一.单多选控件

1.样式

1.名医/点评 主科室/次科室 横向滚动单选按钮控件HorizontalRadioLayoutImpl

HorizontalRadioLayoutImpl这个控件在多个地方都有使用到，比如 CommonHorizontalRadioWidget ，各大页面类似于tabLayout的效果都是这么实现的。

2.横向滚动单选CommonHorizontalRadioWidget

3.多选

这是一个列表BaseCheckSelectorAdapter，列表里面是多选按钮。都是利用 CheckSelectorProxy 与 单选多选 ICheckSelector ， IRadioSelector 进行实现。

CommonCheckFlowLayout：

2.单选实现【代码】

1.基类

1.ISelected

要实现单选的控件，实体类一般要继承该对象

```
public class ISelected {

    private transient boolean isSelected; // 是否选中 - 非json字段

    public boolean isSelected() {
        return isSelected;
    }

    public void setSelected(boolean selected) {
        isSelected = selected;
    }
}
```

2.IRadioSelector

单选接口，我们控件一般要实现该接口来实现我们自己的单选逻辑，也可以利用 已有代理控件进行实现【RadioSelectorProxy】，后面会有实现参考

```
package cn.kidyn.qdmedical160.nybase.view.selector;

import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.util.List;

import androidx.annotation.IntDef;
import androidx.annotation.Nullable;

import static
cn.kidyn.qdmedical160.nybase.view.selector.IRadioSelector.Mode.CANCEL;
import static
cn.kidyn.qdmedical160.nybase.view.selector.IRadioSelector.Mode.DEFAULT;
import static
cn.kidyn.qdmedical160.nybase.view.selector.IRadioSelector.Mode.REPEAT;

/**
 * 单选逻辑代理接口
 * <p>
 * 使用步骤：todo
 * <p>
 * Created by pengxr on 2020/1/19.
 */
public interface IRadioSelector<T extends ISelected> {

    @Retention(RetentionPolicy.SOURCE)
    @IntDef({DEFAULT, CANCEL, REPEAT})
    @interface Mode {
        int DEFAULT = 1;
        int CANCEL = 2; // 支持反选
        int REPEAT = 3; // 支持重复选中
    }

    /**
     * @param data         数据集
     * @param defaultIndex -1：不默认选中
     */
    void setData(@Nullable List<T> data, int defaultIndex);
```

```
    /**
     * @param data          数据集
     * @param defaultIndex -1：不默认选中
     * @param notify        true：回调defaultIndex选中
     */
    void setData(@Nullable List<T> data, int defaultIndex, boolean notify);

    /**
     * @param index -1:取消选中
     */
    void selectOn(int index);

    /**
     * @param index  -1:取消选中
     * @param notify true：回调
     */
    void selectOn(int index, boolean notify);

    /**
     * 获取当前选中
     */
    @Nullable
    T getSelectedItem();

    /**
     * 获取当前选中索引
     *
     * @return -1：未选中
     */
    int getSelectedIndex();

    /**
     * @return true:拦截选中
     */
    boolean beforeSelect(int index, T t);

    /**
     * 模式
     */
    void setMode(@Mode int mode);

    /**
     * 更新
     */
    void updateAll();

}
```

3.RadioSelectorProxy单选代理逻辑

```
package cn.kidyn.qdmedical160.nybase.view.selector;

import java.util.List;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
```

```
import static
cn.kidyn.qdmedical160.nybase.view.selector.IRadioSelector.Mode.DEFAULT;


/**
 * 单选逻辑代理
 * <p>
 * 1、用于 FlowLayout
 * 2、用于 RecyclerView
 * <p>
 * Created by pengxr on 2020/1/14.
 */
public abstract class RadioSelectorProxy<T extends ISelected> implements
IRadioSelector<T> {

    @Mode
    private int mMode = DEFAULT;

    // 数据集
    private List<T> mData;

    // 当前选中
    private int currentIndex = -1;

    @Override
    public void setData(@Nullable List<T> data, int defaultIndex) {
        setData(data, defaultIndex, true);
    }

    /**
     * @param data         数据集
     * @param defaultIndex 默认选中
     * @param notify       true：回调defaultIndex选中
     */
    @Override
    public void setData(@Nullable List<T> data, int defaultIndex, boolean notify)
{
        if (null == data || data.isEmpty() || defaultIndex >= data.size()) {
            // 无效数据
            onDataSet(data);
            return;
        }

        // 重置状态
        currentIndex = -1;

        // 数据集
        this.mData = data;

        // 默认选中
        if (defaultIndex > -1) {
            for (T t : data) {
                t.setSelected(false);
            }
            T currentItem = data.get(defaultIndex);
            currentItem.setSelected(true);
            currentIndex = defaultIndex;
```

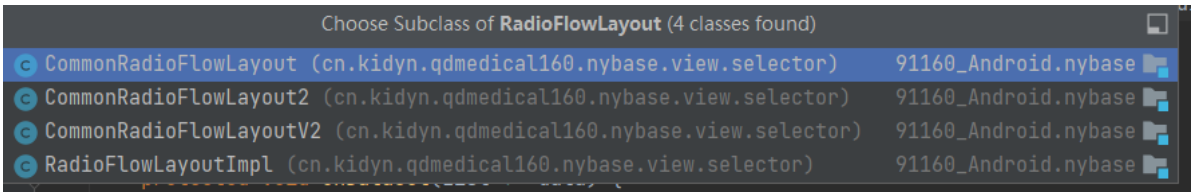
```
        } else {
            for (int index = 0; index < mData.size(); index++) {
                if (mData.get(index).isSelected()) {
                    currentIndex = index;
                    break;
                }
            }
        }
        // 回调
        onDataSet(data);
        // 选中
        if (-1 != currentIndex && notify) { // 使用 defaultIndex，而不是
currentIndex
            onItemSelect(currentIndex, data.get(currentIndex));
        }
    }

    /**
     * @param index -1:取消选中
     */
    @Override
    public void selectOn(int index) {
        selectOn(index, true);
    }

    /**
     * @param index -1:取消选中
     */
    public void selectOn(int index, boolean notify) {
        if (null == mData || index >= mData.size()) {
            return;
        }
        switch (mMode) {
            case Mode.DEFAULT:
                if (currentIndex == index) {
                    // 拦截重复点击
                    return;
                }
                preSelectOn(index, notify);
                break;
            case Mode.CANCEL:
                // 反选
                preSelectOn(currentIndex == index ? -1 : index, notify);
                break;
            case Mode.REPEAT:
                preSelectOn(index, notify);
                break;
        }
    }

    private void preSelectOn(int index, boolean notify) {
        if (-1 != index) {
            T newItem = mData.get(index);
            if (beforeSelect(index, newItem)) {
                // 拦截
                return;
            }
        }
```

```
        if (-1 != index && currentIndex == index && notify) {
            // 重复点击
            T newItem = mData.get(index);
            onItemSelect(index, newItem);
            return;
        }
        doSelectOn(index, notify);
    }

    private void doSelectOn(int index, boolean notify) {
        int oldIndex = currentIndex;
        currentIndex = index;

        if (-1 != index) {
            // 选中
            T newItem = mData.get(index);
            newItem.setSelected(true);
            onItemUpdate(index, newItem);
            if (notify) {
                // 回调
                onItemSelect(index, newItem);
            }
        }

        // 更新
        if (-1 != oldIndex) {
            //反选
            T oldItem = mData.get(oldIndex);
            oldItem.setSelected(false);
            onItemUpdate(oldIndex, oldItem);
            // 反选回调
            if (-1 == index && notify) {
                onItemSelect(oldIndex, null);
            }
        }
    }

    @Override
    public void updateAll() {
        if (null == mData || mData.isEmpty()) {
            return;
        }
        for (int index = 0; index < mData.size(); index++) {
            onItemUpdate(index, mData.get(index));
        }
    }

    @Nullable
    @Override
    public T getSelectedItem() {
        return null == mData || -1 == currentIndex ? null :
mData.get(currentIndex);
    }

    @Override
    public int getSelectedIndex() {
        return currentIndex;
    }
```

```
    @Override
    public void setMode(int mode) {
        this.mMode = mode;
    }

    protected abstract void onDataSet(List<T> data);

    protected abstract void onItemUpdate(int index, @NonNull T t);

    /**
     * @param t null : 反选
     */
    protected abstract void onItemSelect(int index, @Nullable T t);

}
```

4.单选流式布局RadioFlowLayout

这个是结合 流式布局实现的单选控件基类，其他想要实现单选流式的可以继承这个 类进行实现，比如【RadioFlowLayoutImpl，】，



流式布局MyFlowLayout后面解释

```
package cn.kidyn.qdmedical160.nybase.view.selector;

import android.content.Context;
import android.util.AttributeSet;
import android.view.View;

import com.nykj.shareuilib.widget.other.MyFlowLayout;

import java.util.List;

import androidx.annotation.Nullable;

/**
 * 单选FlowLayout
 * <p>
 * Created by pengxr on 2019/11/25.
 */
public abstract class RadioFlowLayout<T extends ISelected> extends MyFlowLayout
implements IRadioSelector<T> {

    private RadioSelectorProxy<T> mProxy = new RadioSelectorProxy<T>() {

        @Override
        protected void onDataSet(List<T> data) {
            removeAllViews();
            if (null == data || data.isEmpty()) {
```

```
                return;
            }
            for (int index = 0; index < data.size(); index++) {
                final int indexTemp = index;
                final T item = data.get(index);
                if (onFilter(item)) {
                    // 过滤
                    continue;
                }
                // 创建
                View itemRoot = onCreateView(item);
                // 绑定
                onUpdateView(itemRoot, item);
                itemRoot.setOnClickListener(new OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        selectOn(indexTemp);
                    }
                });
                addView(itemRoot);
            }
        }

        @Override
        protected void onItemUpdate(int index, T t) {
            View child = getChildAt(index);
            onUpdateView(child, t);
        }

        @Override
        protected void onItemSelect(int index, T t) {
            // 回调
            if (null != mListener) {
                mListener.onSelected(index, t);
            }
        }

        @Override
        public boolean beforeSelect(int index, T t) {
            return RadioFlowLayout.this.beforeSelect(index, t);
        }
    };

    public RadioFlowLayout(Context context) {
        this(context, null);
    }

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

    public RadioFlowLayout(Context context, AttributeSet attrs, int defStyleAttr)
{
        super(context, attrs, defStyleAttr);
    }

    // ---------------------------------------------------------------------------
-----------------
```

```
    // 代理
    // ------------------------------------------------------------------------
-----------------

    public void setData(@Nullable List<T> data, int defaultIndex) {
        mProxy.setData(data, defaultIndex);
    }

    @Override
    public void setData(@Nullable List<T> data, int defaultIndex, boolean notify)
{
        mProxy.setData(data, defaultIndex, notify);
    }

    @Override
    public void selectOn(int index) {
        mProxy.selectOn(index);
    }

    @Override
    public void selectOn(int index, boolean notify) {
        mProxy.selectOn(index, notify);
    }

    @Nullable
    @Override
    public T getSelectedItem() {
        return mProxy.getSelectedItem();
    }

    @Override
    public boolean beforeSelect(int index, T t) {
        return false;
    }

    @Override
    public void setMode(int mode) {
        mProxy.setMode(mode);
    }

    @Override
    public int getSelectedIndex() {
        return mProxy.getSelectedIndex();
    }

    @Override
    public void updateAll() {
        mProxy.updateAll();
    }

    // ------------------------------------------------------------------------
-----------------
    // protected
    // ------------------------------------------------------------------------
-----------------

    protected boolean onFilter(T t) {
        return false;
```

```
    }

    protected abstract View onCreateView(T t);

    protected abstract void onUpdateView(View itemRoot, T t);

    // ------------------------------------------------------------------------
-----------------

    private RadioFlowLayoutListener<T> mListener;

    public void setListener(RadioFlowLayoutListener<T> listener) {
        mListener = listener;
    }

    public interface RadioFlowLayoutListener<T> {

        void onSelected(int index, T e);

    }
}
```

2.实现类

1.RadioFlowLayoutImpl 不可滚动单选实现类

1.源码

2.效果


```
<!-- 医坛星秀模块-->
<com.ny.mqttuikit.widget.BoldTextView
    android:id="@+id/tv_doctor_star_title"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="@dimen/dimen_15dp"
    android:layout_marginTop="@dimen/dimen_40dp"
    android:includeFontPadding="false"
    android:lines="1"
    android:text="@string/txt_doctor_star"
    android:textColor="@color/colorOnPrimary"
    android:textSize="22sp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/banner_loop_ad_ask_doctor" />

<cn.kidyn.qdmedical160.nybase.view.selector.RadioFlowLayoutImpl
    android:id="@+id/radio_group_star_level"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="@+id/tv_doctor_star_title"
    app:layout_constraintLeft_toRightOf="@+id/tv_doctor_star_title"
    app:layout_constraintRight_toLeftOf="@+id/tv_doctor_star_more"
    app:layout_constraintTop_toTopOf="@+id/tv_doctor_star_title" />

<TextView
    android:id="@+id/tv_doctor_star_more"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginRight="@dimen/dimen_15dp"
```

```
    android:drawableRight="@drawable/icon_right_arrow"
    android:drawablePadding="6dp"
    android:drawableTint="@color/colorSecondaryVariant"
    android:includeFontPadding="false"
    android:text="@string/more"
    android:textColor="@color/colorSecondaryVariant"
    android:textSize="14sp"
    app:layout_constraintBottom_toBottomOf="@id/tv_doctor_star_title"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="@id/tv_doctor_star_title" />
```

使用:

```
// 医坛星秀级别
binding.radioGroupStarLevel.setInjector(new
RadioFlowLayoutImpl.Injector<AskDoctorStarLevelItem>() {
    @Override
    public View onCreateView(AskDoctorStarLevelItem item) {
        View itemRoot =
LayoutInflater.from(getContext()).inflate(R.layout.tv_radio_selection_tab,
binding.radioGroupStarLevel, false);
        String text = item.getArea_name();
        ((TextView) itemRoot.findViewById(R.id.tv_tab)).setText(text);
        return itemRoot;
    }

    @Override
    public void onUpdateView(View itemRoot, AskDoctorStarLevelItem item) {
        itemRoot.setSelected(item.isSelected());
        if (item.isSelected()) {

itemRoot.findViewById(R.id.iv_indicator).setVisibility(View.VISIBLE);
            ((TextView)
itemRoot.findViewById(R.id.tv_tab)).setTextSize(TypedValue.COMPLEX_UNIT_SP, 16);
            ((TextView)
itemRoot.findViewById(R.id.tv_tab)).setTypeface(Typeface.DEFAULT_BOLD);
        } else {

itemRoot.findViewById(R.id.iv_indicator).setVisibility(View.INVISIBLE);
            ((TextView)
itemRoot.findViewById(R.id.tv_tab)).setTextSize(TypedValue.COMPLEX_UNIT_SP, 14);
            ((TextView)
itemRoot.findViewById(R.id.tv_tab)).setTypeface(Typeface.DEFAULT);
        }
    }

    @Override
    public boolean onFilter(AskDoctorStarLevelItem item) {
        return false;
    }
});
binding.radioGroupStarLevel.setListener(new
RadioFlowLayout.RadioFlowLayoutListener<AskDoctorStarLevelItem>() {
    @Override
    public void onSelected(int index, AskDoctorStarLevelItem level) {
        getAskDoctorViewModel().fetchDoctorStar(getContext(), level, true);
        binding.tipStar.setTag(true);
```

```
    }
});
```

2.不可滚动通用单选控件CommonRadioFlowLayout

1.源码

这里还有其他另外两个相同的类，CommonRadioFlowLayout2，CommonRadioFlowLayoutV2，他们的区别只是 onCreateView的不同，可以放在BindViewInjector 中将按钮的样式交给 子类去实现处理。

```
package cn.kidyn.qdmedical160.nybase.view.selector;

import android.content.Context;
import androidx.annotation.Nullable;
import android.util.AttributeSet;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.TextView;

import cn.kidyn.qdmedical160.nybase.R;


/**
 * 通用单选样式
 *
 * 使用步骤:
 *
 * 1、见 RadioFlowLayout（抽象类） 的使用步骤
 * 2、实现 BindViewInjector（接口） 进行依赖注入
 *
 * Created by pengxr on 2019/12/19.
 */
public class CommonRadioFlowLayout<T extends ISelected> extends
RadioFlowLayout<T> {

    public CommonRadioFlowLayout(Context context) {
        super(context);
    }

    public CommonRadioFlowLayout(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public CommonRadioFlowLayout(Context context, AttributeSet attrs, int
defStyleAttr) {
        super(context, attrs, defStyleAttr);
    }

    // 依赖注入
    private BindViewInjector<T> mInjector;

    // ------------------------------------------------------------------------
-------------------
    // public
```

```
    // ------------------------------------------------------------------------
------------------

    /**
     * @param injector View绑定注入器
     */
    public void setInjector(@Nullable BindViewInjector<T> injector) {
        this.mInjector = injector;
    }

    public interface BindViewInjector<T> {
        String onBind(T item);

        boolean onFilter(T item);
    }

    // ------------------------------------------------------------------------
------------------
    // RadioFlowLayout
    // ------------------------------------------------------------------------
------------------

    @Override
    protected View onCreateView(T item) {
        return
LayoutInflater.from(getContext()).inflate(R.layout.tv_radio_selection_v2, this,
false);
    }

    @Override
    protected void onUpdateView(View itemRoot, T item) {
        if (null == mInjector) {
            return;
        }
        // 依赖注入
        TextView tv = (TextView) itemRoot;
        String str = mInjector.onBind(item);
        tv.setText(str);
        tv.setSelected(item.isSelected());
    }

    @Override
    protected boolean onFilter(T item) {
        if (null == mInjector) {
            return false;
        }
        // 依赖注入
        return mInjector.onFilter(item);
    }
}
```

2.效果实现


```
<!-- 可下拉 -->
<cn.kidyn.qdmedical160.activity.comment.widget.DropDownFlowLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="@dimen/margin_outer"
    android:layout_marginRight="@dimen/margin_outer"
    android:layout_marginTop="@dimen/margin_outer"
    app:closeLine="2"
    app:layout_constraintTop_toBottomOf="@+id/tv_doctor_comment_list_title">

    <cn.kidyn.qdmedical160.nybase.view.selector.CommonRadioFlowLayout2
        android:id="@+id/container_doctor_comment_list_tag"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:line_divider="@dimen/dimen_10dp"
        app:padding_divider="@dimen/dimen_10dp" />

</cn.kidyn.qdmedical160.activity.comment.widget.DropDownFlowLayout>
```

```
// 依赖注入
mTagContainer.setInjector(new CommonRadioFlowLayout2.BindViewInjector<IllEntity>
() {
    @Override
```

```
    public String onBind(IllEntity item) {
        return item.getIll_name() + " " + item.getCnt();
    }

    @Override
    public boolean onFilter(IllEntity item) {
        return StringUtils.StrIsNull(item.getIll_name()) || 0 == item.getCnt();
    }
});

 mTagContainer.setListener(new RadioFlowLayout.RadioFlowLayoutListener<IllEntity>
() {
            @Override
            public void onSelected(int index, IllEntity e) {
                // 重置标签
                mViewModel.resetIll(e);
            }
        });
```

3.可滚动横向单选HorizontalRadioLayout

1.源码

这个基于 HorizontalScrollView 实现 类型于tabLayout的效果

```
package cn.kidyn.qdmedical160.nybase.view.selector;

import android.content.Context;
import android.content.res.TypedArray;
import android.util.AttributeSet;
import android.util.Log;
import android.view.View;
import android.view.ViewGroup;
import android.widget.HorizontalScrollView;
import android.widget.LinearLayout;

import java.util.List;

import androidx.annotation.Nullable;
import cn.kidyn.qdmedical160.nybase.R;

/**
 * 横向单选
 * <p>
 * Created by pengxr on 2020/12/15.
 */
public abstract class HorizontalRadioLayout<T extends ISelected> extends
HorizontalScrollView implements IRadioSelector<T> {

    private LinearLayout mInnerLayout;

    private int mOutSpace;
    private int mInSpace;

    private boolean mAutoScroll;
```

```
    private RadioSelectorProxy<T> mProxy = new RadioSelectorProxy<T>() {

        @Override
        protected void onDataSet(List<T> data) {
            mInnerLayout.removeAllViews();
            if (null == data || data.isEmpty()) {
                return;
            }
            for (int index = 0; index < data.size(); index++) {
                final int indexTemp = index;
                final T item = data.get(index);
                if (onFilter(item)) {
                    // 过滤
                    continue;
                }
                // 创建
                View itemRoot = onCreateView(item);
                // 绑定
                onUpdateView(itemRoot, item);
                itemRoot.setOnClickListener(new OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        selectOn(indexTemp);
                    }
                });
                MarginLayoutParams layoutParams = (MarginLayoutParams)
itemRoot.getLayoutParams();
                if (0 == index) {
                    // 第一个
                    layoutParams.leftMargin = mOutSpace;
                } else {
                    layoutParams.leftMargin = mInSpace;
                }
                // 最后一个（只有一项item时，即是第一个也是最后一个）
                if (index == data.size() - 1) {
                    layoutParams.rightMargin = mOutSpace;
                }
                mInnerLayout.addView(itemRoot, layoutParams);
            }
        }

        @Override
        protected void onItemUpdate(int index, T t) {
            View child = mInnerLayout.getChildAt(index);
            onUpdateView(child, t);
        }

        @Override
        protected void onItemSelect(int index, T t) {
            // 回调
            if (null != mListener) {
                mListener.onSelected(index, t);
            }
            if (mAutoScroll) {
               post(() -> smoothScrollTo(calculateScrollXForTab(mInnerLayout,
index), 0));
            }
```

```
        }

        @Override
        public boolean beforeSelect(int index, T t) {
            if (mBeforeSelectListener != null) {
                return mBeforeSelectListener.beforeSelect(index, t);
            }
            return HorizontalRadioLayout.this.beforeSelect(index, t);
        }
    };

    public HorizontalRadioLayout(Context context) {
        this(context, null);
    }

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

    public HorizontalRadioLayout(Context context, AttributeSet attrs, int
defStyleAttr) {
        super(context, attrs, defStyleAttr);

        TypedArray array = context.obtainStyledAttributes(attrs,
R.styleable.HorizontalRadioLayout);
        mOutSpace =
array.getDimensionPixelOffset(R.styleable.HorizontalRadioLayout_outSpace, 0);
        mInSpace =
array.getDimensionPixelOffset(R.styleable.HorizontalRadioLayout_inSpace, 0);
        array.recycle();

        mInnerLayout = new LinearLayout(getContext());
        mInnerLayout.setLayoutParams(new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
        mInnerLayout.setOrientation(LinearLayout.HORIZONTAL);


        addView(mInnerLayout);
    }

    // ------------------------------------------------------------------------
-----------------
    // 代理
    // ------------------------------------------------------------------------
-----------------

    public void setData(@Nullable List<T> data, int defaultIndex) {
        mProxy.setData(data, defaultIndex);
    }

    @Override
    public void setData(@Nullable List<T> data, int defaultIndex, boolean notify)
{
        mProxy.setData(data, defaultIndex, notify);
    }

    @Override
```

```
    public void selectOn(int index) {
        mProxy.selectOn(index);
    }

    @Override
    public void selectOn(int index, boolean notify) {
        mProxy.selectOn(index, notify);
    }

    @Nullable
    @Override
    public T getSelectedItem() {
        return mProxy.getSelectedItem();
    }

    @Override
    public boolean beforeSelect(int index, T t) {
        return false;
    }

    @Override
    public void setMode(int mode) {
        mProxy.setMode(mode);
    }

    @Override
    public int getSelectedIndex() {
        return mProxy.getSelectedIndex();
    }

    @Override
    public void updateAll() {
        mProxy.updateAll();
    }

    // ------------------------------------------------------------------------
-----------------
    // protected
    // ------------------------------------------------------------------------
-----------------

    protected boolean onFilter(T t) {
        return false;
    }

    protected abstract View onCreateView(T t);

    protected abstract void onUpdateView(View itemRoot, T t);

    // ------------------------------------------------------------------------
-----------------

    private RadioFlowLayout.RadioFlowLayoutListener<T> mListener;

    public void setListener(RadioFlowLayout.RadioFlowLayoutListener<T> listener)
{
        mListener = listener;
    }
```

```
    private RadioFlowLayoutBeforeSelectListener<T> mBeforeSelectListener;


    public void setBeforeSelectListener(RadioFlowLayoutBeforeSelectListener<T>
beforeSelectListener) {
        this.mBeforeSelectListener = beforeSelectListener;
    }

    /**
     * 选中事件前回调，与RadioFlowLayout.RadioFlowLayoutListener放一起或许会更好，但为不
影响以前代码，再加一个
     */
    public interface RadioFlowLayoutBeforeSelectListener<T> {

        boolean beforeSelect(int index, T e);

    }

    // 参考自TabLayout#calculateScrollXForTab()
    private int calculateScrollXForTab(ViewGroup parent, int position) {
        try {
            View selectedChild = parent.getChildAt(position);
            int selectedWidth = selectedChild != null ? selectedChild.getWidth()
: 0;
            return selectedChild.getLeft() + selectedWidth / 2 - getWidth() / 2;
        }catch (Exception e){
            return 0;
        }
    }

    public void scrollToPosition(int position) {
        smoothScrollTo(calculateScrollXForTab(mInnerLayout, position), 0);
    }

    public void autoScroll(boolean autoScroll) {
        this.mAutoScroll = autoScroll;
    }
}
```

```
package cn.kidyn.qdmedical160.nybase.view.selector;

import android.content.Context;
import androidx.annotation.Nullable;
import android.util.AttributeSet;
import android.view.View;

/**
 * Created by pengxr on 2020/12/15.
 */
public class HorizontalRadioLayoutImpl<T extends ISelected> extends
HorizontalRadioLayout<T> {

    public HorizontalRadioLayoutImpl(Context context) {
```

```
        super(context);
    }

    public HorizontalRadioLayoutImpl(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public HorizontalRadioLayoutImpl(Context context, AttributeSet attrs, int
defStyleAttr) {
        super(context, attrs, defStyleAttr);
    }

    // 依赖注入
    private RadioFlowLayoutImpl.Injector<T> mInjector;

    // ------------------------------------------------------------------------
-------------------
    // public
    // ------------------------------------------------------------------------
-------------------

    /**
     * @param injector View 绑定注入器
     */
    public void setInjector(@Nullable RadioFlowLayoutImpl.Injector<T> injector)
{
        this.mInjector = injector;
    }

    // ------------------------------------------------------------------------
-------------------
    // RadioFlowLayout
    // ------------------------------------------------------------------------
-------------------

    @Override
    protected View onCreateView(T item) {
        if (null == mInjector) {
            throw new IllegalStateException("使用 HorizontalRadioLayoutImpl 必须设
置 Injector");
        }
        return mInjector.onCreateView(item);
    }

    @Override
    protected void onUpdateView(View itemRoot, T item) {
        if (null == mInjector) {
            throw new IllegalStateException("使用 HorizontalRadioLayoutImpl 必须设
置 Injector");
        }
        mInjector.onUpdateView(itemRoot, item);
    }

    @Override
    protected boolean onFilter(T item) {
        if (null == mInjector) {
            throw new IllegalStateException("使用 HorizontalRadioLayoutImpl 必须设
置 Injector");
```

```
        }
        // 依赖注入
        return mInjector.onFilter(item);
    }
}
```

