

Android Application Development



Session: 15

**Android Studio 2.1 and Android
6.0 Marshmallow**

Objectives

- ◆ Identify the updated features of Android Studio 2.1
- ◆ Explain the updated and new features of Android 6.0 Marshmallow

Introduction to Android Studio 2.1

Phones Tablets

Good quality, high performance apps for the Android platform

Android TV

Android Auto

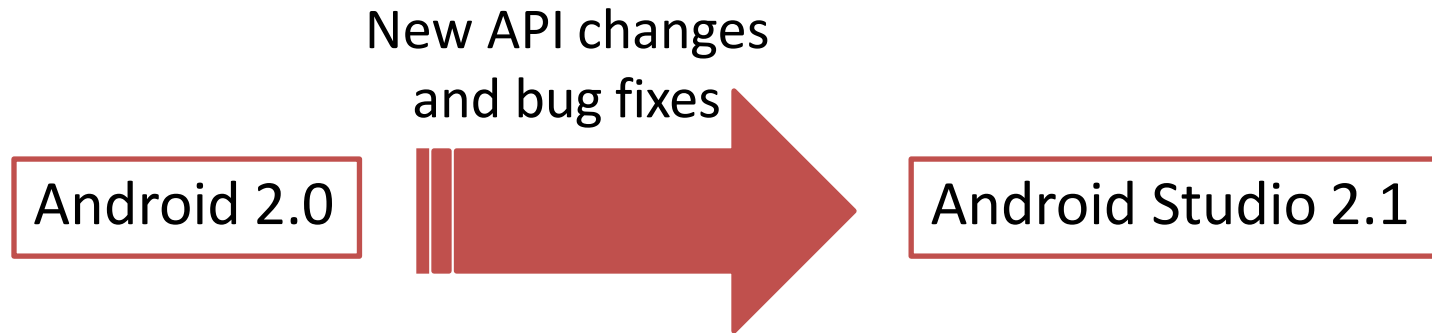
Accelerated emulator Fast build speeds

Latest Android version

Google Play Services

Improved interface and enhanced features

Updated Features in Android Studio 2.1 1-6



Following table describes some updated features:

Feature	Description
API Level	<ul style="list-style-type: none">• Level 7
Bluetooth	<ul style="list-style-type: none">• Bluetooth 2.1• New BT profiles: Object Push Profile (OPP) and Phone Book Access Profile (PBAP)

Updated Features in Android Studio 2.1 2-6

Feature	Description
Built-in Applications	<ul style="list-style-type: none">• Alarm clock• Browser• Calculator• Camera• Contacts• Custom Locale (developer app)• Dev tools (developer app)• Email Gallery• IMEs for Japanese, Chinese, Latin text input• Messaging• Music• Phone• Settings• Spare parts (developer app)

Updated Features in Android Studio 2.1 3-6

Feature	Description
Framework APIs	<ul style="list-style-type: none">• Android Studio 2.1 includes several new developer APIs.
Live Wallpapers	<p>APIs for developing animated wallpapers are provided in:</p> <ul style="list-style-type: none">• New <code>android.service.wallpaper.package</code>• New <code>WallpaperInfo</code> class• Updated <code>WallpaperManager</code>
Media Framework	<ul style="list-style-type: none">• Android Studio 2.1 provides revamped graphics architecture for improved performance.
Telephony	<ul style="list-style-type: none">• The new <code>SignalStrength</code> class provides information about the device's current network signal.

Updated Features in Android Studio 2.1 4-6

Feature	Description
Views	<p>Android Studio 2.1 provides New:</p> <ul style="list-style-type: none">• View methods: <code>isOpaque()</code> and <code>onDrawScrollBars()</code>• RemoteViews methods: <code>addView()</code> and <code>removeAllViews()</code>• ViewGroup methods: <code>isChildrenDrawingOrderEnabled()</code> and <code>setChildrenDrawingOrderEnabled()</code>

Updated Features in Android Studio 2.1 5-6

Feature	Description
Webkit	<p>Android Studio 2.1 provides New:</p> <ul style="list-style-type: none">• <code>WebStorage</code> methods to manipulate Web storage databases.• <code>GeolocationPermissions</code> methods to get and set <code>Geolocation</code> permissions.• <code>WebSettings</code> methods to manage settings for app cache, Web storage, and zooming.• <code>WebChromeClient</code> methods for handling video, browsing history, custom Views and app cache limits.

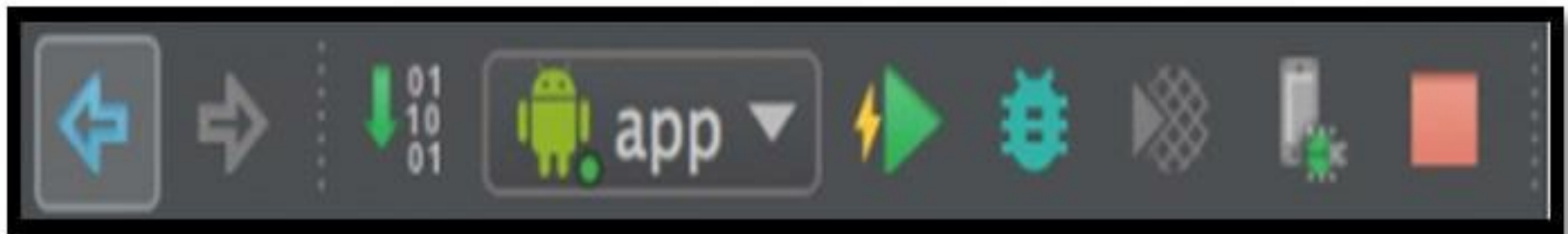
Updated Features in Android Studio 2.1 6-6

Additional Features

- Instant Run
- Android Emulator
- Cloud Test Lab Integration
- App Indexing Code Generation and Test
- GPU Debugger Preview
- IntelliJ 15 Update

