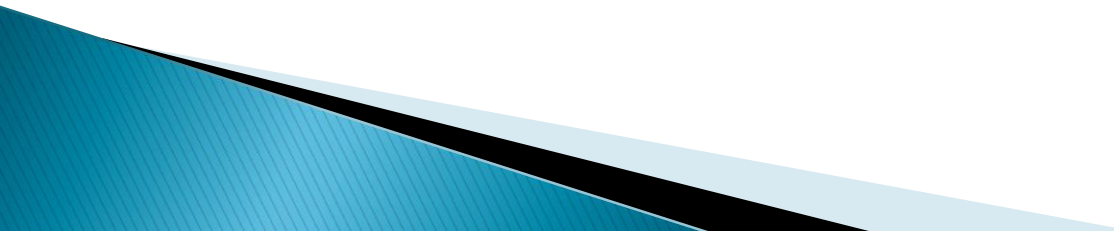


Google Android Development

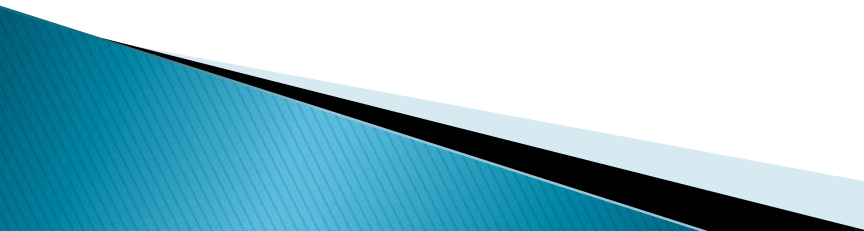
18-848D Sports Technology



Resources

- ▶ <http://developer.android.com/index.html>
 - ▶ <http://developer.android.com/guide/topics/fundamentals.html>
 - ▶ <http://developer.android.com/community/index.html>
 - ▶ Android has excellent documentation. If you can read, you can program Android.
- 

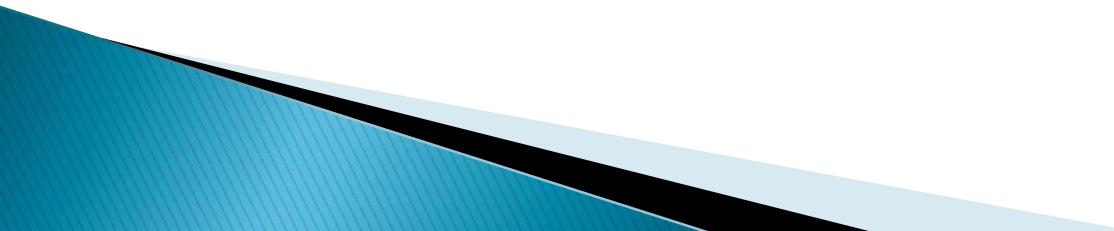
Android Features

- ▶ Application framework
 - ▶ Dalvik virtual machine
 - ▶ Integrated browser
 - ▶ Optimized graphics
 - ▶ SQLite
 - ▶ Media support
 - ▶ GSM Telephony
 - ▶ Bluetooth, EDGE, 3G, and WiFi
 - ▶ Camera, GPS, compass, and accelerometer
 - ▶ Rich development environment
- 

Android Architecture



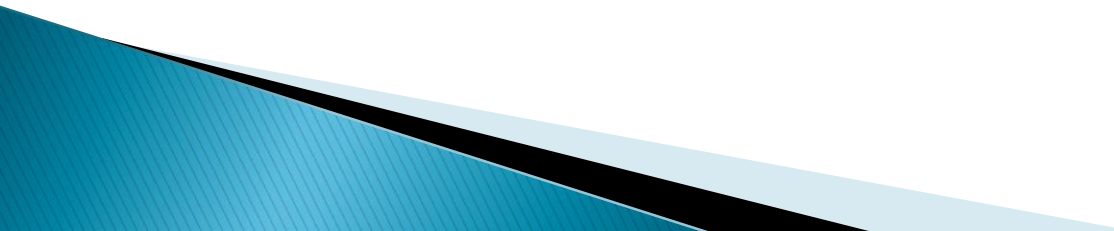
Application Framework

- ▶ Views
 - ▶ Content Providers
 - ▶ Resource Manager
 - ▶ Notification Manager
 - ▶ Activity Manager
- 