2.效果

使用：

```
<cn.kidyn.qdmedical160.nybase.view.selector.HorizontalRadioLayoutImpl
    android:id="@+id/flow_dep_1"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginRight="42dp"
    android:nestedScrollingEnabled="false"
    app:inSpace="0dp"
    app:layout_constraintBottom_toBottomOf="@+id/iv_more_dep1"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:outSpace="0dp"
```

```
    tools:layout_height="48dp" />

<ImageView
    android:layout_width="wrap_content"
    android:layout_height="0dp"
    android:src="@drawable/icon_shadow_left_ask_doctor"
    app:layout_constraintBottom_toBottomOf="@+id/iv_more_dep1"
    app:layout_constraintRight_toLeftOf="@+id/iv_more_dep1"
    app:layout_constraintTop_toTopOf="@+id/iv_more_dep1" />

<ImageView
    android:id="@+id/iv_more_dep1"
    android:layout_width="41dp"
    android:layout_height="48dp"
    android:background="@color/white"
    android:paddingLeft="10dp"
    android:paddingRight="15dp"
    android:scaleType="center"
    android:src="@drawable/icon_more_dep_1"
    app:layout_constraintRight_toRightOf="parent" />
```

```
mFlowDep1.setInjector(new RadioFlowLayoutImpl.Injector<AskDocHotDepEntity>() {
    @Override
    public View onCreateView(AskDocHotDepEntity item) {
        View itemRoot =
LayoutInflater.from(getContext()).inflate(R.layout.tv_radio_selection_tab,
mFlowDep1, false);
        String text = item.getDisplayName();
        ((TextView) itemRoot.findViewById(R.id.tv_tab)).setText(text);
        return itemRoot;
    }

    @Override
    public void onUpdateView(View itemRoot, AskDocHotDepEntity item) {
        itemRoot.setSelected(item.isSelected());
        if (item.isSelected()) {

itemRoot.findViewById(R.id.iv_indicator).setVisibility(View.VISIBLE);
            ((TextView)
itemRoot.findViewById(R.id.tv_tab)).setTextSize(TypedValue.COMPLEX_UNIT_SP, 18);
            ((TextView)
itemRoot.findViewById(R.id.tv_tab)).setTypeface(Typeface.DEFAULT_BOLD);
        } else {

itemRoot.findViewById(R.id.iv_indicator).setVisibility(View.INVISIBLE);
            ((TextView)
itemRoot.findViewById(R.id.tv_tab)).setTextSize(TypedValue.COMPLEX_UNIT_SP, 16);
            ((TextView)
itemRoot.findViewById(R.id.tv_tab)).setTypeface(Typeface.DEFAULT);
        }
    }

    @Override
    public boolean onFilter(AskDocHotDepEntity item) {
        return false;
    }
});
```

```
mFlowDep1.setListener(new
RadioFlowLayout.RadioFlowLayoutListener<AskDocHotDepEntity>() {
    @Override
    public void onSelected(int index, AskDocHotDepEntity item) {
        mDepDispatcher.onLeftSelect(index, item, true);
    }
});
```

3.进一步封装CommonHorizontalRadioWidget

1.源码

```
package cn.kidyn.qdmedical160.activity.home.widget

import android.content.Context
import android.util.AttributeSet
import android.util.Log
import android.view.LayoutInflater
import androidx.constraintlayout.widget.ConstraintLayout
import cn.kidyn.qdmedical160.R
import cn.kidyn.qdmedical160.databinding.LayoutCommonHorizontalRadioBinding
import cn.kidyn.qdmedical160.nybase.view.selector.IRadioSelector
import cn.kidyn.qdmedical160.nybase.view.selector.ISelected
import
cn.kidyn.qdmedical160.nybase.view.selector.RadioFlowLayout.RadioFlowLayoutListen
er
import cn.kidyn.qdmedical160.nybase.view.selector.RadioFlowLayoutImpl
import com.nykj.shareuilib.temp.viewBinding

/**
 * Created by pengxr on 2021/9/10.
 */
class CommonHorizontalRadioWidget<T : ISelected> @JvmOverloads
constructor(context: Context, attrs: AttributeSet? = null, defStyleAttr: Int = 0)
: ConstraintLayout(
    context, attrs, defStyleAttr
), IRadioSelector<T> {

    private val binding by viewBinding(LayoutCommonHorizontalRadioBinding::bind)

    init {

LayoutInflater.from(context).inflate(R.layout.layout_common_horizontal_radio,
this, true)

        init()
    }

    private fun init() {
        with(binding) {
            ivMore.setOnClickListener {
                flows.smoothScrollTo(0, flows.width)
            }
        }
    }
```

```
    override fun setData(data: MutableList<T>?, defaultIndex: Int) {
        binding.flows.setData(data, defaultIndex)
    }

    override fun setData(data: MutableList<T>?, defaultIndex: Int, notify:
Boolean) {
        binding.flows.setData(data, defaultIndex, notify)
    }

    override fun selectOn(index: Int) {
        binding.flows.selectOn(index)
    }

    override fun selectOn(index: Int, notify: Boolean) {
        binding.flows.selectOn(index, notify)
    }

    override fun getSelectedItem(): T? = binding.flows.getSelectedItem() as T?

    override fun getSelectedIndex() = binding.flows.selectedIndex

    override fun beforeSelect(index: Int, t: T) =
binding.flows.beforeSelect(index, t)

    override fun setMode(mode: Int) = binding.flows.setMode(mode)

    override fun updateAll() = binding.flows.updateAll()

    fun setListener(listener: RadioFlowLayoutListener<T>) {
        binding.flows.setListener(listener)
    }

    fun setInjector(injector: RadioFlowLayoutImpl.Injector<T>?) {
        binding.flows.setInjector(injector)
    }

    fun autoScroll(isAutoScroll: Boolean) {
        binding.flows.autoScroll(isAutoScroll)
    }

    fun scrollToPosition(position:Int){
        binding.flows.scrollToPosition(position)
    }
}
```

2.效果

```
<cn.kidyn.qdmedical160.activity.home.widget.CommonHorizontalRadioWidget
    android:id="@+id/tabs_home_activity"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_weight="1"
    android:paddingLeft="@dimen/dimen_7dp"
    tools:layout_height="50dp" />
```


```
// 选项卡（设计样式花样百出，暂不复用了）
tabsHomeActivity.autoScroll(true)
tabsHomeActivity.setInjector(object : RadioFlowLayoutImpl.Injector<HomeTab> {
    override fun onCreateView(item: HomeTab) =
LayoutInflater.from(requireContext())
            .inflate(R.layout.tv_radio_selection_tab_home, tabsHomeActivity,
false).apply {
                (findViewById<View>(R.id.tv_tab) as TextView).text = item.name
            }

    override fun onUpdateView(itemRoot: View, item: HomeTab) {
        with(itemRoot) {
            isSelected = item.isSelected
            if (item.isSelected) {
                findViewById<View>(R.id.iv_indicator).visibility = View.VISIBLE
                (findViewById<View>(R.id.tv_tab) as TextView).setTextSize(
                        TypedValue.COMPLEX_UNIT_SP, 22F
                )
                (findViewById<View>(
                        R.id.tv_tab
                ) as TextView).typeface = Typeface.DEFAULT_BOLD
            } else {
                findViewById<View>(R.id.iv_indicator).visibility =
View.INVISIBLE
                (findViewById<View>(R.id.tv_tab) as TextView).setTextSize(
                        TypedValue.COMPLEX_UNIT_SP, 16F
                )
                (findViewById<View>(
                        R.id.tv_tab
                ) as TextView).typeface = Typeface.DEFAULT
            }
        }
    }

    override fun onFilter(item: HomeTab): Boolean {
        return false
```

```
    }
})
tabsHomeActivity.setListener { index, item ->
    // 切换 Pager
    selectTabByIndex(index)
    // 吸顶
    binding.appBarHomeActivity.setExpanded(false, true)
}
```

4.结合RecyclerView 实现单选BaseRadioSelectorAdapter

1.基类

```
package cn.kidyn.qdmedical160.nybase.view.recycler.adapter;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import java.util.Collections;
import java.util.List;

import androidx.annotation.LayoutRes;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.recyclerview.widget.RecyclerView;
import cn.kidyn.qdmedical160.nybase.view.selector.IRadioSelector;
import cn.kidyn.qdmedical160.nybase.view.selector.ISelected;
import cn.kidyn.qdmedical160.nybase.view.selector.RadioSelectorProxy;
import me.drakeet.multitype.ItemViewBinder;
import me.drakeet.multitype.MultiTypeAdapter;

/**
 * 帮助实现单选RecyclerView
 * <p>
 * Created by pengxr on 2020/1/17.
 */
public abstract class BaseRadioSelectorAdapter<T extends ISelected> extends
MultiTypeAdapter implements IRadioSelector<T> {

    public BaseRadioSelectorAdapter() {
        register(getType(), new BaseRadioSelectorBinder());
    }

    private RadioSelectorProxy<T> mProxy = new RadioSelectorProxy<T>() {
        @Override
        protected void onDataSet(List<T> data) {
            if (null == data || data.isEmpty()) {
                setItems(Collections.EMPTY_LIST);
            } else {
                setItems(data);
            }
            notifyDataSetChanged();
        }
```

```
        @Override
        protected void onItemUpdate(int index, T item) {
            notifyItemChanged(index, -1);
        }

        @Override
        protected void onItemSelect(int index, T item) {
            if (null != mListener) {
                mListener.onItemSelect(item);
            }
        }

        @Override
        public boolean beforeSelect(int index, T t) {
            return BaseRadioSelectorAdapter.this.beforeSelect(index, t);
        }
    };

    // ---------------------------------------------------------------------------
---------
    // 代理
    // ---------------------------------------------------------------------------
---------

    @Override
    public void setData(@Nullable List<T> data, int defaultIndex) {
        mProxy.setData(data, defaultIndex);
    }

    @Override
    public void setData(List<T> data, int defaultIndex, boolean notify) {
        mProxy.setData(data, defaultIndex, notify);
    }

    public void selectOn(int index) {
        mProxy.selectOn(index);
    }

    @Override
    public void selectOn(int index, boolean notify) {
        mProxy.selectOn(index, notify);
    }

    @Nullable
    public T getSelectedItem() {
        return mProxy.getSelectedItem();
    }

    @Override
    public boolean beforeSelect(int index, T t) {
        if (null != mListener) {
            return mListener.beforeItemSelect(t);
        }
        return false;
    }

    @Override
    public void setMode(int mode) {
```

```
        mProxy.setMode(mode);
    }

    @Override
    public int getSelectedIndex() {
        return mProxy.getSelectedIndex();
    }

    @Override
    public void updateAll() {
        mProxy.updateAll();
    }

    // ------------------------------------------------------------------------
---------
    // protected
    // ------------------------------------------------------------------------
---------

    protected abstract Class<T> getType();

    protected abstract BaseListItemHolderImpl generateHolder(View itemRoot);

    @LayoutRes
    protected abstract int getLayoutRes();

    // ------------------------------------------------------------------------
----------------

    private class BaseRadioSelectorBinder extends ItemViewBinder<T,
BaseListItemHolderImpl> {

        @NonNull
        @Override
        protected BaseListItemHolderImpl onCreateViewHolder(@NonNull
LayoutInflater inflater, @NonNull ViewGroup parent) {
            View itemRoot = inflater.inflate(getLayoutRes(), parent, false);
            return generateHolder(itemRoot);
        }

        @Override
        protected void onBindViewHolder(@NonNull BaseListItemHolderImpl holder,
@NonNull T t) {
            holder.bind(t);
        }
    }

    // ------------------------------------------------------------------------
----------------

    public abstract class BaseListItemHolderImpl extends RecyclerView.ViewHolder
{

        protected T mItem;

        public BaseListItemHolderImpl(@NonNull View itemView) {
            super(itemView);
            itemView.setOnClickListener(new View.OnClickListener() {
```

```
                @Override
                public void onClick(View v) {
                    if (null != mItem) {
                        onItemClick(getAdapterPosition(), mItem);
                    }
                }
            });
        }

        public void bind(T item) {
            mItem = item;
            onItemBind(getAdapterPosition(), item);
        }

        protected abstract void onItemBind(int index, T item);

        protected void onItemClick(int index, T item) {
            selectOn(index);
        }
    }

    // ------------------------------------------------------------------------
----------------

    private OnBaseRadioSelectorAdapterListener<T> mListener;

    public void setListener(OnBaseRadioSelectorAdapterListener<T> listener) {
        mListener = listener;
    }

    public interface OnBaseRadioSelectorAdapterListener<T> {

        void onItemSelect(T item);

        boolean beforeItemSelect(T item);

    }
}
```

2.使用效果

基本大部分下拉单选框

使用：

```
static class SortAdapter extends
BaseRadioSelectorAdapter<MPopularSciencTabEntity.SortItem> {

    @Override
    protected Class<MPopularSciencTabEntity.SortItem> getType() {
        return MPopularSciencTabEntity.SortItem.class;
    }

    @Override
    protected BaseListItemHolderImpl generateHolder(View itemRoot) {
        return new SortHolder(itemRoot);
    }

    @Override
    protected int getLayoutRes() {
        return R.layout.recycler_item_hospital_list_line_selection;
    }

    class SortHolder extends BaseListItemHolderImpl {

        SortHolder(@NonNull View itemView) {
            super(itemView);
        }

        @Override
        protected void onItemBind(int index, MPopularSciencTabEntity.SortItem
item) {
            ((TextView) itemView).setText(item.getName());
            itemView.setSelected(item.isSelected());
        }
    }

    void onReset(){
        selectOn(0);
    }
}
```

5.单选RecyclerView 封装第2种 BaseRadioSelectorAdapterV2

1.基类

这个实际跟 BaseRadioSelectorAdapter差不多，只不过增加了点击选中的View，这个view由子类返回实现。

```
package cn.kidyn.qdmedical160.nybase.view.recycler.adapter;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import java.util.Collections;
import java.util.List;

import androidx.annotation.LayoutRes;
import androidx.annotation.NonNull;
```

```
import androidx.annotation.Nullable;
import androidx.recyclerview.widget.RecyclerView;
import cn.kidyn.qdmedical160.nybase.util.LoaderUtil;
import cn.kidyn.qdmedical160.nybase.view.selector.IRadioSelector;
import cn.kidyn.qdmedical160.nybase.view.selector.ISelected;
import cn.kidyn.qdmedical160.nybase.view.selector.RadioSelectorProxy;
import me.drakeet.multitype.ItemViewBinder;
import me.drakeet.multitype.MultiTypeAdapter;

/**
 * 帮助实现单选RecyclerView
 * <p>
 * 原来那个的 API 封装不理想，使用地方较多暂不修改
 * <p>
 * <p>
 * todo : 感觉这几个Radio工具的接口可以再优化下
 * Created by pengxr on 2020/12/2.
 */
public abstract class BaseRadioSelectorAdapterV2<T extends ISelected> extends
MultiTypeAdapter implements IRadioSelector<T> {

    public BaseRadioSelectorAdapterV2() {
        register((Class) LoaderUtil.getGenericTypeForSuper(this, 0),
                new BaseRadioSelectorAdapterV2.BaseRadioSelectorBinder());
    }

    private RadioSelectorProxy<T> mProxy = new RadioSelectorProxy<T>() {
        @Override
        protected void onDataSet(List<T> data) {
            if (null == data || data.isEmpty()) {
                setItems(Collections.EMPTY_LIST);
            } else {
                setItems(data);
            }
            notifyDataSetChanged();
        }

        @Override
        protected void onItemUpdate(int index, @NonNull T item) {
            notifyItemChanged(index, -1);
        }

        @Override
        protected void onItemSelect(int index, @Nullable T item) {
            if (null != mListener) {
                mListener.onItemSelect(index, item);
            }
        }

        @Override
        public boolean beforeSelect(int index, T t) {
            return BaseRadioSelectorAdapterV2.this.beforeSelect(index, t);
        }
    };

    // ------------------------------------------------------------------------
---------
    // 代理
```

```
    // ------------------------------------------------------------------------
---------

    @Override
    public void setData(@Nullable List<T> data, int defaultIndex) {
        mProxy.setData(data, defaultIndex);
    }

    public void setData(List<T> data, int defaultIndex, boolean notify) {
        mProxy.setData(data, defaultIndex, notify);
    }

    public void selectOn(int index) {
        mProxy.selectOn(index);
    }

    @Override
    public void selectOn(int index, boolean notify) {
        mProxy.selectOn(index, notify);
    }

    @Nullable
    public T getSelectedItem() {
        return mProxy.getSelectedItem();
    }

    @Override
    public boolean beforeSelect(int index, T t) {
        if (null != mListener) {
            return mListener.beforeItemSelect(t);
        }
        return false;
    }

    @Override
    public void setMode(int mode) {
        mProxy.setMode(mode);
    }

    @Override
    public int getSelectedIndex() {
        return mProxy.getSelectedIndex();
    }

    @Override
    public void updateAll() {
        mProxy.updateAll();
    }

    // ------------------------------------------------------------------------
---------
    // protected
    // ------------------------------------------------------------------------
---------

    protected abstract BaseRadioSelectorAdapterV2.BaseRadioHolder
generateHolder(View itemRoot);
```

```
    @LayoutRes
    protected abstract int getLayoutRes();

    // ----------------------------------------------------------------------------
----------------

    private class BaseRadioSelectorBinder extends ItemViewBinder<T,
BaseRadioSelectorAdapterV2.BaseRadioHolder> {

        @NonNull
        @Override
        protected BaseRadioSelectorAdapterV2.BaseRadioHolder
onCreateViewHolder(@NonNull LayoutInflater inflater, @NonNull ViewGroup parent)
{
            View itemRoot = inflater.inflate(getLayoutRes(), parent, false);
            return generateHolder(itemRoot);
        }

        @Override
        protected void onBindViewHolder(@NonNull
BaseRadioSelectorAdapterV2.BaseRadioHolder holder, @NonNull T t) {
            holder.bind(t);
        }
    }

    // ----------------------------------------------------------------------------
----------------

    public abstract class BaseRadioHolder extends RecyclerView.ViewHolder {

        protected T mItem;

        public BaseRadioHolder(@NonNull View itemView) {
            super(itemView);

            getTouchAnchor(itemView).setOnClickListener(new
View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    if (null != mItem) {
                        // 选中
                        selectOn(getAdapterPosition());
                    }
                }
            });
        }

        public void bind(T item) {
            if (mItem == item) {
                onItemToggle(getAdapterPosition(), item);
            } else {
                mItem = item;

                onItemBind(getAdapterPosition(), item);
                onItemToggle(getAdapterPosition(), item);
            }
        }
```

```
        /**
         * 数据绑定
         */
        protected abstract void onItemBind(int index, T item);

        /**
         * 选中切换
         */
        protected abstract void onItemToggle(int index, T item);

        /**
         * 点击瞄点
         */
        protected View getTouchAnchor(View root) {
            return root;
        }
    }

    // ------------------------------------------------------------------------
----------------

    private Listener<T> mListener;

    public void setListener(Listener<T> listener) {
        mListener = listener;
    }

    public interface Listener<T> {

        void onItemSelect(int index, T item);

        boolean beforeItemSelect(T item);

    }
}
```

2.LoaderUtil

```
package cn.kidyn.qdmedical160.nybase.util;

import androidx.annotation.Nullable;

import java.lang.reflect.ParameterizedType;
import java.lang.reflect.Type;

/**
 * Create by liangy on 2019/12/11
 */
public class LoaderUtil {

    /**
     * 获取一级类型实参
     * 例如：List<CommentItem>，将返回List<CommentItem>.class
     */
    @Nullable
    public static Type getGenericType(@Nullable Object object, int argIndex) {
        if(null == object){
```

```
            return null;
        }
        // Type可以兼容Object<T>和Object<Object<T>>，例如PageEntity<CommentItem>
        Type[] superClazz = object.getClass().getGenericInterfaces();
        return ((ParameterizedType) superClazz[0]).getActualTypeArguments()
[argIndex];
    }

    public static Type getGenericTypeForSuper(Object object ,int argIndex){
        return ((ParameterizedType)
(object.getClass().getGenericSuperclass())).getActualTypeArguments()[argIndex];
    }

    /**
     * 获取二级类型实参
     * 例如：List<CommentItem>，将返回CommentItem.class
     */
    public static Type getGenericType(Object object, int argIndex1, int
argIndex2) {
        Type first = getGenericType(object, argIndex1);
        return ((ParameterizedType) first).getActualTypeArguments()[argIndex2];
    }

}
```

