File Writing API
Line 2: This is a test of the File Writing API
Line 3: This is a test of the File Writing API

DataManagementFileInternalMemory

```
private void writeFile() throws FileNotFoundException {

    FileOutputStream fos = openFileOutput(fileName, MODE_PRIVATE);

    PrintWriter pw = new PrintWriter(new BufferedWriter(
            new OutputStreamWriter(fos)));

    pw.println("Line 1: This is a test of the File Writing API");
    pw.println("Line 2: This is a test of the File Writing API");
    pw.println("Line 3: This is a test of the File Writing API");

    pw.close();

}
```

DataManagementFileInternalMemory

```
private void readFile(LinearLayout ll) throws IOException {

    FileInputStream fis = openFileInput(fileName);
    BufferedReader br = new BufferedReader(new InputStreamReader(fis));

    String line = "";

    while (null != (line = br.readLine())) {

        TextView tv = new TextView(this);
        tv.setTextSize(24);
        tv.setText(line);

        ll.addView(tv);

    }

    br.close();

}
```

Using External Memory Files

Removable media may appear/disappear without warning

Using External Memory Files

String Environment.

getExternalStorageState()

MEDIA_MOUNTED – present & mounted with read/write access

MEDIA_MOUNTED_READ_ONLY – present & mounted with read-only access

MEDIA_REMOVED – not present

Etc.

