**引言**

今天给大家总结有关自定义对话框的相关内容，前面文章[Android入门——AlertDialog和ProgressDialog总结](http://blog.csdn.net/crazymo_/article/details/50969177) 都在在利用系统提供的函数来实现对话框，但局限性太大，当我们想自己定义Dialog视图的时候，就不能利用系统函数了，就需要我们这里的自定义对话框了来满足产品经理的各种idea。

**一、Dialog部分源码结构**

学习下源码的编程风格和规范

/\*\*

\* Base class for Dialogs.

\* Activity提供了一系列的方法用于dialog的管理：onCreateDialog(int)、onPrepareDialog(int)、showDialog(int)、dismissDialog(int)、

\*当这些方法被调用之后我们就可以通过 getOwnerActivity()方法获取得到对应的Dialog依附的Activity

\*

\*设置在Dialog中隐藏软键盘

\* getWindow().setFlags(WindowManager.LayoutParams.FLAG\_ALT\_FOCUSABLE\_IM,

\* WindowManager.LayoutParams.FLAG\_ALT\_FOCUSABLE\_IM);

\*/

public class Dialog implements DialogInterface, Window.Callback,

KeyEvent.Callback, OnCreateContextMenuListener, Window.OnWindowDismissedCallback {

private Activity mOwnerActivity;

final Context mContext;

final WindowManager mWindowManager;

Window mWindow;

View mDecor;

private ActionBar mActionBar;

protected boolean mCancelable = true;

private String mCancelAndDismissTaken;

private Message mCancelMessage,mDismissMessage,mShowMessage;

private OnKeyListener mOnKeyListener;

private boolean mCreated = false,mShowing = false,mCanceled = false;

private final Handler mHandler = new Handler();

private static final int DISMISS = 0x43,CANCEL = 0x44,SHOW = 0x45;

private Handler mListenersHandler;

private SearchEvent mSearchEvent;

private ActionMode mActionMode;

private int mActionModeTypeStarting = ActionMode.TYPE\_PRIMARY;

private final Runnable mDismissAction = new Runnable() {

public void run() {

dismissDialog();

}

};

public Dialog(@NonNull Context context) {

this(context, 0, true);

}

public Dialog(@NonNull Context context, @StyleRes int themeResId) {

this(context, themeResId, true);

}

Dialog(@NonNull Context context, @StyleRes int themeResId, boolean createContextThemeWrapper) {

if (createContextThemeWrapper) {

if (themeResId == 0) {

final TypedValue outValue = new TypedValue();

context.getTheme().resolveAttribute(R.attr.dialogTheme, outValue, true);

themeResId = outValue.resourceId;

}

mContext = new ContextThemeWrapper(context, themeResId);

} else {

mContext = context;

}

mWindowManager = (WindowManager) context.getSystemService(Context.WINDOW\_SERVICE);

final Window w = new PhoneWindow(mContext);

mWindow = w;

w.setCallback(this);

w.setOnWindowDismissedCallback(this);

w.setWindowManager(mWindowManager, null, null);

w.setGravity(Gravity.CENTER);

mListenersHandler = new ListenersHandler(this);

}

/\*\*

\* @deprecated

\* @hide

\*/

@Deprecated

protected Dialog(@NonNull Context context, boolean cancelable, Message cancelCallback) {

this(context);

mCancelable = cancelable;

mCancelMessage = cancelCallback;

}

protected Dialog(@NonNull Context context, boolean cancelable,

OnCancelListener cancelListener) {

this(context);

mCancelable = cancelable;

setOnCancelListener(cancelListener);

}

/\*\*

\* Retrieve the Context this Dialog is running in.

\* @return Context The Context used by the Dialog.

\*/

@NonNull

public final Context getContext() {

return mContext;

}

/\*\*

\* Sets the Activity that owns this dialog. An example use: This Dialog will

\* use the suggested volume control stream of the Activity.

\* @param activity The Activity that owns this dialog.

