



# Android 课程讲义

SmartPhone Application



# 3、Programming

SmartPhone Application



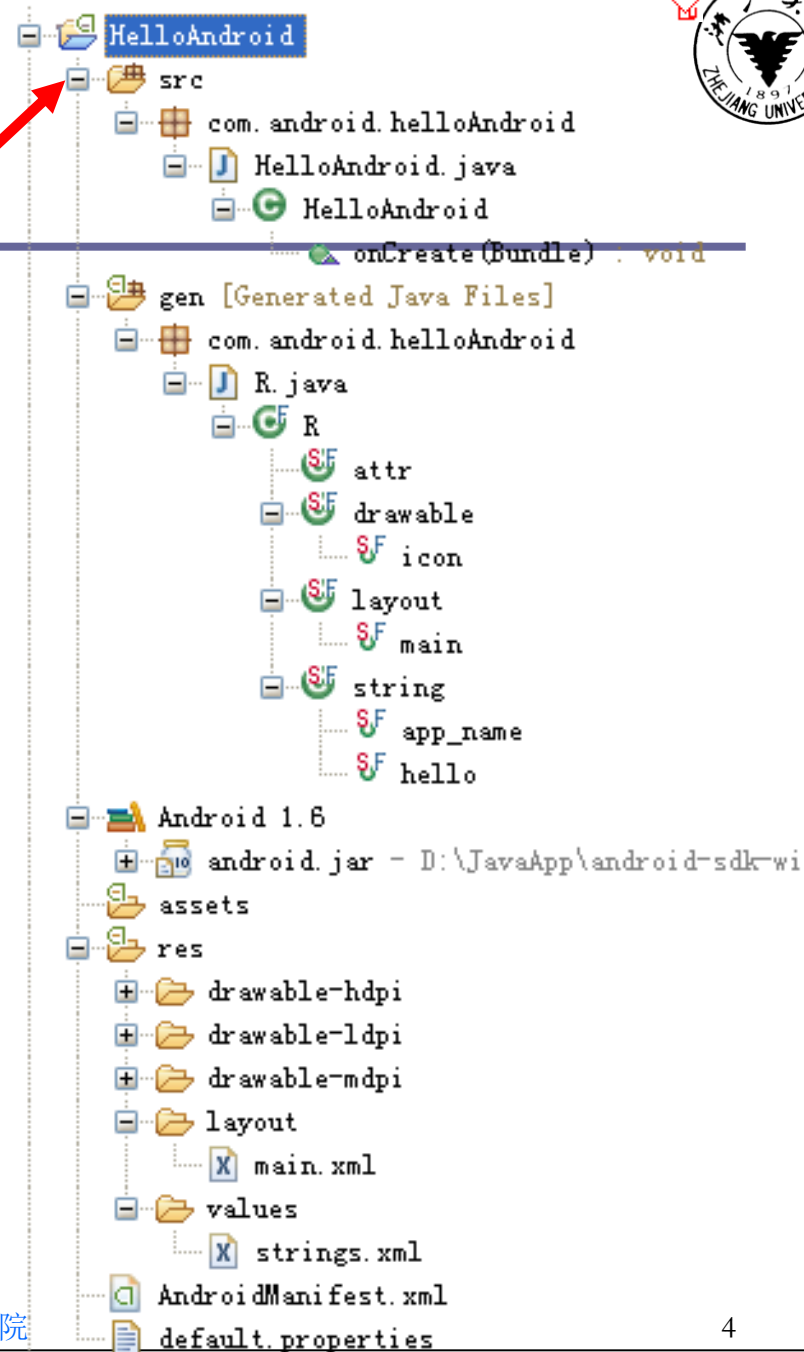
# Hello world!

