# **Toast自定义使用**

## 1.Toast源码分析

老规矩，我们先去看Toast的源码。

Toast有两种显示布局方式，一种最常见调用Toast.makeText(),看源码是这样写的

public static Toast makeText(Context context, CharSequence text, @Duration int duration) {

Toast result = new Toast(context);

LayoutInflater inflate = (LayoutInflater)

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

View v = inflate.inflate(com.android.internal.R.layout.transient\_notification, null);

TextView tv = (TextView)v.findViewById(com.android.internal.R.id.message);

tv.setText(text);

result.mNextView = v;

result.mDuration = duration;

return result;

}

transient\_notification这个布局文件代码是这样的

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"android:layout\_width="match\_parent"android:layout\_height="match\_parent"android:orientation="vertical"android:background="?android:attr/toastFrameBackground">

<TextViewandroid:id="@android:id/message"android:layout\_width="wrap\_content"android:layout\_height="wrap\_content"android:layout\_weight="1"android:layout\_gravity="center\_horizontal"android:textAppearance="@style/TextAppearance.Toast"android:textColor="@color/bright\_foreground\_dark"android:shadowColor="#BB000000"android:shadowRadius="2.75"

/>

</LinearLayout>

那么我们想要修改Toast的文字消息样式，其实就是修改Toast根布局和message这个TextView。

Toast的另外一种显示模式就是自定义布局显示。这个方法不调用Toast.makeText()方法，而是new一个Toast对象，然后调用setView()方法。当然自定义布局就不会加载transient\_notification布局了。

## 2.实现花式Toast

先给大家看下我封装的工具类ToastUtil。

import android.content.Context;import android.view.View;import android.widget.LinearLayout;import android.widget.TextView;import android.widget.Toast;

/\*\*

\* Created by 赵晨璞 on 2016/8/11.

\*/public class ToastUtil {

private Toast toast;private LinearLayout toastView;

/\*\*

\* 修改原布局的Toast

\*/public ToastUtil() {

}

/\*\*

\* 完全自定义布局Toast

\* @param context

\* @param view

\*/public ToastUtil(Context context, View view,int duration){

toast=new Toast(context);

toast.setView(view);

toast.setDuration(duration);

}

/\*\*

\* 向Toast中添加自定义view

\* @param view

\* @param postion

\* @return

\*/public ToastUtil addView(View view,int postion) {

toastView = (LinearLayout) toast.getView();

toastView.addView(view, postion);

return this;

}

/\*\*

\* 设置Toast字体及背景颜色

\* @param messageColor

\* @param backgroundColor

\* @return

\*/public ToastUtil setToastColor(int messageColor, int backgroundColor) {

View view = toast.getView();

if(view!=null){

TextView message=((TextView) view.findViewById(android.R.id.message));

message.setBackgroundColor(backgroundColor);

message.setTextColor(messageColor);

}

return this;

}

/\*\*

\* 设置Toast字体及背景

\* @param messageColor

\* @param background

\* @return

\*/public ToastUtil setToastBackground(int messageColor, int background) {

View view = toast.getView();

if(view!=null){

TextView message=((TextView) view.findViewById(android.R.id.message));

message.setBackgroundResource(background);

message.setTextColor(messageColor);

}

return this;

}

/\*\*

\* 短时间显示Toast

\*/public ToastUtil Short(Context context, CharSequence message){

if(toast==null||(toastView!=null&&toastView.getChildCount()>1)){

toast= Toast.makeText(context, message, Toast.LENGTH\_SHORT);

toastView=null;

}else{

toast.setText(message);

toast.setDuration(Toast.LENGTH\_SHORT);

}

return this;

}

/\*\*

\* 短时间显示Toast

\*/public ToastUtil Short(Context context, int message) {

if(toast==null||(toastView!=null&&toastView.getChildCount()>1)){

toast= Toast.makeText(context, message, Toast.LENGTH\_SHORT);

toastView=null;