Instant Run

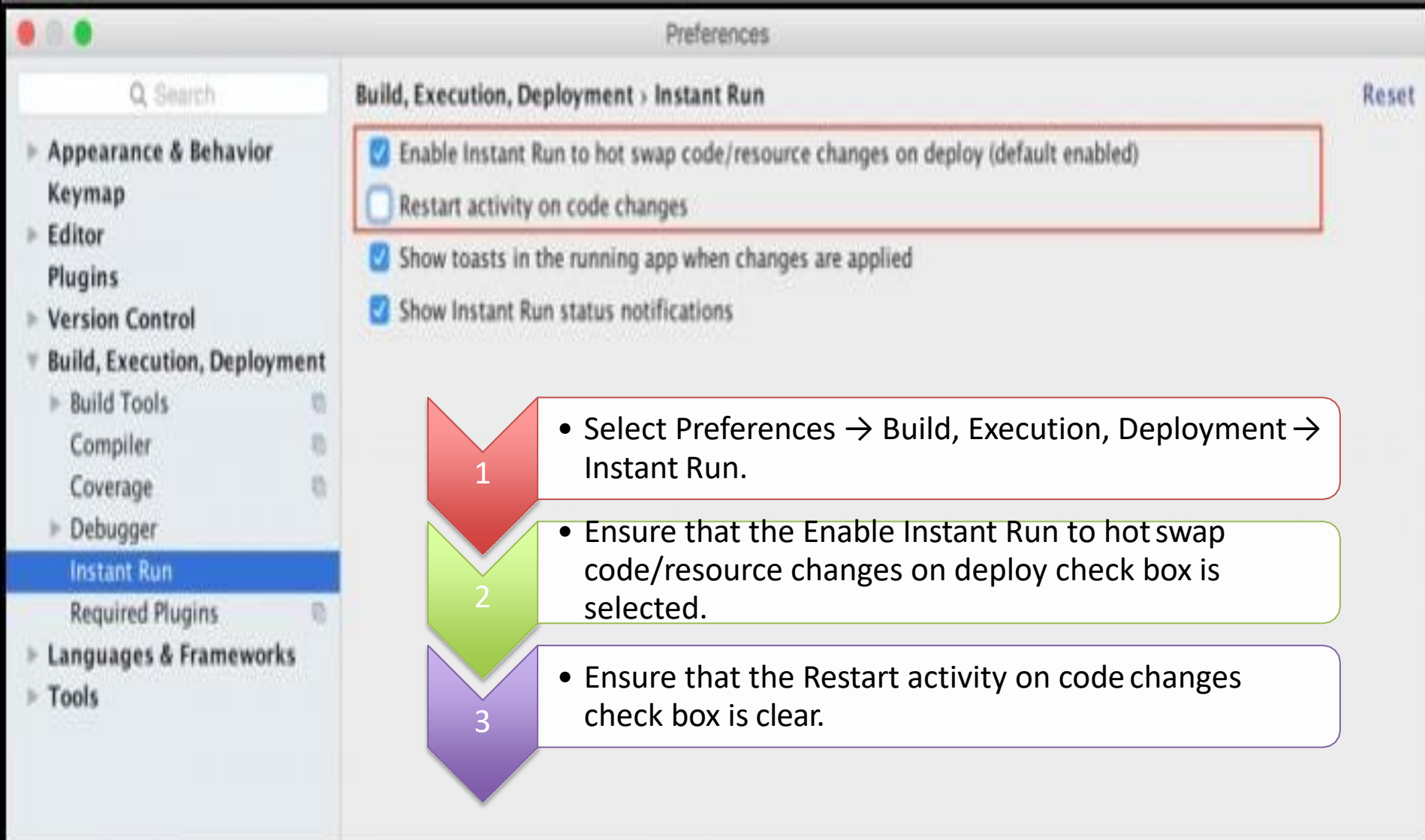
- ☐ Avoids the need to build and deploy an app every time changes are made.
- ☐ Applies updates to a running instance of an app on a device or emulator.



Instant Run

View/Enable Instant Run

To view this setting or to enable the feature manually:

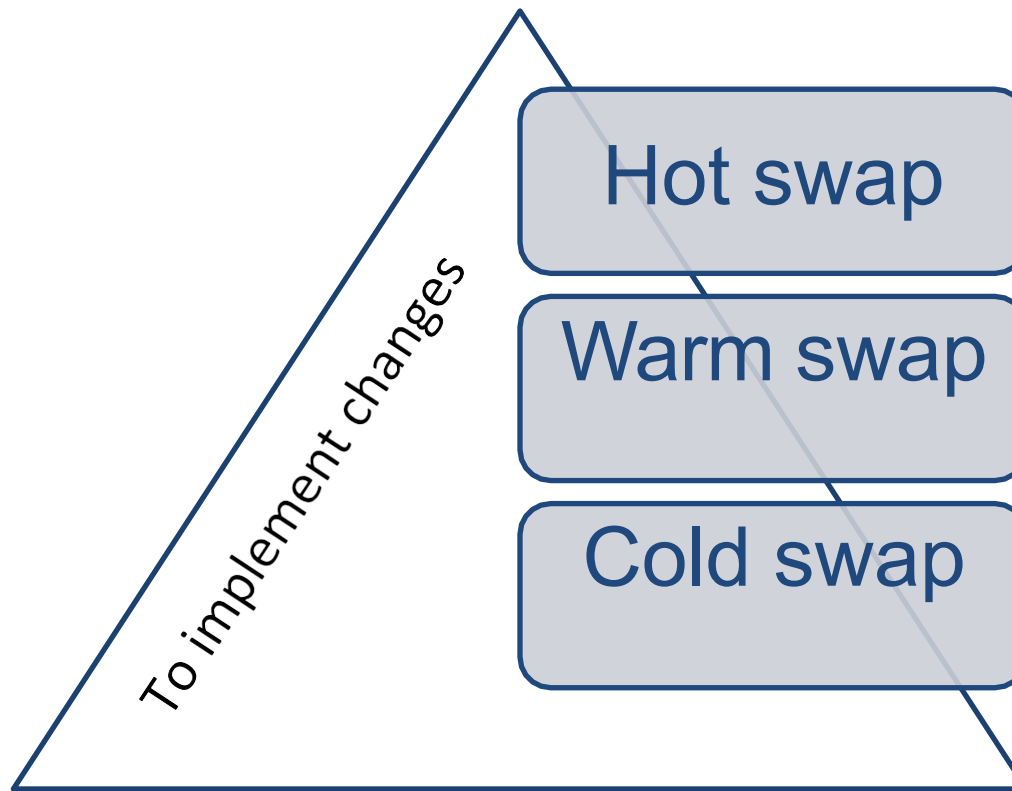


The screenshot shows the 'Preferences' dialog in Android Studio. The left sidebar lists various categories, with 'Build, Execution, Deployment' expanded and 'Instant Run' selected. The right pane shows the 'Instant Run' settings under 'Build, Execution, Deployment > Instant Run'. A red box highlights the 'Enable Instant Run to hot swap code/resource changes on deploy (default enabled)' checkbox, which is checked, and the 'Restart activity on code changes' checkbox, which is unchecked. Below the screenshot, three numbered steps are provided:

1. Select Preferences → Build, Execution, Deployment → Instant Run.
2. Ensure that the Enable Instant Run to hot swap code/resource changes on deploy check box is selected.
3. Ensure that the Restart activity on code changes check box is clear.