\*/

public final void setOwnerActivity(Activity activity) {

mOwnerActivity = activity;

getWindow().setVolumeControlStream(mOwnerActivity.getVolumeControlStream());

}

/\*\*

\* Returns the Activity that owns this Dialog. For example, if

\* {@link Activity#showDialog(int)} is used to show this Dialog, that

\* Activity will be the owner (by default). Depending on how this dialog was

\* created, this may return null.

\*

\* @return The Activity that owns this Dialog.

\*/

public final Activity getOwnerActivity() {

return mOwnerActivity;

}

public boolean isShowing() {

return mShowing;

}

public void create() {

if (!mCreated) {

dispatchOnCreate(null);

}

}

public void show() {

if (DBG) {

Log.d(TAG, "show");

}

if (mShowing) {

if (mDecor != null) {

if (mWindow.hasFeature(Window.FEATURE\_ACTION\_BAR)) {

mWindow.invalidatePanelMenu(Window.FEATURE\_ACTION\_BAR);

}

mDecor.setVisibility(View.VISIBLE);

}

return;

}

mCanceled = false;

if (!mCreated) {

dispatchOnCreate(null);

}

onStart();

mDecor = mWindow.getDecorView();

if (mActionBar == null && mWindow.hasFeature(Window.FEATURE\_ACTION\_BAR)) {

final ApplicationInfo info = mContext.getApplicationInfo();

mWindow.setDefaultIcon(info.icon);

mWindow.setDefaultLogo(info.logo);

mActionBar = new WindowDecorActionBar(this);

}

WindowManager.LayoutParams l = mWindow.getAttributes();

if ((l.softInputMode

& WindowManager.LayoutParams.SOFT\_INPUT\_IS\_FORWARD\_NAVIGATION) == 0) {

WindowManager.LayoutParams nl = new WindowManager.LayoutParams();

nl.copyFrom(l);

nl.softInputMode |=

WindowManager.LayoutParams.SOFT\_INPUT\_IS\_FORWARD\_NAVIGATION;

l = nl;

}

try {

mWindowManager.addView(mDecor, l);

mShowing = true;

sendShowMessage();

} finally {

}

}

/\*\*

\* Hide the dialog, but do not dismiss it.

\*/

public void hide() {

if (mDecor != null) {

mDecor.setVisibility(View.GONE);

}

}

/\*\*

\* 关闭并删除Dialog，线程安全，不能重写

\*/

@Override

public void dismiss() {

if (Looper.myLooper() == mHandler.getLooper()) {

dismissDialog();

} else {

mHandler.post(mDismissAction);

}

}

void dismissDialog() {

if (DBG) {

Log.d(TAG, "dismissDialog");

}

if (mDecor == null || !mShowing) {

return;

}

if (mWindow.isDestroyed()) {

Log.e(TAG, "Tried to dismissDialog() but the Dialog's window was already destroyed!");

return;

}

try {

mWindowManager.removeViewImmediate(mDecor);

} finally {

if (mActionMode != null) {

mActionMode.finish();

}

mDecor = null;

mWindow.closeAllPanels();

onStop();

mShowing = false;

sendDismissMessage();

}

}

private void sendDismissMessage() {

if (mDismissMessage != null) {

Message.obtain(mDismissMessage).sendToTarget();

}

}

private void sendShowMessage() {

if (mShowMessage != null) {

// Obtain a new message so this dialog can be re-used

Message.obtain(mShowMessage).sendToTarget();

}

}

/\*\*

\*与Activity的类似，初始化Dialog,包括调用 setContentView

\*如果Dialog在Activity之前关闭，那么会调用onSaveInstanceState保留状态

\* @param savedInstanceState If this dialog is being reinitalized after a

\*/

protected void onCreate(Bundle savedInstanceState) {

}

/\*\*

\* Called when the dialog is starting.

\*/

protected void onStart() {

if (mActionBar != null) mActionBar.setShowHideAnimationEnabled(true);

}

/\*\*

\* Called to tell you that you're stopping.

