Single Application Persistent Data Storage

Programming the Android Platform

Some Data Storage Options

- Files
- SharedPreferences
- SQLite database

File

- Represents a file system entity identified by a pathname
- Storage areas classified as internal or external
  - Internal memory usually used for application private files
  - External memory used for public files

File API

- boolean isDirectory()
  - Return true if this File represents a directory
- String getAbsolutePath()
  - Returns the absolute path to this File
- boolean setReadable(boolean readable)
  - Sets read permission on this File
- Many others. See documentation.

Writing an Internal Memory File

```
// Open file with ContextWrapper.openFileOutput()
FileOutputStream fos =
                    openFileOutput(fileName, MODE_PRIVATE);
PrintWriter pw = new PrintWriter(
        new BufferedWriter(new OutputStreamWriter(fos)));
// Write to file
pw.println(...);
  ...
// Close file
pw.close();
```

Reading an Internal Memory File

```
// Open file with ContextWrapper.openFileInput()
FileInputStream fis = openFileInput(fileName);
BufferedReader fr =
    new BufferedReader(new InputStreamReader(fis));
String line = "";
// Read from file
while (null != (line = fr.readLine())) {
   // process data
 }
// Close file
fr.close();
```

Cache Files

- Cache files are temporary files that may be deleted by the system when storage is low
- File Context.getCacheDir()
  - Returns absolute path to an application-specific directory that can be used for temporary files
- Files removed when application uninstalled

External Memory Files

- Removable media may appear/disappear without warning
- 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
- Need permission to write external files
  - <uses-permission android:name= “android.permission.WRITE_EXTERNAL_STORAGE" />

Writing an External Memory File

```
public class FileWriteAndReadActivity extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    ...
    if (Environment.MEDIA_MOUNTED.equals(
                                    Environment.getExternalStorageState())) {
     File outFile = new File(getExternalFilesDir(
                                    Environment.DIRECTORY_PICTURES),fileName);
     try {
       BufferedOutputStream os =
            new BufferedOutputStream(new FileOutputStream(outFile));
       BufferedInputStream is =
            new BufferedInputStream(getResources()
                                                .openRawResource(R.drawable.icon));
      copy(is, os);
     } catch (FileNotFoundException e) {}
  }
}
...
```

Writing an External Memory File

```
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) {…
  } finally {
    try {
      is.close();
      os.close();
    } catch (IOException e) {}
…
```

Saving cache files

- Context.getExternalCacheDir() returns a File representing external storage directory for cache files
- Files removed when application uninstalled

SharedPreferences

- A persistent map
  - Holds key-value pairs of simple data types
  - Automatically managed across application uses
- Often used for long-term storage of customizable application data such as user preferences, e.g.,
  - User ID
  - Favorite WiFi networks

Activity SharedPreferences

- Activity.getPreferences (int mode)
  - Mode: MODE_PRIVATE, MODE_WORLD_READABLE or MODE_WORLD_WRITEABLE
- Returns a SharedPreference object for the current Activity

Application SharedPreferences

- Context.getSharedPreferences (
  String name, int mode)
  - name – name of SharedPreference file
  - Mode – MODE_PRIVATE, MODE_WORLD_READABLE or MODE_WORLD_WRITEABLE
- Returns named SharedPreference object for this context

Writing SharedPreferences

- Call SharedPreferences.edit()
  - Returns a SharedPreferences.Editor instance
- Add values with SharedPreferences.Editor
- Commit values with SharedPreferences.Editor.commit()

Reading SharedPreferences

- Use SharedPreferences methods, e.g.,
  - getAll()
  - getBoolean()
  - getString()

SharedPreferences

```
public class SharedPreferenceReadWriteActivity extends Activity {
  private static String HIGH_SCORE = "high_score";
  public void onCreate(Bundle savedInstanceState) {
    ...
    final SharedPreferences prefs = getPreferences(MODE_PRIVATE);
    final Button go = ...
    go.setOnClickListener(new OnClickListener() {
      public void onClick(View v) {
        ...
        int val= ...
        if (val >  prefs.getInt(HIGH_SCORE, 0)) {
          ...
          SharedPreferences.Editor editor = prefs.edit();
          editor.putInt(HIGH_SCORE, val);
          editor.commit();
        }
...
```