## First Android application



# HelloAndroid

- Eclipse自动生成
  - new android project
    - HelloAndroid



# HelloAndroid.java

## □ Java程序

```
package com.android.helloAndroid;
import android.app.Activity;
import android.os.Bundle;
public class HelloAndroid extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

# R.java

## □ Java程序

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 */
package com.android.helloAndroid;
public final class R {
    public static final class attr {}
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

# main.xml

## □ Xml文件

### Resources

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  >
  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
  />
</LinearLayout>
```

# strings.xml

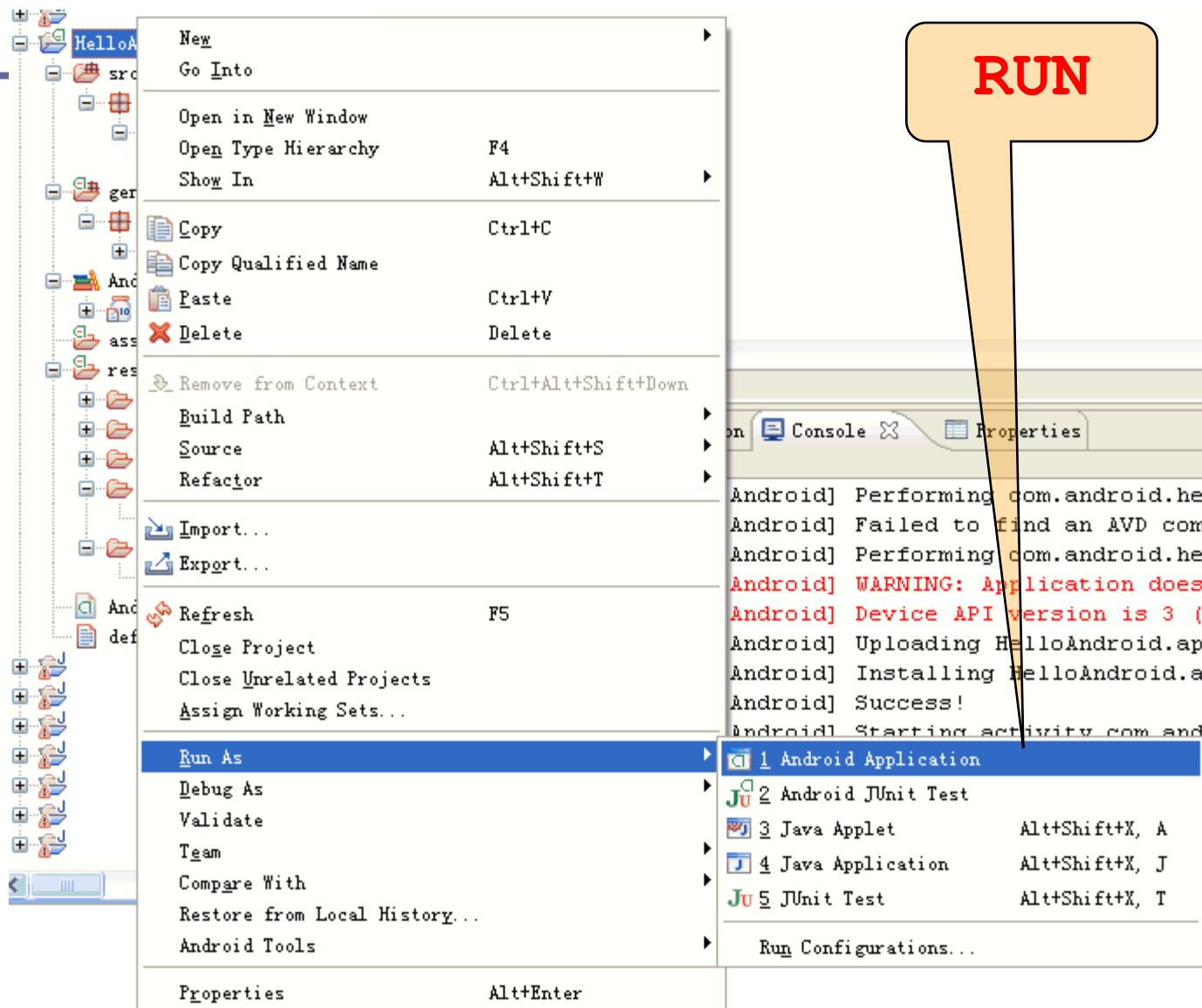
## □ Xml文件

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello Android, 2010.</string>
    <string name="app_name">HelloAndroid</string>
</resources>
```

display:  
Hello Android, 2010.



# RUN



The screenshot shows the 'Run' context menu for an Android project. The menu is open, displaying various options. A large orange arrow points from the word 'RUN' in a box to the 'Run As' option in the menu.

**Run Context Menu Options:**

- New
- Go Into
- Open in New Window
- Open Type Hierarchy F4
- Show In Alt+Shift+W
- Copy Ctrl+C
- Copy Qualified Name
- Paste Ctrl+V
- Delete Delete
- Remove from Context Ctrl+Alt+Shift+Down
- Build Path
- Source Alt+Shift+S
- Refactor Alt+Shift+T
- Import...
- Export...
- Refresh F5
- Close Project
- Close Unrelated Projects
- Assign Working Sets...
- Run As (highlighted)
- Debug As
- Validate
- Team
- Compare With
- Restore from Local History...
- Android Tools
- Properties Alt+Enter

**Run As Sub-menu Options:**

- 1 Android Application
- 2 Android JUnit Test
- 3 Java Applet Alt+Shift+X, A
- 4 Java Application Alt+Shift+X, J
- 5 JUnit Test Alt+Shift+X, T
- Run Configurations...

**Console Output:**

```

Android] Performing com.android.he