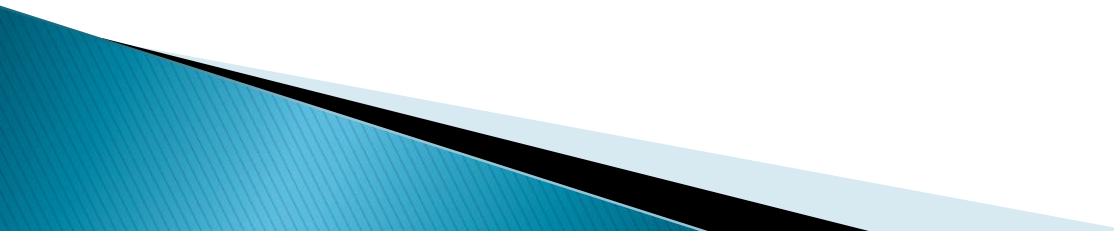
Development Environment Setup

- ▶ Download the SDK:
http://developer.android.com/sdk/1.6_r1/
- ▶ Eclipse Version 3.4 or higher:
Java or RCP version of Eclipse
- ▶ Unpack the SDK
android_sdk_<platform>_<release>
- ▶ Add
android_sdk_<platform>_<release>/tools to
your path
- ▶ Install additional software for Eclipse:
<https://dl-ssl.google.com/android/eclipse/>

Application Components

- ▶ Activity
 - ▶ Services
 - ▶ Broadcast receivers
 - ▶ Content providers
- 

Activity

- ▶ Visual user interface
 - ▶ One or multiple activities per application
 - ▶ Sample text messaging application
 - Activity to view contacts
 - Activity to send a message
 - Activity to change settings
 - Activity to view old messages
- 

