Introducing Mobile Application Development for Android

Presented by:
Ahmed Misbah
Agenda

- Introduction
- Android SDK Features
- Developing an Android Application
- Android Market
- Android Application Trends
INTRODUCTION
What is Android?

- Google’s mobile operating system
- Based on Linux Kernel
- Offers an SDK and NDK
- Latest SDK version is 3.0/3.1 (Honeycomb)
Linux Kernel

Android uses Linux for its memory management, process management, networking, and other operating system services
Native Libraries

- Shared libraries all written in C or C++
- Compiled for the particular hardware architecture used by the phone
- Preinstalled by the phone vendor
- Can be developed using NDK
Native Libraries (cont’d)

- Surface Manager
- 2D, 3D Graphics
- Media Codecs
- SQL Database
- Browser Engine
Android Runtime

**Dalvik VM**
- Google’s implementation of Java
- Optimized for mobile devices
- Runs `.dex` files which are more compact and efficient than standard `.class` files

**Core Java libraries**
- Not those of JSE or JME but have some similarities
Application Framework

- Activity Manager
- Content providers
- Resource Manager
- Location Manager
- Notification Manager
Application Lifecycle

Starting:
1. onCreate()
2. onStart()
3. onRestoreInstanceState() *
4. onResume()

Running:
3. onResume() (optional)
2. onStart() (optional)
1. onRestart() (optional)

Stopped:
1. onSaveInstanceState() *
2. onStop() (optional)

Paused:
1. onSaveInstanceState() *
2. onPause() (optional)

Destroyed:
onDestroy() or <Process killed>

* (optional)
Building Blocks

- **Activities**: User Interface

- **Intent**: A mechanism for describing a specific action

- **Service**: A task that runs in the background without user interaction

- **Content providers**: is a set of data wrapped up in a custom API to read and write it
Application Structure
Resources

- Stored in `res` folder
- Includes all non code information (e.g. localized text and images)
- Resources compiler compresses and packs all resources in a class named `R`
Every application must have an **AndroidManifest.xml** file in its root directory.

Manifest presents essential information about the application to the Android system:

- Java package
- Components of the application (Activities, Services, etc.)
- Permissions the application
- Minimum level of the Android
- Libraries that the application utilizes
Security

Stored in **Android-Manifest.xml**

Contains following permissions:

- INTERNET
- READ_CONTACTS
- WRITE_CONTACTS
- RECEIVE_SMS
- ACCESS_COARSE_LOCATION
- ACCESS_FINE_LOCATION
- WRITE_EXTERNAL_STORAGE
ANDROID SDK FEATURES
Android SDK Features

- User Interface
- Graphics
- Multimedia
- Data Storage
- Networking
- Locating and Sensing
- Telephony, Messaging and Notification
- I18N and Localization
USER INTERFACE
Overview

Design Methods

- Declare UI elements in XML (Declarative design)
- Instantiate UI elements at runtime
Activity Class

- **Activity** class takes care of creating a window in which UI can be placed.
- There is a **one-to-one** relationship between an Activity and a UI screen.
- Activities are made up of subcomponents called **Views**.
Views

Android Views are what your users will see and interact with.
Views (cont’d)
Views (cont’d)
Views (cont’d)
Resources

Some important resource files

- /res/layout/main.xml
- /res/layout-land/main.xml
- /res/values/strings.xml
- /res/values/colors.xml
- /res/values/styles.xml
- /res/menu/menu.xml
Layouts

- Layouts are defined in `/res/layout/main.xml`

- Layouts are automatically converted to a member in the `layout` inner class in `R` class
Layouts (cont’d)

Linear Layout: Arranges its children in a single column or row. This is the most common layout you will use.
Relative Layout: Arranges its children in relation to each other or to the parent. This is often used in forms.
Table Layout: Arranges its children in rows and columns, similar to an HTML table
Tab Activity

dthis is a third tab
Listeners

- Tell Android which object to callback when the user touches or clicks the view

- Use `setOnClickLisener()` method that needs to be passed an object that implements the `OnClickListener` Java interface

- Set `android:onClick` property with the method name that handles the click action
Applying a Theme

Android is packaged with several themes that you can reference by name, or you can make up your own theme by extending existing ones and overriding their default values.

