Android Deep Dive

Appcelerator SDK Fundamentals

In this lesson, you will:

appcelerator

- Identify the strength and weaknesses of the Android platform
- Explore Android components and vocabulary
- Configure Android apps using native configuration techniques
- Implement Android UI APIs
- Implement Android non-visual APIs

PLATFORM CHARACTERISTICS

Android Platform Strengths

appcelerator

- Open nature (hackable)
- Variety of app distribution methods
- Range of devices: low-cost to high-end
- Java-based environment (common skill set)
- Strongly integrated into the Google ecosystem (identity, Google Apps, data sharing)

Android Platform Weaknesses

appcelerator

- Slow OS upgrades on user devices
- Carrier themes add a layer of complexity in UI design
- Large distribution of device screen types, hardware capabilities, etc.
- Open nature (hackable)
- Less active app economy (fewer 99-cent purchases)

Many devices & form factors

appcelerator

- Phones, tablets, various operating system versions, vendor skins, carrier add-ons, forked versions
- Need to test as widely as possible ... on device

ANDROID COMPONENTS

Android Application Key Components

appcelerator

TIP: *Read the official Android fundamentals docs.*
TEST

- Activities
- Services
- Broadcast Receivers
- Intents
  - Pending Intents

It is necessary to understand and implement

these in AppC to provide a truly native experience

Activity

'An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI'
TEST

For example, an email app might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails.

Service

'A Service is an application component representing either an application's desire to perform a longer-running operation while not interacting with the user or to supply functionality for other applications to use.'
TEST

For example, a service might play music in the background while the user is in a different app

Broadcast Receiver

'A broadcast receiver is a component that responds to system-wide broadcast announcements'
TEST

For example, a broadcast announcing that the screen has turned off, the battery is low, or a picture

was captured

Intent

'Three of the core components of an application - activities, services, and broadcast receivers - are activated through messages, called intents.'
TEST

http://developer.android.com/guide/topics/intents/intents-filters.html

Pending Intent

A Pending Intent is an intent you give to another application to perform on your app's behalf at a future time.
TEST

http://developer.android.com/reference/android/app/PendingIntent.html

All of these work in Appcelerator

appcelerator

appcelerator Titanium 3.0

Android

- Global
- Alloy
- Modules
- Titanium
  - Accelerometer
  - Analytics
  - **Android**
    - Calendar
    - NotificationManager
    - ActionBar
    - Activity
    - Intent
    - Menu
    - MenuItem
    - Notification
    - PendingIntent
    - R
    - RemoteViews
    - Service
  - API

Titanium.Android { Submodule of Titanium }

Properties 185 Methods 10 Fil

`Titanium.Proxy > Titanium.Module > Titanium.Android`

The top level Android module.

The currentActivity property provides a reference to the context's current Activity. Activity objects
a new activity can be started from an existing activity using Activity.startActivity or Activity.startAc

Almost all of the constants defined in this module correspond directly to Android constants, and a
documentation. In particular, the ACTION_, EXTRA_ and CATEGORY_ constants are defined in the

See also:

- Intents and Intent Filters in the Android Developer's Guide

ANDROID CONFIGURATION

AndroidManifest.xml

appcelerator

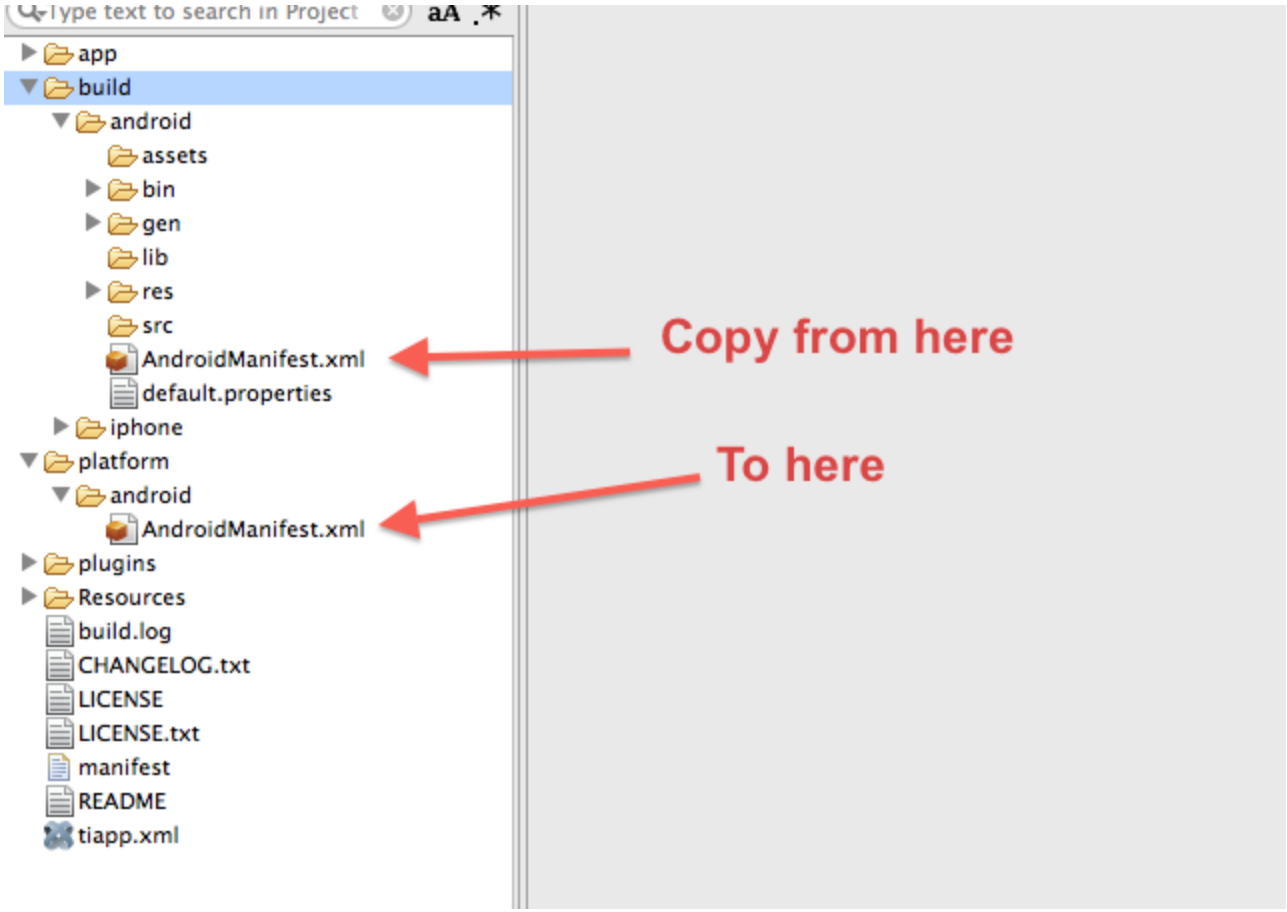
- Native configurations done in `AndroidManifest.xml`
- Most of those properties can be set in the `tiapp.xml`
- Or, use a custom `AndroidManifest.xml`
- Examples: app permissions, orientation handling, version number customization, native themes, tooling, etc.

Configuration in Tiapp.xml

```
48  <android xmlns:android="http://schemas.android.com/apk/res/android">
49      <manifest>
50          <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="17"/>
51          <application android:debuggable="false"
52              android:icon="@drawable/appicon"
53              android:label="URLSchemes" android:name="UrlschemesApplication">
54              <!-- all the activity tags and more go here -->
55          </application>
56      </manifest>
57  </android>
```

1. Build once to generate stock `build/android/AndroidManifest.xml`
2. Copy pertinent tags from generated file (e.g. all of the `<application>` block)
3. In `tiapp.xml`, edit `<android ...>` to be block rather than closed tag
4. Add `<manifest></manifest>` tags
5. Paste in generated code, edit as needed

Custom Android Manifest



Try It

- Create an android manifest and add to the project

ANDROID UI API

Windows and Activities

appcelerator

- Window == Activity
- Windows are default heavyweight (activities). Lightweight (full-screen views) windows
- Menus, title bars, and more are associated with the activity.
- Enter & exit animations on activities
- In a TabGroup, the activity is associated with the TabGroup not the windows

Window Themes

You can set the theme of the window. It can be either a built-in theme or a custom theme.

```javascript
var win = Ti.UI.createWindow({theme: "Theme.AppCompat.Fullscreen"});
win.open({
     activityEnterAnimation: Ti.Android.R.anim.slide_in_left,
     activityExitAnimation: Ti.Android.R.anim.slide_out_right
});
```