Android] Failed to find an AVD com
Android] Performing com.android.he
Android] WARNING: Application does
Android] Device API version is 3 (
Android] Uploading HelloAndroid.ap
Android] Installing HelloAndroid.a
Android] Success!
Android] Starting activity com.and
    
```

# 模拟运行



Hello Android, 2010.

# run

---

- Start emulator
  - emulator -avd android1.6 -skin HVGA-p
- Link to emulator :
  - telnet localhost 5554
- Make a Phone call:
  - gsm call 13810000086
- Send a message:
  - sms send 13810000086 HelloAndroid

# Image display

- add  
ImageView



main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        />
    <ImageView android:id="@+id/ImageView01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/adr"
        />
</LinearLayout>
```

Outline

- LinearLayout
  - TextView
  - ImageView01 (ImageView)

Properties

Property	Value
Scrollbar track vertical	
Scroll X	
Scroll Y	
Sound effects enabled	
Src	@drawable/adr
Style	
Tag	
Tint	
Visibility	
Misc	
Layout gravity	
Layout height	wrap_content
Layout margin	

the content of this ImageView.

Android SDK Content Loader



# Android file type

---

- Java file----source code
  - android 本身相当一部分都是用java 编写而成
  - android 的应用使用java 来开发。
- Class file----Java class file after compiled
  - Google uses DVM(Dalvik virtual machine).
  - Need change .class to .dex。
- Dex file----Executable file on DVM(Android platform)
  - Byte code supported by Android DVM.

# Apk file

- Apk file----Android上的安装文件
  - An .apk file extension denotes an Android Package (APK) file. This file format, a variant of the JAR format, is used for the distribution and installation of bundled components onto the Android mobile device platform.
  - An APK file is an archive that usually contains the following folders:
    - META-INF
    - res
  - and files:
    - AndroidManifest.xml
    - classes.dex
    - resources.arsc



# AndroidManifest.xml

---

- This view shows the actual XML file that results from all the choices made in the other views.