\*/

protected void onStop() {

if (mActionBar != null) mActionBar.setShowHideAnimationEnabled(false);

}

private static final String DIALOG\_SHOWING\_TAG = "android:dialogShowing";

private static final String DIALOG\_HIERARCHY\_TAG = "android:dialogHierarchy";

public Bundle onSaveInstanceState() {

Bundle bundle = new Bundle();

bundle.putBoolean(DIALOG\_SHOWING\_TAG, mShowing);

if (mCreated) {

bundle.putBundle(DIALOG\_HIERARCHY\_TAG, mWindow.saveHierarchyState());

}

return bundle;

}

public void onRestoreInstanceState(Bundle savedInstanceState) {

final Bundle dialogHierarchyState = savedInstanceState.getBundle(DIALOG\_HIERARCHY\_TAG);

if (dialogHierarchyState == null) {

// dialog has never been shown, or onCreated, nothing to restore.

return;

}

dispatchOnCreate(savedInstanceState);

mWindow.restoreHierarchyState(dialogHierarchyState);

if (savedInstanceState.getBoolean(DIALOG\_SHOWING\_TAG)) {

show();

}

}

public Window getWindow() {

return mWindow;

}

public View getCurrentFocus() {

return mWindow != null ? mWindow.getCurrentFocus() : null;

}

@Nullable

public View findViewById(@IdRes int id) {

return mWindow.findViewById(id);

}

public void setContentView(@LayoutRes int layoutResID) {

mWindow.setContentView(layoutResID);

}

public void setContentView(View view) {

mWindow.setContentView(view);

}

public void setContentView(View view, ViewGroup.LayoutParams params) {

mWindow.setContentView(view, params);

}

public void addContentView(View view, ViewGroup.LayoutParams params) {

mWindow.addContentView(view, params);

}

public void setTitle(CharSequence title) {

mWindow.setTitle(title);

mWindow.getAttributes().setTitle(title);

}

public void setTitle(@StringRes int titleId) {

setTitle(mContext.getText(titleId));

}

public boolean onKeyDown(int keyCode, KeyEvent event) {

if (keyCode == KeyEvent.KEYCODE\_BACK) {

event.startTracking();

return true;

}

return false;

}

public boolean onKeyLongPress(int keyCode, KeyEvent event) {

return false;

}

public boolean onKeyUp(int keyCode, KeyEvent event) {

if (keyCode == KeyEvent.KEYCODE\_BACK && event.isTracking()

&& !event.isCanceled()) {

onBackPressed();

return true;

}

return false;

}

public boolean onKeyMultiple(int keyCode, int repeatCount, KeyEvent event) {

return false;

}

public void onBackPressed() {

if (mCancelable) {

cancel();

}

}

public boolean onKeyShortcut(int keyCode, KeyEvent event) {

return false;

}

public boolean onTouchEvent(MotionEvent event) {

if (mCancelable && mShowing && mWindow.shouldCloseOnTouch(mContext, event)) {

cancel();

return true;

}

return false;

}

public boolean onTrackballEvent(MotionEvent event) {

return false;

}

public void onWindowAttributesChanged(WindowManager.LayoutParams params) {

if (mDecor != null) {

mWindowManager.updateViewLayout(mDecor, params);

}

}

public void onContentChanged() {

}

public void onWindowFocusChanged(boolean hasFocus) {

}

public void onAttachedToWindow() {

}

public void onDetachedFromWindow() {

}

/\*\* @hide \*/

@Override

public void onWindowDismissed() {

dismiss();

}

/\*\*

\* 事件传递 ：用于处理各种事件，我们可以重写key events在传递到window之前

\* @param event The key event.