You can define your own custom theme in res/values/styles.xml.
Menus

Android supports three kinds of menus:

- **Options Menu**: the menu you get when you press the physical Menu button

- **Context Menu**: that pops up when you press and hold your finger on the screen

- **Sub Menu**: a floating list of menu items that the user opens by pressing a menu
Menus (cont’d)
Dialogs

A small window that appears in front of the current Activity
Search Activity

- **analogy**: n. drawing a comparison in order to show a similarity.
- **analyst**: n. someone who is skilled at analyzing data.
- **analyze**: v. break down into components or essential features.
- **anchor**: v. fix firmly and stably.
GRAPHICS
Overview

- Android provides a powerful graphics library that supports drawing of 2D shapes and developing animations.

- 2D Graphics since version 3.0 can also be hardware accelerated.

- For 3D Graphics, android provides an implementation based on OpenGL ES 1.0 APIs.
2D Graphics

Android offers a custom 2D graphics library for drawing and animating shapes and images.

The `android.graphics.drawable` and `android.view.animation` packages are where you'll find the common classes used for drawing and animating in two-dimensions.
Drawable class

- A **Drawable** is a general abstraction for “something that can be drawn.”

- Subclasses include **BitmapDrawable**, **ShapeDrawable**, **PictureDrawable**, etc.

- **draw** method takes a **Canvas** which handles drawing of primitive shapes (Bitmap, rectangle, line, circle, etc.)
Animations

Android support 2 animation frameworks:

- **Property Animation**: latest animation framework that allows developers to animate almost anything

- **View Animation**: provides the capability to only animate View objects
Property Animation

- Available since version 3.0

- Changes a property's (a field in an object) value over a specified length of time
View Animation

- Tween Animation: can perform a series of simple transformations (position, size, rotation, and transparency) on the contents of a View object

- Frame Animation: a traditional animation in the sense that it is created with a sequence of different images, played in order, like a roll of film
Live Wallpaper

Introduced in version 2.1

Like any normal application, can use any feature (MapView, Accelerometer, GPS, ...)

Provides an Engine for handling rendering of Wallpaper

Provide “settings screen”
MULTIMEDIA
Audio

Steps for playing Audio:

1. Put sound files in `res/raw` directory
2. Create `android.media.MediaPlayer` instance
3. `mediaPlayer.start()`
   - `stop()`, `pause()`, `reset()`, `prepare()`, `setLooping()`, ...

Useful methods:

- `setVolumeControlStream(AudioManager.STREAM_MUSIC)`
- `setOnCompletionListener()`
Video

Exactly similar to Audio
- MediaPlayer => start(), stop()
- Just add "Surface" to preview the video

Or simply use VideoView:
- video.setVideoPath("/data/samplevideo.3gp");
- video.start();
Preferences

- Out of the box preference screen

- Allows reading and writing application resources

- Preference screen components written in resource XML

- Preference screen loaded from class which extends `PreferenceActivity`
Accessing Internal File System

- Allows access to package private directory created at install time (/data/data/packagename)

- Few helper methods are provided on the Context:
  - `deleteFile()`
  - `fileList()`
  - `openFileInput()`
  - `openFileOutput()`
Accessing SD Card

- Requires WRITE_EXTERNAL_STORAGE permission

- Uses /sdcard/ instead of /data/

```java
// Load and start the movie
video.setVideoPath("/sdcard/samplevideo.3gp");
video.start();
```

- Use standard java.io to access files
Database access

- Android utilizes SQLite

- A SQLite database is just a single file

- Android stores the file in the /data/data/packagename/databases directory

- Uses standard SQL DML and DDL scripts
Database access (cont’d)

- DB is accessible through a class that extends SQLiteOpenHelper

- Provides an object of SQLiteDatabase that exposes methods like:
  
  - `db.execSQL(String sql)`
  - `db.insert(String tablename, String nullColumnHack, ContentValues values);`
  - `db.query (String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit)`
Database access (cont’d)

**query** methods returns an object of **Cursor** class over a result set

Data binding is possible using **ListActivity**
NETWORKING
Available using `ConnectivityManager`:

```java
ConnectivityManager cMgr = (ConnectivityManager) this.getSystemService(Context.CONNECTIVITY_SERVICE);

NetworkInfo netInfo = cMgr.getActiveNetworkInfo();

this.status.setText(netInfo.toString());
```
Sockets

Similar to JSE socket programming
Bluetooth Socket

- Requires permission
  android.permission.BLUETOOTH

Setting up Bluetooth:
- Enabling Bluetooth
- Finding Paired Devices
- Searching for Devices
- Enabling Discoverability
Bluetooth Socket (cont’d)

- You can connect as a Server using `BluetoothServerSocket`

- You can also connect as a client using `BluetoothDevice` and `BluetoothSocket`

- Connections are managed by `BluetoothSocket` using `InputStream` and `OutputStream`
Working with HTTP