Action Bar / Hardware Menu

Action Bar / Hardware Menu

appcelerator

- Menu of options displayed when "hardware" button is pressed
- Associated with activity (heavyweight window)
- The action bar is displayed by default (3.3.x+).
- Pre-ICS: shown at bottom of screen as slide up tray
- ICS: either in Action Bar or in bottom bar as ellipsis button

Menu Example

appcelerator

View

```xml
<Alloy>
    <Window title="My Test App">
        <Menu id="menu" platform="and
roid" title="My XML Menu" onHomeIconI
temSelected="doMenuClick">
            <MenuItem id="item1" titl
e="Settings" onClick="openSettings" /
>
            <MenuItem id="item2" titl
e="Search" onClick="doSearch" />
        </Menu>
        <Label id="label">Welcome!</L
abel>
    </Window>
</Alloy>
```

Stylesheet

```javascript
"MenuItem": {
    showAsAction: Ti.Android.SHOW_AS_A
CTION_ALWAYS
},
"#item1": {
    icon: Ti.Android.R.drawable.ic_men
u_preferences
},
"#item2": {
    icon: Ti.Android.R.drawable.ic_men
u_search
},
"#menu": {
    icon: "/actionicon.png",
    displayHomeAsUp: true,
    backgroundImage: "/actionbackgroun
d.png"

}
```

Menu, Continued

tiapp.xml

```xml
<android xmlns:android="http://schemas.android.com/apk/res/android">
     <manifest>
       <!-- target SDK 14+ (Android 4.0) to get ActionBar style menus
            default if you omit uses-sdk is to target highest installed -->
       <uses-sdk android:minSdkVersion="10" android:targetSdkVersion="14"/>
       <!-- other manifest entries including <application> tag-->
     </manifest>
</android>
```

NOTE: *To get the ActionBar style menu, you just need to build with a current SDK, which is the default.*
TEST

Try It

- Add a menu to previous project
- Use Genymotion if possible

Menus in TabGroups

- TabGroups are the activity (not the windows)
- So menu code goes inside the tab group
- One menu for whole app
- Call Ti.Android.invalidateOptionsMenu() within Window to re-create a menu (i.e. have separate menus for each window)

```xml
<Alloy>
    <TabGroup id='tabGroup'>
        <Require src="Fugitives" id="fugitivesTab"/>
        <Require src="Captured" id="capturedTab"/>
        <Menu>
            <MenuItem title="Add" onClick="addNewFugitive"/>
        </Menu>
    </TabGroup>
</Alloy>
```

