Outline

• Activity Lifecycle
• Services
• Persistence
• Content Providers
Recap

- Android Framework
- Activities
- General UI:
  - Layouts, Handler methods
  - Widgets, Custom UI classes, Handling within the activity
- Specialized UIs:
  - Menus and the Action Bar
  - Special Activities – Preferences
- UI for larger screens:
  - Fragments
The Activity Lifecycle

• Activities are managed by the Android runtime
• Activities have a “lifecycle” consisting of states
  – From creation till death
• Standard (lifecycle) methods on the activity are invoked at each state change (can test by rotating device)
Activity States

• Created: Born to run
• Active: Working 9 to 5
• Paused: I’m about to break
• Resumed: Back to work
• Stopped: Obscured by clouds, vulnerable
Activity Transitions

• Created $\implies$ Active
• Active $\iff$ Paused
• Paused $\implies$ Stopped $\implies$ Active
• Stopped $\implies$ Killed
Timing of Lifecycle Methods
Lifecycle Methods

• **onCreate(Bundle savedInstanceState):** create views, (re) initialize state

• **onStart():** Restore *transient* state or other one-time processing

• **onResume():** Session-specific processing, restore *transient* state

• **onPause():** Save persistent data, release resources, **quickly**! Last method guaranteed to be called.

• **onStop():** Called optionally by runtime

• **onDestroy():** If `finish()` is called, or object is being *temporarily* destroyed. Use `isFinishing()` to distinguish.
Transient State Management Methods

• `onSaveInstanceState(…)`: Called before `onPause()`. Use to save transient state.

• `onRestoreInstanceState(…)`: Called after `onStart()` and `onResume()`. Use to restore state.
public class WhereAmI extends MapActivity implements OnClickListener {
    
    private String whereAmIString = null;
    private static final String WHEREAMISTRING = "WhereAmIString";

    protected void onSaveInstanceState (Bundle outState){
        super.onSaveInstanceState(outState);
        if (whereAmIString != null)
            outState.putString(WHEREAMISTRING, whereAmIString);
    }

    protected void onRestoreInstanceState (Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
        whereAmIString = savedInstanceState.getString(WHEREAMISTRING);
        if (whereAmIString != null)
            myLocationField.setText(whereAmIString);
    }

    ...
Interaction Across Activities

• Activity 1: onPause()
• Activity 2: onCreate(), onStart(), onResume()
• Activity 1: onStop() – if it is being obscured
Starting Activities – Intents and Intent Filters

• Encourages activity reuse among applications
• Uniform mechanism for launching internal and external activities
• Allows loose-coupling between caller and responder
Intent and Intent Filter Components

• Target – fully qualified name (string), direct reference
• Action – standard or custom (string)
• Data (URI)
• Category of the target object
Intent Filters – Examples

<intent-filter> <!-- for SplashScreen activity -->
  <action android:name="android.intent.action.MAIN"/>
  <category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>

<intent-filter> <!-- for Login activity -->
  <action
    android:name="com.wiley.fordummies.androidsdk.tictactoe.Login">
    <category android:name="android.intent.category.DEFAULT" />
  </action>
</intent-filter>
Intent Invocations - Examples

• **Matching component name (Chapter 3)**
  `startActivity(new Intent("com.wiley.fordummies.androidsdk.Login"));`

• **Direct invocation (Chapter 3)**
  `startActivity(new Intent(this, Account.class));`

• **Passing data (Chapter 11)**
  ```
  public void sendScoresViaEmail() {
      Intent emailIntent = new Intent(android.content.Intent.ACTION_SEND);
      emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT, "Look at my AWESOME TicTacToe Score!");
      emailIntent.setType("plain/text");
      emailIntent.putExtra(android.content.Intent.EXTRA_TEXT, 
                          firstPlayerName + " score is " + scorePlayerOne + " and " + secondPlayerName + " score is " + scorePlayerTwo);
      startActivity(emailIntent);
  }
  ```
Tasks and Activity Stacks

- Task: a (conceptual) container for sets of “like-minded” activities
- Roughly corresponds to a Linux process
- Activities are stacked in tasks
- Activities can move across tasks
- Task is requested by calling intent or selected based on taskaffinity attribute in AndroidManifest.xml file
Outline

• Activity Lifecycle
• Services
• Persistence
• Content Providers
Services

• Activities for background, long-term processing
  – Email, music, synchronization
• Local: Accessible by a single application.
• Remote (aka bound): Accessible by all applications on the device
  – See: http://developer.android.com/guide/topics/fundamentals/services.html#CreatingBoundService
• Comparison with threads:
  – For larger-grain processing
  – Lifecycle separated from launching activity
  – More resources consumed
  – More respected 😊 by OS
public class MyPlaybackService extends Service {
    MediaPlayer player;
    @Override
    public IBinder onBind(Intent intent) {...}
    public void onCreate() {
        player = MediaPlayer.create(this, R.raw.sampleaudio); player.setLooping(true);
    }

    public int onStartCommand(Intent intent, int flags, int startId) {
        ...
        Bundle extras = intent.getExtras();
        String audioFileURIString = extras.getString("URIString");
        Uri audioFileURI=Uri.parse(audioFileURIString);
        player.reset(); player.setDataSource(this.getApplicationContext(), audioFileURI);
        player.prepare(); player.start();
        ...
    }

    public void onDestroy() { player.stop(); }
}
Service Invocation

... Intent musicIntent =
    new Intent(this, MyPlaybackService.class);
musicIntent.putExtra("URIString",
    audioFileURI.toString());
startService(musicIntent);
...