Similar to JSE using **HttpURLConnection** and **java.net**

Robust HTTP with **HttpClient**

```java
HttpClient httpclient = new DefaultHttpClient();
HttpPost httppost = new HttpPost("http://www.website.org/service.php");

List<NameValuePair> pairs = new ArrayList<NameValuePair>(2);
pairs.add(new BasicNameValuePair("ID", "VALUE"));
httppost.setEntity(new UrlEncodedFormEntity(pairs));

HttpResponse webServerAnswer = httpclient.execute(httppost);
```
Working with Web Services

- SOAP Web Services can be invoked using 3rd party library such as org.ksoap2

- RESTful Web Service can be implemented using HttpURLConnection and XML parser and/or JSON library
LOCATING AND SENSING
Locating Overview

Supported Providers:
- GPS
- Cell Towers
- WI-FI

Access to location information is protected by Android permissions:
- ACCESS_COARSE_LOCATION
- ACCESS_FINE_LOCATION
Location Manager

- Provides access to the system location services

- Retrieved through `Context.getSystemService(Context.LOCATION_SERVICE)`. 
Useful Methods:

- `getAllProviders()`
- `getBestProvider(Criteria criteria, boolean enabledOnly)`
- `getLastKnownLocation(String provider)`
- `requestLocationUpdates(String provider, long minTime, float minDistance, LocationListener listener)`
Location Listener

Used for receiving notifications from the `LocationManager` when the location is updated

Location Listener methods:
- `onLocationChanged(Location location)`
- `onProviderDisabled(String provider)`
- `onProviderEnabled(String provider)`
- `onStatusChanged(String provider, int status, Bundle extras)`
Geocoding

The process of finding associated geographic coordinates (often expressed as latitude and longitude) from other geographic data, such as street addresses, or zip codes (postal codes)

Reverse Geocoding performs the opposite operation
Geocoding (cont’d)

Geocoding

Address ——> Geocoding ——> Coordinates

Reverse Geocoding

Coordinates ——> Reverse Geocoding ——> Address
Geocoder Class

A class for handling Geocoding and Reverse Geocoding

Useful methods:

- `getFromLocation(double latitude, double longitude, int maxResults)`
- `getFromLocationName(String locationName, int maxResults, double lowerLeftLatitude, double lowerLeftLongitude, double upperRightLatitude, double upperRightLongitude)`
- `getFromLocationName(String locationName, int maxResults)`
Sensors

Android supports many different types of sensor devices:

- **TYPE_ACCELEROMETER**: Measures acceleration in the x-, y-, and z-axes
- **TYPE_LIGHT**: Tells you how bright your surrounding area is
- **TYPE_MAGNETIC_FIELD**: Returns magnetic attraction in the x-, y-, and z-axes
- **TYPE_ORIENTATION**: Measures the yaw, pitch, and roll of the device
- **TYPE_PRESSURE**: Senses the current atmospheric pressure
- **TYPE_PROXIMITY**: Provides the distance between the sensor and some object
- **TYPE_TEMPERATURE**: Measures the temperature of the surrounding area
Sensor Manager

- Allows utilizing the device's sensors

- An instance of this class is retrieved by calling `Context.getSystemService(Context.SENSOR_SERVICE)`

- Specific sensors are retrieved using `getDefaultSensor(Sensor.TYPE_ACCELEROMETER)`
SensorEventListener

- Receives notifications from the **SensorManager** when sensor values are updated

- Callback Methods:
  - `onAccuracyChanged(Sensor sensor, int accuracy)`
  - `onSensorChanged(SensorEvent event)`
TELEPHONY, MESSAGING AND NOTIFICATIONS
Telephony Manager

- Provides access to information about the telephony services on the device

- Requires `READ_PHONE_STATE` permission

- Get an instance of this class by calling `Context.getSystemService(Context.TELEPHONY_SERVICE)`
PhoneStateListener A listener class for monitoring changes in specific telephony states on the device, including service state, signal strength, message waiting indicator (voicemail), and others
SMS Messages Support

Android API supports developing applications that can send and receive SMS messages

*SmsManager* Manages SMS operations such as sending data, text, and PDU SMS messages

Requires *SEND_SMS* permission
Notifications

A **Notification** is a persistent message that not only shows up in the status bar but stays in a notification area until the user deletes it.