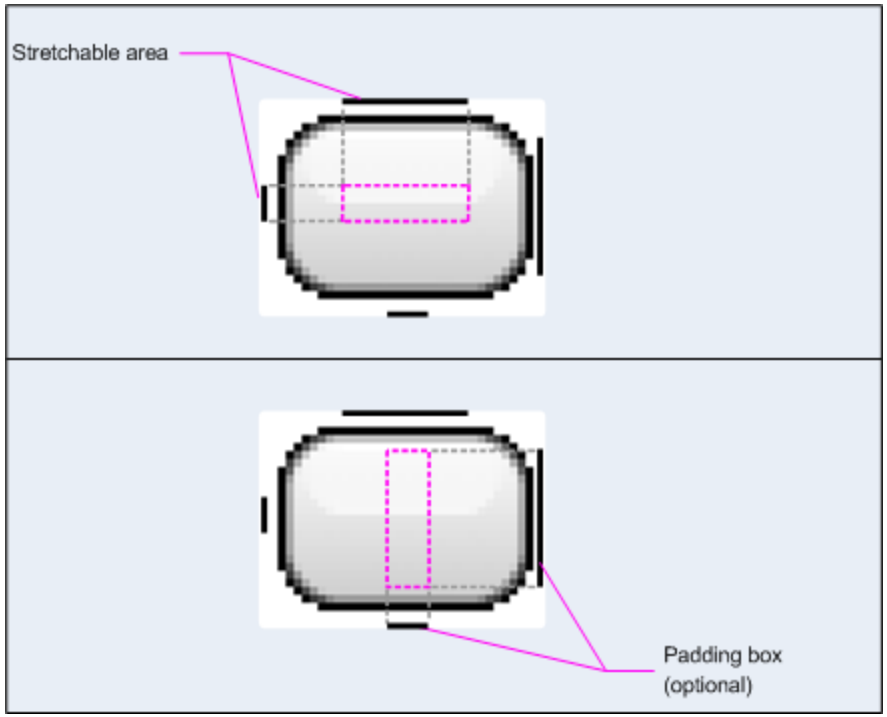
Installing to the SD card

- Must build with SDK tools 8 or higher (default is to use highest installed)
- Set minSdkVersion to 10 or higher
- Options are preferExternal, preferInternal, or auto (default)

```xml
<android xmlns:android="http://schemas.android.com/apk/res/android">
    <manifest android:installLocation="preferExternal">
        <uses-sdk android:minSdkVersion="10" />
    </manifest>
</android>
```

NinePatch images

- Avoid stretched splash screens
- Define stretchable areas of your graphic
- Enable through Android theme modification



Using a NinePatch for the Splash Screen

- Create a NinePatch image using draw9patch utility
- Copy to `platform/android/res/drawable[-xdpi]`
- Rename it to `splash.9.png`
- Create a `theme.xml` file, put it in `platform/android/res/values`

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <style name="Theme.Titanium" parent="
android:Theme">
    <item name="android:windowBackgroun
d"></item>
    <item name="android:windowNoTitle">
true</item>
  </style>
</resources>
```

Try It

- Open draw9patch tool
- Select an image and draw section that will scale

Android Labels

- Use Label.html to apply formatting like you would use AttributedStrings on iOS
- Use Label.autoLink to "linkify" URLs, phone numbers, address, etc.
- autoLink also works on TextAreas

Toast "Pop-up" Notifications

appcelerator

- Simple text display over all activities
- Can control positioning on screen
- Rendering will be different based on OS version and skin

Status Bar Notifications

- Use the NotificationManager module to send and cancel notifications.
- Invoke using Intents
-

Example:http://docs.appcelerator.com/titanium/latest/#!/api/Titanium.Android.NotificationManager

Hijacking the Back Button

appcelerator

- Override default Back button behavior
- Example: wizard interface where you want to go back a screen
- Careful about user expectations
- Demo: AndroidBackDemo

NON-VISUAL APIS

Launching Activities

- You can launch other apps (activities) from JS
- Need to have an intent object to pass
- Many built in intents to use

Forging Titanium #9 - Android Intents Cookbook
http://bit.ly/ryOSW4

Example

```javascript
// create an Android intent whose action is to send plain text data
var intent = Ti.Android.createIntent({
     action: Ti.Android.ACTION_SEND,
     type: 'text/plain'
});
// define two extra fields for the intent
intent.putExtra(Ti.Android.EXTRA_SUBJECT, 'Isn\'t This Cool!');
intent.putExtra(Ti.Android.EXTRA_TEXT, $.message.value);

try {
     Ti.Android.currentActivity.startActivity(intent);
} catch (ex) {
     /* Handle Exception if no suitable apps installed */
     Ti.UI.createNotification({ message : 'No sharing apps installed!' }).show();
}
```

Service Support

Run JavaScript-based services in the background

Must be started by your app, but can survive when you exit the app

Runs on an interval you specify

Can access many non-UI Appcelerator APIs (networking, eventing, etc.)

Example

appcelerator

app/lib/logservice.js

```javascript
// This is the service, use non-UI
Ti APIs Ti.API.info("Hello world, I'm
 a service");
```

```xml
<!-- tiapp.xml -->
<android xmlns:android="http://schemas
.android.com/apk/res/android">
     <services>
    <service url="logservice.js"
         type="interval"/>
     </services>
</android>
```

```javascript
// in index.js or other controller
var SECONDS = 10; // every 10 seconds
var intent = Titanium.Android.createServiceIntent({
     url: 'logservice.js'
});
intent.putExtra('interval', SECONDS * 1000); // Needs to be milliseconds
Ti.Android.stopService(intent); // later, to stop the service!
```

Application Resources

- JS access to R.java - http://developer.android.com/reference/android/R.html
- R.drawable - built in icons for ImageView, etc.
- R.string - OS localized string for 'OK', 'Cancel', etc
- Android docs required to see properties

Summary

In this lesson, you:

- Identified the strength and weaknesses of the Android platform
- Explored Android components and vocabulary
- Configured Android apps using native configuration techniques
- Implemented Android UI APIs
- Implemented Android non-visual APIs

QUESTIONS?