\*/

public boolean dispatchKeyEvent(KeyEvent event) {

if ((mOnKeyListener != null) && (mOnKeyListener.onKey(this, event.getKeyCode(), event))) {

return true;

}

if (mWindow.superDispatchKeyEvent(event)) {

return true;

}

return event.dispatch(this, mDecor != null

? mDecor.getKeyDispatcherState() : null, this);

}

public boolean dispatchKeyShortcutEvent(KeyEvent event) {

if (mWindow.superDispatchKeyShortcutEvent(event)) {

return true;

}

return onKeyShortcut(event.getKeyCode(), event);

}

public boolean dispatchTouchEvent(MotionEvent ev) {

if (mWindow.superDispatchTouchEvent(ev)) {

return true;

}

return onTouchEvent(ev);

}

public final boolean requestWindowFeature(int featureId) {

return getWindow().requestFeature(featureId);

}

public final void setFeatureDrawableResource(int featureId, @DrawableRes int resId) {

getWindow().setFeatureDrawableResource(featureId, resId);

}

public final void setFeatureDrawableUri(int featureId, Uri uri) {

getWindow().setFeatureDrawableUri(featureId, uri);

}

public final void setFeatureDrawable(int featureId, Drawable drawable) {

getWindow().setFeatureDrawable(featureId, drawable);

}

public final void setFeatureDrawableAlpha(int featureId, int alpha) {

getWindow().setFeatureDrawableAlpha(featureId, alpha);

}

public LayoutInflater getLayoutInflater() {

return getWindow().getLayoutInflater();

}

/\*\*

\*各种监听器

\*\*/

public void setOnCancelListener(final OnCancelListener listener) {

if (mCancelAndDismissTaken != null) {

throw new IllegalStateException(

"OnCancelListener is already taken by "

+ mCancelAndDismissTaken + " and can not be replaced.");

}

if (listener != null) {

mCancelMessage = mListenersHandler.obtainMessage(CANCEL, listener);

} else {

mCancelMessage = null;

}

}

public void setOnDismissListener(final OnDismissListener listener) {

if (mCancelAndDismissTaken != null) {

throw new IllegalStateException(

"OnDismissListener is already taken by "

+ mCancelAndDismissTaken + " and can not be replaced.");

}

if (listener != null) {

mDismissMessage = mListenersHandler.obtainMessage(DISMISS, listener);

} else {

mDismissMessage = null;

}

}

public void setOnShowListener(OnShowListener listener) {

if (listener != null) {

mShowMessage = mListenersHandler.obtainMessage(SHOW, listener);

} else {

mShowMessage = null;

}

}

/\*\*

\* Sets the callback that will be called if a key is dispatched to the dialog.

\*/

public void setOnKeyListener(final OnKeyListener onKeyListener) {

mOnKeyListener = onKeyListener;

}

private static final class ListenersHandler extends Handler {

private WeakReference<DialogInterface> mDialog;

public ListenersHandler(Dialog dialog) {

mDialog = new WeakReference<DialogInterface>(dialog);

}

@Override

public void handleMessage(Message msg) {

switch (msg.what) {

case DISMISS:

((OnDismissListener) msg.obj).onDismiss(mDialog.get());

break;

case CANCEL:

((OnCancelListener) msg.obj).onCancel(mDialog.get());

break;

case SHOW:

((OnShowListener) msg.obj).onShow(mDialog.get());

break;