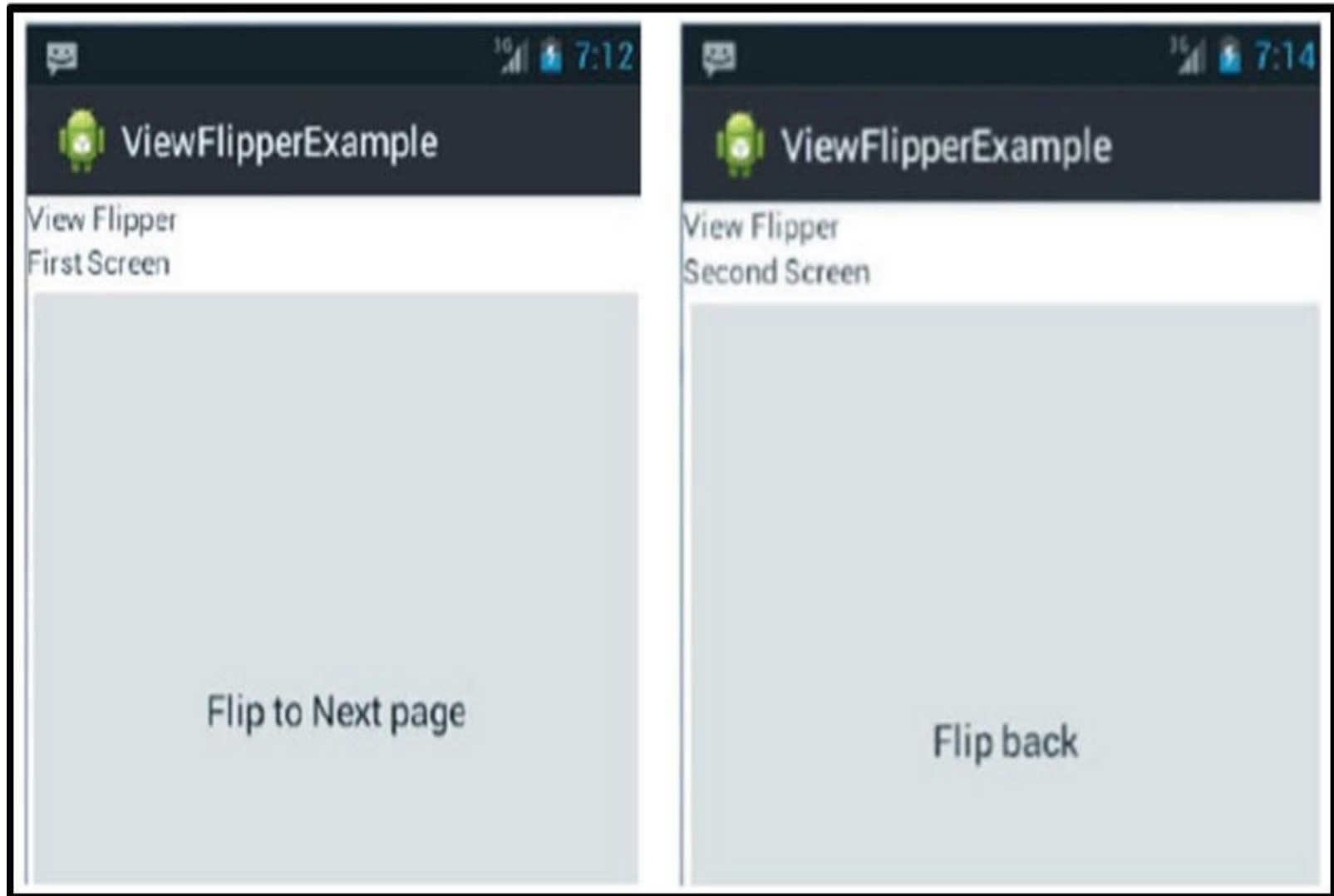
Implementing Updates

Instant Run implements updates in three ways:



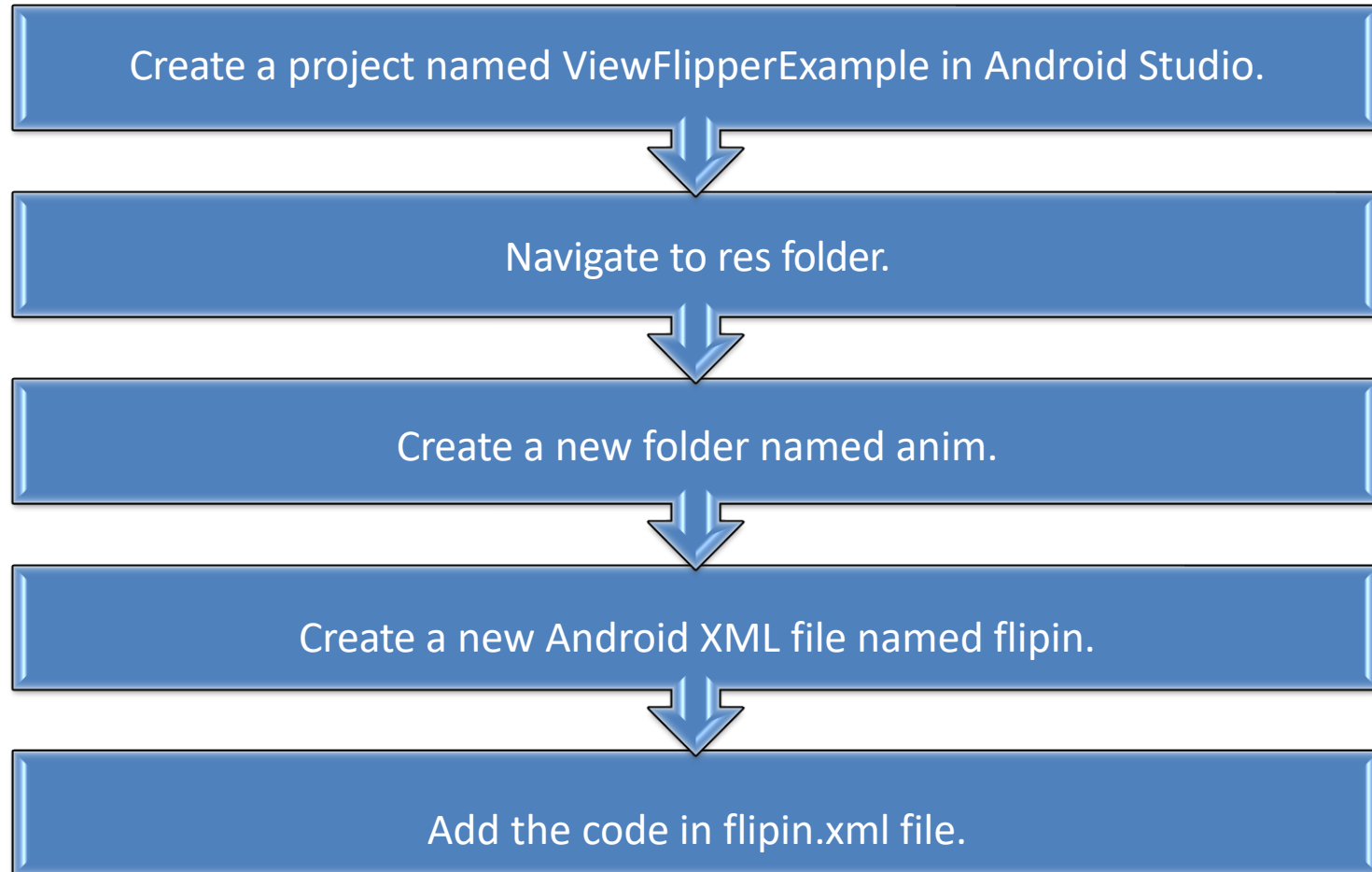
Testing Instant Run

A sample application



Creating Sample Application

To create this application, perform the following steps:



Adding Code

Following code snippet shows how to add the code in flipin.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/decelerate_interpolator" >
    <translate
        android:duration="500"
        android:fromXDelta="-100%"
        android:toXDelta="0%" />
    </set>
```

Modifying Code

To create this application, perform the following steps:

Create another new Android XML file named flipout under res → xml.



Modify the code in flipout.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/decelerate_interpolator" >
<translate
    android:duration="500"
    android:fromXDelta="0%"
    android:toXDelta="100%" />
</set>
```


Modifying code in activity_main.xml File 1-2

Navigate to res → layout folder.