3.使用效果

#1.AppointSchAdapter:

```
class AppointSchAdapter extends BaseRadioSelectorAdapterV2<DocHomeUnit> {

 @Override
    protected BaseRadioHolder generateHolder(View itemRoot) {
        return new Holder(itemRoot);
    }

    @Override
    protected int getLayoutRes() {
        return R.layout.doc_home_appoint_sch_unit;
    }


}
```

#2.RightAdapter

使用:

```
private class RightAdapter extends
BaseRadioSelectorAdapterV2<AskDocHotDepEntity.Child> {

    @Override
    protected BaseRadioHolder generateHolder(View itemRoot) {
        return new RightHolder(itemRoot);
```

```
    }

    @Override
    protected int getLayoutRes() {
        return R.layout.item_ask_doctor_dep_right_list;
    }

    class RightHolder extends BaseRadioHolder {

        private TextView tvName;
        private TextView tvTag;

        RightHolder(@NonNull View itemView) {
            super(itemView);
            tvName = itemView.findViewById(R.id.tv_item_ask_doctor_right_name);
            tvTag = itemView.findViewById(R.id.tv_item_ask_doctor_right_tag);
        }

        @Override
        protected void onItemBind(int index, AskDocHotDepEntity.Child item) {
            tvName.setText(item.getDisplayName());
            if (item.isHotDep()) {
                tvTag.setVisibility(View.VISIBLE);
            } else {
                tvTag.setVisibility(View.INVISIBLE);
            }
        }

        @Override
        protected void onItemToggle(int index, AskDocHotDepEntity.Child item) {
            itemView.setSelected(item.isSelected());
        }
    }
}
```

3.多选实现

1.基类

1.ICheckSelector

多选接口，我们控件一般要实现该接口来实现我们自己的多选逻辑，也可以利用 已有代理控件进行实现【CheckSelectorProxy】，后面会有实现参考

```
package cn.kidyn.qdmedical160.nybase.view.selector;

import androidx.annotation.IntDef;
import androidx.annotation.Nullable;

import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.util.List;
```

```
import static
cn.kidyn.qdmedical160.nybase.view.selector.ICheckSelector.Mode.CANCEL;
import static
cn.kidyn.qdmedical160.nybase.view.selector.ICheckSelector.Mode.DEFAULT;
import static
cn.kidyn.qdmedical160.nybase.view.selector.ICheckSelector.Mode.REPEAT;

/**
 * 多选逻辑代理接口
 *
 * 使用步骤：todo
 * <p>
 * Created by pengxr on 2020/9/4.
 */
public interface ICheckSelector<T extends ISelected> {

    @Retention(RetentionPolicy.SOURCE)
    @IntDef({DEFAULT,CANCEL,REPEAT})
    @interface Mode {
        int DEFAULT = 1;
        int CANCEL = 2; // 支持反选
        int REPEAT = 3; // 支持重复选中
    }

    /**
     * @param data          数据集
     * @param defaultIndex -1：不默认选中
     */
    void setData(@Nullable List<T> data, int defaultIndex);

    /**
     * @param data          数据集
     * @param defaultIndex -1：不默认选中
     * @param notify        true：回调defaultIndex选中
     */
    void setData(@Nullable List<T> data, int defaultIndex, boolean notify);

    /**
     * @param index -1：全部取消选中
     */
    void selectOn(int index);

    /**
     * 获取当前选中列表
     */
    @Nullable
    List<T> getSelectedItems();

    /**
     * 获取数据列表
     */
    @Nullable
    List<T> getData();

    /**
     * @return true:拦截选中
     */
    boolean beforeSelect(int index, T t);
```

```
    /**
     * 模式
     */
    void setMode(@Mode int mode);

    interface Listener<T> {

        void onDataSet(List<T> data);

        void onItemUpdate(int index, T t);

        void onItemSelect(int index, T t);
    }

}
```

2.CheckSelectorProxy多选代理逻辑

```
package cn.kidyn.qdmedical160.nybase.view.selector;

import androidx.annotation.IntRange;
import androidx.annotation.Nullable;

import java.util.ArrayList;
import java.util.List;
import java.util.Set;
import java.util.TreeSet;

import static
cn.kidyn.qdmedical160.nybase.view.selector.ICheckSelector.Mode.DEFAULT;

/**
 * 多选逻辑代理
 * <p>
 * 1、用于 FlowLayout
 * 2、用于 RecyclerView
 * <p>
 * Created by pengxr on 2020/1/14.
 */
public abstract class CheckSelectorProxy<T extends ISelected> implements
ICheckSelector<T>, ICheckSelector.Listener<T> {

    @ICheckSelector.Mode
    private int mMode = DEFAULT;

    // 数据集
    private List<T> mData;

    // 当前选中
    private Set<Integer> currentIndex = new TreeSet<>();

    // ----------------------------------------------------------------
    // ICheckSelector
    // ----------------------------------------------------------------

    @Override
```

```
    public void setData(@Nullable List<T> data, int defaultIndex) {
        setData(data, defaultIndex, true);
    }

    @Override
    public void setData(@Nullable List<T> data, int defaultIndex, boolean notify)
{
        if (null == data || data.isEmpty() || defaultIndex >= data.size()) {
            // 无效数据
            onDataSet(data);
            return;
        }

        // 重置状态
        currentIndex.clear();

        // 数据集
        this.mData = data;

        // 默认选中
        if (defaultIndex > -1) {
            T currentItem = data.get(defaultIndex);
            currentItem.setSelected(true);
            currentIndex.add(defaultIndex);
        }

        // 回调
        onDataSet(data);
        // 回调选中
        if (-1 != defaultIndex && notify) {
            onItemSelect(defaultIndex, data.get(defaultIndex));
        }
    }

    @Override
    public void selectOn(int index) {
        if (-1 == index) {
            doClear();
            return;
        }

        switch (mMode) {
            case Mode.DEFAULT:
                if (currentIndex.contains(index)) {
                    // 拦截重复点击
                    return;
                }
                preSelectOn(index);
                break;
            case Mode.CANCEL:
                if (currentIndex.contains(index)) {
                    // 反选
                    doUnSelectOn(index);
                } else {
                    // 选中
                    preSelectOn(index);
                }
                break;
```

```
            case Mode.REPEAT:
                preSelectOn(index);
                break;
        }
    }

    private void preSelectOn(int index) {
        T item = mData.get(index);
        if (beforeSelect(index, item)) {
            // 拦截
            return;
        }
        if (currentIndex.contains(index)) {
            // 重复点击
            onItemSelect(index, item);
            return;
        }
        doSelectOn(index);
    }

    private void doSelectOn(@IntRange(from = 0) int index) {
        // 选中
        T item = mData.get(index);
        item.setSelected(true);
        onItemUpdate(index, item);
        // 回调
        currentIndex.add(index);
        onItemSelect(index, item);
    }

    private void doUnSelectOn(@IntRange(from = 0) int index) {
        // 反选
        T item = mData.get(index);
        item.setSelected(false);
        onItemUpdate(index, item);
        // 回调
        currentIndex.remove(index);
        onItemSelect(index, item);
    }

    private void doClear() {
        for (Integer index : currentIndex) {
            T item = mData.get(index);
            item.setSelected(false);
            onItemUpdate(index, item);
        }
        currentIndex.clear();
    }

    @Nullable
    @Override
    public List<T> getSelectedItems() {
        if (null == mData) {
            return null;
        }
        List<T> selected = new ArrayList<>();
        for (Integer index : currentIndex) {
            selected.add(mData.get(index));
```

```
        }
        return selected;
    }

    @Nullable
    @Override
    public List<T> getData() {
        return mData;
    }

    @Override
    public void setMode(int mode) {
        this.mMode = mode;
    }
}
```

3.多选流式布局CheckFLowLayout

这个是结合 流式布局实现的多选控件基类，其他想要实现多选流式的可以继承这个 类进行实现。

```
package cn.kidyn.qdmedical160.nybase.view.selector;

import android.content.Context;
import androidx.annotation.Nullable;
import android.util.AttributeSet;
import android.view.View;

import com.nykj.shareuilib.widget.other.MyFlowLayout;

import java.util.List;

/**
 * 多选FlowLayout
 * <p>
 * Created by pengxr on 2020/9/4.
 */
public abstract class CheckFLowLayout<T extends ISelected> extends MyFlowLayout
implements ICheckSelector<T> {

    private CheckSelectorProxy<T> mProxy = new CheckSelectorProxy<T>() {

        @Override
        public boolean beforeSelect(int index, T t) {
            return false;
        }

        @Override
        public void onDataSet(List<T> data) {
            removeAllViews();
            if (null == data || data.isEmpty()) {
                return;
            }
            for (int index = 0; index < data.size(); index++) {
                final int indexTemp = index;
                final T item = data.get(index);
```

```
                if (onFilter(item)) {
                    // 过滤
                    continue;
                }
                // 创建
                View itemRoot = onCreateView(item);
                // 绑定
                onUpdateView(itemRoot, item);
                itemRoot.setOnClickListener(new OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        selectOn(indexTemp);
                    }
                });
                addView(itemRoot);
            }
        }

        @Override
        public void onItemUpdate(int index, T t) {
            View child = getChildAt(index);
            onUpdateView(child, t);
        }

        @Override
        public void onItemSelect(int index, T t) {
            // 回调
            if (null != mListener) {
                mListener.onSelected(t);
            }
        }
    };


    public CheckFLowLayout(Context context) {
        this(context, null);
    }

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

    public CheckFLowLayout(Context context, AttributeSet attrs, int defStyleAttr)
{
        super(context, attrs, defStyleAttr);
    }

    // ----------------------------------------------------------------------------
-----------------
    // 代理
    // ----------------------------------------------------------------------------
-----------------

    @Override
    public void setData(@Nullable List<T> data, int defaultIndex) {
        mProxy.setData(data, defaultIndex);
    }
```

```
    @Override
    public void setData(@Nullable List<T> data, int defaultIndex, boolean notify)
{
        mProxy.setData(data, defaultIndex, notify);
    }

    @Override
    public void selectOn(int index) {
        mProxy.selectOn(index);
    }

    @Nullable
    @Override
    public List<T> getSelectedItems() {
        return mProxy.getSelectedItems();
    }

    @Override
    public boolean beforeSelect(int index, T t) {
        return mProxy.beforeSelect(index, t);
    }

    @Override
    public void setMode(int mode) {
        mProxy.setMode(mode);
    }

    @Nullable
    @Override
    public List<T> getData() {
        return mProxy.getData();
    }

    // ------------------------------------------------------------------------
-----------------
    // protected
    // ------------------------------------------------------------------------
-----------------

    protected boolean onFilter(T t) {
        return false;
    }

    protected abstract View onCreateView(T t);

    protected abstract void onUpdateView(View itemRoot, T t);

    // ------------------------------------------------------------------------
-----------------

    private Listener<T> mListener;

    public void setListener(Listener<T> listener) {
        mListener = listener;
    }

    public interface Listener<T> {
```

```
        void onSelected(T e);

    }
}
```

2.实现类

1.CommonCheckFlowLayout通用多选流式控件

1.源码

```
package cn.kidyn.qdmedical160.nybase.view.selector;

import android.content.Context;
import androidx.annotation.Nullable;
import android.util.AttributeSet;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.TextView;

import cn.kidyn.qdmedical160.nybase.R;

/**
 * 通用多选FlowLayout
 *
 * 使用步骤:
 *
 * 1、见 CheckFLowLayout（抽象类） 的使用步骤
 * 2、实现 BindViewInjector（接口） 进行依赖注入
 *
 * Created by pengxr on 2020/9/4.
 */
public class CommonCheckFlowLayout<T extends ISelected> extends
CheckFLowLayout<T> {

    public CommonCheckFlowLayout(Context context) {
        this(context,null);
    }

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

    public CommonCheckFlowLayout(Context context, AttributeSet attrs, int
defStyleAttr) {
        super(context, attrs, defStyleAttr);
    }

    // 依赖注入
    private BindViewInjector<T> mInjector;

    // ------------------------------------------------------------------------
-------------------
    // public
    // ------------------------------------------------------------------------
-------------------
```

```
    /**
     * @param injector View绑定注入器
     */
    public void setInjector(@Nullable BindViewInjector<T> injector) {
        this.mInjector = injector;
    }

    public interface BindViewInjector<T> {

        String onBind(T item);

        boolean onFilter(T item);
    }

    // ---------------------------------------------------------------------------
-------------------
    // CheckFLowLayout
    // ---------------------------------------------------------------------------
-------------------

    @Override
    protected boolean onFilter(T item) {
        if (null == mInjector) {
            return false;
        }
        // 依赖注入
        return mInjector.onFilter(item);
    }

    @Override
    protected View onCreateView(T t) {
        return
LayoutInflater.from(getContext()).inflate(R.layout.tv_radio_selection_v2, this,
false);
    }

    @Override
    protected void onUpdateView(View itemRoot, T item) {
        if (null == mInjector) {
            return;
        }
        // 依赖注入
        TextView tv = (TextView) itemRoot;
        String str = mInjector.onBind(item);
        tv.setText(str);
        tv.setSelected(item.isSelected());
    }
}
```

2.实现


```
<cn.kidyn.qdmedical160.nybase.view.selector.CommonCheckFlowLayout
    android:id="@+id/flow_ask_allergy"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="15dp"
    android:layout_marginRight="15dp"
    app:layout_constraintTop_toBottomOf="@+id/title_ask_allergy"
    app:line_divider="10dp"
    app:padding_divider="10dp" />
```

```
// 标签
flowAskAllergy.setMode(ICheckSelector.Mode.CANCEL);
flowAskAllergy.setInjector(new
CommonCheckFlowLayout.BindViewInjector<PayAskInitDataEntity.Options>() {
    @Override
    public String onBind(PayAskInitDataEntity.Options item) {
        return item.getName();
    }

    @Override
    public boolean onFilter(PayAskInitDataEntity.Options item) {
        return TextUtils.isEmpty(item.getName());
    }
});
flowAskAllergy.setListener(new
CheckFLowLayout.Listener<PayAskInitDataEntity.Options>() {
    @Override
    public void onSelected(PayAskInitDataEntity.Options e) {
        callListener(new AllergyInputParam(getTags(),
editAskAllergy.getTextString()));
    }
});
```

2.多选RecyclerView.Adapter

1.效果

这是一个列表BaseCheckSelectorAdapter，列表里面是多选按钮。都是利用 CheckSelectorProxy 与 单选多选 ICheckSelector ， IRadioSelector 进行实现。

2.源码基类

```
package cn.kidyn.qdmedical160.nybase.view.recycler.adapter;

import androidx.annotation.LayoutRes;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.recyclerview.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import java.util.Collections;
import java.util.List;

import cn.kidyn.qdmedical160.nybase.view.selector.CheckSelectorProxy;
import cn.kidyn.qdmedical160.nybase.view.selector.ICheckSelector;
import cn.kidyn.qdmedical160.nybase.view.selector.ISelected;
import me.drakeet.multitype.ItemViewBinder;
import me.drakeet.multitype.MultiTypeAdapter;

/**
 * 多选RecyclerView.Adapter
 *
 * Created by pengxr on 2020/9/7.
 */
public abstract class BaseCheckSelectorAdapter<T extends ISelected> extends
MultiTypeAdapter implements ICheckSelector<T> {

    public BaseCheckSelectorAdapter() {
        register(getType(), new BaseCheckSelectorBinder());
```

```
    }

    private CheckSelectorProxy<T> mProxy = new CheckSelectorProxy<T>() {
        @Override
        public boolean beforeSelect(int index, T t) {
            return false;
        }

        @Override
        public void onDataSet(List<T> data) {
            if (null == data || data.isEmpty()) {
                setItems(Collections.EMPTY_LIST);
            } else {
                setItems(data);
            }
            notifyDataSetChanged();
        }

        @Override
        public void onItemUpdate(int index, T t) {
            notifyItemChanged(index, -1);
        }

        @Override
        public void onItemSelect(int index, T t) {
            if (null != mListener) {
                mListener.onItemSelect(t);
            }
        }
    };

    // ------------------------------------------------------------------------
---------
    // 代理
    // ------------------------------------------------------------------------
---------

    @Override
    public void setData(@Nullable List<T> data, int defaultIndex) {
        mProxy.setData(data, defaultIndex);
    }

    @Override
    public void setData(@Nullable List<T> data, int defaultIndex, boolean notify)
{
        mProxy.setData(data, defaultIndex, notify);
    }

    @Override
    public void selectOn(int index) {
        mProxy.selectOn(index);
    }

    @Nullable
    @Override
    public List<T> getSelectedItems() {
        return mProxy.getSelectedItems();
    }
```

```
    @Override
    public boolean beforeSelect(int index, T t) {
        if (null != mListener) {
            return mListener.beforeItemSelect(t);
        }
        return mProxy.beforeSelect(index, t);
    }

    @Override
    public void setMode(int mode) {
        mProxy.setMode(mode);
    }

    @Nullable
    @Override
    public List<T> getData() {
        return mProxy.getData();
    }

    // ------------------------------------------------------------------------
---------
    // protected
    // ------------------------------------------------------------------------
---------

    protected abstract Class<T> getType();

    protected abstract BaseListItemHolderImpl generateHolder(View itemRoot);

    @LayoutRes
    protected abstract int getLayoutRes();

    // ------------------------------------------------------------------------
----------------

    private class BaseCheckSelectorBinder extends ItemViewBinder<T,
BaseListItemHolderImpl> {

        @NonNull
        @Override
        protected BaseListItemHolderImpl onCreateViewHolder(@NonNull
LayoutInflater inflater, @NonNull ViewGroup parent) {
            View itemRoot = inflater.inflate(getLayoutRes(), parent, false);
            return generateHolder(itemRoot);
        }

        @Override
        protected void onBindViewHolder(@NonNull BaseListItemHolderImpl holder,
@NonNull T t) {
            holder.bind(t);
        }
    }

    // ------------------------------------------------------------------------
----------------
```

```
    public abstract class BaseListItemHolderImpl extends RecyclerView.ViewHolder
{

        protected T mItem;

        public BaseListItemHolderImpl(@NonNull View itemView) {
            super(itemView);
            itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    if (null != mItem) {
                        onItemClick(getAdapterPosition(), mItem);
                    }
                }
            });
        }

        public void bind(T item) {
            mItem = item;
            onItemBind(getAdapterPosition(), item);
        }

        protected abstract void onItemBind(int index, T item);

        protected void onItemClick(int index, T item) {
            selectOn(index);
        }
    }

    // ------------------------------------------------------------------------
----------------

    private OnCheckListener<T> mListener;

    public void setListener(OnCheckListener<T> listener) {
        mListener = listener;
    }

    public interface OnCheckListener<T> {

        void onItemSelect(T item);

        boolean beforeItemSelect(T item);

    }
}
```

3.实现类MajorDiseaseAdapter

这个就是使用类，具体的查看项目91160_android.

二.通用底部弹框选择器CommonBottomSheetFragment

这个控件在 图文咨询页面使用较多。

1.使用

1.选择患者

实现CommonBottomSheetFragment.OnItemSelectListener，并构造实例数据以及注入对象

```
int dialogHeight = (int) (ViewCommonUtils.getScreenHeight(activity) * 0.7F);
CommonBottomSheetFragment.BindViewInjector<PatientEntity> injector = new
CommonBottomSheetFragment.BindViewInjector<PatientEntity>() {
    @Override
    public void inject(TextView tv, PatientEntity item) {
        tv.setText(item.getTruename());
        if (ConstantConfig.HANDLER_CALLBACKCODE_0 != item.getTextColor()) {

tv.setTextColor(mContext.getResources().getColor(item.getTextColor()));
        } else {

tv.setTextColor(mContext.getResources().getColor(R.color.text_black_color));
        }
    }
};
CommonBottomSheetFragment<PatientEntity> impl =
CommonBottomSheetFragment.newInstance(data, dialogHeight, true);
impl.setInjector(injector);
impl.setListener(ChooseConsultantController.this);
impl.show(activity);
```

2.底部相册选择弹框


```
/**
 * 获取图片途径选择：拍照或相册
 */
public static void fetchImage(final FragmentActivity activity, @Nullable
FetchImageParam param, @NonNull final IFetchImage.Listener callback) {
    CaptureImageAction captureAction = new CaptureImageAction(param);
    SelectImageAction selectAction = new SelectImageAction(param);

    ImageButton captureButton = new ImageButton(R.string.capture_image,
captureAction);
    ImageButton selectButton = new ImageButton(R.string.select_image,
selectAction);

    List<ImageButton> buttons = Arrays.asList(captureButton, selectButton);

    // 这里使用的是通用底部弹框控件，具体参考【通用底部弹框选择器
CommonBottomSheetFragment】
    CommonBottomSheetFragment<ImageButton> dialog =
CommonBottomSheetFragment.newInstance(buttons, -1, false);
    dialog.setInjector(new
CommonBottomSheetFragment.BindViewInjector<ImageButton>() {
        @Override
        public void inject(TextView tv, ImageButton item) {
```

```
            tv.setText(item.textRes);
        }
    });

    dialog.setListener(new
CommonBottomSheetFragment.OnItemSelectListener<ImageButton>() {
        @Override
        public void onSelect(DialogFragment dialog, ImageButton object) {
            object.action.fetchImage(activity, callback);
        }
    });

    dialog.show(activity);
}
```

这里利用 CommonBottomSheetFragment 注入的方式进行底部弹框实现。

2.源码分析

1.ui布局

```
<?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="wrap_content"
    android:background="#f5f5f5"
    android:gravity="bottom"
    android:orientation="vertical">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:animationCache="false"
        android:cacheColorHint="#00000000"
        android:scrollingCache="false" />

    <Space
        android:layout_width="match_parent"
        android:layout_height="12dp" />

    <Button
        android:id="@+id/btn_cancel"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:background="#ffffff"
        android:text="取消"
        android:textColor="#666666"
        android:textSize="16sp" />

</LinearLayout>
```