# 一个**AndroidManifest.xml**文件的例子

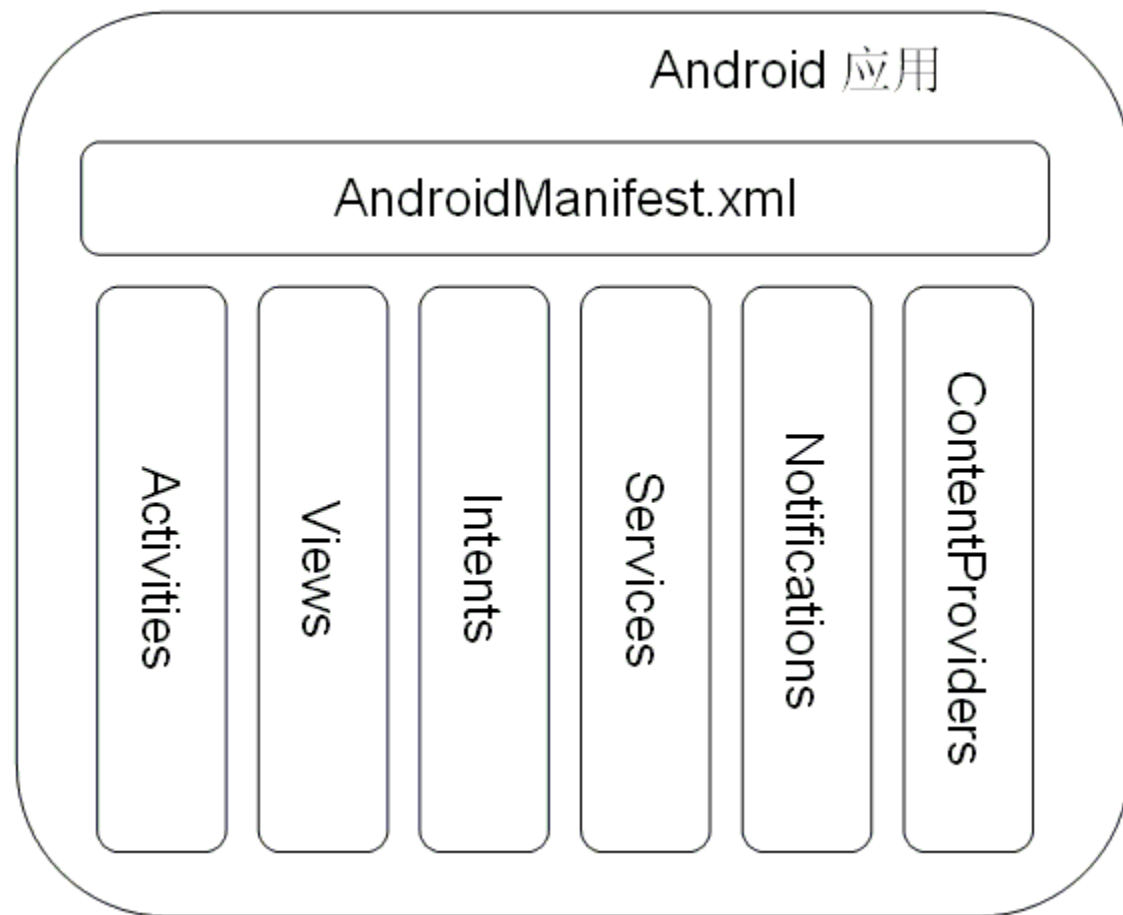
---

```
<?xml version="1.0" encoding="utf-8"?>
  <manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.my_domain.app.helloactivity">
    <application android:label="@string/app_name">
      <activity class=".HelloActivity">
        <intent-filter>
          <action
            android:value="android.intent.action.MAIN"/>
          <category
            android:value="android.intent.category.LAUNCHER"/>
        </intent-filter>
      </activity>
    </application>
  </manifest>
```





# Adroid应用



# Activities

---

- Activities are pieces of executable code that come and go in time, instantiated by either the user or the operating system and running as long as they are needed. They can interact with the user and request data or services from other activities or services via queries or Intents (discussed in a moment).
- Activities usually correspond to display screens: each Activity shows one screen to the user. When it is not actively running, an Activity can be killed by the operating system to conserve memory.



# Services

---

- These are analogous to services or daemons in desktop and server operating systems. They are executable pieces of code that usually run in the background from the time of their instantiation until the mobile handset is shut down. They generally don't expose a user interface.
- The classic example of a Service is an MP3 player that needs to keep playing queued files, even while the user has gone on to use other applications. Your application may need to implement Services to perform background tasks that persist without a user interface.



