Android™

Notes for Professionals

Chapter 28: Creating Custom Views

Section 28.1: Creating Custom Views

If you need a completely customized view, you'll need to subclass View (the superclass of all A… provide your custom sizing (onMeasure(...)) and drawing (onDraw(...)) methods:

1. **Create your custom view skeleton:** this is basically the same for every custom view. … skeleton for a custom view that can draw a smiley, called SmileyView:

```
public class SmileyView extends View {
    private Paint mCirclePaint;
    private Paint mEyeAndMouthPaint;

    private float mCenterX;
    private float mCenterY;
    private float mRadius;
    private RectF mArcBounds = new RectF();

    public SmileyView(Context context) {
        this(context, null, 0);
    }

    public SmileyView(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
    }

    public SmileyView(Context context, AttributeSet attrs, int de
        super(context, attrs, defStyleAttr);
        initPaints();
    }

    private void initPaints() {/* ... */}

    @Override
    protected void onMeasure(int widthMeasureSpec, int heightM

    @Override
    protected void onDraw(Canvas canvas) {/* ... */}
}
```

2. **Initialize your paints:** the Paint objects are the brushes of your virtu… objects are rendered (e.g. color, fill and stroke style, etc.). Here we cre… for the circle and one black stroke paint for the eyes and the mouth:

```
private void initPaints() {
    mCirclePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    mCirclePaint.setStyle(Paint.Style.FILL);
    mCirclePaint.setColor(Color.YELLOW);
    mEyeAndMouthPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    mEyeAndMouthPaint.setStyle(Paint.Style.STROKE);
    mEyeAndMouthPaint.setStrokeWidth(16 * getResources().getD
    mEyeAndMouthPaint.setStrokeCap(Paint.Cap.ROUND);
    mEyeAndMouthPaint.setColor(Color.BLACK);
}
```

3. **Implement your own onMeasure(...) method:** this is required so that the parent layouts (e.g.

Android™ Notes for Professionals 212

Chapter 91: Menu

Parameter	Description
inflate(int menuRes, Menu menu)	Inflate a menu hierarchy from the specified XML resource.
getMenuInflater ()	Returns a MenuInflater with this context.
onCreateOptionsMenu (Menu menu)	Initialize the contents of the Activity's standard options menu. You should place your menu items in to menu.
onOptionsItemSelected (MenuItem item)	This method is called whenever an item in your options menu is selected

Section 91.1: Options menu with dividers

In Android there is a default options menu, which can take a number of options. If a larger number of options needs to be displayed, then it makes sense to group those options in order to maintain clarity. Options can be grouped by putting dividers (i.e. horizontal lines) between them. In order to allow for dividers, the following theme can be used:

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
    <item name="android:dropDownListViewStyle">@style/PopupMenuListView</item>
</style>
<style name="PopupMenuListView" parent="@style/Widget.AppCompat.ListView.DropDown">
    <item name="android:divider">@color/black</item>
    <item name="android:dividerHeight">1dp</item>
</style>
```

By changing the theme, dividers can be added to a menu.

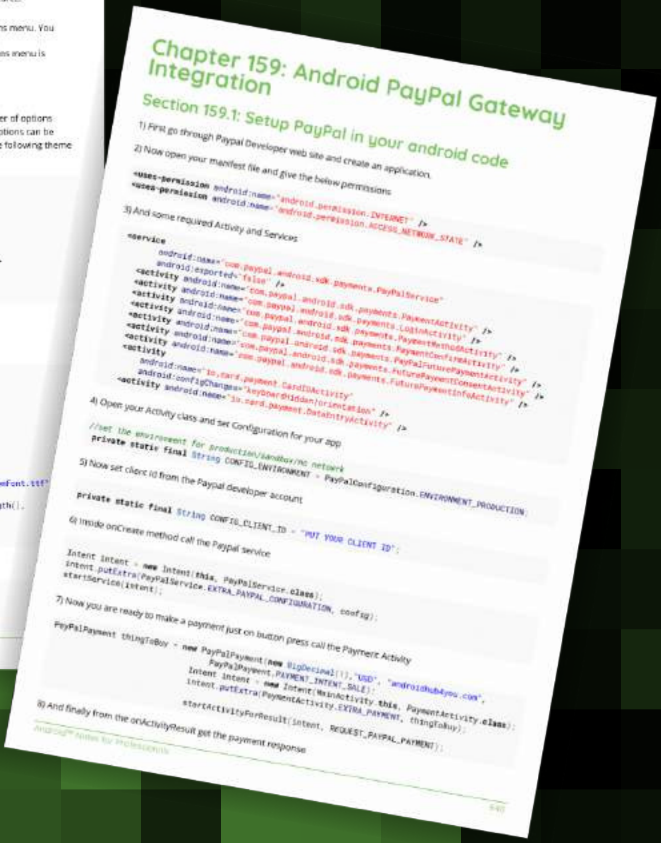
Section 91.2: Apply custom font to Menu