Outline

- Activity Lifecycle
- Services
- Persistence
- Content Providers
Saving Persistent Data – Files

- Same as in any Java app
- Internal storage (Linux file system)
  - Local to app
  - Destroyed upon uninstall
- External storage (e.g., SD card)
  - Stored to SD card or flash memory (device-dependent)
  - Shared across applications
  - E.g.:

```java
File ScoresFile = new File(getExternalFilesDir(null), "Scores.txt");
```
Saving Persistent Data – SQLite

• Sqlite.org: “Most widely deployed SQL database engine in the world” 😊
  – Lightweight, transactional

• Helper class for database creation

• Methods for operating on SQL statements:
  – Creating a statement
  – Binding data
  – Executing
SQLite Example – Creating a Database

```java
public class DatabaseHelper {
    private static final String DATABASE_NAME = "TicTacToe.db";
    private static final int DATABASE_VERSION = 1;
    private static final String TABLE_NAME = "Accounts";
    private Context context;
    private SQLiteDatabase db;
    ...
    public DatabaseHelper(Context context) {
        this.context = context;
        TicTacToeOpenHelper openHelper =
            new TicTacToeOpenHelper(this.context); // this creates the DB
        this.db = openHelper.getWritableDatabase();
        this.insertStmt = this.db.compileStatement(INSERT);
    }
}
private static class TicTacToeOpenHelper extends SQLiteOpenHelper
    
    TicTacToeOpenHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE " + TABLE_NAME + " (id INTEGER PRIMARY KEY, name TEXT, password TEXT)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        Log.w("Example", "Upgrading database; this drops & recreates tables.");
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
    }
}
private SQLiteDatabase db;
private SQLiteStatement insertStmt;
private static final String INSERT = "insert into " + TABLE_NAME + "(name, password) values (?, ?)";

public long insert(String name, String password) {
    this.insertStmt.bindString(1, name);
    this.insertStmt.bindString(2, password);
    return this.insertStmt.executeInsert();
}

public void deleteAll() {
    this.db.delete(TABLE_NAME, null, null);
}
/// SQLite Example – Database Operations (2)

public List<String> selectAll(String username, String password) {
  List<String> list = new ArrayList<String>();
  Cursor cursor =
    this.db.query(TABLE_NAME,
    new String[] { "name", "password" },
    "name = "+ username +" AND password= "+ password+",
    null, null, null, "name desc");
  if (cursor.moveToFirst()) do {
    list.add(cursor.getString(0));
    list.add(cursor.getString(1));
  } while (cursor.moveToNext());
  if (cursor != null && !cursor.isClosed()) {
    cursor.close();
  }
  return list;
}
Outline

• Activity Lifecycle
• Services
• Persistence
• Content Providers
Sharing Content – Content Providers

- Standard content providers:
  - Contacts, dictionary, media
- Referred to by URI prefix: `content://`
  - Standard providers
- Permission must be requested in the manifest file:
  
  ```xml
  <uses-permission android:name="android.permission.READ_CONTACTS"/>
  ```
public class ContactsView extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.contacts);
        TextView contactView = (TextView) findViewById(R.id.contactsView);
        Cursor cursor = getContacts();
        while (cursor.moveToNext()) {
            String displayName = cursor.getString(cursor.getColumnIndex(ContactsContract.Data.DISPLAY_NAME));
            contactView.append("Name: ");
            contactView.append(displayName);
            contactView.append("\n");
        }
    }
}
...
private Cursor getContacts() {
    // Run query
    Uri uri = ContactsContract.Contacts.CONTENT_URI;
    String[] Columns =
        new String[] { ContactsContract.Contacts._ID,
                       ContactsContract.Contacts.DISPLAY_NAME };
    String Selection =
        ContactsContract.Contacts.IN_VISIBLE_GROUP + " = '" +
        ("1") + "'";
    String[] selectionArgs = null;
    String sortOrder = ContactsContract.Contacts.DISPLAY_NAME +
        " COLLATE LOCALIZED ASC";
    return this.managedQuery(uri, Columns, Selection, selectionArgs, sortOrder);
}
Even More?! 

- Yes!
- More widgets, styles and themes
- Maps
- Email, media, telephony
- Sensors
Summary

• Activities: “One screen” of content that user sees
  – Can be paused, resumed by users
  – Managing Activity lifecycle is crucial

• Services: Long-running tasks

• Persistence:
  – Files (on SD card)
  – SQLite database

• Content Providers: mechanism for sharing data