}

}

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55
* 56
* 57
* 58
* 59
* 60
* 61
* 62
* 63
* 64
* 65
* 66
* 67
* 68
* 69
* 70
* 71
* 72
* 73
* 74
* 75
* 76
* 77
* 78
* 79
* 80
* 81
* 82
* 83
* 84
* 85
* 86
* 87
* 88
* 89
* 90
* 91
* 92
* 93
* 94
* 95
* 96
* 97
* 98
* 99
* 100
* 101
* 102
* 103
* 104
* 105
* 106
* 107
* 108
* 109
* 110
* 111
* 112
* 113
* 114
* 115
* 116
* 117
* 118
* 119
* 120
* 121
* 122
* 123
* 124
* 125
* 126
* 127
* 128
* 129
* 130
* 131
* 132
* 133
* 134
* 135
* 136
* 137
* 138
* 139
* 140
* 141
* 142
* 143
* 144
* 145
* 146
* 147
* 148
* 149
* 150
* 151
* 152
* 153
* 154
* 155
* 156
* 157
* 158
* 159
* 160
* 161
* 162
* 163
* 164
* 165
* 166
* 167
* 168
* 169
* 170
* 171
* 172
* 173
* 174
* 175
* 176
* 177
* 178
* 179
* 180
* 181
* 182
* 183
* 184
* 185
* 186
* 187
* 188
* 189
* 190
* 191
* 192
* 193
* 194
* 195
* 196
* 197
* 198
* 199
* 200
* 201
* 202
* 203
* 204
* 205
* 206
* 207
* 208
* 209
* 210
* 211
* 212
* 213
* 214
* 215
* 216
* 217
* 218
* 219
* 220
* 221
* 222
* 223
* 224
* 225
* 226
* 227
* 228
* 229
* 230
* 231
* 232
* 233
* 234
* 235
* 236
* 237
* 238
* 239
* 240
* 241
* 242
* 243
* 244
* 245
* 246
* 247
* 248
* 249
* 250
* 251
* 252
* 253
* 254
* 255
* 256
* 257
* 258
* 259
* 260
* 261
* 262
* 263
* 264
* 265
* 266
* 267
* 268
* 269
* 270
* 271
* 272
* 273
* 274
* 275
* 276
* 277
* 278
* 279
* 280
* 281
* 282
* 283
* 284
* 285
* 286
* 287
* 288
* 289
* 290
* 291
* 292
* 293
* 294
* 295
* 296
* 297
* 298
* 299
* 300
* 301
* 302
* 303
* 304
* 305
* 306
* 307
* 308
* 309
* 310
* 311
* 312
* 313
* 314
* 315
* 316
* 317
* 318
* 319
* 320
* 321
* 322
* 323
* 324
* 325
* 326
* 327
* 328
* 329
* 330
* 331
* 332
* 333
* 334
* 335
* 336
* 337
* 338
* 339
* 340
* 341
* 342
* 343
* 344
* 345
* 346
* 347
* 348
* 349
* 350
* 351
* 352
* 353
* 354
* 355
* 356
* 357
* 358
* 359
* 360
* 361
* 362
* 363
* 364
* 365
* 366
* 367
* 368
* 369
* 370
* 371
* 372
* 373
* 374
* 375
* 376
* 377
* 378
* 379
* 380
* 381
* 382
* 383
* 384
* 385
* 386
* 387
* 388
* 389
* 390
* 391
* 392
* 393
* 394
* 395
* 396
* 397
* 398
* 399
* 400
* 401
* 402
* 403
* 404
* 405
* 406
* 407
* 408
* 409
* 410
* 411
* 412
* 413
* 414
* 415
* 416
* 417
* 418
* 419
* 420
* 421
* 422
* 423
* 424
* 425
* 426
* 427
* 428
* 429
* 430
* 431
* 432
* 433
* 434
* 435
* 436
* 437
* 438
* 439
* 440
* 441
* 442
* 443
* 444
* 445
* 446
* 447
* 448
* 449
* 450
* 451
* 452
* 453
* 454
* 455
* 456
* 457
* 458
* 459
* 460
* 461
* 462
* 463
* 464
* 465
* 466
* 467
* 468
* 469
* 470
* 471
* 472
* 473
* 474
* 475
* 476
* 477
* 478
* 479
* 480
* 481
* 482
* 483
* 484
* 485
* 486
* 487
* 488

**二、扩展系统Dialog步骤**

* **定义Dialog将要显示的布局xml文件**
* **继承Dialog 并实现相关的构造方法**
* **重写相关父类的生命周期方法，比如说在onCreate里通过LayoutInflater的LayoutInflater对象的inflate方法把xml布局文件映射成为自定义的view，获取我们自定义对话框的view，然后利用setContentView**

**1、实现普通的自定义Dialog**

定义自定义Dialog 的布局xml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<TextView

style="@style/TitleStyle"

android:layout\_width="match\_parent"

android:padding="5dp"

android:layout\_height="wrap\_content"

android:gravity="center"

android:text="自定义的Dialog"

android:drawableLeft="@mipmap/ic\_red\_launcher"/>

<ProgressBar

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" />

</LinearLayout>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18

继承Dialog并实现相关构造方法并重写onCreate：

/\*\*

\* Created by cmo on 16-5-5.