Modify the code in activity_main.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android" android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical">
<TextView android:layout_width="fill_parent"
android:layout_height="wrap_content" android:text="View Flipper"
/>
<ViewFlipper android:id="@+id/viewflipper"
android:layout_width="fill_parent"
android:layout_height="fill_parent" >
<LinearLayout android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical" >
<TextView android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="First Screen" />
```

Modifying code in activity_main.xml File 2-2

```
= "Flip to Next page" />
</LinearLayout>
<LinearLayout android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Second Screen" /><Button
android:id="@+id/button2"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="Flipback" />
</LinearLayout>
</ViewFlipper>
</LinearLayout>
```

Adding Code in MainActivity.java File 1-3

Add the code into the MainActivity.java file.

```
package com.example.viewflipperexample;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.ViewFlipper;
public class MainActivity extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final ViewFlipper FlipIt = (ViewFlipper)
        findViewById(R.id.viewflipper);
```

Adding Code in MainActivity.java File 2-3

```
Button flip1 = (Button) findViewById(R.id.button1);
Button flip2 = (Button) findViewById(R.id.button2);
Animation FlipIn = AnimationUtils.loadAnimation (this, R.anim.
flipin);
Animation FlipOut = AnimationUtils.loadAnimation(this, R.anim.
flipout);
FlipIt.setInAnimation(FlipIn);
FlipIt.setOutAnimation(FlipOut);
flip1.setOnClickListener(new Button. OnClickListener()
{
@Override
public void onClick(View arg0)
{
// TODO Auto-generated method stub
FlipIt.showNext();
}
});
```

Adding Code in MainActivity.java File 3-3

```
flip2.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        FlipIt.showPrevious();
    }
});
@Override
public boolean onCreateOptionsMenu(Menu menu)
{
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}
}
```

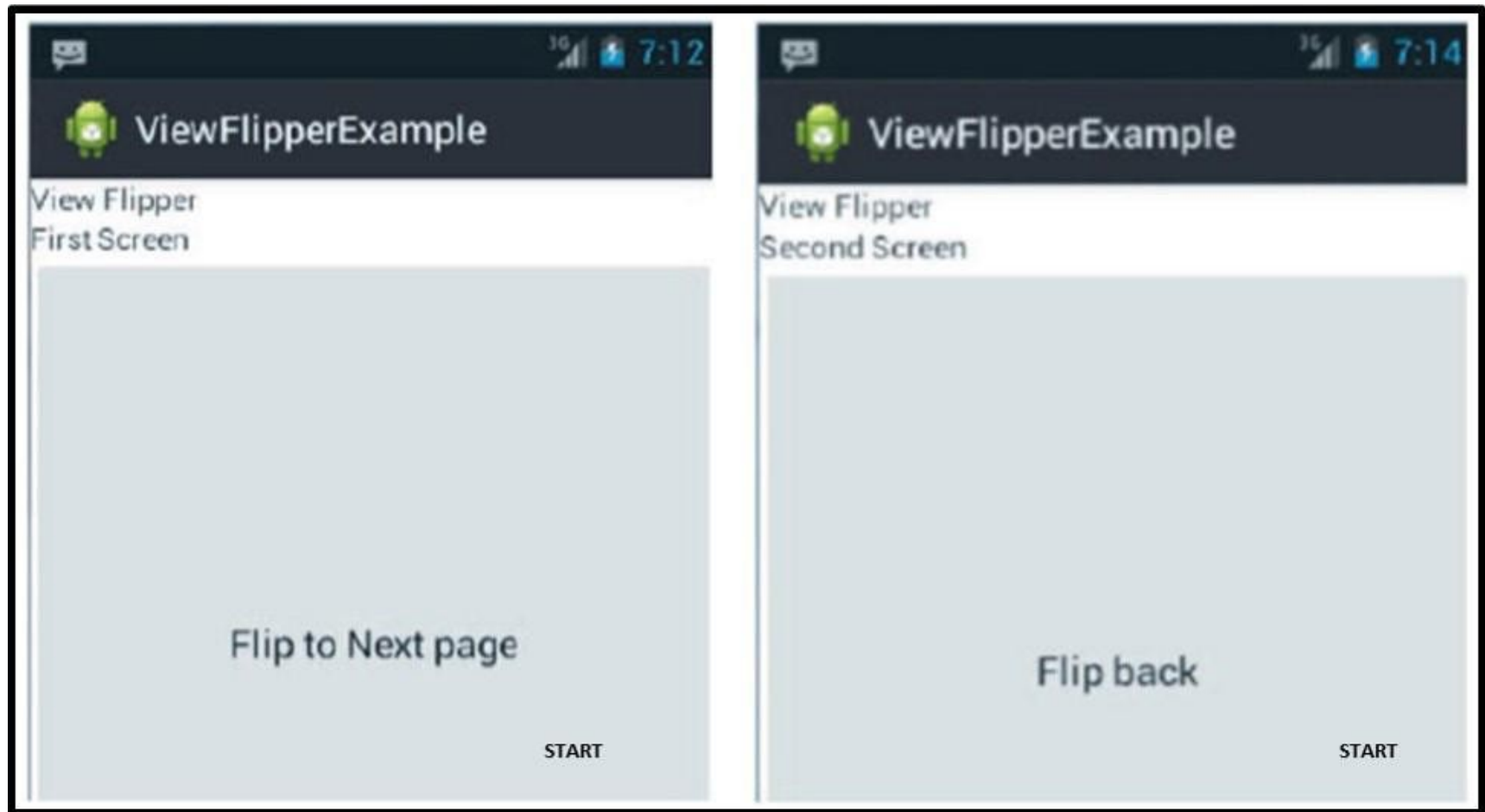
Replacing Code Using Hot Swap

Replacing code for onClick () method

```
public void onClick(View v)
{
    //add code to display new message is displayed on selecting
    // the action button
    . . .
}
```

Updated App

The flip back happens when user clicks **START**.



Changing App Background

To change the gray app background to pale yellow using warm swap method:

```
import android.view.View.OnClickListener;
import android.widget.Button;
import android.graphics.Color;
@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_new_contact);
    Button button = (Button)findViewById(R.id.button);
    button.setOnClickListener(addButtonListener);
    RelativeLayout colorLayout = (RelativeLayout) findViewById(R.
id.colorLayout);
    colorLayout.setBackgroundColor(Color.YELLOW);
}
```


Replacing Code Using Cold Swap

Following code shows the onCreate () method code replaced for cold swap update.

```
/ A complete restart is triggered by a cold swap.  
//Just add a new method after onCreate()  
//for a complete restart of currently running application  
public void demoMethod()  
{  
}  
/  

```

New Interface Features

- ❑ Resizable emulator
- ❑ Possibility of multi-touch actions
- ❑ Toolbar to provide easy-access controls
- ❑ Quick installation of APKs
- ❑ New features in GUI version

Knowing Battery Status

Following code snippet displays the battery status in percentage on screen:

```
package com.example.test;
import java.io.IOException;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.res.AssetFileDescriptor;
import android.widget.ProgressBar;
setContentView(R.layout.activity_main);
registerReceiver(mbc, new IntentFilter(Intent.ACTION_BATTERY_
CHANGED));
}
}
```

Adding Code to XML File 1-2

Following code is added to the XML file to display the battery status in percentage on screen:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/  
android"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:orientation="vertical"  
    android:background="#abc" >  
<TextView  
    android:id="@+id/textView1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text=""  
    android:textSize="25sp"  
    android:layout_gravity="center"  
</>
```

Adding Code to XML File 2-2

```
<ProgressBar
  android:id="@+id/progressBar1"
  style="?android:attr/progressBarStyleHorizontal"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_gravity="center"
  android:minHeight="100dp"
  android:minWidth="200dp" />
</LinearLayout>
```

Making Changes in Battery Status

To make changes in the battery status:

The Battery Tab

Open the Extended controls window and choose the Battery tab.