2.封装代码

```
package com.nykj.shareuilib.widget.dialog;

import android.app.Activity;
import android.graphics.Color;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AbsListView;
import android.widget.TextView;

import com.ny.jiuyi160_doctor.common.util.DensityUtil;
import com.nykj.shareuilib.R;
import com.nykj.shareuilib.widget.recyclerview.SpacingDecoration;

import java.util.List;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.DialogFragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

/**
 * Created by pengxr on 2020/4/28.
 */
public class CommonBottomSheetFragment<T> extends BaseBottomSheetFragment {

    private List<T> mData;
    private int mDialogHeight;
    private boolean mShouldStackFromBottom;

    private OnItemSelectListener<T> mListener;

    // 依赖注入
    private BindViewInjector<T> mInjecter;

    /**
     * @param data                  数据
     * @param dialogHeight          弹窗高度
     * @param shouldStackFromBottom true：列表初始化是滚动到底部
     * @param <T>                   数据泛型
     */
    public static <T> CommonBottomSheetFragment<T> newInstance(@NonNull List<T>
data, int dialogHeight, boolean shouldStackFromBottom) {
        CommonBottomSheetFragment<T> dialog = new CommonBottomSheetFragment<>();
        dialog.init(data, dialogHeight, shouldStackFromBottom);
        return dialog;
    }

    private void init(List<T> data, int dialogHeight, boolean
shouldStackFromBottom) {
        this.mData = data;
        this.mDialogHeight = dialogHeight;
        this.mShouldStackFromBottom = shouldStackFromBottom;
    }
```

```
    // ------------------------------------------------------------------------
-------------
    // BaseBottomSheetFragment
    // ------------------------------------------------------------------------
-------------

    @Override
    protected int getLayoutRes() {
        return R.layout.mqtt_dialog_bottom_sheet_common;
    }

    @Override
    public int getDialogHeight(Activity activity) {
        return mDialogHeight;
    }

    @Override
    protected void init(View root) {
        root.findViewById(R.id.btn_cancel).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // 退出
                dismiss();
            }
        });

        RecyclerView rv = root.findViewById(R.id.list);
        LinearLayoutManager linearLayoutManager = new
LinearLayoutManager(getContext(), LinearLayoutManager.VERTICAL, false);
        linearLayoutManager.setStackFromEnd(mShouldStackFromBottom);
        rv.setLayoutManager(linearLayoutManager);
        rv.addItemDecoration(new SpacingDecoration(getContext(), 1));

        TextAdapter adapter = new TextAdapter();
        rv.setAdapter(adapter);

    }

    /**
     * @param injector View绑定注入器
     */
    public void setInjector(@Nullable BindViewInjector<T> injector) {
        this.mInjecter = injector;
    }

    /**
     * @param listener 点击监听器
     */
    public void setListener(@NonNull OnItemSelectListener<T> listener) {
        this.mListener = listener;
    }

    // ------------------------------------------------------------------------
-------------

    public interface OnItemSelectListener<T> {
```

```
        void onSelect(DialogFragment dialog, T object);
    }

    public interface BindViewInjector<T> {
        void inject(TextView tv, T item);
    }

    private class TextAdapter extends RecyclerView.Adapter<TextViewHolder> {

        @NonNull
        @Override
        public TextViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup,
int type) {
            TextView textView = new TextView(viewGroup.getContext());
            textView.setLayoutParams(new
AbsListView.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
            textView.setTextSize(16);
            textView.setGravity(Gravity.CENTER);
            textView.setTextColor(0xff333333);
            textView.setBackgroundColor(Color.WHITE);
            textView.setPadding(0, DensityUtil.dip2px(getContext(), 14), 0,
DensityUtil.dip2px(getContext(), 14));
            return new TextViewHolder(textView);
        }

        @Override
        public void onBindViewHolder(@NonNull TextViewHolder holder, int
position) {
            T item = mData.get(position);
            holder.bind(item);
        }

        @Override
        public int getItemCount() {
            return null != mData ? mData.size() : 0;
        }
    }

    private class TextViewHolder extends RecyclerView.ViewHolder {

        private TextView mTextView;
        private T mItem;

        TextViewHolder(View itemView) {
            super(itemView);
            mTextView = (TextView) itemView;
            itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    if (null != mItem) {
                        mListener.onSelect(CommonBottomSheetFragment.this,
mItem);
                        dismiss();
                    }
                }
            });
        }
```

```
        void bind(T item) {
            mItem = item;
            if (null != mInjecter) {
                mInjecter.inject(mTextView, item);
            } else {
                mTextView.setText(item.toString());
            }
        }
    }


}
```

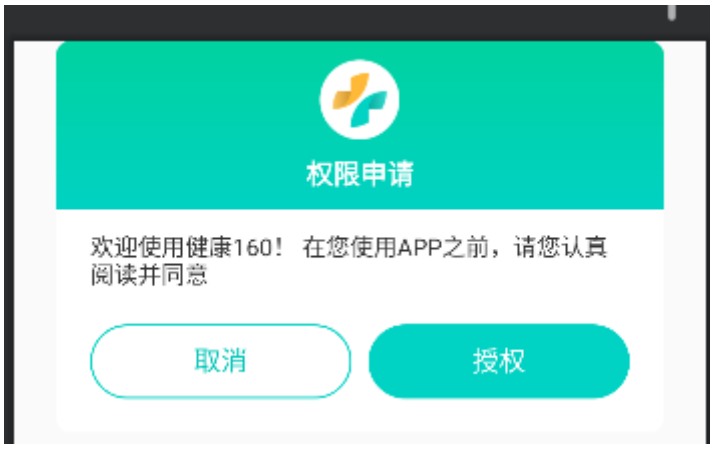
三.Dialog

1.无边框中心弹框

1.ui效果

我们经常要实现下面的弹框效果：


5:54
总店-中医科
张哥中医理疗馆
国家重点
要组成部分，物理因素通过
科室介绍
07月26日排班在今日17:07放出，是否有号以医
全部
今天
6
周四
周六
9
周日
10
科室公告
总店-中医科科室公告内容：中医理疗是康复治疗的重要组成部分，物理因素通过人体局部直接作用，和对神经、体液的间接作用引起人体反应，从而调整了血液循环，加快了新陈代谢，促进对细胞组织的修复，调节神经系统的功能，提高免疫功能，消除致病因素，改善病理过程，达到治病目的。
我知道了
镇万新
主任医师
9.5分
预约量 982
点评量 14
擅长：骨科及脊柱外科临床工作,颈、胸、腰、...
张健康


权限申请
欢迎使用健康160！ 在您使用APP之前，请您认真阅读并同意
取消
授权


工作时间
周一至周四
周一至周四
周一至周四
历史工作时间：
09:00~12:00/14:00-18:00
知道了

这里已经有封装好的基类，我们直接继承实现就可以了。

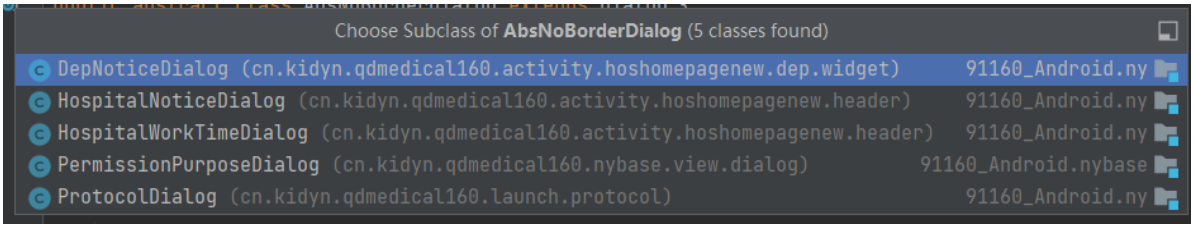
2.代码实现

1.基类AbsNoBorderDialog

```
package cn.kidyn.qdmedical160.nybase.view.dialog;

import android.app.Dialog;
import android.content.Context;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.view.Gravity;
import android.view.Window;
import android.view.WindowManager;

/**
 * 无边框对话框
 * Create by liangy on 2019/12/27
 */
public abstract class AbsNoBorderDialog extends Dialog {
    public AbsNoBorderDialog(Context context) {
        super(context);
        init();
    }

    public AbsNoBorderDialog(Context context, int theme) {
        super(context, theme);
        init();
    }

    protected AbsNoBorderDialog(Context context, boolean cancelable,
OnCancelListener cancelListener) {
        super(context, cancelable, cancelListener);
        init();
    }

    private void init(){
        removeTitle();
    }

    private void removeTitle(){
        requestWindowFeature(Window.FEATURE_NO_TITLE);
    }

    @Override
    public void show() {
        super.show();
        removeDialogBorder(this);
    }

    private static void removeDialogBorder(Dialog dialog){
        Window window = dialog.getWindow();
        if (window != null){
            WindowManager.LayoutParams layoutParams = window.getAttributes();
```

```
            layoutParams.gravity= Gravity.CENTER;
            layoutParams.width= WindowManager.LayoutParams.MATCH_PARENT;
            layoutParams.height= WindowManager.LayoutParams.WRAP_CONTENT;
            layoutParams.dimAmount = 0.6f;
            window.getDecorView().setPadding(0, 0, 0, 0);
            window.getDecorView().setBackgroundColor(Color.TRANSPARENT);
            window.setAttributes(layoutParams);
            window.setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));
        }
    }
}
```

2.实现类



弹框实现都差不多，我们挑选一个看看即可：

科室公告弹框：

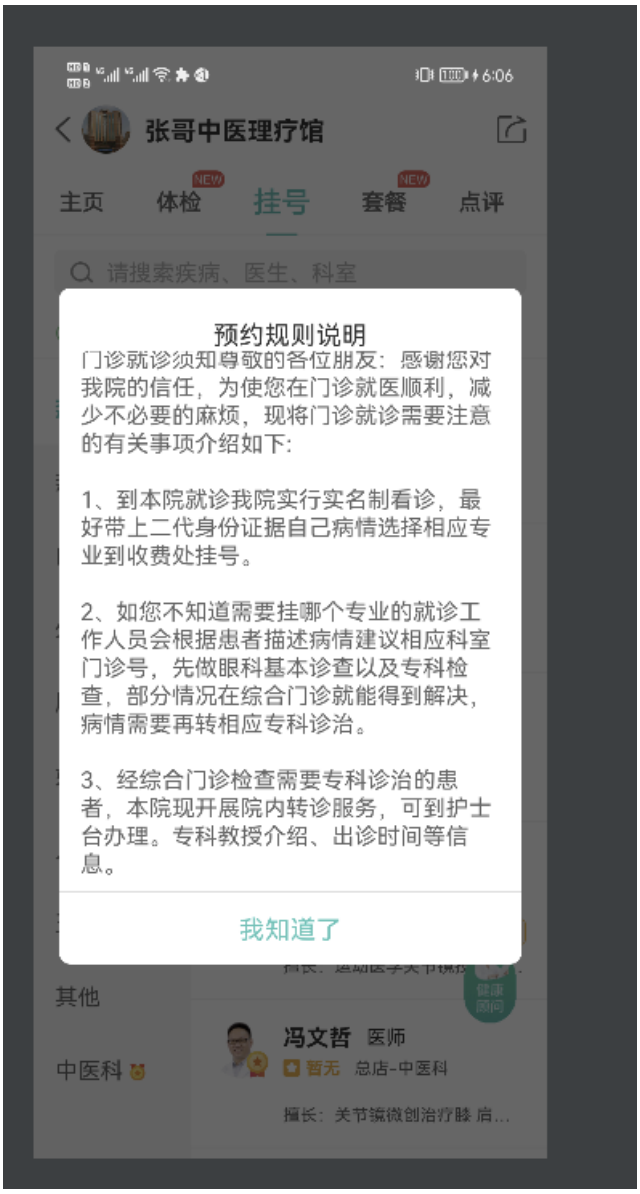
布局代码R.layout.dialog_dep_notice 这么不贴，有需要的可以自己去对应的项目代码 91160_android 查看

```
package cn.kidyn.qdmedical160.activity.hoshomepagenew.dep.widget;

import android.content.Context;
import android.os.Bundle;
import android.text.Html;
import android.text.TextUtils;
import android.view.View;
import android.widget.TextView;

import java.text.SimpleDateFormat;
import java.util.Locale;

import cn.kidyn.qdmedical160.QDApplicationContext;
import cn.kidyn.qdmedical160.R;
import cn.kidyn.qdmedical160.nybase.util.StringUtils;
import cn.kidyn.qdmedical160.nybase.view.dialog.AbsNoBorderDialog;
import cn.kidyn.qdmedical160.util.WebLinkMethod;
import cn.kidyn.qdmedical160.view.html.HtmlTagHandler;
import cn.kidyn.qdmedical160.view.html.HtmlImageGetter;

/**
 * 科室公告新版弹窗【6.4.3】
```

```
 * Create by qulj on 2019/12/24
 */
public class DepNoticeDialog extends AbsNoBorderDialog {
    private String title;
    private String updateTime;
    private String content;

    public DepNoticeDialog(Context context,String title,String updateTime, String
htmlCode) {
        super(context);
        this.title=title;
        this.updateTime=updateTime;
        this.content = htmlCode;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.dialog_dep_notice);
        TextView tv_ok = findViewById(R.id.tv_ok);
        TextView tv_title = findViewById(R.id.tv_title);
        TextView tv_published_time = findViewById(R.id.tv_published_time);
        TextView tv_content = findViewById(R.id.tv_content);
        tv_ok.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                dismiss();
            }
        });
        if (!TextUtils.isEmpty(title)){
            tv_title.setText(title);
        }else {

tv_title.setText(QDApplicationContext.getInstance().getText(R.string.dep_notice)
);
        }
        if(!StringUtils.StrIsNull(updateTime)) {
            try {
                tv_published_time.setVisibility(View.VISIBLE);
                //以防哪天公共接口（悄悄）改成了组装数据，格式化会失败。则直接展示原数据。
                SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd",
Locale.CHINA);
                String time = sdf.format(sdf.parse(updateTime));//.format(time);

tv_published_time.setText(String.format(getContext().getString(R.string.hos_home
_notice_publish_time),time));
            } catch (Exception e){
                tv_published_time.setText(updateTime);
            }
        } else {
            tv_published_time.setVisibility(View.INVISIBLE);
        }
        HtmlTagHandler myTagHandler=new HtmlTagHandler(getContext());
        tv_content.setText(Html.fromHtml(content,new HtmlImageGetter(tv_content,
tv_content.getContext()),myTagHandler));//v6.5.6又要改回使用html的样式

tv_content.setMovementMethod(WebLinkMethod.getInstance(this.getContext()));
    }
```

```
}
```

2.DialogFactory工厂创建方式

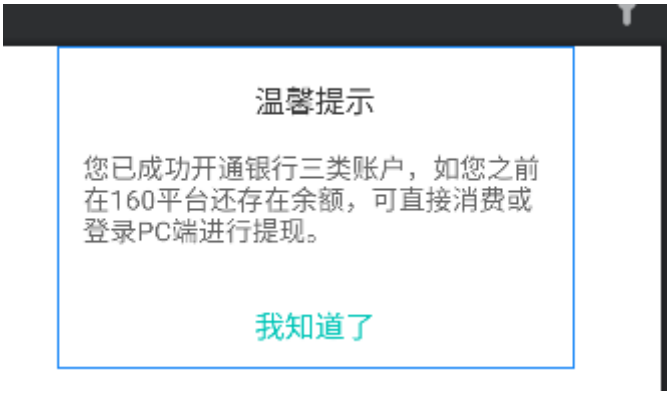


设置方式:

```
final DialogFactory dialogCompleteMember = new DialogFactory().create((Activity)
mContext, R.layout.dialog_common_i_know);
((TextView)
dialogCompleteMember.findViewById(R.id.tv_dialog_content)).setMovementMethod(new
ScrollingMovementMethod());
dialogCompleteMember
        .setConfirm(R.id.tv_confirm, new DialogFactory.OnClickListener() {
            @Override
            public void onClick() {
                if (null != dialogCompleteMember) {
                    dialogCompleteMember.dismiss();
                }
            }
        })
        .setText(R.id.tv_confirm, mContext.getString(R.string.notice_know))
```

```
        .setText(R.id.tv_dialog_title,
mContext.getString(R.string.hos_home_yuyueguize))
        .setViewVisiable(R.id.iv_cancle_right, View.GONE)
        .setText(R.id.tv_dialog_content, Html.fromHtml(ruleInfo))
        .setCustomCanceledOnTouchOutside(false)
        .show();
```

代码布局自己定义：




```
<?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="wrap_content"
    android:layout_marginLeft="@dimen/dimen_53dp"
    android:layout_marginRight="@dimen/dimen_53dp"
    android:background="@drawable/wallet_bg_white_corner"
    android:orientation="vertical">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TextView
            android:id="@+id/tv_dialog_title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="@dimen/dimen_20dp"
            android:gravity="center"
            android:text="@string/common_kindly_reminder"
            android:textColor="@color/color_black_333333"
            android:textSize="@dimen/dimen_18sp" />


        <ImageView
            android:id="@+id/iv_cancle_right"
            android:layout_width="@dimen/dimen_26dp"
            android:layout_height="@dimen/dimen_26dp"
            android:layout_alignParentRight="true"
            android:layout_marginTop="@dimen/dimen_10dp"
            android:layout_marginRight="@dimen/dimen_10dp"
            android:padding="@dimen/dimen_6dp"
            android:scaleType="fitXY"
            android:src="@drawable/btn_pop_close"
            android:visibility="gone" />
```

```
    </RelativeLayout>


    <cn.kidyn.qdmedical160.activity.hoshomepagenew.header.NoticeScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:scrollbars="vertical"
        android:scrollbarSize="@dimen/dimen_2dp">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">
            <TextView
                android:id="@+id/tv_dialog_content"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:drawablePadding="@dimen/dimen_20dp"
                android:layout_marginLeft="@dimen/dimen_15dp"
                android:layout_marginTop="@dimen/dimen_17dp"
                android:layout_marginRight="@dimen/dimen_15dp"
                android:layout_marginBottom="@dimen/dimen_20dp"
                android:maxLines="26"
                android:text="您已成功开通银行三类账户，如您之前在160平台还存在余额，可直
接消费或登录PC端进行提现。"
                android:textColor="@color/color_black_ff666666"
                android:textSize="@dimen/dimen_16sp" />
        </LinearLayout>
    </cn.kidyn.qdmedical160.activity.hoshomepagenew.header.NoticeScrollView>

    <View
        android:id="@+id/view_divider"
        android:layout_width="match_parent"
        android:layout_height="@dimen/line_height"
        android:background="@color/grey_line" />

    <TextView
        android:id="@+id/tv_confirm"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="@dimen/dimen_13dp"
        android:layout_marginBottom="@dimen/dimen_13dp"
        android:gravity="center"
        android:text="@string/notice_know"
        android:textColor="@color/colorAccent"
        android:textSize="@dimen/dimen_18sp" />
</LinearLayout>
```

四.跑马灯

这是常见的跑马灯效果，利用TextView进行实现

1.MarqueeText

1.UI效果

> 善病理过程，达到治病目的

2.代码实现

1.布局使用

```
<cn.kidyn.qdmedical160.nybase.view.MarqueeText
    android:id="@+id/tv_notice"
    style="@style/hospital_home_dep_topmenu_text"
    android:drawableLeft="@drawable/drawable_dep_icon_selector"
    android:ellipsize="marquee"
    android:focusable="true"
    android:focusableInTouchMode="true"
    android:marqueeRepeatLimit="marquee_forever"
    android:maxWidth="@dimen/dimen_200dp"
    android:paddingLeft="@dimen/dimen_0dp"
    android:paddingRight="@dimen/dimen_10dp"
    android:singleLine="true"
    android:text="@string/dep_notice"
    android:enabled="false"
    android:textColor="@color/color_dep_selector" />
```

```
if (noticeEntity != null &&
noticeEntity.isShowNotice()&&noticeEntity.hasContent()) {
    NoticeShowUtils.Companion.scroll(tv_notice,false);

tv_notice.setText(QDApplicationContext.getInstance().getString(R.string.dep_noti
ce_2));
    tv_notice.setEnabled(true);
    if (noticeEntity.isShowDialogNotice()) {
        if (noticeEntity.isAlwaysShowDialogNotice() ||
!NoticeShowUtils.Companion.isLast24HoursShowNoticeDialog(key)) {
            if (noticeEntity.isOnlyShowDialogNotice()){
                NoticeShowUtils.Companion.updateNoticeDialogShowTime(key,false);
            }
            showNoticeDialog(noticeEntity);
        }
    } else if (noticeEntity.isShowRollNotice()&&noticeEntity.hasContentNoHtml())
{
        //滚动展示
        NoticeShowUtils.Companion.scroll(tv_notice,true);
        tv_notice.setText(noticeEntity.getContentNoHtml());
    }
} else {
    //公告不展示或者公告内容为空时   直接返回
    tv_notice.setEnabled(false);
}
```

```
fun scroll(text: TextView, isScroll: Boolean){
    text.maxLines=1
    if (isScroll){
        text.ellipsize= TextUtils.TruncateAt.MARQUEE
        text.isFocusable=true
        text.isFocusableInTouchMode=true;
        text.marqueeRepeatLimit= -1
    }else{
        text.ellipsize= TextUtils.TruncateAt.END
        text.marqueeRepeatLimit= 0
    }
}
```

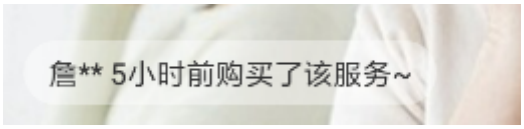
2.源码

```
package cn.kidyn.qdmedical160.nybase.view;

import android.content.Context;
import android.util.AttributeSet;

import androidx.appcompat.widget.AppCompatTextView;

public class MarqueeText extends AppCompatTextView {

public MarqueeText(Context context) {

super(context);

}

public MarqueeText(Context context, AttributeSet attrs) {

super(context, attrs);

}

public MarqueeText(Context context, AttributeSet attrs, int defStyleAttr) {

super(context, attrs, defStyleAttr);

}

//重写isFocused方法，让其一直返回true

public boolean isFocused() {

return true;

}

@Override

public void onWindowFocusChanged(boolean hasWindowFocus) {

}
```

```
}
```

2.上下滚动广告条AutoRollViewSwitcher

1.UI效果




```
public class AutoRollViewSwitcher extends ViewSwitcher {}
```

利用ViewSwitcher进行实现，当时实现方式还有其他的：

TextSwitcher实现文本自动垂直滚动；

ViewFlipper - 左右上下自动滚动切换；

2.代码实现

1.布局使用

```
<cn.kidyn.qdmedical160.view.AutoRollViewSwitcher
    android:id="@+id/view_switcher"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="@dimen/dimen_16dp"
    android:inAnimation="@anim/adv_text_translate_in"
    android:outAnimation="@anim/adv_text_translate_out"
    android:visibility="gone"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@id/ll_dep_top_menu" />
```

```
@BindView(R.id.view_switcher)
AutoRollViewSwitcher switcher;

  @Override
    public void onResume() {
        super.onResume();
        if (switcher.getVisibility() == View.VISIBLE) {
            switcher.startLooping();
        }
    }

    @Override
    public void onPause() {
        super.onPause();
        if (switcher.getVisibility() == View.VISIBLE) {
```

```
            switcher.stopLooping();
        }
    }

    设置数据
    private void updateSwitcher(List<DepSchNoticeEntity> list) {
        if (null == list || list.isEmpty()) {
            switcher.setVisibility(View.GONE);
            return;
        }
        switcher.setVisibility(View.VISIBLE);
        DepSchNoticeAdapter adapter = new DepSchNoticeAdapter(getContext());
        switcher.setAdapter(adapter);
        adapter.setDataSource(list);
        switcher.setInterval(3000);
        switcher.startLooping();
    }
```

view_registration_dynamic 这个布局就不贴了。很简单

```
package cn.kidyn.qdmedical160.activity.hoshomepagenew.dep.view;

import android.content.Context;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.LinearLayout;
import android.widget.TextView;

import java.util.List;

import cn.kidyn.qdmedical160.R;
import cn.kidyn.qdmedical160.entity.department.DepSchNoticeEntity;
import cn.kidyn.qdmedical160.util.ComprehensiveJumpUntil;
import cn.kidyn.qdmedical160.util.config.ConstantConfig;
import cn.kidyn.qdmedical160.view.AutoRollViewSwitcher;

public class DepSchNoticeAdapter extends
AutoRollViewSwitcher.BaseAutoRollViewAdapter<DepSchNoticeEntity> {


    public DepSchNoticeAdapter(Context context) {
        super(context);
    }

    @Override
    public View onCreateView() {
        LinearLayout layout = new LinearLayout(getContext());
        layout.setOrientation(LinearLayout.VERTICAL);
        for (int i = 0; i < getLines(); i++) {
```

```
            View v =
LayoutInflater.from(getContext()).inflate(R.layout.view_registration_dynamic,
null);
            layout.addView(v);
        }
        return layout;
    }

    @Override
    public void onBindView(View view, List<DepSchNoticeEntity> obj) {
        for (int i = 0; i < obj.size(); i++) {
            View content = ((ViewGroup) view).getChildAt(i);
            TextView titleTv = content.findViewById(R.id.tv_title);
            TextView tv_go_search = content.findViewById(R.id.tv_go_search);

            final DepSchNoticeEntity each = obj.get(i);
            titleTv.setText(each.getFanghao_tip());
            boolean is_redirect =
ConstantConfig.ONE.equals(each.getIs_redirect());
            tv_go_search.setVisibility(is_redirect ? View.VISIBLE : View.GONE);
            titleTv.setGravity(Gravity.LEFT);//is_redirect ? Gravity.LEFT :
Gravity.CENTER);//v6.6.4放号提醒要改加居左，因为有可能有多行
            content.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    //2019/12/23 放号提醒+附近有号
                    ComprehensiveJumpUntil.gotoWebView2Activity(getContext(),
each.getNosch_redirect_url(), ConstantConfig.IS_NULL);
                }
            });
        }
    }
}
```