}else{

toast.setText(message);

toast.setDuration(Toast.LENGTH\_SHORT);

}

return this;

}

/\*\*

\* 长时间显示Toast

\*/public ToastUtil Long(Context context, CharSequence message){

if(toast==null||(toastView!=null&&toastView.getChildCount()>1)){

toast= Toast.makeText(context, message, Toast.LENGTH\_LONG);

toastView=null;

}else{

toast.setText(message);

toast.setDuration(Toast.LENGTH\_LONG);

}

return this;

}

/\*\*

\* 长时间显示Toast

\*

\* @param context

\* @param message

\*/public ToastUtil Long(Context context, int message) {

if(toast==null||(toastView!=null&&toastView.getChildCount()>1)){

toast= Toast.makeText(context, message, Toast.LENGTH\_LONG);

toastView=null;

}else{

toast.setText(message);

toast.setDuration(Toast.LENGTH\_LONG);

}

return this;

}

/\*\*

\* 自定义显示Toast时间

\*

\* @param context

\* @param message

\* @param duration

\*/public ToastUtil Indefinite(Context context, CharSequence message, int duration) {

if(toast==null||(toastView!=null&&toastView.getChildCount()>1)){

toast= Toast.makeText(context, message,duration);

toastView=null;

}else{

toast.setText(message);

toast.setDuration(duration);

}

return this;

}

/\*\*

\* 自定义显示Toast时间

\*

\* @param context

\* @param message

\* @param duration

\*/public ToastUtil Indefinite(Context context, int message, int duration) {

if(toast==null||(toastView!=null&&toastView.getChildCount()>1)){

toast= Toast.makeText(context, message,duration);

toastView=null;

}else{

toast.setText(message);

toast.setDuration(duration);

}

return this;

}

/\*\*

\* 显示Toast

\* @return

\*/public ToastUtil show (){

toast.show();

return this;

}

/\*\*

\* 获取Toast

\* @return

\*/public Toast getToast(){

return toast;

}

}

修改Toast背景色的使用法方法如下：

ToastUtil toastUtil=new ToastUtil();

toastUtil.Short(MainActivity.this,"自定义message字体、背景色").setToastColor(Color.WHITE, getResources().getColor(R.color.colorAccent)).show();

方形的Toast看上去有些呆板，我自定义了一个名为toast\_radius.xml的背景，代码如下：

<?xml version="1.0" encoding="utf-8"?><shapexmlns:android="http://schemas.android.com/apk/res/android"android:shape="rectangle"><!-- 填充的颜色 --><solid android:color="#ffc107" />

<!-- android:radius 弧形的半径 --><corners android:radius="20dip" />

</shape>

然后上面设置背景的代码改成:

toastUtil.Short(MainActivity.this,"自定义message字体颜色和背景").setToastBackground(Color.WHITE,R.drawable.toast\_radius).show();

虽然官方认为Toast和Snackbar都应该是短文本的形式，不能包含图标，但是个人感觉加上图标还是挺好玩的...

向Toast中添加图标可以这样:

ImageView toastImage = new ImageView(getApplicationContext());

toastImage.setImageResource(R.mipmap.ic\_launcher);

toastUtil.Short(MainActivity.this,"向Toast添加了一个ImageView").setToastBackground(Color.WHITE,R.drawable.toast\_radius).addView(toastImage,0).show();

如果你想要Toast显示自定义的布局，可以这样：

View view = LayoutInflater.from(MainActivity.this).inflate(R.layout.image,null);

new ToastUtil(MainActivity.this,view,Toast.LENGTH\_SHORT).show();

大家都知道，连续触发Toast的show()方法的时候，Toast就会排着队连续展示，感觉上不太友好。所以我先判断了toast是否没被创建或者是否被添加了额外的view，如果是的话就重新生成一个toast对象；如果否的话就只修改message文字和显示时间。