The Battery Level

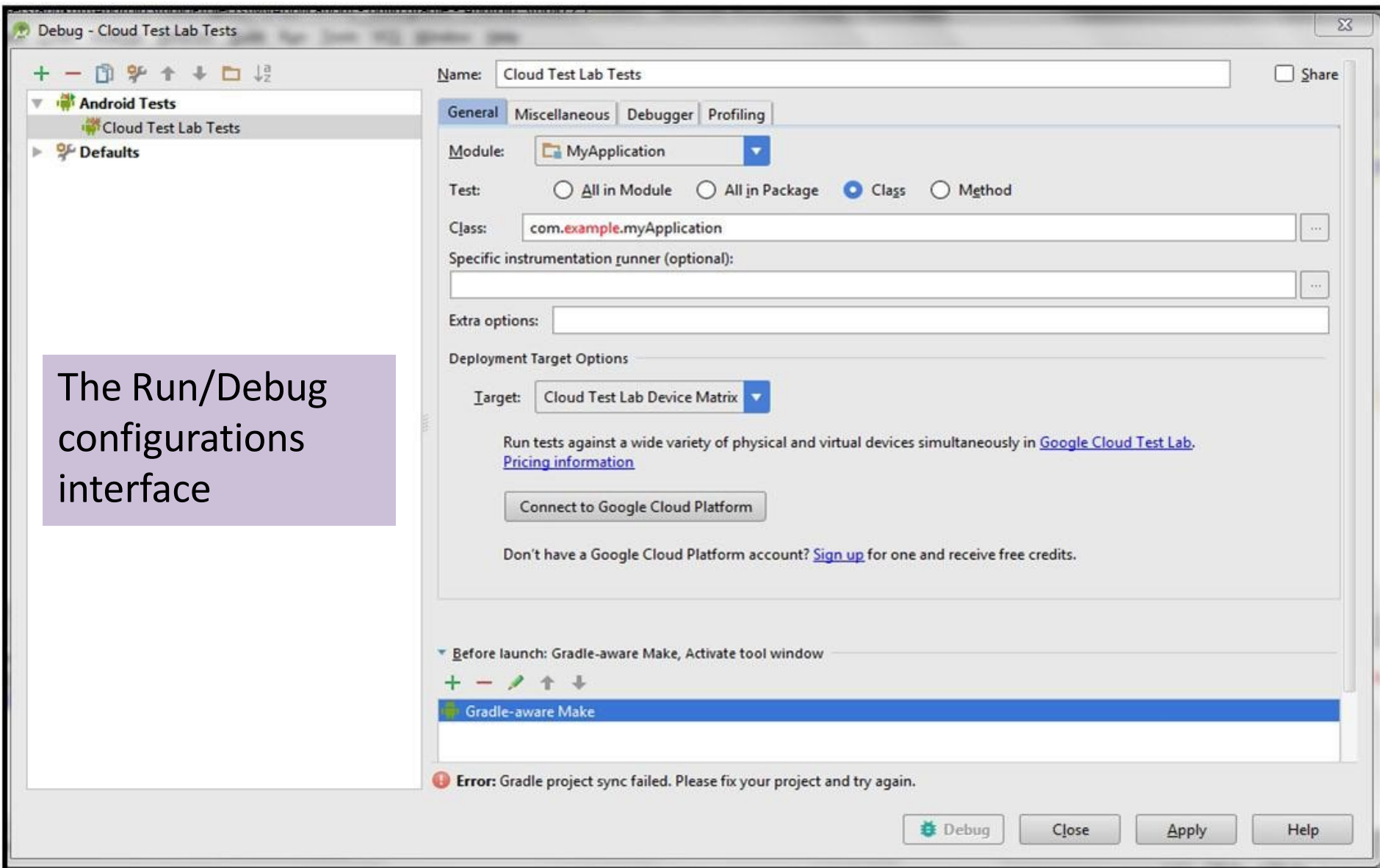
Select the Charge level slider to change the battery level.

The Battery Status

Choose Full from the Battery status drop-down list for setting the battery status.