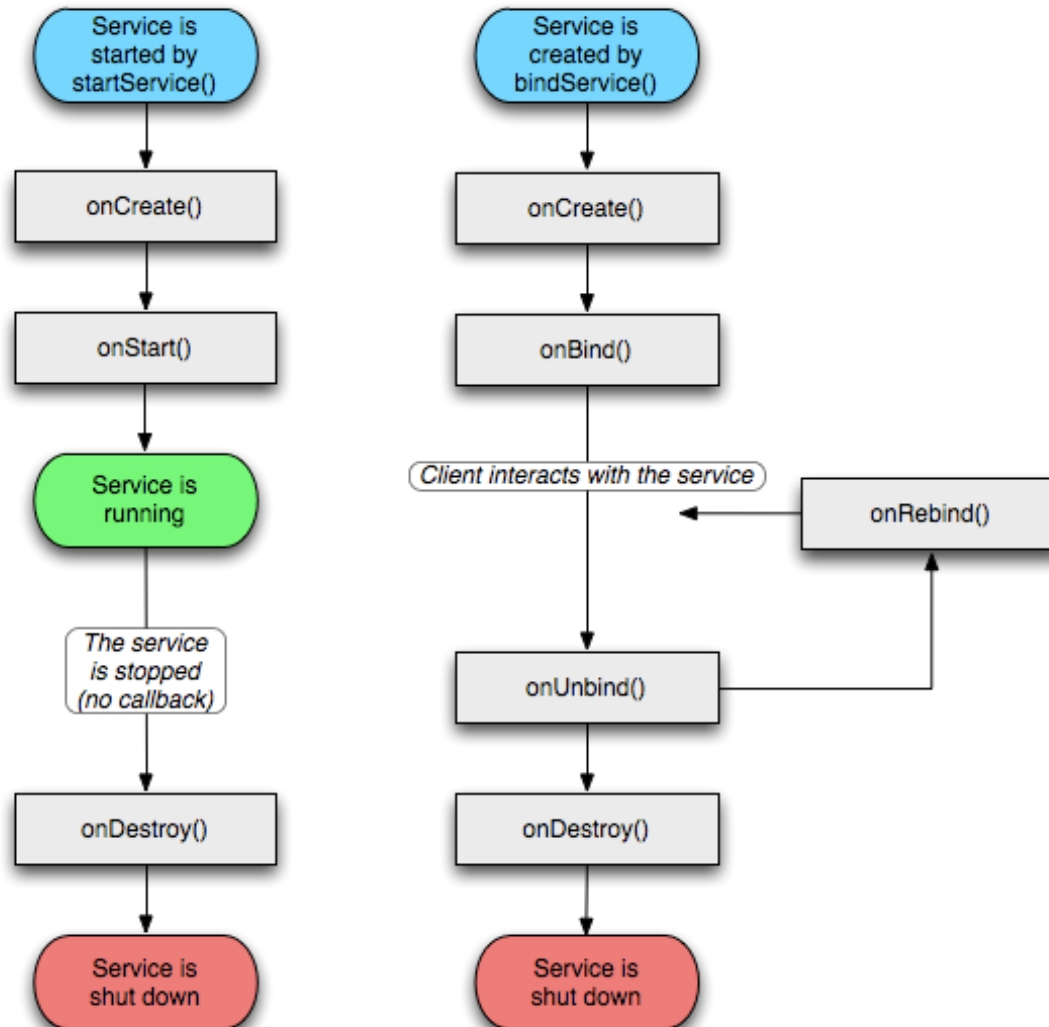
Activity life cycle

Method	Description	Killable?	Next
<u>onCreate()</u>	Called when the activity is first created. This is where you should do all of your normal static set up — create views, bind data to lists, and so on. This method is passed a Bundle object containing the activity's previous state, if that state was captured (see <u>Saving Activity State</u> , later). Always followed by <code>onStart()</code> .	No	<code>onStart()</code>
<u>onRestart()</u>	Called after the activity has been stopped, just prior to it being started again. Always followed by <code>onStart()</code>	No	<code>onStart()</code>
<u>onStart()</u>	Called just before the activity becomes visible to the user. Followed by <code>onResume()</code> if the activity comes to the foreground, or <code>onStop()</code> if it becomes hidden.	No	<code>onResume()</code> or <code>onStop()</code>
<u>onResume()</u>	Called just before the activity starts interacting with the user. At this point the activity is at the top of the activity stack, with user input going to it. Always followed by <code>onPause()</code> .	No	<code>onPause()</code>
<u>onPause()</u>	Called when the system is about to start resuming another activity. This method is typically used to commit unsaved changes to persistent data, stop animations and other things that may be consuming CPU, and so on. It should do whatever it does very quickly, because the next activity will not be resumed until it returns. Followed either by <code>onResume()</code> if the activity returns back to the front, or by <code>onStop()</code> if it becomes invisible to the user.	Yes	<code>onResume()</code> or <code>onStop()</code>
<u>onStop()</u>	Called when the activity is no longer visible to the user. This may happen because it is being destroyed, or because another activity (either an existing one or a new one) has been resumed and is covering it. Followed either by <code>onRestart()</code> if the activity is coming back to interact with the user, or by <code>onDestroy()</code> if this activity is going away.	Yes	<code>onRestart()</code> or <code>onDestroy()</code>
<u>onDestroy()</u>	Called before the activity is destroyed. This is the final call that the activity will receive. It could be called either because the activity is finishing (someone called <code>finish()</code> on it), or because the system is temporarily destroying this instance of the activity to save space. You can distinguish between these two scenarios with the <u>isFinishing()</u> method.	Yes	<i>nothing</i>