Using External Memory Files

Permission to write external files

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

manifest.xml

DataManagement
FileExternalMemory

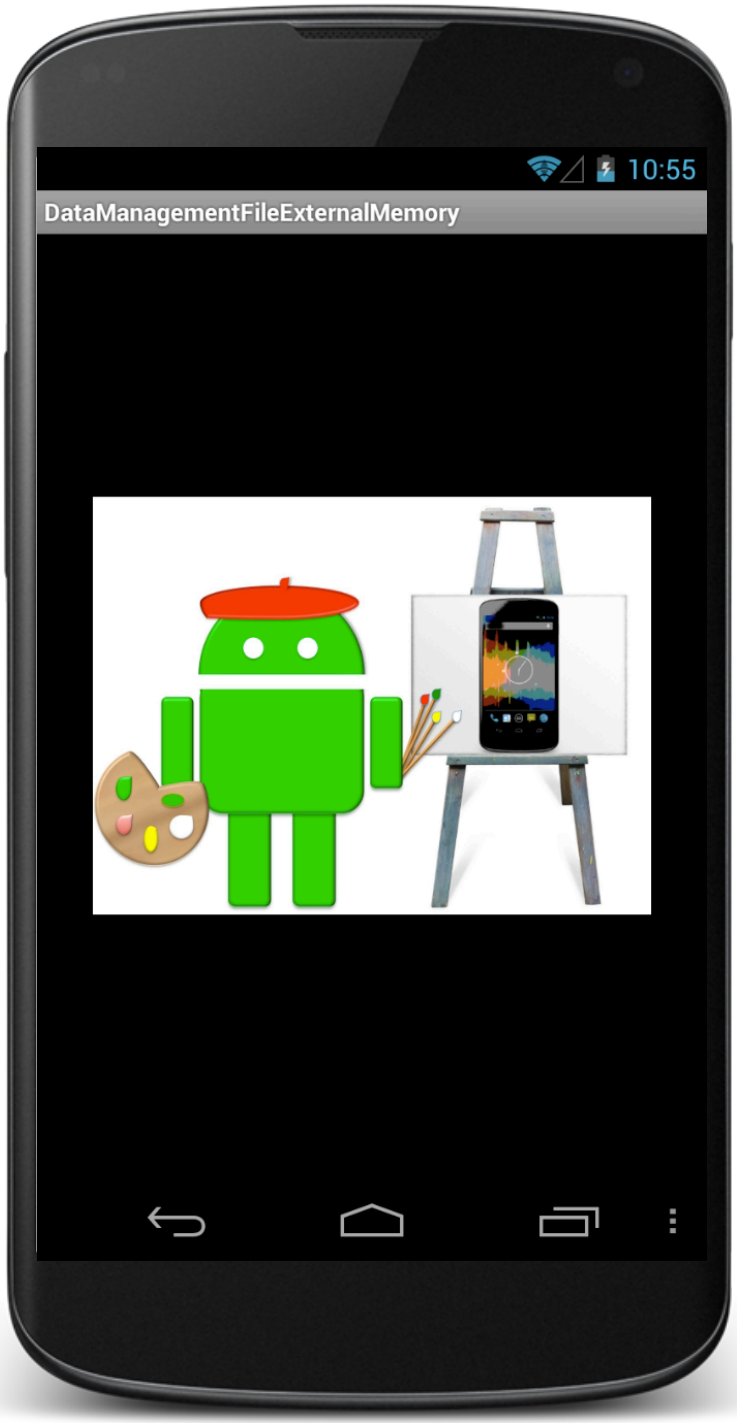
Example

Application reads an image file from the resources directory

Copies that file to external storage

Reads image data from the file in external storage

then displays the image


10:55
DataManagementFileExternalMemory

DataManagementFileExternalMemory

```
private void copyImageToMemory(File outFile) {
    try {

        BufferedOutputStream os = new BufferedOutputStream(
                new FileOutputStream(outFile));

        BufferedInputStream is = new BufferedInputStream(getResources()
                .openRawResource(R.raw.painter));

        copy(is, os);

    } catch (FileNotFoundException e) {
        Log.e(TAG, "FileNotFoundException");
    }
}
```

DataManagementFileExternalMemory

```
private void copy(InputStream is, OutputStream os) {
    final byte[] buf = new byte[1024];
    int numBytes;
    try {
        while (-1 != (numBytes = is.read(buf))) {
            os.write(buf, 0, numBytes);
        }
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        try {
            is.close();
            os.close();
        } catch (IOException e) {
            Log.e(TAG, "IOException");

        }
    }
}
```

Cache Files

Temporary files that may be deleted by the system when storage is low

Files removed when application uninstalled

Cache Files

File Context.getCacheDir()

Returns absolute path to an application-specific directory that can be used for temporary files

Saving cache files

Context.getExternalCacheDir()

returns a File representing external storage directory for cache files

SQLite

SQLite provides in-memory database

Designed to operate within a very small footprint (<300kB)

Implements most of SQL92

Supports ACID transactions

- Atomic, Consistent, Isolated & Durable

Using a Database

Recommended method relies on a helper class called SQLiteOpenHelper

Using a Database

Subclass SQLiteOpenHelper

Call super() from subclass constructor to initialize underlying database

Using a Database

Override onCreate()

Override onUpgrade()

Execute CREATE TABLE commands

Using a Database

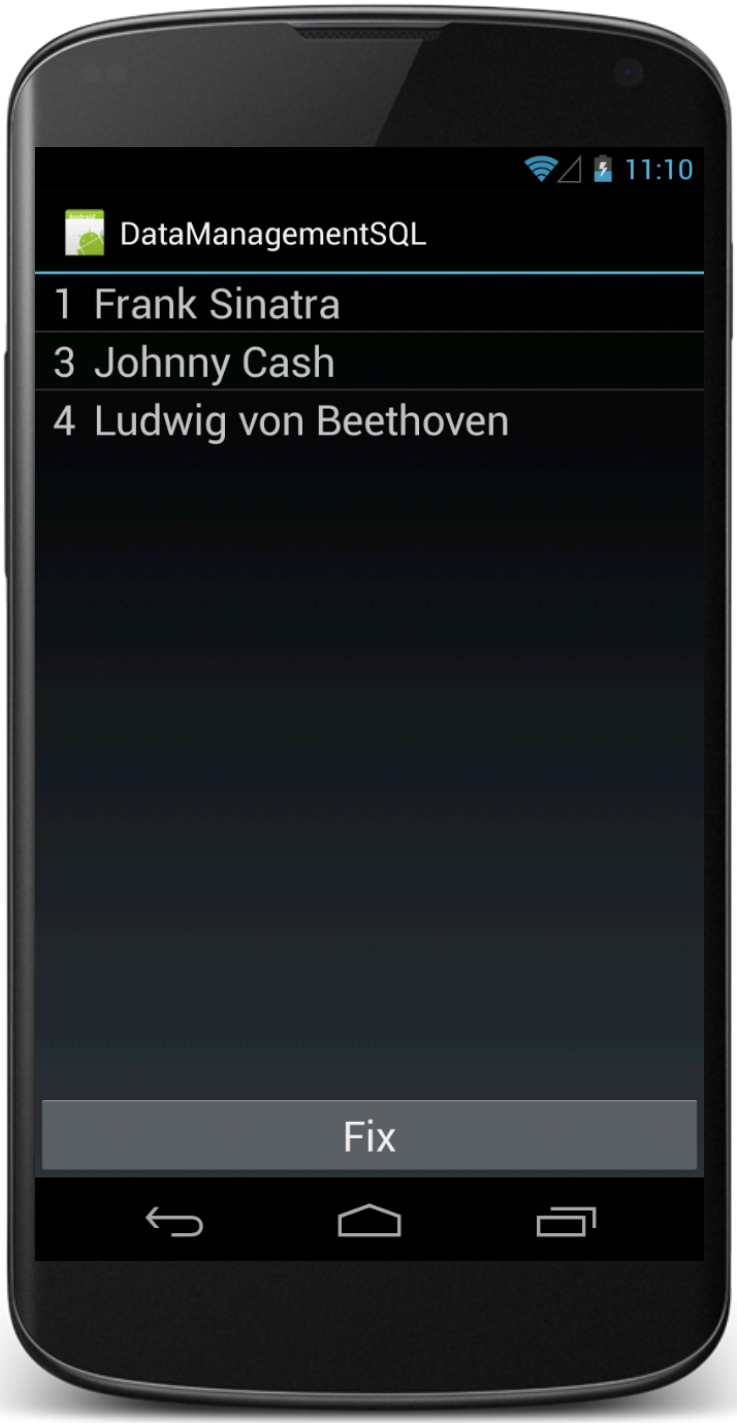
Use SQLiteOpenHelper methods to open & return underlying database