\*/

public class MyCostomDialog extends Dialog {

Context mContext;

public MyCostomDialog(Context context) {

super(context);

mContext=context;

}

/\*\*

\*另一种形式的使用Theme

\* public MyCostomDialog(Context context){

super(context,R.style.myCustomdialog);

mContext=context;

}

\*/

//使用到Theme时调用

public MyCostomDialog(Context context, int theme) {

super(context, theme);

mContext = context;

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

LayoutInflater inflater = (LayoutInflater) mContext

.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

View layout = inflater.inflate(R.layout.dialog\_my\_costom, null);

this.setContentView(layout);

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30

styles.xml:

<resources>

<!-- Base application theme. -->

<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>

</style>

<style name="TitleStyle">

<item name="android:textColor">@color/colorGreen</item>

<item name="android:textSize">@dimen/activity\_title\_size</item>

</style>

<!--

android:windowFrame:界面对应的前景图片；

android:background:背景图片

android:windowIsFloating:表示浮在屏幕上的，如果在这里使用了，整个layout就会在 屏幕中心，相当于浮在屏幕上，所以这个只适用于dialog

android:windowContentOverlay:表示标题栏的阴影部分的样式，使用图片或者颜色

android:windowNoTitle:标题栏是否隐藏

-->

<style name="myCustomdialog" parent="android:Theme.Dialog">

<item name="android:windowFrame">@null</item>

<item name="android:windowIsFloating">true</item>

<item name="android:windowContentOverlay">@null</item>

<item name="android:windowNoTitle">true</item>

<item name="android:background">@drawable/shape\_dialog\_bcg</item>

</style>

</resources>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28

public class MainActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

//调用代码显示无Theme的Dialog

public void showDialog(View view){

MyCostomDialog dialog = new MyCostomDialog(MainActivity.this);

///MyCostomDialog dialog = new MyCostomDialog(MainActivity.this, R.style.myCustomdialog);显示自定义Theme的的Dialog

dialog.show();

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13

**2、显示具有交互功能的Dialog**

布局很简单就不贴了，既然需要交互，那肯定得注册监听，so 参考Activity上的处理肯定是在界面初始化的时候设置对应的监听

/\*\*

\* Created by cmo on 16-5-5.

\*/

public class MyCostomDialog extends Dialog {

Context mContext;

public MyCostomDialog(Context context) {

super(context);

mContext=context;

}

//使用到Theme时调用

public MyCostomDialog(Context context, int theme) {

super(context, theme);

mContext = context;

}

//设置RadioGroup的监听

private void setRadioListener(final View view){

final RadioGroup radioGroup= (RadioGroup) view.findViewById(R.id.id\_radiogroup);

radioGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener(){

@Override

public void onCheckedChanged(RadioGroup group, int checkedId) {

RadioButton checkedRadio=(RadioButton)findViewById(radioGroup.getCheckedRadioButtonId());

switch (checkedId){

case R.id.id\_android\_radiobtn:

case R.id.id\_ios\_radiobtn:

case R.id.id\_java\_radiobtn:

Log.d("TAG",checkedRadio.getText().toString());

break;

default:

break;

}

}

});

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

LayoutInflater inflater = (LayoutInflater) mContext

.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

View layout = inflater.inflate(R.layout.dialog\_my\_costom, null);

setRadioListener(layout);//设置监听

this.setContentView(layout);

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47

**3、与Activity之间的通信**

**3.1、通过构造函数把Activity里的数据传递至Dialog**

package com.crazymo.costomstyledialog;

/\*\*

\* Created by cmo on 16-5-5.