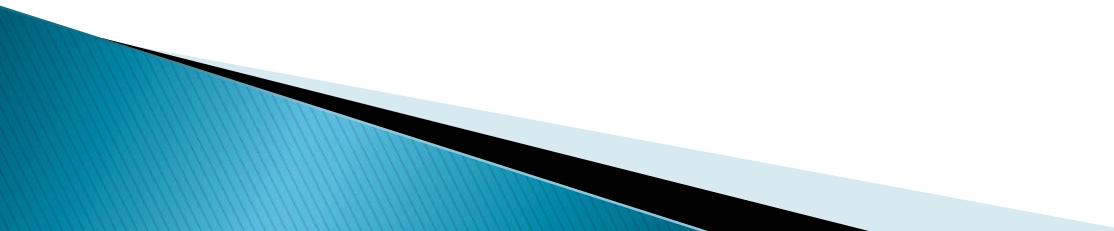
Service

- ▶ Runs in the background
- ▶ Examples
 - Playing music in the background
 - Fetching data from the network
 - Computing results
- ▶ Run in the main thread of the application
- ▶ For time consuming tasks should spawn a different thread

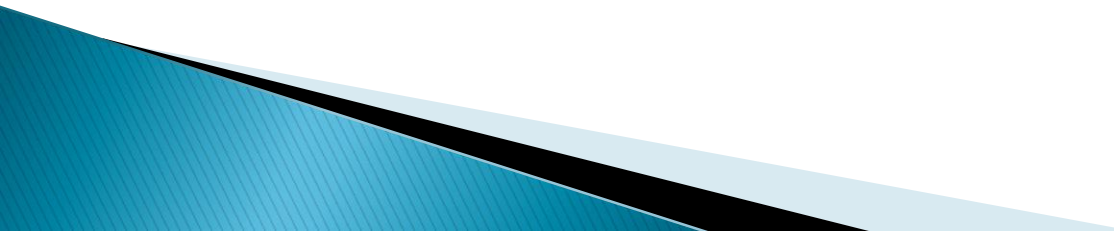
Service life cycle



Broadcast receivers

- ▶ Receive and react to broadcasts
 - ▶ Many originate in system code
 - Low battery
 - Picture has been taken
 - Change in timezone
 - Change in language
 - ▶ Do not have a user interface
 - ▶ Can start an activity that interacts with the user
- 

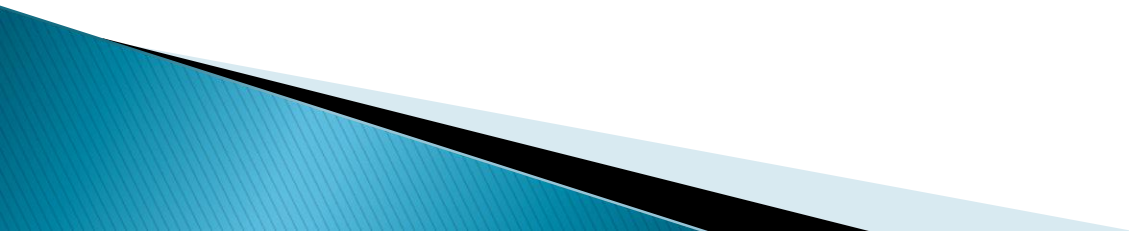
Content providers

- ▶ Makes a specific set of application data available to other application
 - ▶ Examples
 - Data stored in filesystem
 - Data stored in SQLite db
 - ▶ Implement a set of methods to retrieve and store data
 - ▶ A ContentResolver is used to call the methods in a content provider
- 

Data storage

- ▶ Data for application is private only to that application
- ▶ Content providers are used to share data
- ▶ Four means of storing data
 - Preferences
 - Files
 - Databases
 - Network

Demo



Questions