Execute operations on underlying database

DataManagementSQL

Application creates an SQLite database and inserts records, some with errors, into it

When user presses the Fix button, the application deletes, updates and redisplays the corrected database records


11:10
DataManagementSQL
1 Frank Sinatra
3 Johnny Cash
4 Ludwig von Beethoven
Fix

DataManagementSQL

```
// Insert several artist records
private void insertArtists() {

    ContentValues values = new ContentValues();

    values.put(DatabaseOpenHelper.ARTIST_NAME, "Frank Sinatra");
    mDbHelper.getWritableDatabase().insert(DatabaseOpenHelper.TABLE_NAME, null, values);

    values.clear();

    values.put(DatabaseOpenHelper.ARTIST_NAME, "Lady Gaga");
    mDbHelper.getWritableDatabase().insert(DatabaseOpenHelper.TABLE_NAME, null, values);

    values.clear();

    values.put(DatabaseOpenHelper.ARTIST_NAME, "Jawny Cash");
    mDbHelper.getWritableDatabase().insert(DatabaseOpenHelper.TABLE_NAME, null, values);

    values.clear();

    values.put(DatabaseOpenHelper.ARTIST_NAME, "Ludwig von Beethoven");
    mDbHelper.getWritableDatabase().insert(DatabaseOpenHelper.TABLE_NAME, null, values);
}
```

DataManagementSQL

```
// Returns all artist records in the database
private Cursor readArtists() {
    return mDbHelper.getWritableDatabase().query(DatabaseOpenHelper.TABLE_NAME,
            DatabaseOpenHelper.columns, null, new String[] {}, null, null,
            null);
}
```

DataManagementSQL

```
// Modify the contents of the database
private void fix() {

    // Sorry Lady Gaga :-(
    mDbHelper.getWritableDatabase().delete(DatabaseOpenHelper.TABLE_NAME,
            DatabaseOpenHelper.ARTIST_NAME + "=?",
            new String[] { "Lady Gaga" });

    // fix the Man in Black
    ContentValues values = new ContentValues();
    values.put(DatabaseOpenHelper.ARTIST_NAME, "Johnny Cash");

    mDbHelper.getWritableDatabase().update(DatabaseOpenHelper.TABLE_NAME, values,
            DatabaseOpenHelper.ARTIST_NAME + "=?",
            new String[] { "Jawny Cash" });

}
```

DataManagementSQL

```java
// Delete all records
private void clearAll() {

    mDbHelper.getWritableDatabase().delete(DatabaseOpenHelper.TABLE_NAME, null, null);

}

// Close database
@Override
protected void onDestroy() {

    mDbHelper.getWritableDatabase().close();
    mDbHelper.deleteDatabase();

    super.onDestroy();

}
```

Examining the Database Remotely

Databases stored in

/data/data/<package name>/databases/

Can examine database with sqlite3

adb -s emulator-5554 shell

sqlite3 /data/data/
course.examples.DataManagement.Data
BaseExample/databases/artist_dbd

Next Time

ContentProvider