PreferenceActivity

- Class that supports displaying & modifying user preferences

SharedPreferences (cont.)

```
public class DataManagementPreferencesActivity extends Activity {
   SharedPreferences prefs;
   final static String USERNAME = "uname";
   public void onCreate(Bundle savedInstanceState) {
      …
     prefs = PreferenceManager.
                  getDefaultSharedPreferences(getApplicationContext());
     final Button button  = …;
     button.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
           startActivity(new Intent
                                (DataManagementPreferencesActivity.this,
                                 LoadPreferencesActivity.class));
        }
    });
}
…
```

SharedPreferences (cont.)

```
public class LoadPreferencesActivity extends PreferenceActivity {
  protected void onCreate(Bundle savedInstanceState) {
    ...
    final SharedPreferences prefs =
                    PreferenceManager.getDefaultSharedPreferences(this);
    addPreferencesFromResource(R.xml.user_prefs);
    final EditTextPreference uNamePref = (EditTextPreference)
                        getPreferenceScreen().findPreference(USERNAME);
    uNamePref.setSummary(prefs.getString(USERNAME, ""));
    prefs.registerOnSharedPreferenceChangeListener(
     new OnSharedPreferenceChangeListener() {
       public void onSharedPreferenceChanged(
                         SharedPreferences sharedPreferences, String key) {
          uNamePref.setSummary(prefs.getString(USERNAME, ""));
       }
    });
...
```

user_prefs.xml

```
<PreferenceScreen …>
   <EditTextPreference
         android:dialogMessage="Enter Your User Name"
         android:dialogTitle="Change User Name"
         android:positiveButtonText="Submit"
         android:negativeButtonText="Cancel"
         android:title="User Name"
         android:key="uname">
    </EditTextPreference>
</PreferenceScreen>
```

SQLite

- SQLite provides in-memory database
- Designed to operate within a very small footprint (<300kB) within a single cross-platform disk file
- Implements most of SQL92
- Supports ACID transactions
  - ACID: atomic, consistent, isolated & durable

Opening a Database

- Recommended method relies on a helper class called SQLiteOpenHelper
- Create a subclass SQLiteOpenHelper
  - Override onCreate()
  - Execute CREATE TABLE command
- Use Constructor to instantiate subclass
- Use SQLiteOpenHelper methods to open & return underlying database

Opening a Database (cont.)

```
public class DatabaseOpenHelper extends SQLiteOpenHelper {
  final private static String CREATE_CMD =
          "CREATE TABLE artists (”
                  + “_id”+ “ INTEGER PRIMARY KEY   AUTOINCREMENT, ”
                  + “name” + " TEXT NOT NULL)”;
  public DatabaseOpenHelper(Context context) {
    super(context, “artist_db”, null, 1);
  }
  public void onCreate(SQLiteDatabase db) {
    db.execSQL(CREATE_CMD);
  }
…
```

Using a Database

```
public class DatabaseExampleActivity extends ListActivity {
  final static String[] columns ={ "_id", "name"};
  static SQLiteDatabase db = null;
  public void onCreate(Bundle savedInstanceState) {
    ...
    DatabaseOpenHelper dbHelper = new DatabaseOpenHelper(this);
    db = dbHelper.getWritableDatabase();
    insertArtists();
    Cursor c = readArtists();
    deleteLadyGaga();
    setListAdapter(new SimpleCursorAdapter(
                                        this, R.layout.list_layout, c, columns,
                                        new int[] { R.id._id, R.id.name }));

}
```

Insertion

```
private void insertArtists() {
  ContentValues values = new ContentValues();
  values.put(“name”, "Lady Gaga");
  db.insert("artists", null, values);
  values.clear();
  values.put(“name”, "Johnny Cash");
  db.insert("artists", null, values);
  values.clear();
  values.put(“name”, "Ludwig von Beethoven");
  db.insert("artists", null, values);
}
```

Deletion

```
private int deleteLadyGaga() {
  return db.delete(
                    "artists", "name" +"=?", new String [] {"Lady Gaga"});
}
```

Querying

```
private Cursor readArtists() {
// returns all rows
return db.query( "artists", new String [] {“_id”,”name”}, null,
                                        new String[] {}, null, null, null);
}
```

Display

```
…
Cursor c = readArtists();
setListAdapter(
     new SimpleCursorAdapter( this,  R.layout.list_layout, c,
                                          new String [] {“_id”, “name”},
                                          new int[] { R.id._id, R.id.name }));
…
```

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.DataBaseExample/databases/artist_db

Source Code Examples

- DataManagementFileInternalMemory
- DataManagementFileExternalMemory
- DataManagementSharedPreference
- DataManagementPreferenceActivity
- DataManagementSQL