2.源码

```
package cn.kidyn.qdmedical160.view;

import android.content.Context;
import android.util.AttributeSet;
import android.util.Log;
import android.view.View;
import android.widget.ViewSwitcher;

import java.util.ArrayList;
import java.util.List;

/**
 * @author maohui
 * @date Created on 2019/5/13.
 */
public class AutoRollViewSwitcher extends ViewSwitcher {

    private int mInterval;

    private int mCurrentItem;
```

```
    private BaseAutoRollViewAdapter mAdapter;

    private final Runnable runnable = new Runnable() {
        @Override
        public void run() {
            if (getItemCount() == 0) {
                mCurrentItem = 0;
            } else {
                mCurrentItem = parseDataIndex(mCurrentItem +
mAdapter.getLines());
            }
            showNext();
            startLooping();
        }
    };

    public AutoRollViewSwitcher(Context context) {
        super(context);
    }

    public AutoRollViewSwitcher(Context context, AttributeSet attrs) {
        super(context, attrs);
    }


    private List info = new ArrayList<>();

    private void onShowView() {
        info.clear();
        if (mAdapter != null) {
            for (int i = 0; i < mAdapter.getLines(); i++) {
                info.add(mAdapter.getItem(parseDataIndex(mCurrentItem + i)));
            }
            mAdapter.onBindView(getCurrentView(), info);
        }
    }

    private int parseDataIndex(int index) {
        if (getItemCount() != 0) {
            return index % getItemCount();
        }
        return 0;
    }

    public void setInterval(int interval) {
        mInterval = interval;
    }

    private int getItemCount() {
        return mAdapter.getItemCount();
    }

    public void startLooping() {
        if (mAdapter == null) {
            return;
        }
        removeCallbacks(runnable);
        if (getItemCount() >= 2 && getVisibility() == VISIBLE) {
```

```
            postDelayed(runnable, mInterval);
            if (mCurrentItem == 0) {
                onShowView();
            }
        } else {
            onShowView();
        }
    }

    public void stopLooping() {
        removeCallbacks(runnable);
    }

    @Override
    public void showNext() {
        super.showNext();
        onShowView();
    }

    public BaseAutoRollViewAdapter getAdapter() {
        return mAdapter;
    }

    public void setAdapter(BaseAutoRollViewAdapter mAdapter) {
        if (mAdapter != null) {
            removeCallbacks(runnable);
            this.mAdapter = mAdapter;
            removeAllViews();
            setFactory(() -> mAdapter.onCreateView());
            if (mAdapter.getItemCount() > 0) {
                onShowView();
            }
        }
    }

    public interface AutoRollAdapter<T> {

        View onCreateView();

        void onBindView(View view, List<T> obj);

        int getItemCount();

        T getItem(int position);

        int getLines();
    }

    /**
     * Base class for an Adapter
     *
     * @param <T> the data TYPE
     */
    public static abstract class BaseAutoRollViewAdapter<T> implements
AutoRollViewSwitcher.AutoRollAdapter<T> {

        //默认一次展示数据数量
        private static final int DEFAULT_LINES = 1;
```

```
        private final Context mContext;

        private List<T> dataSource;

        public BaseAutoRollViewAdapter(Context context) {
            mContext = context;
        }

        public void setDataSource(List<T> dataSource) {
            this.dataSource = dataSource;
        }

        public List<T> getDataSource() {
            return dataSource;
        }

        public Context getContext() {
            return mContext;
        }

        @Override
        public int getItemCount() {
            return dataSource != null ? dataSource.size() : 0;
        }

        @Override
        public T getItem(int position) {
            if (position >= 0 && position < getItemCount()) {
                return getDataSource().get(position);
            } else {
                return null;
            }
        }

        @Override
        public int getLines() {
            return DEFAULT_LINES;
        }
    }

}
```

3.左右自动滚动AutoScrollLayout

使用

```
<cn.kidyn.qdmedical160.activity.hoshomepagenew.maintab.view.AutoScrollLayout
    android:id="@+id/tv_asl_hospital_name"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginLeft="@dimen/dimen_10dp"
    android:layout_marginRight="@dimen/dimen_5dp"
    android:elevation="@dimen/dimen_1dp"
    app:layout_constraintBottom_toBottomOf="@id/close_background"
    app:layout_constraintLeft_toRightOf="@id/avatar_hospital"
    app:layout_constraintRight_toLeftOf="@id/iv_share_2"
    app:layout_constraintTop_toTopOf="@id/close_background">

    <TextView
        android:id="@+id/tv_inner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:ellipsize="end"
        android:textStyle="bold"
        android:lines="1"
        android:textColor="@color/color_black_333333"
        android:textSize="@dimen/dimen_18sp"
        tools:text="暂无暂无暂无暂无暂无暂无暂无暂无暂无暂无暂无暂无" />
```

```
</cn.kidyn.qdmedical160.activity.hoshomepagenew.maintab.view.AutoScrollLayout>
```

源码：

```
package cn.kidyn.qdmedical160.activity.hoshomepagenew.maintab.view;

import android.content.Context;
import android.os.Handler;
import android.os.Looper;
import android.util.AttributeSet;
import android.view.View;
import android.widget.HorizontalScrollView;
import android.widget.TextView;

import androidx.annotation.Nullable;
import cn.kidyn.qdmedical160.R;
import cn.kidyn.qdmedical160.util.ViewCommonUtils;

/**
 * 自动滚动
 * Create by liangy on 2020/1/9
 */
public class AutoScrollLayout extends HorizontalScrollView {

    private Handler handler = new Handler(Looper.getMainLooper());
    private View innerView;
    private int translation = 0;
    private boolean detached = false;

    // 停靠多少个像素（到达边缘时的等待时间）
    private static final int DECOR_RANGE = 50;
    // 刷新率
    private static final int FRAME_INTERVAL = 15;
    // 一次刷新移动多少像素
    private static final int SPEED_IN_DIP = 1;

    // 翻转滚动
    private int reverseRemove = -1;

    // 保存当前的leftMargin
    private int leftMargin = 0;

    public AutoScrollLayout(Context context) {
        super(context);
        init();
    }

    public AutoScrollLayout(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    public AutoScrollLayout(Context context, @Nullable AttributeSet attrs, int
defStyleAttr) {
        super(context, attrs, defStyleAttr);
        init();
    }
```

```
private void init(){
    setHorizontalScrollBarEnabled(false);
    setOverScrollMode(OVER_SCROLL_NEVER);
    setClipChildren(true);
    restartTimer();
}

private void restartTimer(){
    handler.removeCallbacks(r);
    handler.postDelayed(r, FRAME_INTERVAL);
}

@Override
protected void onDetachedFromWindow() {
    super.onDetachedFromWindow();
    handler.removeCallbacksAndMessages(null);
    detached = true;
}

private Runnable r = new Runnable() {
    @Override
    public void run() {
        if (detached){
            return;
        }

        int t = getTextViewWidth();
        int m = getMeasuredWidth();
        // 文字较长时进入循环
        if (t > m) {
            onEffect();
        } else if (t < m && leftMargin != 0) {
            // 一旦宽度变化后有可能保留当前的leftMargin,修复回最左端显示
            fixLeftMargin();
        }

        restartTimer();
    }
};

private void onEffect() {
    translation += ViewCommonUtils.dipToPx(SPEED_IN_DIP) * reverseRemove;
    int maxMargin = getTextViewWidth() - getMeasuredWidth();
    if (translation < -(maxMargin + DECOR_RANGE)) {
        translation = -(maxMargin + DECOR_RANGE);
        reverseRemove *= -1;
    } else if (translation >= DECOR_RANGE) {
        translation = DECOR_RANGE;
        reverseRemove *= -1;
    }

    // 超出部分不取，只保留最大/小值，取值区间：【-maxMargin(最右边),0（最左边）】
    int margin = translation;
    if (margin < -maxMargin) {
        margin = -maxMargin;
    } else if (margin > 0) {
        margin = 0;
```

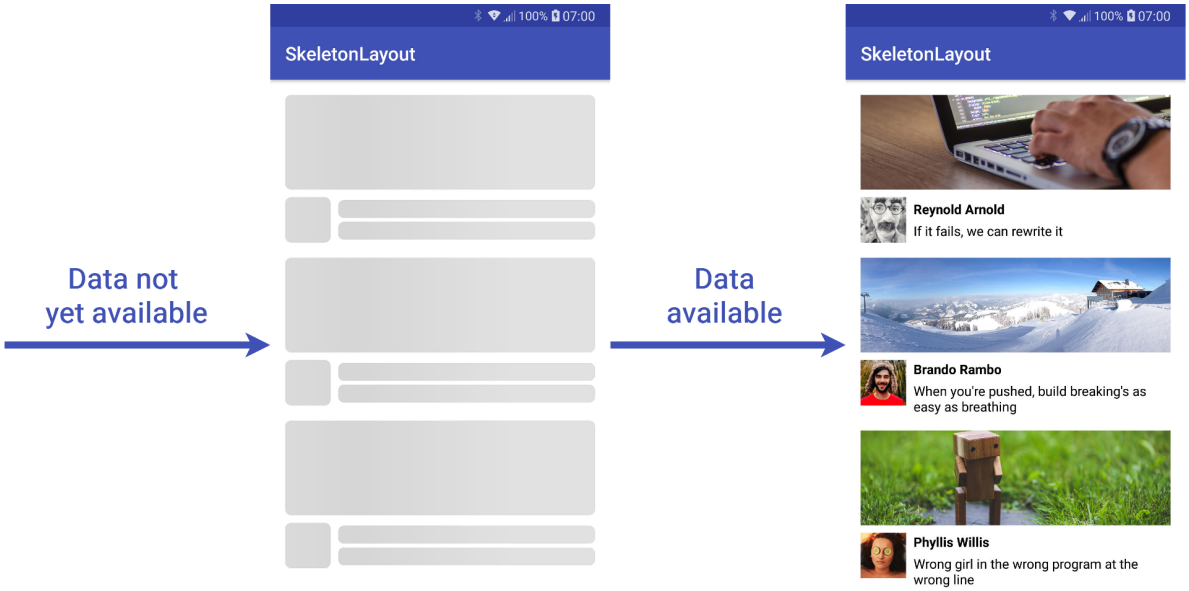
```
        }

        LayoutParams lp = (LayoutParams) getInnerTextView().getLayoutParams();
        if (lp != null) {
            lp.leftMargin = margin;
            leftMargin = margin;
            getInnerTextView().setLayoutParams(lp);
        }
    }

    private void fixLeftMargin() {
        leftMargin += ViewCommonUtils.dipToPx(SPEED_IN_DIP);

        if (leftMargin > 0) {
            leftMargin = 0;
        }
        LayoutParams lp = (LayoutParams) getInnerTextView().getLayoutParams();
        if (lp != null) {
            lp.leftMargin = leftMargin;
            getInnerTextView().setLayoutParams(lp);
        }
    }

    private int getTextViewWidth() {
        return getInnerTextView().getMeasuredWidth();
    }

    public TextView getInnerTextView() {
        if (innerView == null) {
            innerView = findViewById(R.id.tv_inner);
        }
        return (TextView) innerView;
    }
}
```

```
/**
 * px转换成dip
 */
public static int dipToPx(float dipValue) {
    // V6.3.5 2019/8/30-pengxr-修复
    // java.lang.NullPointerException: Attempt to invoke virtual method
'android.content.Context android.app.Activity.getApplicationContext()' on a null
object reference
    // at cn.kidyn.qdmedical160.util.ViewCommonUtils.getDensity(Unknown
Source:0)
    // at cn.kidyn.qdmedical160.util.ViewCommonUtils.dipToPx(Unknown Source:14)
    Context context =
QDApplicationContext.getInstance().getApplicationContext();
    return (int) (dipValue * context.getResources().getDisplayMetrics().density +
0.5f);
}
```

五.骨架屏

主要是基于 【基于开源项目4.0.0版本,此模块也与开源版本号保持一致
https://github.com/Faltenreich/SkeletonLayout/releases/tag/version%2F4.0.0）】

进行更改，主要修改是对 NextButton样式的修改，其余代码都是一样的。



1.效果【首页问医】

2.使用

1.针对某一个View

1.首先自己 需要对单个view单独写一套xml布局来实现每个骨架屏效果

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/ll_ask_doctor_skeleton"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="@dimen/dimen_70dp"
    android:background="@color/white"
    android:visibility="visible"
    android:orientation="vertical">
   <!-- <com.nykj.uikits.widget.button.NyTextButton
        android:layout_width="match_parent"
        android:layout_height="@dimen/dimen_40dp"
        android:layout_marginTop="@dimen/dimen_15dp"
        android:layout_marginRight="@dimen/dimen_15dp"
        android:layout_marginLeft="@dimen/dimen_15dp"
        app:uikits_cornersRadius="@dimen/dimen_22dp"
        app:uikits_normalBackgroundColor="@color/color_F5F5F5" />-->

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="110dp"
        android:layout_marginLeft="@dimen/dimen_15dp"
```

```
        android:layout_marginTop="@dimen/dimen_20dp"
        android:layout_marginRight="@dimen/dimen_3dp"
        android:orientation="horizontal">

        <com.nykj.uikits.widget.button.NyTextButton
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_marginRight="12dp"
            android:layout_weight="1.0"
            app:uikits_cornersRadius="@dimen/dimen_14dp"
            app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

        <com.nykj.uikits.widget.button.NyTextButton
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_marginRight="11dp"
            android:layout_weight="1.0"
            app:uikits_cornersRadius="@dimen/dimen_14dp"
            app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

        <com.nykj.uikits.widget.button.NyTextButton
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_marginRight="11dp"
            android:layout_weight="1.0"
            app:uikits_cornersRadius="@dimen/dimen_14dp"
            app:uikits_normalBackgroundColor="@color/color_F5F5F5" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="@dimen/dimen_15dp"
        android:layout_marginLeft="@dimen/dimen_8dp"
        android:layout_marginRight="@dimen/dimen_8dp"
        android:orientation="horizontal">

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1.0"
            android:gravity="center"
            android:orientation="vertical">

            <com.nykj.uikits.widget.button.NyTextButton
                android:layout_width="62dp"
                android:layout_height="36dp"
                app:uikits_cornersRadius="@dimen/dimen_6dp"
                app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

            <com.nykj.uikits.widget.button.NyTextButton
                android:layout_width="62dp"
                android:layout_height="13dp"
                android:layout_marginTop="@dimen/dimen_6dp"
                app:uikits_cornersRadius="@dimen/dimen_6dp"
                app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

        </LinearLayout>
```

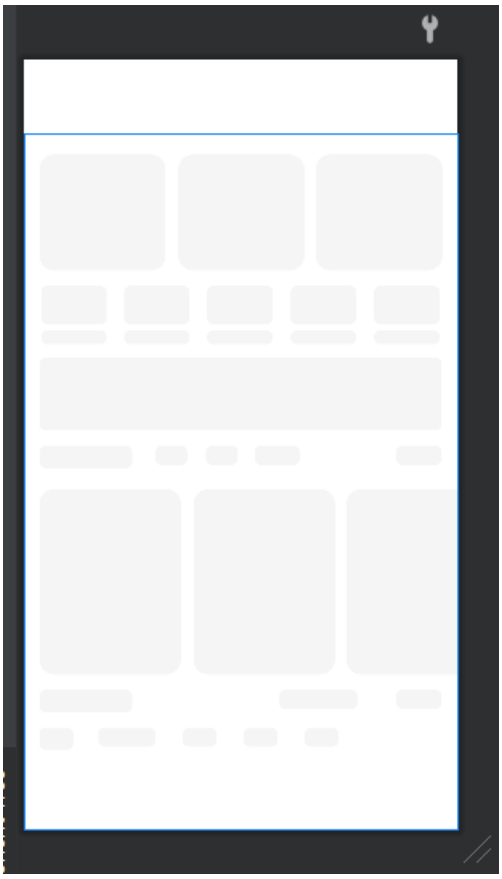
```
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1.0"
            android:gravity="center"
            android:orientation="vertical">

            <com.nykj.uikits.widget.button.NyTextButton
                android:layout_width="62dp"
                android:layout_height="36dp"
                app:uikits_cornersRadius="@dimen/dimen_6dp"
                app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

            <com.nykj.uikits.widget.button.NyTextButton
                android:layout_width="62dp"
                android:layout_height="13dp"
                android:layout_marginTop="@dimen/dimen_6dp"
                app:uikits_cornersRadius="@dimen/dimen_6dp"
                app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

        </LinearLayout>

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1.0"
            android:gravity="center"
            android:orientation="vertical">

            <com.nykj.uikits.widget.button.NyTextButton
                android:layout_width="62dp"
                android:layout_height="36dp"
                app:uikits_cornersRadius="@dimen/dimen_6dp"
                app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

            <com.nykj.uikits.widget.button.NyTextButton
                android:layout_width="62dp"
                android:layout_height="13dp"
                android:layout_marginTop="@dimen/dimen_6dp"
                app:uikits_cornersRadius="@dimen/dimen_6dp"
                app:uikits_normalBackgroundColor="@color/color_F5F5F5" />


        </LinearLayout>

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1.0"
            android:gravity="center"
            android:orientation="vertical">

            <com.nykj.uikits.widget.button.NyTextButton
                android:layout_width="62dp"
                android:layout_height="36dp"
                app:uikits_cornersRadius="@dimen/dimen_6dp"
                app:uikits_normalBackgroundColor="@color/color_F5F5F5" />
```

```
        <com.nykj.uikits.widget.button.NyTextButton
            android:layout_width="62dp"
            android:layout_height="13dp"
            android:layout_marginTop="@dimen/dimen_6dp"
            app:uikits_cornersRadius="@dimen/dimen_6dp"
            app:uikits_normalBackgroundColor="@color/color_F5F5F5" />


    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1.0"
        android:gravity="center"
        android:orientation="vertical">

        <com.nykj.uikits.widget.button.NyTextButton
            android:layout_width="62dp"
            android:layout_height="36dp"
            app:uikits_cornersRadius="@dimen/dimen_6dp"
            app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

        <com.nykj.uikits.widget.button.NyTextButton
            android:layout_width="62dp"
            android:layout_height="13dp"
            android:layout_marginTop="@dimen/dimen_6dp"
            app:uikits_cornersRadius="@dimen/dimen_6dp"
            app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

    </LinearLayout>
</LinearLayout>

<com.nykj.uikits.widget.button.NyTextButton
    android:layout_width="match_parent"
    android:layout_height="69dp"
    android:layout_marginLeft="@dimen/dimen_15dp"
    android:layout_marginTop="@dimen/dimen_13dp"
    android:layout_marginRight="@dimen/dimen_15dp"
    app:uikits_cornersRadius="@dimen/dimen_6dp"
    app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom"
    android:layout_marginLeft="@dimen/dimen_15dp"
    android:layout_marginTop="@dimen/dimen_15dp"
    android:layout_marginRight="@dimen/dimen_15dp"
    android:orientation="horizontal">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        android:orientation="horizontal">
```

```
            <com.nykj.uikits.widget.button.NyTextButton
                android:layout_width="88dp"
                android:layout_height="21dp"
                app:uikits_cornersRadius="@dimen/dimen_6dp"
                app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

            <com.nykj.uikits.widget.button.NyTextButton
                android:layout_width="30dp"
                android:layout_height="18dp"
                android:layout_marginLeft="@dimen/dimen_22dp"
                app:uikits_cornersRadius="@dimen/dimen_6dp"
                app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

            <com.nykj.uikits.widget.button.NyTextButton
                android:layout_width="30dp"
                android:layout_height="18dp"
                android:layout_marginLeft="@dimen/dimen_18dp"
                app:uikits_cornersRadius="@dimen/dimen_6dp"
                app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

            <com.nykj.uikits.widget.button.NyTextButton
                android:layout_width="43dp"
                android:layout_height="18dp"
                android:layout_marginLeft="@dimen/dimen_16dp"
                app:uikits_cornersRadius="@dimen/dimen_6dp"
                app:uikits_normalBackgroundColor="@color/color_F5F5F5" />
        </LinearLayout>

        <com.nykj.uikits.widget.button.NyTextButton
            android:layout_width="43dp"
            android:layout_height="18dp"
            android:layout_alignParentRight="true"
            app:uikits_cornersRadius="@dimen/dimen_6dp"
            app:uikits_normalBackgroundColor="@color/color_F5F5F5" />
    </RelativeLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="176dp"
        android:layout_marginLeft="@dimen/dimen_15dp"
        android:layout_marginTop="@dimen/dimen_20dp"
        android:orientation="horizontal">

        <com.nykj.uikits.widget.button.NyTextButton
            android:layout_width="134dp"
            android:layout_height="match_parent"
            android:layout_marginRight="12dp"
            app:uikits_cornersRadius="@dimen/dimen_14dp"
            app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

        <com.nykj.uikits.widget.button.NyTextButton
            android:layout_width="134dp"
            android:layout_height="match_parent"
            android:layout_marginRight="11dp"
            app:uikits_cornersRadius="@dimen/dimen_14dp"
            app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

        <com.nykj.uikits.widget.button.NyTextButton
```

```
            android:layout_width="134dp"
            android:layout_height="match_parent"
            android:layout_marginRight="11dp"
            app:uikits_cornersRadius="@dimen/dimen_14dp"
            app:uikits_normalBackgroundColor="@color/color_F5F5F5" />
    </LinearLayout>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        android:layout_marginLeft="@dimen/dimen_15dp"
        android:layout_marginTop="@dimen/dimen_15dp"
        android:layout_marginRight="@dimen/dimen_15dp"
        android:orientation="horizontal">
        <com.nykj.uikits.widget.button.NyTextButton
            android:layout_width="88dp"
            android:layout_height="21dp"
            app:uikits_cornersRadius="@dimen/dimen_6dp"
            app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

        <com.nykj.uikits.widget.button.NyTextButton
            android:layout_width="74dp"
            android:layout_height="18dp"
            android:layout_alignParentRight="true"
            android:layout_marginLeft="@dimen/dimen_22dp"
            android:layout_marginRight="@dimen/dimen_80dp"
            app:uikits_cornersRadius="@dimen/dimen_6dp"
            app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

        <com.nykj.uikits.widget.button.NyTextButton
            android:layout_width="43dp"
            android:layout_height="18dp"
            android:layout_alignParentRight="true"
            android:layout_marginLeft="@dimen/dimen_18dp"
            app:uikits_cornersRadius="@dimen/dimen_6dp"
            app:uikits_normalBackgroundColor="@color/color_F5F5F5" />
    </RelativeLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="@dimen/dimen_15dp"
        android:layout_marginTop="@dimen/dimen_15dp"
        android:layout_marginRight="@dimen/dimen_15dp"
        android:orientation="horizontal">

        <com.nykj.uikits.widget.button.NyTextButton
            android:layout_width="32dp"
            android:layout_height="21dp"
            app:uikits_cornersRadius="@dimen/dimen_6dp"
            app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