```
public static void applyFontToMenu(Menu m, Context mContext){
    for(int i=0;i<m.size();i++) {
        applyFontToMenuItem(m.getItem(i),mContext);
    }
}
public static void applyFontToMenuItem(MenuItem mi, Context mContext) {
    if(mi.hasSubMenu())
        for(int i=0;i<mi.getSubMenu().size();i++) {
            applyFontToMenuItem(mi.getSubMenu().getItem(i),mContext);
        }
    Typeface font = Typeface.createFromAsset(mContext.getAssets(), "fonts/yourCustomFont.ttf"
    SpannableString mNewTitle = new SpannableString(mi.getTitle());
    mNewTitle.setSpan(new CustomTypefaceSpan("", font, mContext), 0, mNewTitle.length(),
Spannable.SPAN_INCLUSIVE_INCLUSIVE);
    mi.setTitle(mNewTitle);
}
```

and then in the Activity:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    applyFontToMenu(menu,this);
```

Android™ Notes for Professionals


Chapter 159: Android PayPal Gateway Integration

Section 159.1: Setup PayPal in your android code

1) First go through Paypal Developer web site and create an application.

2) Now open your manifest file and give the below permissions

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

3) And some required Activity and Services

```
<service
    android:name="com.paypal.android.sdk.payments.PayPalService"
    android:exported="false" />
<activity android:name="com.paypal.android.sdk.payments.PaymentActivity" />
<activity android:name="com.paypal.android.sdk.payments.LoginActivity" />
<activity android:name="com.paypal.android.sdk.payments.PaymentMethodActivity" />
<activity android:name="com.paypal.android.sdk.payments.PaymentConfirmActivity" />
<activity android:name="com.paypal.android.sdk.payments.PayPalFuturePaymentActivity" />
<activity android:name="com.paypal.android.sdk.payments.FuturePaymentConsentActivity" />
<activity android:name="com.paypal.android.sdk.payments.FuturePaymentInfoActivity" />
<activity
    android:name="io.card.payment.CardIOActivity"
    android:configChanges="keyboardHidden|orientation" />
<activity android:name="io.card.payment.DataEntryActivity" />
```

4) Open your Activity class and set Configuration for your app

```
//set the environment for production/sandbox/no netowrk
private static final String CONFIG_ENVIRONMENT = PayPalConfiguration.ENVIRONMENT_PRODUCTION;
```

5) Now set client id from the Paypal developer account

```
private static final String CONFIG_CLIENT_ID = "PUT YOUR CLIENT ID";
```

6) Inside onCreate method call the Paypal service

```
Intent intent = new Intent(this, PayPalService.class);
intent.putExtra(PayPalService.EXTRA_PAYPAL_CONFIGURATION, config);
startService(intent);
```