# Broadcast

---

- A Broadcast Receiver responds to a system-wide announcement of an event. These announcements can come from Android itself (e.g., battery low) or from any program running on the system.
- An Activity or Service provides other applications with access to its functionality by executing an Intent Receiver, a small piece of executable code that responds to requests for data or services from other activities.



# Intent Receivers

---

- The requesting (client) activity issues an Intent, leaving it up to the Android framework to figure out which application should receive and act on it.
- Intents are one of the key architectural elements in Android that facilitate the creation of new applications from existing applications. You will use Intents in your application to interact with other applications and services that provide information needed by your application.



# Content providers

---

- These are created to share data with other activities or services. A content provider uses a standard interface in the form of a URI to fulfill requests for data from other applications that may not even know which content provider they are using.
- The operating system looks to see which applications have registered themselves as content providers for the given URI, and sends the request to the appropriate application (starting the application if it is not already running). If there is more than one content provider registered for the requested URI, the operating system asks the user which one he wants to use.



- 
- An application doesn't have to use all of the Android components, but a well-written application will make use of the mechanisms provided, rather than reinventing functionality or hardcoding references to other applications.

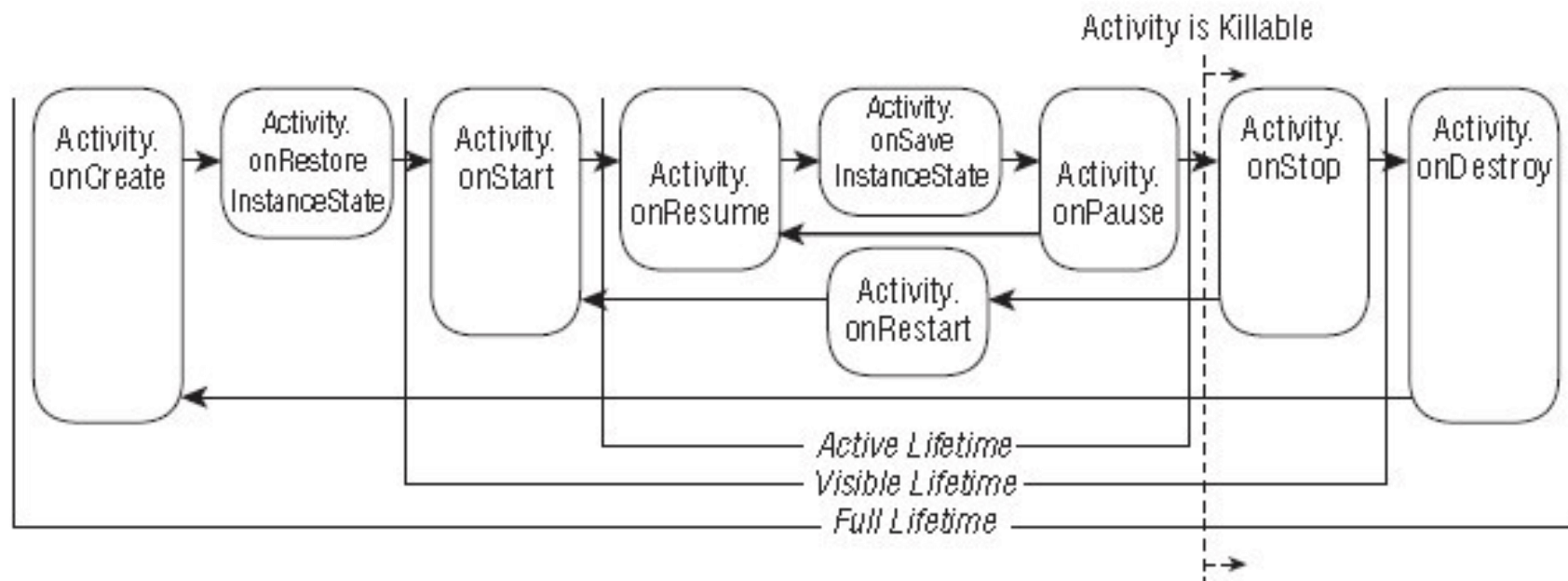
# Activity

---

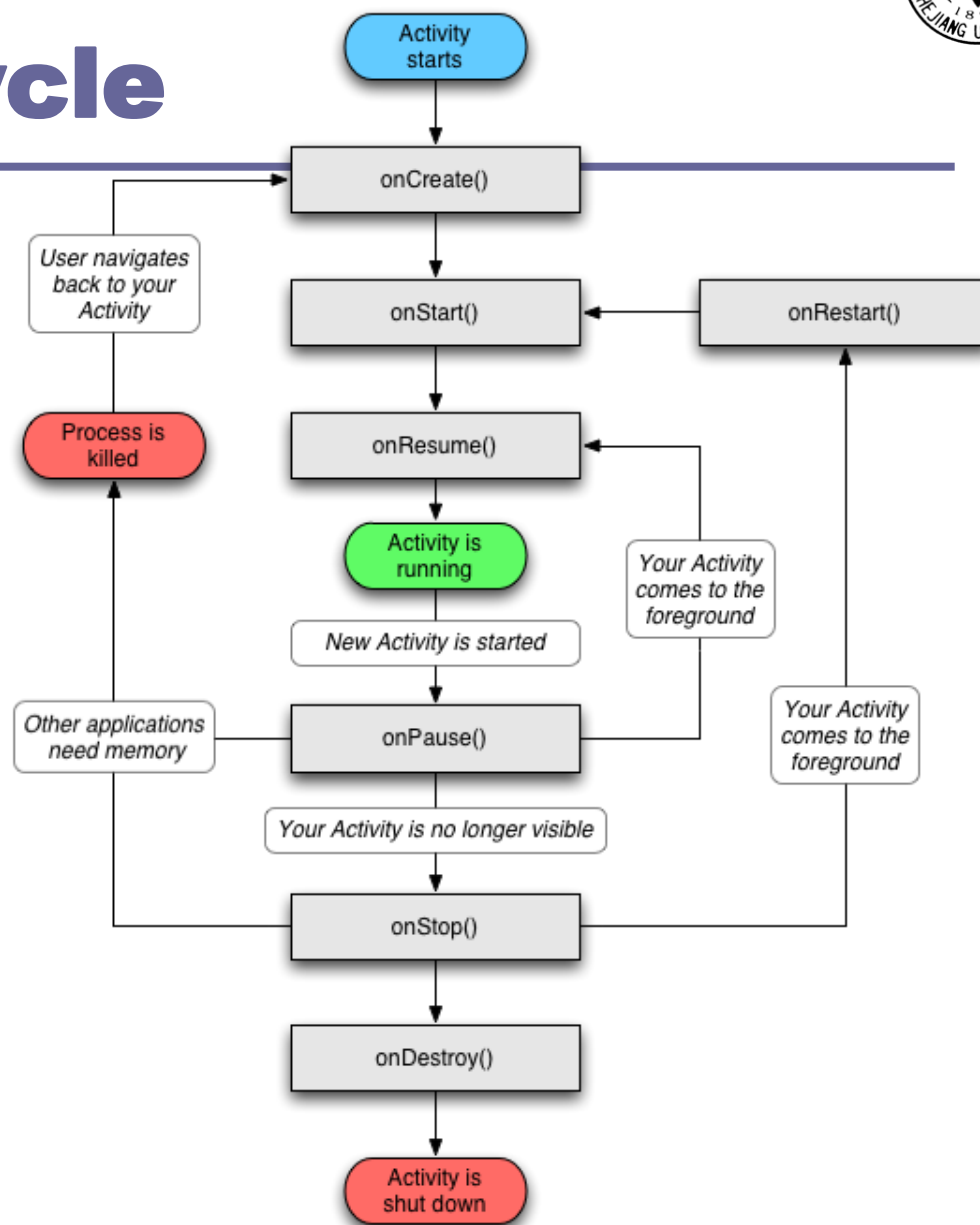
- ❑ Android is designed around the unique requirements of mobile applications. In particular, Android recognizes that resources (memory and battery, for example) are limited on most mobile devices, and provides mechanisms to conserve those resources.
- ❑ The mechanisms are evident in the Android Activity Lifecycle, which defines the states or events that an activity goes through from the time it is created until it finishes running.