\*/

public class MyCostomDialog extends Dialog {

Context mContext;

String mText;

public MyCostomDialog(Context context) {

super(context);

mContext=context;

}

//使用到Theme时调用

public MyCostomDialog(Context context, int theme) {

super(context, theme);

mContext = context;

}

//其中txt则是从Activity传递至DIalog的数据

public MyCostomDialog(Context context, String txt,int theme) {

super(context,theme);

mContext = context;

mText=txt;

}

private void setRadioListener(final View view){

final RadioGroup radioGroup= (RadioGroup) view.findViewById(R.id.id\_radiogroup);

radioGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener(){

@Override

public void onCheckedChanged(RadioGroup group, int checkedId) {

RadioButton checkedRadio=(RadioButton)findViewById(radioGroup.getCheckedRadioButtonId());

switch (checkedId){

case R.id.id\_android\_radiobtn:

case R.id.id\_ios\_radiobtn:

case R.id.id\_java\_radiobtn:

Log.d("TAG", checkedRadio.getText().toString());

break;

default:

break;

}

}

});

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

LayoutInflater inflater = (LayoutInflater) mContext

.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

View layout = inflater.inflate(R.layout.dialog\_my\_costom, null);

setRadioListener(layout);

TextView tv = (TextView)layout.findViewById(R.id.id\_dialog\_title\_txt);

tv.setText(mText);//设置Activity传递来的数据

this.setContentView(layout);

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55

在MainActivity里显示：

public void showDialog(View view){

MyCostomDialog dialog = new MyCostomDialog(MainActivity.this, "我来自Activity",R.style.myCustomdialog);

dialog.show();

}

* 1
* 2
* 3
* 4

**3.2、通过回调方式把Dialog里的数据回传至Activity**

package com.crazymo.costomstyledialog;

/\*\*

\* Created by cmo on 16-5-5.

\*/

public class MyCostomDialog extends Dialog {

Context mContext;

String mText;

private IShowPicByIndexListener mIShowPicByIndexListener;//声明一个回调接口函数变量，用于调用

//定义一个回调接口，用于接收Activity返回的值

public interface IShowPicByIndexListener{

public void onShowPicIndex(int index);

}

public MyCostomDialog(Context context) {

super(context);

mContext=context;

}

//使用到Theme时调用

public MyCostomDialog(Context context, int theme) {

super(context, theme);

mContext = context;

}

//把回调也传递到构造方法中

public MyCostomDialog(Context context, String txt,int theme,IShowPicByIndexListener listener) {

super(context,theme);

mContext = context;

mText=txt;

mIShowPicByIndexListener=listener;

}

public MyCostomDialog(Context context, String txt,int theme) {

super(context,theme);

mContext = context;

mText=txt;

}

private void setRadioListener(final View view){

final RadioGroup radioGroup= (RadioGroup) view.findViewById(R.id.id\_radiogroup);

radioGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener(){

@Override

public void onCheckedChanged(RadioGroup group, int checkedId) {

int index=0;

RadioButton checkedRadio=(RadioButton)findViewById(radioGroup.getCheckedRadioButtonId());

switch (checkedId){

case R.id.id\_android\_radiobtn:

index=1;

break;

case R.id.id\_ios\_radiobtn:

index=2;

break;

case R.id.id\_java\_radiobtn:

index=3;

break;

default:

index=0;

}

mIShowPicByIndexListener.onShowPicIndex(index);

}

});

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

LayoutInflater inflater = (LayoutInflater) mContext

.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

View layout = inflater.inflate(R.layout.dialog\_my\_costom, null);

setRadioListener(layout);

TextView tv = (TextView)layout.findViewById(R.id.id\_dialog\_title\_txt);

tv.setText(mText);

this.setContentView(layout);

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55
* 56
* 57
* 58
* 59
* 60
* 61
* 62
* 63
* 64
* 65
* 66
* 67
* 68
* 69
* 70
* 71
* 72
* 73
* 74

package com.crazymo.costomstyledialog;

public class MainActivity extends Activity {

private int[] imgs={R.mipmap.ic\_blue\_launcher,R.mipmap.ic\_red\_launcher,R.mipmap.ic\_toy,R.mipmap.ic\_launcher};

private RadioGroup mRadioGroup;

private ImageView mImageView;

private int index=0;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

init();

}

private void init(){

getViews();

}

private void getViews(){

mImageView= (ImageView) findViewById(R.id.id\_pic\_imgv);