        <com.nykj.uikits.widget.button.NyTextButton
            android:layout_width="54dp"
            android:layout_height="18dp"
            android:layout_marginLeft="@dimen/dimen_24dp"
            app:uikits_cornersRadius="@dimen/dimen_6dp"
```

```
            app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

        <com.nykj.uikits.widget.button.NyTextButton
            android:layout_width="32dp"
            android:layout_height="18dp"
            android:layout_marginLeft="@dimen/dimen_26dp"
            app:uikits_cornersRadius="@dimen/dimen_6dp"
            app:uikits_normalBackgroundColor="@color/color_F5F5F5" />

        <com.nykj.uikits.widget.button.NyTextButton
            android:layout_width="32dp"
            android:layout_height="18dp"
            android:layout_marginLeft="@dimen/dimen_26dp"
            app:uikits_cornersRadius="@dimen/dimen_6dp"
            app:uikits_normalBackgroundColor="@color/color_F5F5F5" />
        <com.nykj.uikits.widget.button.NyTextButton
            android:layout_width="32dp"
            android:layout_height="18dp"
            android:layout_marginLeft="@dimen/dimen_26dp"
            app:uikits_cornersRadius="@dimen/dimen_6dp"
            app:uikits_normalBackgroundColor="@color/color_F5F5F5" />
    </LinearLayout>
</LinearLayout>
<!--</com.nykj.skeletonlayout.SkeletonLayout>-->
```



定义好骨架布局后，对于 SkeletonLayout的使用方式有两种，一个是 直接在原布局中使用，另外一种是 直接利用代码方式创建。

1.原布局中引入布局方式

在原布局中自己引入骨架屏以及骨架屏布局xml

```
<com.nykj.skeletonlayout.SkeletonLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/sl_ask_doctor_skeleton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:visibility="visible"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/iv_bg_top"
    >
    <include layout="@layout/fragment_ask_doctor_skeleton_1"/>
</com.nykj.skeletonlayout.SkeletonLayout>
```

加载数据时候开启骨架屏，加载完数据后，调用 hideSkeleton()隐藏骨架屏

```
var skeleton1 : SkeletonLayout?= null
fun initSkeleton() {
    try {
        llAskDoctorSkeleton =
getLayoutView()?.findViewById(R.id.ll_ask_doctor_skeleton)
        skeleton1 = binding.slAskDoctorSkeleton;
        skeleton1?.maskCornerRadius = 6f
        skeleton1?.showSkeleton()
    } catch (e:Exception) {
    }
}

fun hideSkeleton() {
    try {
        skeleton1.let {
            if (skeleton1?.isSkeleton() == true) {
                skeleton1?.showOriginal()
            }
        }
        binding.coorActivityTabAskDoctor.visibility = VISIBLE
        llAskDoctorSkeleton?.visibility = GONE
    } catch (e:Exception) {
    }
}
```

2.直接代码创建骨架屏view方式

```
// 针对View
@JvmOverloads
fun View.createSkeleton(
    config: SkeletonConfig = SkeletonConfig.default(context)
): Skeleton {
    // If this View already has a parent, we need to replace it with the
SkeletonLayout
    val parent = (parent as? ViewGroup)
```

```
    val index = parent?.indexOfChild(this) ?: 0
    val params = layoutParams

    parent?.removeView(this)

    val skeleton = SkeletonLayout(this, config)

    if (params != null) {
        skeleton.layoutParams = params
    }
    parent?.addView(skeleton, index)

    return skeleton
}

使用:
llAskDoctorSkeleton = getLayoutView()?.findViewById(R.id.ll_ask_doctor_skeleton)
llAskDoctorSkeleton.createSkeleton()
```

2.针对列表View

先定义对应的ItemView布局：【R.layout.activity_famous_doctor_home_page_skeleton_2】

然后可以同时针对不同的view进行分块使用即可：

```
    //针对RecyclerView 【View，itemView,数量】

Skeleton skeleton1, skeleton2;
public void initSkeleton(){

    skeleton1 = SkeletonLayoutUtils.applySkeleton(ad_department_list,
R.layout.activity_famous_doctor_home_page_skeleton_1, 1);
    skeleton1.setMaskCornerRadius(70);
    skeleton1.setMaskColor(getResources().getColor(R.color.f5));
    skeleton1.showSkeleton();


    skeleton2 = SkeletonLayoutUtils.applySkeleton(doctor_list,
R.layout.activity_famous_doctor_home_page_skeleton_2, 3);
    skeleton2.setMaskCornerRadius(80);
    skeleton2.setMaskColor(getResources().getColor(R.color.f5));
    skeleton2.showSkeleton();
}
```

3.实现原理

1.SkeletonLayout针对某个view

```
open class SkeletonLayout @JvmOverloads constructor(
    context: Context,
    attrs: AttributeSet? = null,
    defStyleAttr: Int = 0,
    originView: View? = null,
    private val config: SkeletonConfig = SkeletonConfig.default(context)
) : FrameLayout(context, attrs, defStyleAttr), Skeleton, SkeletonStyle by config
{}

整体 是一个 帧布局，并且其创建方式是利用 类委托方式进行创建。
```

1.创建方式

1.直接xml布局中引入的话，使用的是默认的创建方式

```
fun default(context: Context): SkeletonConfig {
    return SkeletonConfig(
        maskColor = ContextCompat.getColor(context,
SkeletonLayout.DEFAULT_MASK_COLOR),
        maskCornerRadius = SkeletonLayout.DEFAULT_MASK_CORNER_RADIUS,
        showShimmer = SkeletonLayout.DEFAULT_SHIMMER_SHOW,
        shimmerColor = ContextCompat.getColor(context,
SkeletonLayout.DEFAULT_SHIMMER_COLOR),
        shimmerDurationInMillis =
SkeletonLayout.DEFAULT_SHIMMER_DURATION_IN_MILLIS,
        shimmerDirection = SkeletonLayout.DEFAULT_SHIMMER_DIRECTION,
        shimmerAngle = SkeletonLayout.DEFAULT_SHIMMER_ANGLE
    )
}
```

当然还有其他创建方式

```
@file:JvmName("SkeletonLayoutUtils")

package com.nykj.skeletonlayout

import android.view.View
import android.view.ViewGroup
import androidx.annotation.ColorInt
import androidx.annotation.LayoutRes
import androidx.core.content.ContextCompat
import androidx.recyclerview.widget.RecyclerView
import androidx.viewpager2.widget.ViewPager2
import com.nykj.skeletonlayout.mask.SkeletonShimmerDirection
import com.nykj.skeletonlayout.recyclerview.SkeletonRecyclerView
import com.nykj.skeletonlayout.viewpager2.SkeletonViewPager2

private const val LIST_ITEM_COUNT_DEFAULT = 3

@JvmOverloads
```

```
fun View.createSkeleton(
    config: SkeletonConfig = SkeletonConfig.default(context)
): Skeleton {
    // If this View already has a parent, we need to replace it with the
SkeletonLayout
    val parent = (parent as? ViewGroup)
    val index = parent?.indexOfChild(this) ?: 0
    val params = layoutParams

    parent?.removeView(this)

    val skeleton = SkeletonLayout(this, config)

    if (params != null) {
        skeleton.layoutParams = params
    }
    parent?.addView(skeleton, index)

    return skeleton
}
```

这种方式就是可以定义好骨架屏xml之后，调用createSkeleton函数即可。

2.showSkeleton()

展示骨架屏入口是showSkeleton()

```
private var mask: SkeletonMask? = null

override fun showSkeleton() {
    isSkeleton = true

    if (isRendered) {
        if (childCount > 0) {
            // 设置当前子view暂时不可见
            views().forEach { it.visibility = View.INVISIBLE }
            // 设置ViewGroup可以绘制自身
            setWillNotDraw(false)
            // 绘制覆盖mask
            invalidateMask()
            // 调用mask view的绘制方法
            mask?.invalidate()

        } else {
            Log.i(tag(), "No views to mask")
        }
    }
}

private fun invalidateMask() {
        if (isRendered) {
            // 停止绘制
            mask?.stop()
            mask = null
            if (isSkeleton) {
```

```
                if (width > 0 && height > 0) {
                //创建并给当前view应用Mask
                    mask = SkeletonMaskFactory
                        .createMask(this, config)
                        .also { mask -> mask.mask(this, maskCornerRadius) }
                } else {
                    Log.e(tag(), "Failed to mask view with invalid width and
height")
                }
            }
        } else {
            Log.e(tag(), "Skipping invalidation until view is rendered")
        }
    }
```

从showSkeleton()函数来看，SkeletonLayout 更像是一个 容器，而骨架屏覆盖物以及其动画效果实际是在 SkeletonMask。

3.SkeletonMask

SkeletonMask 是一个 工具类，mask()函数主要是遍历传入进来的Group【这里是 SkeletonLayout 】中的子view，然后给其子view上面覆盖一个 半透明的遮罩。

```
package com.nykj.skeletonlayout.mask

import android.graphics.*
import android.util.Log
import android.view.View
import android.view.ViewGroup
import androidx.annotation.ColorInt
import androidx.recyclerview.widget.RecyclerView
import androidx.viewpager2.widget.ViewPager2
import com.nykj.skeletonlayout.R
import com.nykj.skeletonlayout.R.*
import com.nykj.skeletonlayout.tag
import com.nykj.skeletonlayout.views
import com.nykj.uikits.widget.button.NyTextButton

internal abstract class SkeletonMask(protected val parent: View, @ColorInt color:
Int) : SkeletonMaskable {

    var color: Int = color
        set(value) {
            paint.color = value
            field = value
        }

    private val bitmap: Bitmap by lazy { createBitmap() }
    private val canvas: Canvas by lazy { createCanvas() }
    protected val paint: Paint by lazy { createPaint() }

    protected open fun createBitmap(): Bitmap = Bitmap.createBitmap(parent.width,
parent.height, Bitmap.Config.ALPHA_8)

    protected open fun createCanvas(): Canvas = Canvas(bitmap)
```

```
    protected open fun createPaint(): Paint = Paint().also { it.color = color }

    fun draw(canvas: Canvas) {
        canvas.drawBitmap(bitmap, 0f, 0f, paint)
    }

    private fun draw(rectF: RectF, cornerRadius: Float, paint: Paint) {
        canvas.drawRoundRect(rectF, cornerRadius, cornerRadius, paint)
    }

    private fun draw(rect: Rect, paint: Paint) {
        canvas.drawRect(rect, paint)
    }

    fun mask(viewGroup: ViewGroup, maskCornerRadius: Float) {
        val xferPaint = Paint().apply {
            xfermode = PorterDuffXfermode(PorterDuff.Mode.SRC)
            isAntiAlias = maskCornerRadius > 0
        }
        mask(viewGroup, viewGroup, xferPaint, maskCornerRadius)
    }

    private fun mask(view: View, root: ViewGroup, paint: Paint, maskCornerRadius:
Float) {
        (view as? ViewGroup)?.let { viewGroup ->
            viewGroup.views().forEach { view -> mask(view, root, paint,
maskCornerRadius) }
        } ?: maskView(view, root, paint, maskCornerRadius)
    }

    private fun maskView(view: View, root: ViewGroup, paint: Paint,
maskCornerRadius: Float) {
        validate(view)

        val rect = Rect()
        view.getDrawingRect(rect)
        root.offsetDescendantRectToMyCoords(view, rect)

        if (maskCornerRadius > 0) {
            val rectF = RectF(rect.left.toFloat(), rect.top.toFloat(),
rect.right.toFloat(), rect.bottom.toFloat())
            (view as? NyTextButton) ?.let {
                val cornersRadius = view.getCornersRadius()
                draw(rectF, cornersRadius, paint) } ?:
            draw(rectF, maskCornerRadius, paint)
        } else {
            draw(rect, paint)
        }
    }

    private fun validate(view: View) {
        when (view) {
            is RecyclerView, is ViewPager2 -> Log.w(tag(), "Passing ViewGroup
with reusable children to SkeletonLayout - consider using applySkeleton()
instead")
        }
    }
```

```
}
```

他的子类有：SkeletonMaskShimmer 与 SkeletonMaskSolid， 而起到 光晕动画效果的实际是：SkeletonMaskShimmer 。

4.SkeletonMaskShimmer 光晕动画

这个类实际是通过handler不断的刷新当前矩阵的偏移度，然后给view进行绘制改变透明度。

```
package com.nykj.skeletonlayout.mask

import android.graphics.LinearGradient
import android.graphics.Matrix
import android.graphics.Paint
import android.graphics.Shader
import android.os.Handler
import android.view.View
import androidx.annotation.ColorInt
import com.nykj.skeletonlayout.isAttachedToWindowCompat
import com.nykj.skeletonlayout.refreshRateInSeconds
import kotlin.math.cos
import kotlin.math.floor
import kotlin.math.sin

internal class SkeletonMaskShimmer(
    parent: View,
    @ColorInt maskColor: Int,
    @ColorInt private val shimmerColor: Int,
    private val durationInMillis: Long,
    private val shimmerDirection: SkeletonShimmerDirection,
    private val shimmerAngle: Int
) : SkeletonMask(parent, maskColor) {

    private val refreshIntervalInMillis: Long by lazy { ((1000f /
parent.context.refreshRateInSeconds()) * .9f).toLong() }
    private val width: Float = parent.width.toFloat()
    private val matrix: Matrix = Matrix()
    private val shimmerGradient by lazy {
        val radians = Math.toRadians(shimmerAngle.toDouble())
        LinearGradient(
            0f,
            0f,
            cos(radians.toFloat()) * width,
            sin(radians.toFloat()) * width,
            intArrayOf(color, shimmerColor, color),
            null,
            Shader.TileMode.CLAMP
        )
    }

    private var animation: Handler? = null
    private var animationTask: Runnable? = null

    override fun invalidate() {
```

```
        when {
            parent.isAttachedToWindowCompat() && parent.visibility ==
View.VISIBLE -> start()
            else -> stop()
        }
    }

    override fun start() {
        if (animation == null) {
            animation = Handler()
            animationTask = object : Runnable {
                override fun run() {
                    updateShimmer()
                    animation?.postDelayed(this, refreshIntervalInMillis)
                }
            }
            animationTask?.let { task -> animation?.post(task) }
        }
    }

    override fun stop() {
        animationTask?.let { task -> animation?.removeCallbacks(task) }
        animation = null
    }

    override fun createPaint(): Paint {
        return Paint().also {
            it.shader = shimmerGradient
            it.isAntiAlias = true
        }
    }

    private fun updateShimmer() {
        matrix.setTranslate(currentOffset(), 0f)
        paint.shader.setLocalMatrix(matrix)
        parent.invalidate()
    }

    private fun currentOffset(): Float {
        val progress = when (shimmerDirection) {
            SkeletonShimmerDirection.LEFT_TO_RIGHT -> currentProgress()
            SkeletonShimmerDirection.RIGHT_TO_LEFT -> 1 - currentProgress()
        }
        val offset = width * 2
        val min = -offset
        val max = width + offset
        return progress * (max - min) + min
    }

    // Progress is time-dependent to support synchronization between uncoupled
views
    private fun currentProgress(): Float {
        val millis = System.currentTimeMillis()
        val current = millis.toDouble()
        val interval = durationInMillis
        val divisor = floor(current / interval)
        val start = interval * divisor
        val end = start + interval
```

```
        val percentage = (current - start) / (end - start)
        return percentage.toFloat()
    }
}
```

5.整体流程

```
SkeletonLayout:
    showSkeleton()--》
    invalidateMask()-》
    mask = SkeletonMaskFactory
                        .createMask(this, config)
                        .also { mask -> mask.mask(this, maskCornerRadius) }
==》创建 SkeletonMaskShimmer
=》 给骨架屏容器view创建一个渐变遮罩 mask.mask(this, maskCornerRadius)
=》invalidate()==》start() 开启handler 定时任务
=》不断绘制
 private fun updateShimmer() {
        matrix.setTranslate(currentOffset(), 0f)
        paint.shader.setLocalMatrix(matrix)
        parent.invalidate()
    }
```

2.SkeletonRecyclerView针对列表View

```
//创建RecycleView的骨架屏
@JvmOverloads
fun RecyclerView.applySkeleton(
    @LayoutRes listItemLayoutResId: Int,
    itemCount: Int = LIST_ITEM_COUNT_DEFAULT,
    config: SkeletonConfig = SkeletonConfig.default(context)
): Skeleton = SkeletonRecyclerView(this, listItemLayoutResId, itemCount, config)

//创建ViewPager2的骨架屏
@JvmOverloads
fun ViewPager2.applySkeleton(
    @LayoutRes listItemLayoutResId: Int,
    itemCount: Int = LIST_ITEM_COUNT_DEFAULT,
    config: SkeletonConfig = SkeletonConfig.default(context)
): Skeleton = SkeletonViewPager2(this, listItemLayoutResId, itemCount, config)
```

1.SkeletonRecyclerView

针对列表view的实现思路是 替换RecyclerView的ViewHolder，骨架屏ViewHolder以SkeletonLayout来实现光晕动画

```
package com.nykj.skeletonlayout.recyclerview

import androidx.annotation.LayoutRes
```

```
import androidx.recyclerview.widget.RecyclerView
import com.nykj.skeletonlayout.Skeleton
import com.nykj.skeletonlayout.SkeletonConfig
import com.nykj.skeletonlayout.SkeletonStyle

internal class SkeletonRecyclerView(
    private val recyclerView: RecyclerView,
    @LayoutRes layoutResId: Int,
    itemCount: Int,
    config: SkeletonConfig
) : Skeleton, SkeletonStyle by config {

    private val originalAdapter = recyclerView.adapter
    private var skeletonAdapter = SkeletonRecyclerViewAdapter(layoutResId,
itemCount, config)

    init {
        config.addValueObserver { skeletonAdapter.notifyDataSetChanged() }
    }

    override fun showOriginal() {
        recyclerView.adapter = originalAdapter
    }

    override fun showSkeleton() {
        recyclerView.adapter = skeletonAdapter
    }

    override fun isSkeleton(): Boolean {
        return recyclerView.adapter == skeletonAdapter
    }
}
```

```
package com.nykj.skeletonlayout.recyclerview

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.annotation.LayoutRes
import androidx.recyclerview.widget.RecyclerView
import com.nykj.skeletonlayout.SkeletonConfig
import com.nykj.skeletonlayout.SkeletonLayout
import com.nykj.skeletonlayout.createSkeleton

internal class SkeletonRecyclerViewAdapter(
    @LayoutRes private val layoutResId: Int,
    private val itemCount: Int,
    private val config: SkeletonConfig
) : RecyclerView.Adapter<SkeletonRecyclerViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
SkeletonRecyclerViewHolder {
        val originView = LayoutInflater.from(parent.context).inflate(layoutResId,
parent, false)
        val skeleton = originView.createSkeleton(config) as SkeletonLayout
        skeleton.layoutParams = originView.layoutParams
```

```
        skeleton.showSkeleton()
        return SkeletonRecyclerViewHolder(skeleton)
    }

    override fun onBindViewHolder(holder: SkeletonRecyclerViewHolder, position:
Int) = Unit

    override fun getItemCount() = itemCount
}
```

六.状态栏处理SystemUIDisplayer。

这个工具栏主要用于设置状态栏颜色变化的

1.使用

```
new SystemUIDisplayer(this)
        .setLightStatusBar(true)
        .setMode(MODE_IMMERSIVE)
        .setStatusBarColor(Color.TRANSPARENT)
        .display();
```

2.源码

1.ExceptionUtil

```
package com.ny.jiuyi160_doctor.common.util;

import java.io.PrintWriter;
import java.io.StringWriter;
import java.io.Writer;

public class ExceptionUtil {

    public static String getStackTraceString(Throwable tr) {
        if (tr == null) {
            return "";
        }

        Writer writer = new StringWriter();
        tr.printStackTrace(new PrintWriter(writer));
        return writer.toString();
    }
}
```

2.StatusBarHelper

```
package com.ny.jiuyi160_doctor.common.ui;

import android.view.Window;
import android.view.WindowManager;

import com.ny.jiuyi160_doctor.common.util.ExceptionUtil;
import com.ny.jiuyi160_doctor.common.util.TraceUtil;

import java.lang.reflect.Field;
import java.lang.reflect.Method;

/**
 * 顶部状态栏帮助类
 * Created by liangy on 2017/1/11.
 */
public class StatusBarHelper {

    /**
     * 设置状态栏图标为深色和魅族特定的文字风格
     * 可以用来判断是否为Flyme用户
     *
     * @param window 需要设置的窗口
     * @param dark   是否把状态栏字体及图标颜色设置为深色
     * @return boolean 成功执行返回true
     */
    static boolean setStatusBarContentModeFlyme(Window window, boolean dark) {
        boolean result = false;
        if (window != null) {
            try {
                WindowManager.LayoutParams lp = window.getAttributes();
                Field darkFlag = WindowManager.LayoutParams.class
                        .getDeclaredField("MEIZU_FLAG_DARK_STATUS_BAR_ICON");
                Field meizuFlags = WindowManager.LayoutParams.class
                        .getDeclaredField("meizuFlags");
                darkFlag.setAccessible(true);
                meizuFlags.setAccessible(true);
                int bit = darkFlag.getInt(null);
                int value = meizuFlags.getInt(lp);
                if (dark) {
                    value |= bit;
                } else {
                    value &= ~bit;
                }
                meizuFlags.setInt(lp, value);
                window.setAttributes(lp);
                result = true;
            } catch (Exception e) {
                TraceUtil.i("widget", ExceptionUtil.getStackTraceString(e));
            }
        }
        return result;
    }

    /**
     * 设置状态栏字体图标为深色，需要MIUIV6以上
```

```
     *
     * @param window 需要设置的窗口
     * @param dark   是否把状态栏字体及图标颜色设置为深色
     * @return boolean 成功执行返回true
     */
    static boolean setStatusBarContentModeMiui(Window window, boolean dark) {
        boolean result = false;
        if (window != null) {
            Class clazz = window.getClass();
            try {
                int darkModeFlag = 0;
                Class layoutParams =
Class.forName("android.view.MiuiWindowManager$LayoutParams");
                Field field =
layoutParams.getField("EXTRA_FLAG_STATUS_BAR_DARK_MODE");
                darkModeFlag = field.getInt(layoutParams);
                Method extraFlagField = clazz.getMethod("setExtraFlags",
int.class, int.class);
                if (dark) {
                    extraFlagField.invoke(window, darkModeFlag, darkModeFlag);//
状态栏透明且黑色字体
                } else {
                    extraFlagField.invoke(window, 0, darkModeFlag);//清除黑色字体
                }
                result = true;
            } catch (Exception e) {

            }
        }
        return result;
    }
}
```

3.SystemUIDisplayer

```
package com.ny.jiuyi160_doctor.common.ui;

import android.app.Activity;
import android.graphics.Color;
import android.os.Build;
import android.view.Window;

import com.ny.jiuyi160_doctor.common.system.RomHelper;
import com.readystatesoftware.systembartint.SystemBarTintManager;

import static android.view.View.SYSTEM_UI_FLAG_FULLSCREEN;
import static android.view.View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN;
import static android.view.View.SYSTEM_UI_FLAG_LAYOUT_STABLE;
import static android.view.View.SYSTEM_UI_FLAG_LIGHT_STATUS_BAR;
import static
android.view.WindowManager.LayoutParams.FLAG_DRAWS_SYSTEM_BAR_BACKGROUNDS;
import static android.view.WindowManager.LayoutParams.FLAG_TRANSLUCENT_STATUS;
import static
com.ny.jiuyi160_doctor.common.ui.StatusBarHelper.setStatusBarContentModeFlyme;
import static
com.ny.jiuyi160_doctor.common.ui.StatusBarHelper.setStatusBarContentModeMiui;
```