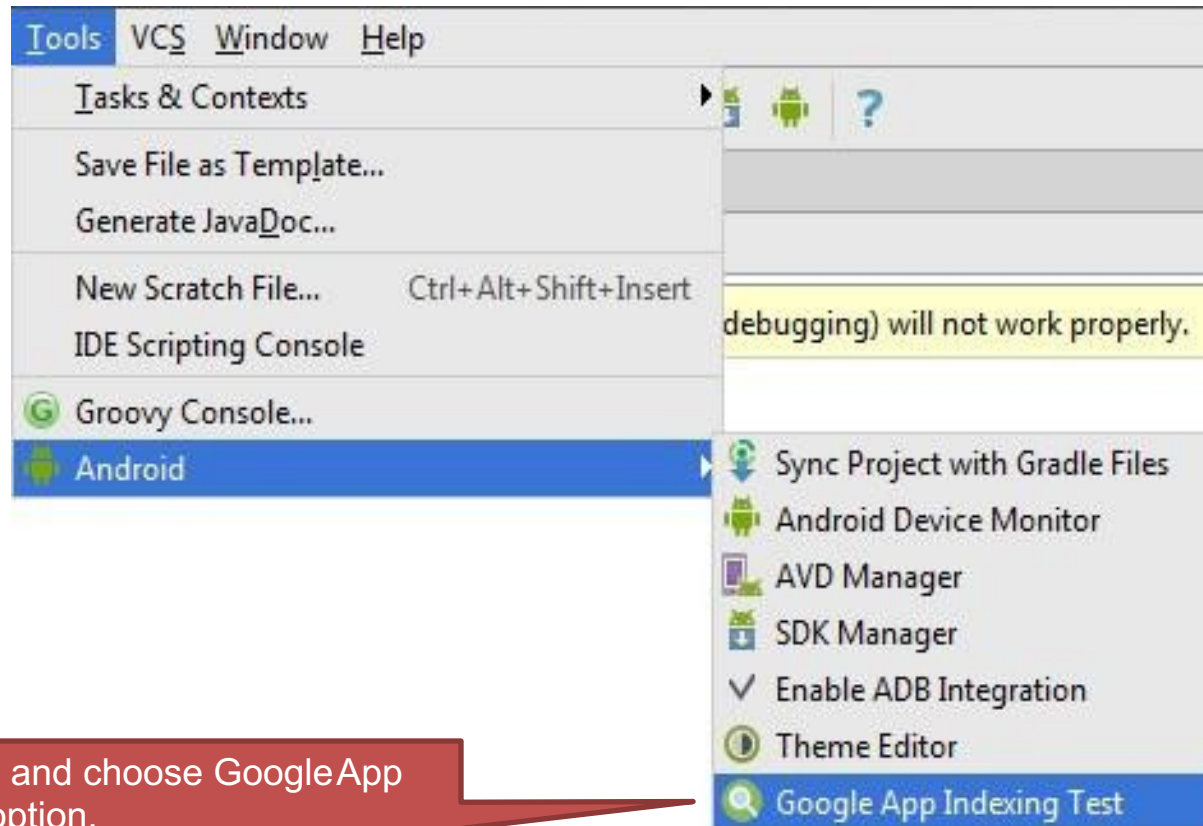
Cloud Test Lab Integration

The Run/Debug configurations interface

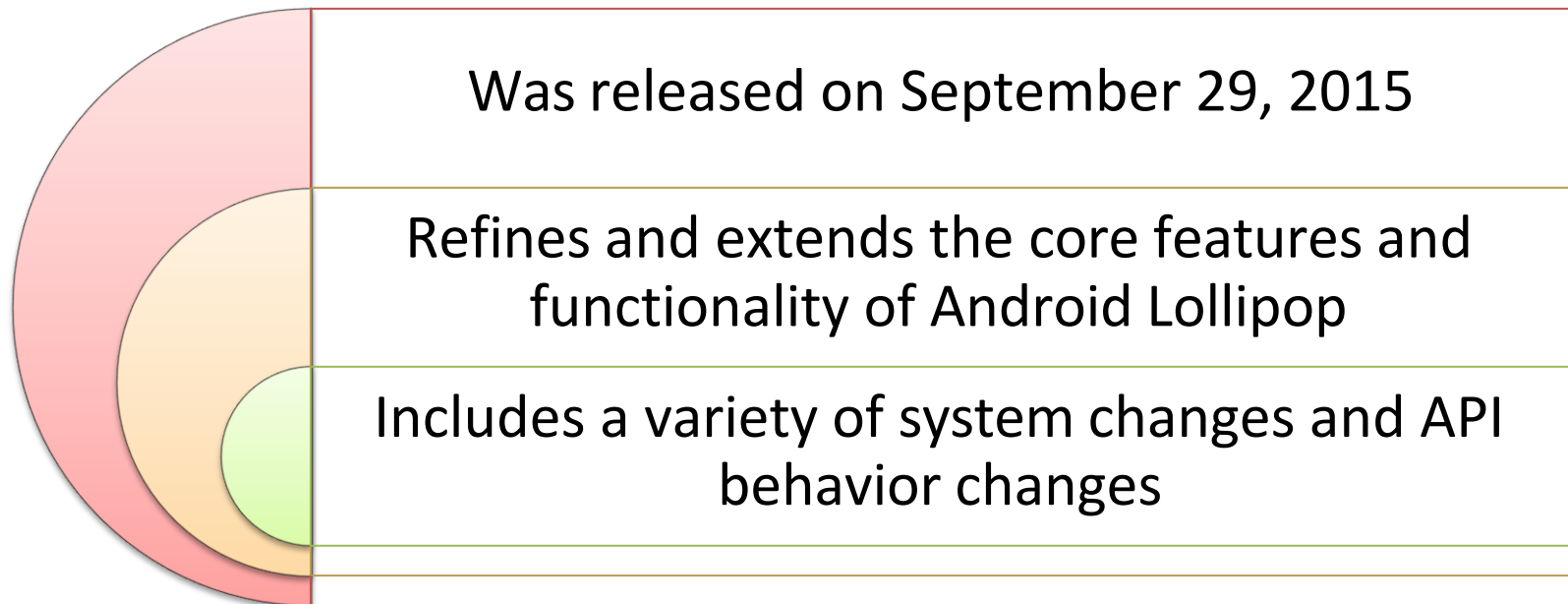


App Indexing Code Generation and Test

Click Tools.



Select Android and choose GoogleApp Indexing Test option.



Updated Features

Some of the key changes are:

- ◆ Doze
- ◆ App Standby
- ◆ Enhanced HTTP support
- ◆ Boring SSL
- ◆ Access to hardware identifier
 - `WifiManager.getScanResults()`
 - `BluetoothDevice.ACTION_FOUND`
 - `BluetoothLeScanner.startScan()`

Android Run Time Permissions 1-2

If the app lists a normal permission in its manifest, the system grants the permission automatically.

To determine if the app has been granted a permission

Call the new `checkSelfPermission()` method

If the app has the permission

Method returns
`PackageManager.PERMISSION_GRANTED`
App can proceed with the operation

If the app does not have permission

Method returns `PERMISSION_DENIED`
App has to explicitly ask the user for permission

Android Run Time Permissions 2-2

- ◆ Users should give explicit approval to the app for access to confidential data.
- ◆ The new `requestPermissions()` method can be used to request permissions for the app at runtime.

Checking App's Permission to Data Access 1-2

Following code checks if the app has permission to read user's contacts and requests the permission:

```
// Here, thisActivity is the current activity
if (ContextCompat.checkSelfPermission(thisActivity, Manifest.
permission.READ_CONTACTS) != PackageManager.PERMISSION_GRANTED)
{
/* Check whether explanation needs to be shown */
if (ActivityCompat.
shouldShowRequestPermissionRationale(thisActivity, Manifest.
permission.READ_CONTACTS))
{
/* Show an explanation to the user and wait for the user's
response
After user has noticed the explanation, request permission
again.*/
}
```

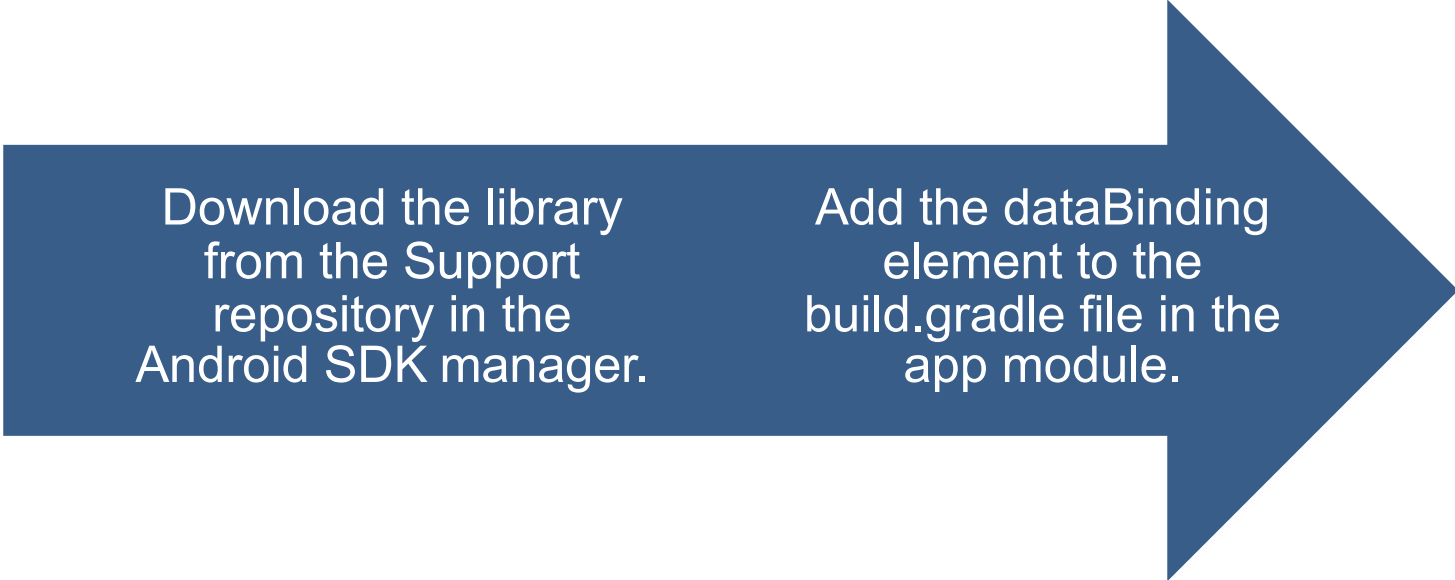
Checking App's Permission to Data Access 2-2