7) Now you are ready to make a payment just on button press call the Payment Activity

```
PayPalPayment thingToBuy = new PayPalPayment(new BigDecimal(1),"USD", "androidhub4you.com",
            PayPalPayment.PAYMENT_INTENT_SALE);
        Intent intent = new Intent(MainActivity.this, PaymentActivity.class);
        intent.putExtra(PaymentActivity.EXTRA_PAYMENT, thingToBuy);
        startActivityForResult(intent, REQUEST_PAYPAL_PAYMENT);
```

8) And finally from the onActivityResult get the payment response

Android™ Notes for Professionals 640

1000+ pages

of professional hints and tricks

GoalKicker.com
Free Programming Books

Disclaimer
This is an unofficial free book created for educational purposes and is not affiliated with official Android™ group(s) or company(s).
All trademarks and registered trademarks are the property of their respective owners

Contents

About ... 1
Chapter 1: Getting started with Android ... 2
- Section 1.1: Creating a New Project ... 2
- Section 1.2: Setting up Android Studio ... 13
- Section 1.3: Android programming without an IDE ... 14
- Section 1.4: Application Fundamentals ... 18
- Section 1.5: Setting up an AVD (Android Virtual Device) ... 19

Chapter 2: Android Studio ... 23
- Section 2.1: Setup Android Studio ... 23
- Section 2.2: View And Add Shortcuts in Android Studio ... 23
- Section 2.3: Android Studio useful shortcuts ... 24
- Section 2.4: Android Studio Improve performance tip ... 25
- Section 2.5: Gradle build project takes forever ... 26
- Section 2.6: Enable/Disable blank line copy ... 26
- Section 2.7: Custom colors of logcat message based on message importance ... 27
- Section 2.8: Filter logs from UI ... 28
- Section 2.9: Create filters configuration ... 29
- Section 2.10: Create assets folder ... 30

Chapter 3: Instant Run in Android Studio ... 32
- Section 3.1: Enabling or disabling Instant Run ... 32
- Section 3.2: Types of code Swaps in Instant Run ... 32
- Section 3.3: Unsupported code changes when using Instant Run ... 33

Chapter 4: TextView ... 34
- Section 4.1: Spannable TextView ... 34
- Section 4.2: Strikethrough TextView ... 35
- Section 4.3: TextView with image ... 36
- Section 4.4: Make RelativeSizeSpan align to top ... 36
- Section 4.5: Pinchzoom on TextView ... 38
- Section 4.6: Textview with different Textsize ... 39
- Section 4.7: Theme and Style customization ... 39
- Section 4.8: TextView customization ... 41
- Section 4.9: Single TextView with two different colors ... 44

Chapter 5: AutoCompleteTextView ... 46
- Section 5.1: AutoComplete with CustomAdapter, ClickListener and Filter ... 46
- Section 5.2: Simple, hard-coded AutoCompleteTextView ... 49

Chapter 6: Autosizing TextViews ... 50
- Section 6.1: Granularity ... 50
- Section 6.2: Preset Sizes ... 50

Chapter 7: ListView ... 52
- Section 7.1: Custom ArrayAdapter ... 52
- Section 7.2: A basic ListView with an ArrayAdapter ... 53
- Section 7.3: Filtering with CursorAdapter ... 53

Chapter 8: Layouts ... 55
- Section 8.1: LayoutParams ... 55
- Section 8.2: Gravity and layout gravity ... 58
- Section 8.3: CoordinatorLayout Scrolling Behavior ... 60