# Activity状态管理



# Activity Lifecycle





- An activity monitors and reacts to these events by instantiating methods that override the Activity class methods for each event:
- onCreate
  - Called when an activity is first created. This is the place you normally create your views, open any persistent datafiles your activity needs to use, and in general initialize your activity.
  - When calling onCreate, the Android framework is passed a Bundle object that contains any activity state saved from when the activity ran before.
- onStart
  - Called just before an activity becomes visible on the screen. Once onStart completes, if your activity can become the foreground activity on the screen, control will transfer to onResume.
  - If the activity cannot become the foreground activity for some reason, control transfers to the onStop method.

## □ onResume

- Called right after onStart if your activity is the foreground activity on the screen. At this point your activity is running and interacting with the user. You are receiving keyboard and touch inputs, and the screen is displaying your user interface.
- onResume is also called if your activity loses the foreground to another activity, and that activity eventually exits, popping your activity back to the foreground. This is where your activity would start (or resume) doing things that are needed to update the user interface.

## □ onPause

- Called when Android is just about to resume a different activity, giving that activity the foreground. At this point your activity will no longer have access to the screen, so you should stop doing things that consume battery and CPU cycles unnecessarily.
- If you are running an animation, no one is going to be able to see it, so you might as well suspend it until you get the screen back. Your activity needs to take advantage of this method to store any state that you will need in case your activity gains the foreground again—and it is not guaranteed that your activity will resume.
- Once you exit this method, Android may kill your activity at any time without returning control to you.



## □ onStop

- Called when your activity is no longer visible, either because another activity has taken the foreground or because your activity is being destroyed.

## □ onDestroy

- The last chance for your activity to do any processing before it is destroyed. Normally you'd get to this point because the activity is done and the framework called its finish method. But as mentioned earlier, the method might be called because Android has decided it needs the resources your activity is consuming.



# Android Service Lifecycle

---

- The lifecycle for a service:
- onCreate and onStart differences
  - Services can be started when a client calls the `Context.startService(Intent)` method. If the service isn't already running, Android starts it and calls its `onCreate` method followed by the `onStart` method. If the service is already running, its `onStart` method is invoked again with the new intent. So it's quite possible and normal for a service's `onStart` method to be called repeatedly in a single run of the service.
- onResume, onPause, and onStop are not needed
  - Recall that a service generally has no user interface, so there isn't any need for the `onPause`, `onResume`, or `onStop` methods. Whenever a service is running, it is always in the background.

## □ onBind

- If a client needs a persistent connection to a service, it can call the Context.bindService method. This creates the service if it is not running, and calls onCreate but not onStart. Instead, the onBind method is called with the client's intent, and it returns an IBind object that the client can use to make further calls to the service.

## □ onDestroy

- the onDestroy method is called when the service is about to be terminated. Android will terminate a service when there are no more clients starting or bound to it.
- As with activities, Android may also terminate a service when memory is getting low. If that happens, Android will attempt to restart the service when the memory pressure passes, so if your service needs to store persistent information for that restart, it's best to do so in the onStart method.





# Permissions

- Android controls what applications are allowed to do by requiring that they ask for permission to perform critical actions. When the application is installed on a real device, the user can choose whether to allow the requested permissions and proceed with installation, or to reject installation (in the emulator environment, it is assumed all permission requests are granted).
  - `<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />` allows us to use fine-grained location providers, such as GPS.
  - `<uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />` allows us to access additional location commands.
  - `<uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION" />` allows the creation of mock location providers.
  - `<uses-permission android:name="android.permission.INTERNET" />` allows us to access the Internet.
  - `<uses-permission android:name="android.permission.CALL_PHONE" />` allows the application to place telephone calls.

# Connecting the phone

---

- >adb devices
  - List of devices attached
  - emulator-5554 device
  - HT840GZ12968 device

---

Thank you!