Following code checks if the app has permission to read user's contacts and requests the permission:

```
else
{
// Request permission directly without explanation.
ActivityCompat.requestPermissions(thisActivity, new
String[]{Manifest.permission.READ_CONTACTS}, GRANT_REQUESTS_TO_
READ_CONTACTS);
/* GRANT_REQUESTS_TO_READ_CONTACTS is an app-defined int
constant.
The callback method gets the result of the request. */
}
}
/* When user responds, the system passes response to the app's
onRequestPermissionsResult() method. */
@Override
public void onRequestPermissionsResult(int requestCode, String
accessRights[], int[] accessResults)
{
switch (requestCode)
```

Android Data Binding Library

To configure the app to use data binding:



Download the library
from the Support
repository in the
Android SDK manager.

Add the dataBinding
element to the
build.gradle file in the
app module.

Configuring Data Binding

To configure the app to use data binding:



Use the code snippet to configure data binding.

```
android
{
    ....
    dataBinding
    {
        enabled = true
    }
}
```


Layout File without Data Binding

Following code snippet shows layout file without data binding.

```
<LinearLayout .....">
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="DataBound"
    android:id="@+id/dataBoundText"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true"/>
</LinearLayout>
```

Adding DataBindingHelper Class

Following code snippet shows adding of DataBindingHelper class to the project:

```
package com.databindingexample;
public class DataBindingHelper
{
    private String information;
    public static DataBindingHelper get(String information) {
        return new DataBindingHelper(information);
    }
    public DataBindingHelper(String information)
    {
        this.information = information;
    }
    public String getMessage()
    {
        return String.format("Welcome %s", information);
    }
}
```

Layout Files 1-2

Following code snippet shows the layout files will have the root tag of layout:

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android"
    android:id="@+id/Layout01"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
<data>
<variable
    name="databindingidentifier"
    type="com.databindingexample"/>
</data>
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@{databindingidentifier.message}"
```

Layout Files 2-2

```
android:layout_centerVertical="true"  
android:layout_centerHorizontal="true"  
>  
</LinearLayout>
```

Method References

Following code snippet shows how method references can be used:

```
<Button ...  
android:onClick="@{isTrainer ? handler.trainerClick :  
handler.studentClick}" />  
// First search the setOnClick() method whenever event is fired by  
clicking on the button.  
@BindingMethods  
({@BindingMethod(type = View.class, attribute = "android:onClick",  
method = "setOnClickListener")}  
// Search the OnClickListener for abstract method  
void onClick(View v1);  
// Search the View for setOnClickListener  
void setOnClickListener(View.OnClickListener l)
```

Listener Bindings

Following code snippet shows how to modify the code to use `onClickListener` to handle events:


```
static class Implemantion1OnClickListener implements
OnClickListener
{
public Handler mouseHandler;
@Override
public void onClick(android.view.View arg0)
{
mouseHandler.trainerClick(arg0);
}
}

static class Implemantion2OnClickListener implements
OnClickListener
{
public Handler mouseHandler;
@Override
public void onClick(android.view.View arg0)
{
mouseHandler.studentClick(arg0);
}}
}}
```


Fingerprint Authentication for Payments

To implement fingerprint scan-based authentication feature:

Ensure that the app is running with a fingerprint sensor.



Include the standard Android fingerprint icon in the UI.



Add the `USE_FINGERPRINT` permission in the manifest.

```
<uses-permission  
android:name="android.permission.USE_FINGERPRINT" />
```

Fingerprint Authentication for Online Purchases 1-16

Following code snippet shows how to implement fingerprint authentication for making online purchases:

```
package com.example.android.fingerprintdialog;
import android.app.Activity;
import android.app.DialogFragment;
import android.content.SharedPreferences;
import android.hardware.fingerprint.FingerprintManager;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.inputmethod.EditorInfo;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ImageView;
```


Fingerprint Authentication for Online Purchases 2-16

```
import android.widget.TextView;
import javax.inject.Inject;
public class FingerprintAuthenticationDialogFragment extends
DialogFragment
implements TextView.OnEditorActionListener, FingerprintUiHelper.
Callback {
private Button mCancelButton;
private Button mSecondDialogButton;
private View mFingerprintContent;
private View mBackupContent;
```

Fingerprint Authentication for Online Purchases 3-16

```
private EditText mPassword;
private CheckBox mUseFingerprintFutureCheckBox;
private TextView mPasswordDescriptionTextView;
private TextView mNewFingerprintEnrolledTextView;
private Stage mStage = Stage.FINGERPRINT;
private FingerprintManager.CryptoObject mCryptoObject;
private FingerprintUiHelper mFingerprintUiHelper;
private MainActivity mActivity;
@Inject FingerprintUiHelper.FingerprintUiHelperBuilder
mFingerprintUiHelperBuilder;
@Inject InputMethodManager mInputMethodManager;
@Inject SharedPreferences mSharedPreferences;
@Inject
public FingerprintAuthenticationDialogFragment() {}
@Override
public void onCreate(Bundle savedInstanceState) {
```

Fingerprint Authentication for Online Purchases 4-16

```
// During re-instantiated of activity do not a create a new
fragment
setRetainInstance(true);
super.onCreate(savedInstanceState);
setStyle(DialogFragment.STYLE_NORMAL, android.R.style.
Theme_Material_Light_Dialog);
}
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup
container,
Bundle savedInstanceState) {
getDialog().setTitle(getString(R.string.sign_in));
View v = inflater.inflate(R.layout.fingerprint_dialog_container,
```

Fingerprint Authentication for Online Purchases 5-16

```
container, false);  
mCancelButton = (Button) v.findViewById(R.id.cancel_  
button);  
mCancelButton.setOnClickListener(new View.  
OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        dismiss();  
    }  
});  
mSecondDialogButton = (Button) v.findViewById(R.id.second_  
dialog_button);
```

Fingerprint Authentication for Online Purchases 6-16

```
mSecondDialogButton.setOnClickListener(new View.  
OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        if (mStage == Stage.FINGERPRINT) {  
            goToBackup();  
        } else {  
            verifyPassword();  
        }  
    }  
});  
mFingerprintContent = v.findViewById(R.id.fingerprint_  
container);  
mBackupContent = v.findViewById(R.id.backup_container);  
mPassword = (EditText) v.findViewById(R.id.password);  
mPassword.setOnEditorActionListener(this);  
mPasswordDescriptionTextView = (TextView) v.findViewById(R.  
id.password_description);
```

Fingerprint Authentication for Online Purchases 7-16