Section 8.4: Percent Layouts ... 62
Section 8.5: View Weight ... 63
Section 8.6: Creating LinearLayout programmatically ... 64
Section 8.7: LinearLayout ... 65
Section 8.8: RelativeLayout ... 66
Section 8.9: FrameLayout ... 68
Section 8.10: GridLayout ... 69
Section 8.11: CoordinatorLayout ... 71
Chapter 9: ConstraintLayout ... 73
Section 9.1: Adding ConstraintLayout to your project ... 73
Section 9.2: Chains ... 74
Chapter 10: TextInputLayout ... 75
Section 10.1: Basic usage ... 75
Section 10.2: Password Visibility Toggles ... 75
Section 10.3: Adding Character Counting ... 75
Section 10.4: Handling Errors ... 76
Section 10.5: Customizing the appearance of the TextInputLayout ... 76
Section 10.6: TextInputEditText ... 77
Chapter 11: CoordinatorLayout and Behaviors ... 79
Section 11.1: Creating a simple Behavior ... 79
Section 11.2: Using the SwipeDismissBehavior ... 80
Section 11.3: Create dependencies between Views ... 80
Chapter 12: TabLayout ... 82
Section 12.1: Using a TabLayout without a ViewPager ... 82
Chapter 13: ViewPager ... 83
Section 13.1: ViewPager with a dots indicator ... 83
Section 13.2: Basic ViewPager usage with fragments ... 85
Section 13.3: ViewPager with PreferenceFragment ... 86
Section 13.4: Adding a ViewPager ... 87
Section 13.5: Setup OnPageChangeListener ... 88
Section 13.6: ViewPager with TabLayout ... 89
Chapter 14: CardView ... 92
Section 14.1: Getting Started with CardView ... 92
Section 14.2: Adding Ripple animation ... 93
Section 14.3: Customizing the CardView ... 93
Section 14.4: Using Images as Background in CardView (Pre-Lollipop device issues) ... 94
Section 14.5: Animate CardView background color with TransitionDrawable ... 96
Chapter 15: NavigationView ... 97
Section 15.1: How to add the NavigationView ... 97
Section 15.2: Add underline in menu elements ... 101
Section 15.3: Add seperators to menu ... 102
Section 15.4: Add menu Divider using default DividerItemDecoration ... 103
Chapter 16: RecyclerView ... 105
Section 16.1: Adding a RecyclerView ... 105
Section 16.2: Smoother loading of items ... 106
Section 16.3: RecyclerView with DataBinding ... 107
Section 16.4: Animate data change ... 108
Section 16.5: Popup menu with recyclerView ... 112
Section 16.6: Using several ViewHolders with ItemViewType ... 114

Section 16.7: Filter items inside RecyclerView with a SearchView ... 115
Section 16.8: Drag&Drop and Swipe with RecyclerView ... 116
Section 16.9: Show default view till items load or when data is not available ... 117
Section 16.10: Add header/footer to a RecyclerView ... 119
Section 16.11: Endless Scrolling in Recycleview ... 122
Section 16.12: Add divider lines to RecyclerView items ... 122
Chapter 17: RecyclerView Decorations ... 125
Section 17.1: Add divider to RecyclerView ... 125
Section 17.2: Drawing a Separator ... 127
Section 17.3: How to add dividers using and DividerItemDecoration ... 128
Section 17.4: Per-item margins with ItemDecoration ... 128
Section 17.5: ItemOffsetDecoration for GridLayoutManager in RecycleView ... 129
Chapter 18: RecyclerView onClickListeners ... 131
Section 18.1: Kotlin and RxJava example ... 131
Section 18.2: RecyclerView Click listener ... 132
Section 18.3: Another way to implement Item Click Listener ... 133
Section 18.4: New Example ... 135
Section 18.5: Easy OnLongClick and OnClick Example ... 136
Section 18.6: Item Click Listeners ... 139
Chapter 19: RecyclerView and LayoutManagers ... 141
Section 19.1: Adding header view to recyclerview with gridlayout manager ... 141
Section 19.2: GridLayoutManager with dynamic span count ... 142
Section 19.3: Simple list with LinearLayoutManager ... 144
Section 19.4: StaggeredGridLayoutManager ... 148
Chapter 20: Pagination in RecyclerView ... 151
Section 20.1: MainActivity.java ... 151
Chapter 21: ImageView ... 156
Section 21.1: Set tint ... 156
Section 21.2: Set alpha ... 157
Section 21.3: Set Scale Type ... 157
Section 21.4: ImageView ScaleType - Center ... 162
Section 21.5: ImageView ScaleType - CenterCrop ... 164
Section 21.6: ImageView ScaleType - CenterInside ... 166
Section 21.7: ImageView ScaleType - FitStart and FitEnd ... 168
Section 21.8: ImageView ScaleType - FitCenter ... 172
Section 21.9: Set Image Resource ... 174
Section 21.10: ImageView ScaleType - FitXy ... 175
Section 21.11: MLRoundedImageView.java ... 177
Chapter 22: VideoView ... 180
Section 22.1: Play video from URL with using VideoView ... 180
Section 22.2: VideoView Create ... 180
Chapter 23: Optimized VideoView ... 181
Section 23.1: Optimized VideoView in ListView ... 181
Chapter 24: WebView ... 193
Section 24.1: Troubleshooting WebView by printing console messages or by remote debugging ... 193
Section 24.2: Communication from Javascript to Java (Android) ... 194
Section 24.3: Communication from Java to Javascript ... 195
Section 24.4: Open dialer example ... 195
Section 24.5: Open Local File / Create dynamic content in Webview ... 196