```
public class SystemUIDisplayer {

    public static final int MODE_IMMERSIVE = 0;
    public static final int MODE_FULLSCREEN = 1;

    private Activity activity;
    private int mode = MODE_IMMERSIVE;
    private boolean lightStatusBar = false;
    private Integer statusBarColor = null;

    public SystemUIDisplayer(Activity activity) {
        if (activity == null) {
            throw new IllegalArgumentException();
        }
        this.activity = activity;
    }

    public SystemUIDisplayer setMode(int mode) {
        this.mode = mode;
        return this;
    }

    public SystemUIDisplayer setLightStatusBar(boolean lightStatusBar) {
        this.lightStatusBar = lightStatusBar;
        return this;
    }

    public SystemUIDisplayer setStatusBarColor(int statusBarColor) {
        this.statusBarColor = statusBarColor;
        return this;
    }

    public void display() {
        Window window = activity.getWindow();
        boolean fullScreen = mode == MODE_FULLSCREEN;
        boolean immersive = mode == MODE_IMMERSIVE;

        int sysUIFlag = 0;

        if (fullScreen) {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.P) {
                // 这部分代码解决androidP刘海屏问题
                sysUIFlag = SYSTEM_UI_FLAG_FULLSCREEN |
SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN | SYSTEM_UI_FLAG_LAYOUT_STABLE;

                window.setFlags(FLAG_TRANSLUCENT_STATUS, 0);
                window.setFlags(FLAG_DRAWS_SYSTEM_BAR_BACKGROUNDS,
FLAG_DRAWS_SYSTEM_BAR_BACKGROUNDS);
                window.setStatusBarColor(statusBarColor != null ? statusBarColor
: Color.TRANSPARENT);
            } else {
                // 无刘海及androidO刘海不用把布局扩展到顶部
                sysUIFlag = SYSTEM_UI_FLAG_FULLSCREEN;
            }
        } else if (immersive) {
            /**
             * 窗口标记部分
             */
```

```
            // 窗口标记：透明状态栏
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
                int windowFlag_TRANSLUCENT_STATUS;
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                    // android21新方法，状态栏透明需要清除窗口透明标记
                    // android23浅色状态栏也需要把它清除
                    windowFlag_TRANSLUCENT_STATUS = 0;
                } else {
                    // android19需要设置窗口透明+tint使状态栏透明
                    windowFlag_TRANSLUCENT_STATUS = FLAG_TRANSLUCENT_STATUS;
                }
                window.setFlags(windowFlag_TRANSLUCENT_STATUS,
FLAG_TRANSLUCENT_STATUS);
            }

            // 窗口标记：绘制系统工具条背景
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                // android21引入，其他版本不设置
                window.setFlags(FLAG_DRAWS_SYSTEM_BAR_BACKGROUNDS,
FLAG_DRAWS_SYSTEM_BAR_BACKGROUNDS);
            }

            // 状态栏颜色
            if (statusBarColor != null) {
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                    window.setStatusBarColor(statusBarColor);
                } else if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT)
{
                    SystemBarTintManager tintManager = new
SystemBarTintManager(activity);
                    tintManager.setStatusBarTintEnabled(true);
                    tintManager.setStatusBarTintColor(statusBarColor);
                }
            }

            /**
             * 系统UI标记部分
             */

            // 系统UI标记：全屏
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN) {
                int sysUIFlag_FullScreen = 0;
                sysUIFlag |= sysUIFlag_FullScreen;
            }

            // 系统UI标记：布局扩展到全屏
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN) {
                int sysUIFlag_LAYOUT_FULLSCREEN =
SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN;
                sysUIFlag |= sysUIFlag_LAYOUT_FULLSCREEN;
            }

            // 系统UI标记：浅色状态栏。先处理厂商定制部分，失败了fallback到安卓M
            boolean fallback = true;
            if (RomHelper.isMiui()) {
                /**
```

```
                 * <a href="https://dev.mi.com/console/doc/detail?pId=1159">MIUI
9 & 10“状态栏黑色字符”实现方法变更通知</a>
                 */
                try {
                    if (!RomHelper.isMiuiVerGreaterThanOrEqual(2017, 7, 13)) {
                        fallback = !setStatusBarContentModeMiui(window,
lightStatusBar);
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                    fallback = true;
                }
            } else if (RomHelper.isFlyme()) {
                fallback = !setStatusBarContentModeFlyme(window,
lightStatusBar);
            }
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                if (fallback) {
                    int sysUIFlag_LIGHT_STATUS_BAR = lightStatusBar ?
SYSTEM_UI_FLAG_LIGHT_STATUS_BAR : 0;
                    sysUIFlag |= sysUIFlag_LIGHT_STATUS_BAR;
                }
            }

            // 系统UI标记：布局固定
            sysUIFlag |= SYSTEM_UI_FLAG_LAYOUT_STABLE;
        }

        window.getDecorView().setSystemUiVisibility(sysUIFlag);
    }
}
```

七.EditTextVisibleScrollView 滚动列表控制键盘弹出时不遮挡EditText

1.EditTextVisibleScrollView

使用：当作一个父容器使用

```
<cn.kidyn.qdmedical160.activity.payconsultation.order.widget.EditTextVisibleScro
llView
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:background="@color/colorGrayOnWhite"
    app:layout_constraintBottom_toTopOf="@+id/layout_btn_next"
    app:layout_constraintTop_toTopOf="parent">
```

源码：

```
package cn.kidyn.qdmedical160.activity.payconsultation.order.widget;

import android.content.Context;
import android.graphics.Rect;
```

```
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.core.widget.NestedScrollView;
import android.util.AttributeSet;
import android.view.View;
import android.view.ViewTreeObserver;

import org.greenrobot.eventbus.Subscribe;
import org.greenrobot.eventbus.ThreadMode;

import cn.kidyn.ny_common.util.EventBusUtil;
import cn.kidyn.qdmedical160.util.ViewCommonUtils;

/**
 * 控制键盘弹出时不遮挡 EditText
 * <p>
 * Created by pengxr on 2020/9/8.
 */
public class EditTextVisibleScrollView extends NestedScrollView implements
ViewTreeObserver.OnGlobalLayoutListener,
ViewTreeObserver.OnGlobalFocusChangeListener {

    private boolean isSoftKeyboardOpened;

    private int mOffset = -1;

    public EditTextVisibleScrollView(@NonNull Context context) {
        this(context, null);
    }

    public EditTextVisibleScrollView(@NonNull Context context, @Nullable
AttributeSet attrs) {
        this(context, attrs, 0);
    }

    public EditTextVisibleScrollView(@NonNull Context context, @Nullable
AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);

        getViewTreeObserver().addOnGlobalLayoutListener(this);
        getViewTreeObserver().addOnGlobalFocusChangeListener(this);
    }

    @Override
    public void onAttachedToWindow() {
        super.onAttachedToWindow();
        EventBusUtil.eventBusRegister(this);
    }

    @Override
    protected void onDetachedFromWindow() {
        super.onDetachedFromWindow();
        EventBusUtil.eventBusUnregister(this);
    }

    // ---------------------------------------------------------------------

    public static class WindowPopEvent {
```

```
        private int height; // -1 close

        public WindowPopEvent(int height) {
            this.height = height;
        }
    }

    @Subscribe(threadMode = ThreadMode.MAIN)
    public void onEventMainThread(WindowPopEvent event) {
        onWindowPop(event.height);
    }

    // ------------------------------------------------------------------------

    @Override
    public void onGlobalLayout() {
        final Rect windowRect = new Rect();
        getWindowVisibleDisplayFrame(windowRect);

        final int heightDiff = getRootView().getHeight() - (windowRect.bottom -
windowRect.top);
        if (!isSoftKeyboardOpened && heightDiff > 300) {
            isSoftKeyboardOpened = true;
            onSoftBoardPop();
        } else if (isSoftKeyboardOpened && heightDiff < 300) {
            isSoftKeyboardOpened = false;
            onClose();
        }
    }

    @Override
    public void onGlobalFocusChanged(View oldFocus, View newFocus) {
        onSoftBoardPop();
    }

    /**
     * 软键盘弹出
     */
    private void onSoftBoardPop() {
        final Rect windowRect = new Rect();
        getWindowVisibleDisplayFrame(windowRect);
        onPop(windowRect);
    }

    /**
     * 其他窗口弹出
     */
    private void onWindowPop(int height) {
        if (-1 == height) {
            onClose();
        } else {
            final Rect windowRect = new Rect();
            getWindowVisibleDisplayFrame(windowRect);
            windowRect.bottom -= height;
            onPop(windowRect);
        }
    }
```

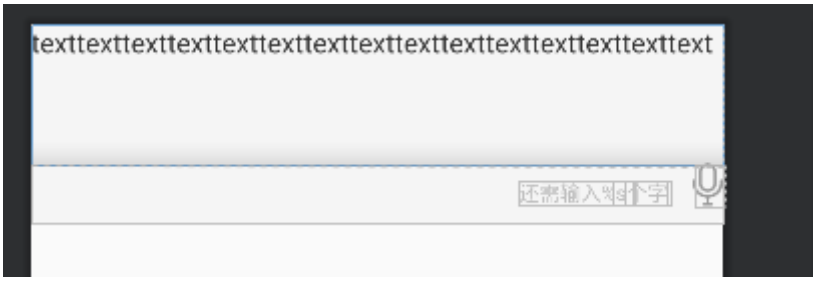
```
    private void onPop(Rect windowRect) {
        View focus = findFocus();
        if (null != focus) {
            // 保证父布局显示
            focus = (View) focus.getParent();
            final Rect editRect = new Rect();
            focus.getGlobalVisibleRect(editRect);
            // 计算偏移
            int offset = editRect.bottom - windowRect.bottom +
ViewCommonUtils.dipToPx(10);
//            Logger.showToastShort("" + offset);
            if (offset > 0) {
                mOffset = offset;
                smoothScrollBy(0, mOffset);
            }
        }
    }

    private void onClose() {
        mOffset = -1;
    }


}
```

这个控件主要是：

1.在重布局【onGlobalLayout】的时候，判断rootView的总部大于整个窗口的高度，如果是，表示 软键盘已经弹出，这时候调用 onPop()，查找 当前获取焦点的 控件，如果偏移量 大于 0，则利用 滚动，稍微往上滚动一点。

2.接收 WindowPopEvent 事件，比如 录音：这个看LimitEditTextV2这个控件

2.LimitEditTextV2 封装输入框



UI代码略过。

LimitEditTextV2 基于 ContainsEmojiEditText 封装了一些常用事件，比如 底部录音框的弹出 Audio2TextDialogFragment。

```
private void showAudio() {
    final FragmentActivity activity = (FragmentActivity)
ViewHelper.getActivityFromView(this);
    if (null != activity) {
        // 录音权限
        PermissionHelper.getInstance().requestRecordPermission(activity, new
PermissionHelper.PermissionGrantListener() {
            @Override
            public void onGranted() {
                Audio2TextDialogFragment.start(activity, editLimit, new
DialogInterface.OnDismissListener() {
                    @Override
```

```
                    public void onDismiss(DialogInterface dialog) {
                        EventBusUtil.post(new
EditTextVisibleScrollView.WindowPopEvent(-1));
                    }
                });
                editLimit.requestFocus();
                EventBusUtil.post(new
EditTextVisibleScrollView.WindowPopEvent(ViewCommonUtils.dipToPx(getContext(),
210)));
            }

            @Override
            public void onDenied(boolean shouldToSetting) {
                if (shouldToSetting) {
                    PermissionHelper.showSettingDialog(activity,
getContext().getString(R.string.permission_record_dialog_content));
                } else {
                    Logger.showToastShort(getContext(),
getContext().getString(R.string.permission_record_denied_hint));
                }
            }
        });
    }
}
```

下面是 ContainsEmojiEditText的源码，有兴趣可以看下。

```
/**
 * @title:ContainsEmojiEditText.java
 * TODO包含类名列表
 * Copyright (C) Shenzhen nykj Technology Co.Ltd.All right reserved.
 * @author : developer name
 * @version：v1.0,2015-6-17
 */


import android.content.Context;
import androidx.annotation.NonNull;
import android.text.InputFilter;
import android.text.SpannableString;
import android.text.Spanned;
import android.text.TextUtils;
import android.util.AttributeSet;
import android.util.Log;



public class ContainsEmojiEditText extends EditText {

    /**
     * 方法二
     */
    int maxLength = -1;
```

```
    public ContainsEmojiEditText(Context context, AttributeSet attrs, int
defStyle) {
        super(context, attrs, defStyle);
        addListener(attrs);
    }

    public ContainsEmojiEditText(Context context, AttributeSet attrs) {
        super(context, attrs);
        addListener(attrs);
    }

    public ContainsEmojiEditText(Context context) {
        super(context);
        addListener(null);
    }

    private void addListener(AttributeSet attrs) {
        try {
            if (attrs != null)
                maxLength =
attrs.getAttributeIntValue("http://schemas.android.com/apk/res/android",
"maxLength", -1);
            InputFilter filter = getInputFilter();
            // 输入框长度限制
            if (maxLength > 0)
                setFilters(new InputFilter[] { filter, new
InputFilter.LengthFilter(maxLength) });
            else
                setFilters(new InputFilter[] { filter });
        } catch (Exception e) {
            // TODO: handle exception
        }
    }

    @NonNull
    public InputFilter getInputFilter() {
        // 过滤输入法表情
        return new InputFilter() {
            @Override
            public CharSequence filter(CharSequence source, int start, int end,
Spanned dest, int dstart, int dend) {
                StringBuffer buffer = new StringBuffer();
                for (int i = start; i < end; i++) {
                    char c = source.charAt(i);
                    Log.i("char", (int) c + "");
                    // 第一个字符为以下时，过滤掉
                    if (c == 55356 || c == 55357 || c == 10060 || c == 9749 || c
== 9917 || c == 10067
                            || c == 10024 || c == 11088 || c == 9889 || c == 9729
|| c == 11093 || c == 9924) {
                        i++;
                        continue;
                    } else {
                        buffer.append(c);
                    }
                }
                if (source instanceof Spanned) {
                    SpannableString sp = new SpannableString(buffer);
```

```
                    TextUtils.copySpansFrom((Spanned) source, start, end, null,
sp, 0);
                    return sp;
                } else {
                    return buffer;
                }
            }
        };
    }
}
```

###

八.图片选择封装组件【封装思想参考】

1.SelectMediaWidget 入口组件

入口：医生主页/图文咨询， 问医页面/图文咨询，免费咨询

1.UI效果

2.源码分析

1.使用

```
<cn.kidyn.qdmedical160.media.select.SelectMediaWidget
    android:id="@+id/widget_select_media"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:column="4"
    app:maxImgCount="8"
    app:maxVideoCount="1"
    app:showOriginMode="true"/>
```

自定义属性:

```
<declare-styleable name="SelectMediaWidget">
    <attr name="maxImgCount" format="integer" />
    <attr name="maxVideoCount" format="integer" />
    <attr name="column" format="integer"/>
    <attr name="headViewVisible" format="integer">
        <enum name="visible"  value="0"/>
        <enum name="gone"  value="1" />
    </attr>
    <attr name="showTipsText" format="boolean"/>
    <attr name="showOriginMode" format="boolean"/>
</declare-styleable>
```

2.源码【部分解释在源码里面，2022.07.12版本】

```
package cn.kidyn.qdmedical160.media.select;

import android.app.Activity;
import android.content.Context;
import android.content.res.TypedArray;
import android.graphics.Color;
import android.text.Spannable;
import android.text.SpannableString;
import android.text.TextUtils;
import android.text.style.ForegroundColorSpan;
import android.util.AttributeSet;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import com.netease.nim.avchatkit.common.util.TimeUtil;
import com.ny.jiuyi160_doctor.common.ui.ViewHelper;
import com.nykj.onresult.IActivityResult;
import com.nykj.shareuilib.widget.recyclerview.VH;
import com.nykj.videoupload.base.VideoUploadParams;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.constraintlayout.widget.Group;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.FragmentActivity;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import cn.kidyn.qdmedical160.R;
import cn.kidyn.qdmedical160.activity.consultation.HistoryImageActivity;
import cn.kidyn.qdmedical160.activity.department.util.CollectionUtil;
import cn.kidyn.qdmedical160.activity.picture.preview.PicturePreviewActivity;
import cn.kidyn.qdmedical160.entity.consultation.HistoryImageEntity;
import cn.kidyn.qdmedical160.entity.session.ImageEntity;
import cn.kidyn.qdmedical160.entity.session.VideoEntity;
import cn.kidyn.qdmedical160.media.FetchMediaUtil;
import cn.kidyn.qdmedical160.media.PlayMediaUtil;
import cn.kidyn.qdmedical160.media.image.FetchImageParam;
import cn.kidyn.qdmedical160.media.image.IFetchImage;
import cn.kidyn.qdmedical160.media.video.FetchVideoParam;
import cn.kidyn.qdmedical160.media.video.IFetchVideo;
import cn.kidyn.qdmedical160.nybase.util.PreferencesHelper;
import cn.kidyn.qdmedical160.util.ComprehensiveJumpUntil;
import cn.kidyn.qdmedical160.util.ImageLoaderKit;
import cn.kidyn.qdmedical160.util.ViewCommonUtils;
import cn.kidyn.qdmedical160.view.dialog.DialogFactory;
```

```
import me.drakeet.multitype.ItemViewBinder;
import me.drakeet.multitype.MultiTypeAdapter;

/**
 * 选择 / 拍摄照片 & 视频
 *
 *
 * <p>
 * Created by pengxr on 2020/9/1.
 这是一个单独的控件
 */
public class SelectMediaWidget extends ConstraintLayout {

    private RecyclerView rvSelectMedia;
    private View btnHistory;
    private View tvTitleChooseMedia;
    private View headView;

    private int mImgWidth;
    private int mImgHeight;
    private int visible;

    //拍照或者选择图片配置的参数
    private FetchImageParam mImageParam = new FetchImageParam();
    private FetchVideoParam mVideoParam = new FetchVideoParam();

    // 选择的图片适配器
    private MediaAdapter mAdapter;

    public SelectMediaWidget(Context context) {
        this(context, null);
    }

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

    public SelectMediaWidget(Context context, AttributeSet attrs, int
defStyleAttr) {
        super(context, attrs, defStyleAttr);

        setBackgroundColor(Color.WHITE);

        init(attrs);
    }

    // ----------------------------------------------------------------------
----

    private String mHelpUrl;

    public void setHelpUrl(String url) {
        mHelpUrl = url;
    }

    // ----------------------------------------------------------------------
----
```

```
    private void init(AttributeSet attrs) {
        View root =
LayoutInflater.from(getContext()).inflate(R.layout.widget_choose_media, this,
true);
        headView = root.findViewById(R.id.head_view);
        rvSelectMedia = root.findViewById(R.id.rv_select_media);
        btnHistory = root.findViewById(R.id.tv_return_his_text);
        tvTitleChooseMedia = root.findViewById(R.id.tv_title_choose_media);

        initAttr(attrs);
        initView();
    }

    private void initAttr(AttributeSet attrs) {
        TypedArray typedArray = getContext().obtainStyledAttributes(attrs,
R.styleable.SelectMediaWidget);
        maxImgCount =
typedArray.getInteger(R.styleable.SelectMediaWidget_maxImgCount, 3);
        maxVideoCount =
typedArray.getInteger(R.styleable.SelectMediaWidget_maxVideoCount, 0);
        column = typedArray.getInteger(R.styleable.SelectMediaWidget_column, 3);
        visible =
typedArray.getInteger(R.styleable.SelectMediaWidget_headViewVisible, 0);
        showTipsText =
typedArray.getBoolean(R.styleable.SelectMediaWidget_showTipsText, true);
        showOriginMode =
typedArray.getBoolean(R.styleable.SelectMediaWidget_showOriginMode, false);
        typedArray.recycle();

        // 最大可选图片数
        mVideoParam.setSelectVideoCount(maxVideoCount);
        // 最大可选图片数
        mImageParam.setSelectImageCount(maxImgCount);
        // 是否显示原图按钮
        mImageParam.setOriginMode(showOriginMode);
    }

    private void initView() {
        headView.setVisibility(visible == 0 ? VISIBLE : GONE);

        // 图片尺寸
        int padding = ViewCommonUtils.dipToPx(getContext(), 25 + 5 * (column -
1)); // 15 + 10 + 5 *  (column - 1)
        mImgWidth = (ViewCommonUtils.getScreenWidth(getContext()) - padding) /
column;
        mImgHeight = mImgWidth + ViewCommonUtils.dipToPx(getContext(), 5);

        btnHistory.setOnClickListener(v -> {
            // 历史图片
            toSelectHistoryImg();
        });
        tvTitleChooseMedia.setOnClickListener(v -> {
            if (!TextUtils.isEmpty(mHelpUrl)) {
                ComprehensiveJumpUntil.gotoWebView2Activity(v.getContext(),
mHelpUrl, "");
            }
        });
```

```
        if (maxImgCount > 0) {
            // 选择图片
            mData.add(new SelectImageType());
        }
        if (maxVideoCount > 0) {
            // 选择视频
            mData.add(new SelectVideoType());
        }
        if ((maxImgCount > 0 || maxVideoCount > 0) && showTipsText) {
            // 提示
            mData.add(new TipType());
        }

        mAdapter = new MediaAdapter();
        mAdapter.register(SelectImageType.class, new SelectImageBinder());
        mAdapter.register(SelectVideoType.class, new SelectVideoBinder());
        mAdapter.register(ImageType.class, new ImageBinder());
        mAdapter.register(VideoType.class, new VideoBinder());
        mAdapter.register(TipType.class, new TipBinder());
        mAdapter.setItems(mData);

        GridLayoutManager layoutManager = new GridLayoutManager(getContext(),
column);
        layoutManager.setSpanSizeLookup(new GridLayoutManager.SpanSizeLookup() {
            @Override
            public int getSpanSize(int position) {
                if (mData.get(position) instanceof TipType) {
                    return 2;
                }
                return 1;
            }
        });

        rvSelectMedia.setLayoutManager(layoutManager);
        rvSelectMedia.setItemAnimator(null);
        rvSelectMedia.setAdapter(mAdapter);

    }

    // ------------------------------------------------------------------------
----

    /**
     * 选择图片底部弹框
     */
    private void toSelectImg() {
        FetchMediaUtil.fetchImage((FragmentActivity) getContext(), mImageParam,
new IFetchImage.Listener() {
            @Override
            public void onFail() {
                // do nothing
            }

            @Override
            public void onSuccess(@NonNull List<ImageEntity> entities, boolean
isOrigin) {
                for (int index = 0; index < entities.size(); index++) {
```

```
                    String fid =
PreferencesHelper.getInstance().getValue(PreferencesHelper.FID);
                    final String fileName = fid + "_" +
System.currentTimeMillis() + "_" + index;
                    ImageType imageType = new ImageType();
                    imageType.fileName = fileName; // 图片名（用于上传）
                    imageType.url = entities.get(index).getFilePath(); // 本地图片
地址（用于显示、上传）
                    addImage(imageType);
                }
            }
        });
    }

    /**
     * 选择视频底部弹框
     */
    private void toSelectVideo() {
        FetchMediaUtil.fetchVideo((FragmentActivity) getContext(), mVideoParam,
new IFetchVideo.Listener() {
            @Override
            public void onFail() {
                // do nothing
            }

            @Override
            public void onSuccess(@NonNull List<VideoEntity> entities, boolean
isOrigin) {
                for (int index = 0; index < entities.size(); index++) {
                    VideoEntity entity = entities.get(index);
                    VideoType videoType = new VideoType();
                    videoType.url = entity.getFilePath(); // 本地地址（用于上传）
                    videoType.duration = entity.getDuration(); // 视频时长（用于显
示、下单）
                    videoType.thumb_url = entity.getThumbPath(); // 本地缩略图地址
（用于显示、上传）
                    addVideo(videoType);
                }
            }
        });
    }