```
mUseFingerprintFutureCheckBox = (CheckBox)
v.findViewById(R.id.use_fingerprint_in_future_check);
mNewFingerprintEnrolledTextView = (TextView)
v.findViewById(R.id.new_fingerprint_enrolled_
description);
mFingerprintUiHelper = mFingerprintUiHelperBuilder.
build(
    (ImageView) v.findViewById(R.id.fingerprint_icon),
    (TextView) v.findViewById(R.id.fingerprint_
status), this);
updateStage();
// If fingerprint authentication is not available, switch
immediately to the backup
// (password) screen.
```

Fingerprint Authentication for Online Purchases 8-16

```
if (!mFingerprintUiHelper.isFingerprintAuthAvailable())
{
    goToBackup();
}
return v;
}
@Override
public void onResume() {
    super.onResume();
    If (mStage == Stage.FINGERPRINT) {
        mFingerprintUiHelper.startListening(mCryptoObject);
    }
}
```

Fingerprint Authentication for Online Purchases 9-16

```
public void setStage(Stage stage) {
    mStage = stage;
}
@Override
public void onPause() {
    super.onPause();
    mFingerprintUiHelper.stopListening();
}
@Override
public void onAttach(Activity activity) {
    super.onAttach(activity);
    mActivity = (MainActivity) activity;
}
/**
 * Sets the crypto object to be passed in when authenticating
 * with fingerprint.
 */
```


Fingerprint Authentication for Online Purchases 10-16

```
public void setCryptoObject(FingerprintManager.CryptoObject
cryptoObject) {
    mCryptoObject = cryptoObject;
}
// After Unsuccessfull attempt for authenticating lets switch to
the other authentication method
private void goToBackup() {
    mStage = Stage.PASSWORD;
    updateStage();
    mPassword.requestFocus();
    // Enable keyboard.
    mPassword.postDelayed(mShowKeyboardRunnable, 500);
    // Fingerprint is not used anymore. Stop listening for it.
    mFingerprintUiHelper.stopListening();
}
```

Fingerprint Authentication for Online Purchases 11-16

```
//After Successful authentication dismisses the dialog
private void verifyPassword() {
    if (!checkPassword(mPassword.getText().toString())) {
        return;
    }
    if (mStage == Stage.NEW_FINGERPRINT_ENROLLED) {
        SharedPreferences.Editor editor = mSharedPreferences.
            edit();
        editor.putBoolean(getString(R.string.use_fingerprint_
            to_authenticate_key),
            mUseFingerprintFutureCheckBox.isChecked());
        editor.apply();
        if (mUseFingerprintFutureCheckBox.isChecked()) {
            // Re-create the key so that fingerprints including
```

Fingerprint Authentication for Online Purchases 12-16

```
new ones are validated.
mActivity.createKey();
mStage = Stage.FINGERPRINT;
}
}
mPassword.setText("");
mActivity.onPurchased(false /* without Fingerprint */);
dismiss();
}
/**
 * @return true if {@code password} is correct, false otherwise
 */
private boolean checkPassword(String password) {
    // Authentication process needs to be verified at server side.
    return password.length() > 0;
}
```

Fingerprint Authentication for Online Purchases 13-16

```
private final Runnable mShowKeyboardRunnable = new Runnable() {  
    @Override  
    public void run() {  
        mInputMethodManager.showSoftInput(mPassword, 0);  
    }  
};  
  
private void updateStage() {  
    switch (mStage) {  
        case FINGERPRINT:  
            mCancelButton.setText(R.string.cancel);  
            mSecondDialogButton.setText(R.string.use_password);  
            mFingerprintContent.setVisibility(View.VISIBLE);  
            mBackupContent.setVisibility(View.GONE);  
            break;  
        case NEW_FINGERPRINT_ENROLLED:  
            // Intentional fall through  
        case PASSWORD:
```

Fingerprint Authentication for Online Purchases 14-16

```
mCancelButton.setText(R.string.cancel);
mSecondDialogButton.setText(R.string.ok);
mFingerprintContent.setVisibility(View.GONE);
mBackupContent.
setVisibility(View.VISIBLE);
if (mStage == Stage.NEW_FINGERPRINT_ENROLLED) {
mPasswordDescriptionTextView.
setVisibility(View.GONE);
mNewFingerprintEnrolledTextView.
setVisibility(View.VISIBLE);
mUseFingerprintFutureCheckBox.
setVisibility(View.VISIBLE);
}
break;
}
}
```

Fingerprint Authentication for Online Purchases 15-16

```
@Override
public boolean onEditorAction(TextView v, int actionId,
KeyEvent event) {
    if (actionId == EditorInfo.IME_ACTION_GO) {
        verifyPassword();
        return true;
    }
    return false;
}

@Override
public void onAuthenticated() {
    // After the authentication was successfull, call the
    UiHelper
    mActivity.onPurchased(true /* withFingerprint */);
    dismiss();
}
```

Fingerprint Authentication for Online Purchases 16-16

```
@Override
public void onError() {
    goToBackup();
}
//Types of Authentication Methods which are used to authenticate
the user.
public enum Stage {
    FINGERPRINT,
    NEW_FINGERPRINT_ENROLLED,
    PASSWORD
}
}
```

Confirm Credential Functionality

Following code snippet shows inclusion of Confirm Credential Functionality:

```
private void showAuthenticationScreen()
{
    // Description and Title are customizable, this
    // creates the Confirm Credentials screen Or if it is left
    null,
    // a generic one shall be provided
    Intent i = mKeyguardManager.
    createConfirmDeviceCredentialIntent(null, null);
    if (i != null)
    { startActivityForResult(i, REQUEST_CODE_CONFIRM_DEVICE_
    CREDENTIALS);
    }
}
```


Direct Share 1-2

To enable direct share targets, developers need to perform following steps:

1. Define a class that extends the `ChooserTargetService` class. Declare this service in the manifest, as shown in code snippet.

```
<service android:name=".ChooserTargetService" android:label="@string/service_name" android:permission="android.permission.BIND_CHOOSER_TARGET_SERVICE">
<intent-filter> <action android:name="android.service.chooser.ChooserTargetService" />
</intent-filter>
</service>
```

2. For each activity that needs to be exposed to ChooserTargetService, add a `<meta-data>` element with the name “android.service.chooser.chooser_target_service” in the app manifest, as shown in the code snippet.

```
<activity android:name=".MyShareActivity"
android:label="@string/share_activity_label">
<intent-filter>
<action android:name="android.intent.action.SEND" />
</intent-filter>
<meta-data
android:name="android.service.chooser.chooser_target_
service"
android:value=".ChooserTargetService" />
</activity>
```

Summary

- ◆ Android Studio 2.1 has many new features to develop applications faster and efficiently.
- ◆ The Instant Run feature makes changes made to the app live instantly in the running app.
- ◆ The new emulator has rich new features to manage calls, battery, network, GPS, and more.
- ◆ The Cloud Test Lab Integration feature allows developers to run and test the app on a wide range of physical Android devices in the Cloud Test Lab from within Android Studio.
- ◆ The new App Indexing feature in Android Studio allows the developers to add indexable URL links that they can test all within the IDE.
- ◆ Android 6.0 Marshmallow has doze and app standby features to save power and resources.
- ◆ Fingerprint and Confirm Credential APIs help developers to incorporate fingerprint scan-based authentication and accelerate payments and sign-in operations in the app.
- ◆ The Direct Share feature of Marshmallow allows developers to define direct share targets, such as contacts within other apps.