Section 24.6: JavaScript alert dialogs in WebView - How to make them work 196
Chapter 25: SearchView 198
Section 25.1: Setting Theme for SearchView 198
Section 25.2: SearchView in Toolbar with Fragment 198
Section 25.3: Appcompat SearchView with RxBindings watcher 200
Chapter 26: BottomNavigationView 203
Section 26.1: Basic implemetation 203
Section 26.2: Customization of BottomNavigationView 204
Section 26.3: Handling Enabled / Disabled states 204
Section 26.4: Allowing more than 3 menus 205
Chapter 27: Canvas drawing using SurfaceView 207
Section 27.1: SurfaceView with drawing thread 207
Chapter 28: Creating Custom Views 212
Section 28.1: Creating Custom Views 212
Section 28.2: Adding attributes to views 214
Section 28.3: CustomView performance tips 216
Section 28.4: Creating a compound view 217
Section 28.5: Compound view for SVG/VectorDrawable as drawableRight 220
Section 28.6: Responding to Touch Events 223
Chapter 29: Getting Calculated View Dimensions 224
Section 29.1: Calculating initial View dimensions in an Activity 224
Chapter 30: Adding a FuseView to an Android Project 225
Section 30.1: hikr app, just another android.view.View 225
Chapter 31: Supporting Screens With Different Resolutions, Sizes 232
Section 31.1: Using configuration qualifiers 232
Section 31.2: Converting dp and sp to pixels 232
Section 31.3: Text size and different android screen sizes 233
Chapter 32: ViewFlipper 234
Section 32.1: ViewFlipper with image sliding 234
Chapter 33: Design Patterns 235
Section 33.1: Observer pattern 235
Section 33.2: Singleton Class Example 235
Chapter 34: Activity 237
Section 34.1: Activity launchMode 237
Section 34.2: Exclude an activity from back-stack history 238
Section 34.3: Android Activity LifeCycle Explained 238
Section 34.4: End Application with exclude from Recents 241
Section 34.5: Presenting UI with setContentView 242
Section 34.6: Up Navigation for Activities 243
Section 34.7: Clear your current Activity stack and launch a new Activity 244
Chapter 35: Activity Recognition 246
Section 35.1: Google Play ActivityRecognitionAPI 246
Section 35.2: PathSense Activity Recognition 248
Chapter 36: Split Screen / Multi-Screen Activities 250
Section 36.1: Split Screen introduced in Android Nougat implemented 250
Chapter 37: Material Design 251
Section 37.1: Adding a Toolbar 251
Section 37.2: Buttons styled with Material Design 252

Click here to download full PDF material