}

public void showDialog(View view){

MyCostomDialog dialog = new MyCostomDialog(MainActivity.this, "我来自Activity", R.style.myCustomdialog, new MyCostomDialog.IShowPicByIndexListener() {

@Override

public void onShowPicIndex(int index) {

mImageView.setImageResource(imgs[index]);

}

});

dialog.show();

}

private void setImageViewBcg(int index){

mImageView.setImageResource(imgs[index]);

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33

**4、自主控制自定义Dialog**

private void showVolumeDialog() {

if ("".equals(AppUtil.getValue(getBaseContext(), ConstantUtil.IP\_CONFIG, ConstantUtil.NAO\_IP, ""))) {

ToastUtil.getInstance().showErrorToast(getBaseContext(), "请先设置NAO的IP地址");

return;

}

voiceIml.getCurrentVolumne();

if (volumDialog == null) {

volumDialog = new Dialog(this);

}

if (volumeDialogView == null) {

volumeDialogView = LayoutInflater.from(this).inflate(R.layout.volume\_dialog\_layout, null, false);

}

SeekBar seekBar = (SeekBar) volumeDialogView.findViewById(R.id.volume\_seek\_bar);

ImageView imvCancel = (ImageView) volumeDialogView.findViewById(R.id.iv\_btn\_cancel);

seekBar.setProgress(VoiceActivity.currVolum);

seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {

@Override

public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {

VoiceActivity.currVolum=progress;

}

@Override

public void onStartTrackingTouch(SeekBar seekBar) {

return;

}

@Override

public void onStopTrackingTouch(SeekBar seekBar) {

voiceIml.sendVolumeToNAO(VoiceActivity.currVolum);

return;

}

});

//取消

imvCancel.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

volumDialog.dismiss();

}

});

volumDialog.setContentView(volumeDialogView);

Window dialogWindow = volumDialog.getWindow();

volumDialog.show();

dialogWindow.setLayout(ScreenUtil.getScreenSize(VoiceActivity.this)[0], ScreenUtil.getScreenSize(VoiceActivity.this)[0] \* 3 / 5+80);

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45

**4.1、动态设置Dialog的显示位置**

对话框默认显示于屏幕中心位置，但是我们可以通过Window对象来动态设置Dialog的位置

public void showDialog(View view){

MyCostomDialog dialog = new MyCostomDialog(MainActivity.this, "我来自Activity", R.style.myCustomdialog, new MyCostomDialog.IShowPicByIndexListener() {

@Override

public void onShowPicIndex(int index) {

mImageView.setImageResource(imgs[index]);

}

});

Window window=dialog.getWindow();//获取Window对象

window.setGravity(Gravity.TOP|Gravity.LEFT);//设置显示位置

dialog.show();

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11

**4.2、改变对话框的透明度**

改变对话框的透明度也可以通过设置Window的alpha属性。

Window window=dialog.getWindow();//先获取Window对象

WindowManager.LayoutParams layoutParams=window.getAttributes();//再获取属性集

layoutParams.alpha=0.6f;//设置alpha属性

window.setAttributes(layoutParams);

* 1
* 2
* 3
* 4

**4.3、自主控制对话框的关闭状态**

public void showDialog(View view){

MyCostomDialog dialog = new MyCostomDialog(MainActivity.this, "我来自Activity", R.style.myCustomdialog, new MyCostomDialog.IShowPicByIndexListener() {

@Override

public void onShowPicIndex(int index) {

mImageView.setImageResource(imgs[index]);

}

});

Window window=dialog.getWindow();

window.setGravity(Gravity.TOP | Gravity.LEFT);

WindowManager.LayoutParams layoutParams=window.getAttributes();

layoutParams.alpha=0.6f;

window.setAttributes(layoutParams);

try {

Field field=dialog.getClass().getSuperclass().getDeclaredField("mShowing");

field.setAccessible(true);

try {

field.set(dialog,false);

} catch (IllegalAccessException e) {

e.printStackTrace();

}

} catch (NoSuchFieldException e) {

e.printStackTrace();

}

dialog.show();

}