- Managed by **Notification** and **NotificationManager** Classes.
I18N AND LOCALIZATION
Localization

All resources in Android can be configured to support localization

Example:
- Default (English): `res/values/strings.xml`
- Arabic: `res/values-ar/strings.xml`
- French: `res/values-fr/strings.xml`

Use Android context to change locale

```
Locale locale = context.getResources().getConfiguration().locale
```
DEVELOPING AN ANDROID APP
SDK

Contains Dalvik VM, Java libraries and Emulator
An Android plugin, called Android Development Tools (ADT) (https://dl-ssl.google.com/android/eclipse/), is available for Eclipse IDE.

MotoDev is an Eclipse based IDE with tremendous features for Android Development.
Create an AVD
Create new project
Development Checklist

- Memory
- App speed
- Sending texts or accessing the Internet costs money for some mobile users.
- Internet access: The Net is not always available for the mobile user - and it's typically slow.
- Usage fees
- Display capabilities
- User input tools (keyboard, mouse, display, etc.). Some mobile devices use arrows, some have extra buttons, and others have touch screens.

Mobile devices have limited memory and a lot of it is already allocated.

Mobile device users don't have a lot of time, so apps need to load and work quickly.
Debugging

<table>
<thead>
<tr>
<th>Time</th>
<th>pid</th>
<th>tag</th>
<th>Message</th>
</tr>
</thead>
</table>
Package and deploy

- Sign application using Eclipse Export Wizard
- Choose a strong password to sign your application
- Application is exported to an APK file
Publish to market

Publishing checklist:

1. Test your application extensively on an actual device
2. Consider adding an End User License Agreement in your application
3. Consider adding licensing support
4. Specify an icon and label in the application's manifest
5. Turn off logging and debugging and clean up data/files
Publish to market (cont’d)

6. Version your application
7. Obtain a suitable cryptographic key
8. Register for a Maps API Key, if your application is using MapView elements
9. Sign your application
10. Obfuscate your code using ProGuard

Follow MotoDev publishing steps
Support and Resources

Android Developers
(http://developer.android.com/index.html)

Offers SDK downloads, Reference (JAVADOCs), Resources and Dev Guide
Overview

Android’s application repository

Similar to Apple’s App Store and Nokia’s Ovi Store

By August 2010, there were over 80,000 applications available for download, with over 1 billion application downloads.
Overview (cont’d)
Overview (cont’d)
Publishing on Android Market

1. Create a developer profile using a Google account

2. Pay a registration fee of 25$

3. For paid applications, Google receives a 30% transaction fee

4. Google handles version updates
ANDROID APPLICATION TRENDS
What are analysts saying?

“Android Is Destroying Everyone, Especially RIM -- iPhone Dead In Water” - Business Insider

“Android market share to near 50 percent” - Gartner

“Android's Market Share Soars Ahead Of Apple iPhone's” - The Huffington Post
Market Share

Data collected on Q4 2010
Market Share (cont’d)

Worldwide smartphone market share
Millions of units shipped

- Google*: 33.3 (615.1% increase from 4Q 2009), 31.0 (85.9% increase from 4Q 2009)
- Nokia: 23.9 (30.0% increase from 4Q 2009), 16.2 (36.0% increase from 4Q 2009)
- Apple: 8.7 (20.3% decrease from 4Q 2009), 6.2
- RIM: 10.7 (36.0% increase from 4Q 2009), 14.6
- Microsoft: 3.9, 3.1
- Others: 1.8, 3.0

* Google numbers include Android and OMS, Tapas platform variants
Usage Share

Data collected on May 2011
During 2010, a total of 170,000 applications were published on Google’s Android Market. 75% of them were still active and available at the close of the year.

78% were updated by the developer in the last 6 months.
27% were updated in the last month.
Only a third of the applications available are paid.

Among the paid applications, the most common price is $1. Over half the paid applications are offered at this price.

$1 is also the minimum price at which applications are sold.

Conclusion: Half the paid applications are sold at the minimum price.
Category Analysis
Category Analysis (cont’d)
Key factors for 2010

- Entertainment category will remain most popular
- Free applications will continue to dominate
- The rise of books and reference categories
Future of Android Apps

- Localized content
- More mature business applications
- Applications for Tablet devices
- Applications utilizing location and maps
- Social Network aggregators
- Satellite Systems (SSTL)
- Software Development process for mobile applications
Gartner Top 10 Mobile Applications for 2012

- Mobile Money Transfer
- Location-Based Services
- Mobile Search
- Mobile Browsing
- Mobile Health Monitoring
- Mobile Payment
- Near Field Communication Services
- Mobile Advertising
- Mobile Instant Messaging
- Mobile Music
Thank you