    /**
     * 选择历史图片
     */
    private void toSelectHistoryImg() {
        HistoryImageActivity.start(getContext(),
mImageParam.getSelectImageCount(), new IActivityResult<List<HistoryImageEntity>>
() {
            @Override
            public void onResult(@Nullable List<HistoryImageEntity> entities) {
                if (!CollectionUtil.isEmpty(entities)) {
                    for (int i = 0; i < entities.size(); i++) {
                        HistoryImageEntity imageItem = entities.get(i);
                        ImageType imageType = new ImageType();
                        imageType.url = imageItem.getImageUrl(); // 链接地址（用于
显示）
```

```
                        imageType.originImg = imageItem.origin_image; // （用于下
单）
                        addImage(imageType);
                    }
                }
            }
        });
    }

    // ------------------------------------------------------------------------
----
    // 数据
    // ------------------------------------------------------------------------
----

    private LinkedList<Object> mData = new LinkedList<>();

    // 最大可选图片数量
    private int maxImgCount;
    // 最大可选视频数量
    private int maxVideoCount;
    // 列
    private int column;

    /**
     * 是否显示提示文字 默认true
     */
    private boolean showTipsText;

    // 已选择图片数量
    private int selectedImgCount = 0;
    // 已选择视频数量
    private int selectedVideoCount = 0;

    /**
     * 是否显示原图按钮 默认false
     */
    private boolean showOriginMode;

    // 排列顺序：图片 - 视频 - 提示（初始状态有） - 选择图片 - 选择视频

    public void addImage(ImageType item) {
        mAdapter.add(selectedImgCount, item);

        selectedImgCount++;

        callback();

        int leftImgCount = maxImgCount - selectedImgCount;
        if (leftImgCount > 0) {
            mImageParam.setSelectImageCount(leftImgCount);
        } else {
            // 移除“选择图片”
            mAdapter.remove(new SelectImageType());
            btnHistory.setVisibility(View.GONE);
        }

        mAdapter.remove(new TipType());
```

```
    }

    private void deleteImage(ImageType item, int index) {
        mAdapter.remove(index);

        selectedImgCount--;

        callback();

        int leftImgCount = maxImgCount - selectedImgCount;
        mImageParam.setSelectImageCount(leftImgCount);
        if (!mData.contains(new SelectImageType())) {
            // 插入到“选择视频之前”
            int indexOfSelectVideo = mData.indexOf(new SelectVideoType());
            // 追加“选择图片”
            if (-1 == indexOfSelectVideo) {
                mAdapter.add(new SelectImageType());
            } else {
                mAdapter.add(indexOfSelectVideo, new SelectImageType());
            }
            btnHistory.setVisibility(View.VISIBLE);
        }

        if (0 == selectedImgCount && 0 == selectedVideoCount) {
            mAdapter.add(new TipType());
        }
    }

    private void addVideo(VideoType item) {
        mAdapter.add(selectedImgCount + selectedVideoCount, item);

        selectedVideoCount++;

        callback();

        int leftVideoCount = maxVideoCount - selectedVideoCount;
        if (leftVideoCount > 0) {
            mVideoParam.setSelectVideoCount(leftVideoCount);
        } else {
            // 移除“选择视频”
            mAdapter.remove(new SelectVideoType());
        }

        mAdapter.remove(new TipType());
    }

    private void deleteVideo(VideoType item, int index) {
        mAdapter.remove(index);

        selectedVideoCount--;

        callback();

        int leftVideoCount = maxVideoCount - selectedVideoCount;
        mVideoParam.setSelectVideoCount(leftVideoCount);
        if (!mData.contains(new SelectVideoType())) {
            // 追加“选择视频”
            mAdapter.add(new SelectVideoType());
```

```
        }

        if (0 == selectedImgCount && 0 == selectedVideoCount) {
            mAdapter.add(new TipType());
        }
    }

    // 文件
    public Map<String, String> getLocalImageFile() {
        Map<String, String> fileMap = new HashMap<>();
        for (Object type : mData) {
            if (type instanceof ImageType) {
                ImageType localImageType = (ImageType) type;
                if (localImageType.isLocal()) {
                    fileMap.put(localImageType.fileName, localImageType.url);
                }
            }
        }
        return fileMap;
    }

    /**
     * 获取已选择的图片数量
     * @return int:数量
     */
    public int getSelectedImgCount() {
        return selectedImgCount;
    }

    // 历史图片
    public List<String> getHisImageUrl() {
        List<String> hisUrls = new ArrayList<>();
        for (Object type : mData) {
            if (type instanceof ImageType) {
                ImageType localImageType = (ImageType) type;
                if (!localImageType.isLocal()) {
                    hisUrls.add(localImageType.originImg);
                }
            }
        }
        return hisUrls;
    }

    // todo
    public List<VideoUploadParams> getLocalVideoFile() {
        List<VideoUploadParams> videoPaths = new ArrayList<>();
        for (Object type : mData) {
            if (type instanceof VideoType) {
                VideoType localVideoType = (VideoType) type;
                if (localVideoType.isLocal()) {
                    VideoUploadParams entity = new VideoUploadParams();
                    entity.setDuration(localVideoType.duration);
                    entity.setVideoPath(localVideoType.url);
                    entity.setCoverPath(localVideoType.thumb_url);
                    videoPaths.add(entity);
                }
            }
        }
```

```
        return videoPaths;
    }

    /**
     * 预览图片
     */
    public ArrayList<String> getPreviewImgUrl() {
        ArrayList<String> urls = new ArrayList<>();
        for (Object type : mData) {
            if (type instanceof ImageType) {
                ImageType localImageType = (ImageType) type;
                urls.add(localImageType.url);
            }
        }
        return urls;
    }


    public String getHisImageUrlStr() {
        return TextUtils.join("|", getHisImageUrl());
    }

    // ------------------------------------------------------------------------
----

    private class MediaAdapter extends MultiTypeAdapter {

        void add(Object o) {
            mData.add(o);
            notifyItemInserted(mData.size() - 1);
        }

        void add(int index, Object o) {
            mData.add(index, o);
            notifyItemInserted(index);
        }

        void remove(int index) {
            mData.remove(index);
            notifyItemRemoved(index);
        }

        void remove(Object o) {
            int index = mData.indexOf(o);
            if (-1 != index) {
                remove(index);
            }
        }

        MediaAdapter() {

        }

    }

    private static class SelectImageType {

        @Override
```

```
        public boolean equals(@Nullable Object obj) {
            return obj instanceof SelectImageType;
        }
    }

    private static class SelectVideoType {

        @Override
        public boolean equals(@Nullable Object obj) {
            return obj instanceof SelectVideoType;
        }
    }

    // 提示（没有选择图片或视频时才显示）
    private static class TipType {

        @Override
        public boolean equals(@Nullable Object obj) {
            return obj instanceof TipType;
        }
    }

    /**
     * @see HistoryImageEntity
     */
    public static class ImageType {
        public String fileName; // 图片名（用于图片上传）
        public String url; // 图片本地地址或链接地址（用于显示图片）
        public String originImg; // 原始图片地址（用于下单），为空说明是本地图片

        private boolean isLocal() {
            return null == originImg;
        }
    }

    /**
     * 将来可能会增加历史视频，因此不依赖VideoEntity
     */
    public static class VideoType {
        public String url; // 视频本地地址
        public int duration; // 视频时长，秒
        public String thumb_url;
        public String originVideo; // （v6.5.9暂无历史视频）

        private boolean isLocal() {
            return null == originVideo;
        }
    }

    private class SelectImageBinder extends ItemViewBinder<SelectImageType,
ImgViewHolder> {

        @NonNull
        @Override
        protected ImgViewHolder onCreateViewHolder(@NonNull LayoutInflater
layoutInflater, @NonNull ViewGroup viewGroup) {
```

```
            return new
ImgViewHolder(layoutInflater.inflate(R.layout.item_photo_gridview, viewGroup,
false));
        }

        @Override
        protected void onBindViewHolder(@NonNull ImgViewHolder vh, @NonNull
SelectImageType item) {
            vh.image.setImageResource(R.drawable.icon_select_photo);
            vh.iv_close.setVisibility(View.INVISIBLE);
            vh.itemView.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                    // 选择图片
                    toSelectImg();
                }
            });
        }
    }

    private class SelectVideoBinder extends ItemViewBinder<SelectVideoType,
ImgViewHolder> {

        @NonNull
        @Override
        protected ImgViewHolder onCreateViewHolder(@NonNull LayoutInflater
layoutInflater, @NonNull ViewGroup viewGroup) {
            return new
ImgViewHolder(layoutInflater.inflate(R.layout.item_photo_gridview, viewGroup,
false));
        }

        @Override
        protected void onBindViewHolder(@NonNull ImgViewHolder vh, @NonNull
SelectVideoType item) {
            vh.image.setImageResource(R.drawable.icon_select_video);
            vh.iv_close.setVisibility(View.INVISIBLE);
            vh.itemView.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                    // 选择视频
                    toSelectVideo();
                }
            });
        }
    }

    private class ImageBinder extends ItemViewBinder<ImageType, ImgViewHolder> {

        private ImageType mItem;

        @NonNull
        @Override
        protected ImgViewHolder onCreateViewHolder(@NonNull LayoutInflater
layoutInflater, @NonNull ViewGroup viewGroup) {
            return new
ImgViewHolder(layoutInflater.inflate(R.layout.item_photo_gridview, viewGroup,
false));
```

```
        }

        @Override
        protected void onBindViewHolder(@NonNull final ImgViewHolder vh, @NonNull
ImageType item) {
            mItem = item;
            ImageLoaderKit.display(vh.image, item.url);
            vh.iv_close.setVisibility(View.VISIBLE);
            vh.iv_close.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                    // 删除图片
                    Activity activity = ViewHelper.getActivityFromView(v);
                    final DialogFactory dialogFactory = new
DialogFactory().create(activity, R.layout.dialog_common);
                    dialogFactory.setText(R.id.tv_dialog_content,
getResources().getString(R.string.free_consultation_delete_picture))
                            .setText(R.id.tv_cancel,
getResources().getString(R.string.cancel))
                            .setText(R.id.tv_confirm,
getResources().getString(R.string.confirm))
                            .setConfirm(R.id.tv_confirm, new
DialogFactory.OnClickListener() {
                                @Override
                                public void onClick() {
                                    dialogFactory.dismiss();
                                    deleteImage(mItem, vh.getAdapterPosition());
                                }
                            })
                            .setCancel(R.id.tv_cancel, new
DialogFactory.OnClickListener() {
                                @Override
                                public void onClick() {
                                    dialogFactory.dismiss();
                                }
                            })
                            .show();
                }
            });
            vh.itemView.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                    // 预览图片
                    Activity activity = ViewHelper.getActivityFromView(v);
                    if (null != activity) {
                        PicturePreviewActivity.start(activity,
getPreviewImgUrl(), vh.getAdapterPosition());
                    }
                }
            });
        }
    }

    private class VideoBinder extends ItemViewBinder<VideoType, ImgViewHolder> {

        private VideoType mItem;

        @NonNull
```

```
        @Override
        protected ImgViewHolder onCreateViewHolder(@NonNull LayoutInflater
layoutInflater, @NonNull ViewGroup viewGroup) {
            return new
ImgViewHolder(layoutInflater.inflate(R.layout.item_photo_gridview, viewGroup,
false));
        }

        @Override
        protected void onBindViewHolder(@NonNull final ImgViewHolder vh, @NonNull
VideoType item) {
            mItem = item;
            ImageLoaderKit.display(vh.image, item.thumb_url);
            vh.iv_close.setVisibility(View.VISIBLE);
            vh.iv_close.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                    Activity activity = ViewHelper.getActivityFromView(v);
                    final DialogFactory dialogFactory = new
DialogFactory().create(activity, R.layout.dialog_common);
                    dialogFactory.setText(R.id.tv_dialog_content,
getResources().getString(R.string.free_consultation_delete_video))
                            .setText(R.id.tv_cancel,
getResources().getString(R.string.cancel))
                            .setText(R.id.tv_confirm,
getResources().getString(R.string.confirm))
                            .setConfirm(R.id.tv_confirm, new
DialogFactory.OnClickListener() {
                                @Override
                                public void onClick() {
                                    dialogFactory.dismiss();
                                    deleteVideo(mItem, vh.getAdapterPosition());
                                }
                            })
                            .setCancel(R.id.tv_cancel, new
DialogFactory.OnClickListener() {
                                @Override
                                public void onClick() {
                                    dialogFactory.dismiss();
                                }
                            })
                            .show();
                }
            });
            vh.group_layer_video.setVisibility(View.VISIBLE);
            vh.tv_desc.setVisibility(View.VISIBLE);
            vh.tv_desc.setText(TimeUtil.secToTimeShort(item.duration));
            vh.itemView.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                    // 预览视频
                    PlayMediaUtil.playVideo(v.getContext(), mItem.url);
                }
            });
        }
    }

    private class ImgViewHolder extends RecyclerView.ViewHolder {
```

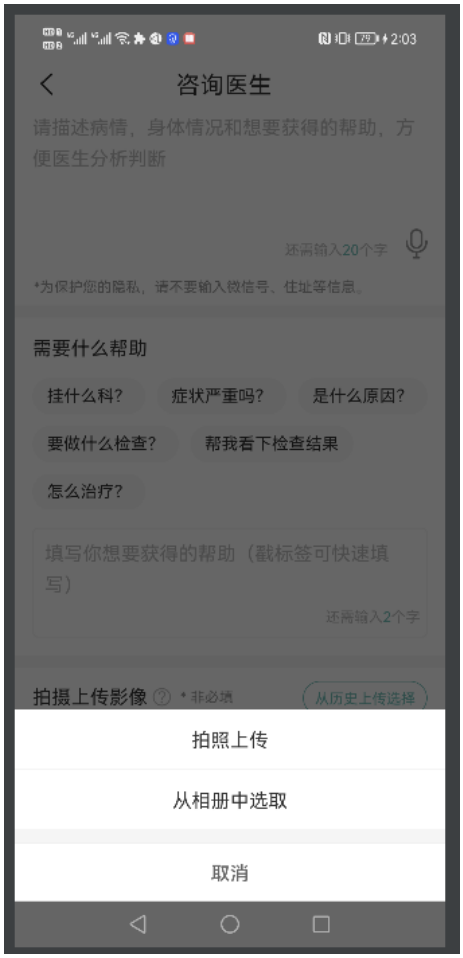
```
        private ImageView image;
        private ImageView iv_close;
        private TextView tv_desc;
        private Group group_layer_video;

        ImgViewHolder(View itemView) {
            super(itemView);
            image = itemView.findViewById(R.id.item_grida_image);
            iv_close = itemView.findViewById(R.id.iv_close);
            tv_desc = itemView.findViewById(R.id.tv_media_desc);
            group_layer_video = itemView.findViewById(R.id.group_layer_video);

            ViewGroup.LayoutParams params = itemView.getLayoutParams();
            params.height = mImgHeight;
            params.width = mImgWidth;
            itemView.setLayoutParams(params);
        }
    }

    private class TipBinder extends ItemViewBinder<TipType, VH> {

        @NonNull
        @Override
        protected VH onCreateViewHolder(@NonNull LayoutInflater layoutInflater,
@NonNull ViewGroup viewGroup) {
            View itemRoot =
layoutInflater.inflate(R.layout.item_photot_tip_gridview, viewGroup, false);
            TextView tvImage = itemRoot.findViewById(R.id.tv_tip_image);
            TextView tvVideo = itemRoot.findViewById(R.id.tv_tip_video);
            if (maxImgCount > 0) {
                String numStr = "" + maxImgCount;
                String tip =
getContext().getString(R.string.select_media_image_limit, numStr);
                int color = ContextCompat.getColor(getContext(),
R.color.color_FFFF9F4F);
                SpannableString spannableStr = new SpannableString(tip);
                spannableStr.setSpan(new ForegroundColorSpan(color), 3, 3 +
numStr.length(), Spannable.SPAN_INCLUSIVE_EXCLUSIVE);
                tvImage.setText(spannableStr);
                tvImage.setVisibility(View.VISIBLE);
            }
            if (maxVideoCount > 0) {
                String numStr = "30";
                String tip =
getContext().getString(R.string.select_media_video_limit, numStr);
                int color = ContextCompat.getColor(getContext(),
R.color.color_FFFF9F4F);
                SpannableString spannableStr = new SpannableString(tip);
                spannableStr.setSpan(new ForegroundColorSpan(color), 6, 6 +
numStr.length(), Spannable.SPAN_INCLUSIVE_EXCLUSIVE);
                tvVideo.setText(spannableStr);
                tvVideo.setVisibility(View.VISIBLE);
            }

            // 尺寸 间距12dp
            if (mImgWidth > 0) {
                ViewGroup.LayoutParams imageParam = itemRoot.getLayoutParams();
```

```
                imageParam.height = mImgWidth -
ViewCommonUtils.dipToPx(getContext(), 6);
                itemRoot.setLayoutParams(imageParam);
            }
            return new VH(itemRoot);
        }

        @Override
        protected void onBindViewHolder(@NonNull VH vh, @NonNull TipType tipType)
{
            // do nothing
        }
    }

    // ---------------------------------------------------------------------------
----

    private Listener mListener;

    private void callback() {
        if (null != mListener) {
            mListener.onUpdate();
        }
    }

    public void setListener(Listener listener) {
        this.mListener = listener;
    }

    public interface Listener {

        void onUpdate();

    }

}
```

2.选择图片或者拍照底部弹框

1.选择拍照【变化的丢出去外面，相同的封装】

1.UI



2.源码

对应的代码如下：

```
/**
 * 选择图片
 */
private void toSelectImg() {
    FetchMediaUtil.fetchImage((FragmentActivity) getContext(), mImageParam, new
IFetchImage.Listener() {
        @Override
        public void onFail() {
            // do nothing
        }

        @Override
        public void onSuccess(@NonNull List<ImageEntity> entities, boolean
isOrigin) {
            for (int index = 0; index < entities.size(); index++) {
                String fid =
PreferencesHelper.getInstance().getValue(PreferencesHelper.FID);
                final String fileName = fid + "_" + System.currentTimeMillis() +
"_" + index;
                ImageType imageType = new ImageType();
                imageType.fileName = fileName; // 图片名（用于上传）
                imageType.url = entities.get(index).getFilePath(); // 本地图片地址
（用于显示、上传）
                addImage(imageType);
```

```
            }
        }
    });
}
```

这边是构造了两个操作动作，并利用【通用底部弹框进行实现CommonBottomSheetFragment】

```
/**
 * 获取图片途径选择：拍照或相册
 */
public static void fetchImage(final FragmentActivity activity, @Nullable
FetchImageParam param, @NonNull final IFetchImage.Listener callback) {
    CaptureImageAction captureAction = new CaptureImageAction(param);
    SelectImageAction selectAction = new SelectImageAction(param);

    ImageButton captureButton = new ImageButton(R.string.capture_image,
captureAction);
    ImageButton selectButton = new ImageButton(R.string.select_image,
selectAction);

    List<ImageButton> buttons = Arrays.asList(captureButton, selectButton);

    CommonBottomSheetFragment<ImageButton> dialog =
CommonBottomSheetFragment.newInstance(buttons, -1, false);
    dialog.setInjector(new
CommonBottomSheetFragment.BindViewInjector<ImageButton>() {
        @Override
        public void inject(TextView tv, ImageButton item) {
            tv.setText(item.textRes);
        }
    });

    dialog.setListener(new
CommonBottomSheetFragment.OnItemSelectListener<ImageButton>() {
        @Override
        public void onSelect(DialogFragment dialog, ImageButton object) {
            object.action.fetchImage(activity, callback);
        }
    });

    dialog.show(activity);
}
```

3.ImageButton操作类

```
private static class ImageButton {
    @StringRes
    private int textRes;
    private IFetchImage action;

    ImageButton(int textRes, IFetchImage action) {
        this.textRes = textRes;
        this.action = action;
    }
}
```

4.IFetchImage

```
public interface IFetchImage {

    /**
     * 拍摄图片
     */
    void fetchImage(@NonNull final Context context, @NonNull final Listener
callback);

    interface Listener {

        void onFail();

        /**
         * @param entities 结果集 非空、非空集
         * @param isOrigin 是否原图（暂时只在图片选择时有意义，图片拍摄恒为false）
         */
        void onSuccess(@NonNull List<ImageEntity> entities, boolean isOrigin);
    }
}
```

5.BaseFetchImageAction 操作基类

```
package cn.kidyn.qdmedical160.media.image;

import android.content.Context;

import java.util.Collections;

import androidx.annotation.CallSuper;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import cn.kidyn.qdmedical160.entity.session.ImageEntity;
import cn.kidyn.qdmedical160.media.FetchMediaUtil;

/**
 * Created by pengxr on 2020/9/10.
 */
public class BaseFetchImageAction implements IFetchImage {

    protected Context context;
    protected Listener callback;

    @NonNull
```

```
    protected FetchImageParam param = new FetchImageParam();

    BaseFetchImageAction() {
    }

    // temp
    public BaseFetchImageAction(@Nullable FetchImageParam param) {
        if (null != param) {
            this.param = param;
        }
    }

    public void setParam(@Nullable FetchImageParam param) {
        if (null != param) {
            this.param = param;
        }
    }

    @Override
    @CallSuper
    public void fetchImage(@NonNull Context context, @NonNull Listener callback)
{
        this.context = context;
        this.callback = callback;
    }

    /**
     * 裁剪
     */
    protected void crop(@NonNull ImageEntity entity) {
        if (param.isCrop()) {
            // 开启裁剪
            param.getCropParam().setSourceFilePath(entity.getFilePath());
            FetchMediaUtil.cropImage(context, param.getCropParam(), new
ICropImage.Listener() {
                @Override
                public void onFail() {
                    callback.onFail();
                }

                @Override
                public void onSuccess(@NonNull ImageEntity cropEntity) {
                    callback.onSuccess(Collections.singletonList(cropEntity),
false);
                }
            });
        } else {
            callback.onSuccess(Collections.singletonList(entity), false);
        }
    }
}
```

6.拍照

```
package cn.kidyn.qdmedical160.media.image;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.content.pm.ResolveInfo;
import android.net.Uri;
import android.provider.MediaStore;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;

import com.ny.jiuyi160_doctor.common.ui.ViewHelper;
import com.nykj.onresult.ActivityIntentLauncher;
import com.nykj.onresult.ActivityResult;

import java.io.File;
import java.util.List;
import java.util.UUID;

import cn.kidyn.qdmedical160.R;
import cn.kidyn.qdmedical160.database.base.AsyncQueryHandler;
import cn.kidyn.qdmedical160.entity.session.ImageEntity;
import cn.kidyn.qdmedical160.media.util.CompressUtil;
import cn.kidyn.qdmedical160.media.util.MediaUtil;
import cn.kidyn.qdmedical160.nybase.util.Logger;
import cn.kidyn.qdmedical160.nybase.util.PermissionHelper;
import cn.kidyn.qdmedical160.nybase.view.dialog.PermissionPurposeDialog;

/**
 * 拍照行为
 * <p>
 * Created by pengxr on 2020/9/9.
 */
public class CaptureImageAction extends BaseFetchImageAction {

    public CaptureImageAction(@Nullable FetchImageParam param) {
        super(param);
    }

    private String tempCameraFilePath;

    /**
     * 参考自 ImageChooser#chooseCamera()
     */
    @Override
    public void fetchImage(@NonNull final Context context, @NonNull final
Listener callback) {
        super.fetchImage(context, callback);
        // 1. 权限
        final Activity activity = ViewHelper.getWrappedActivity(context);
        if (null == activity) {
            return;
        }
```

```
        PermissionHelper.getInstance().requestCameraPermission(activity, new
PermissionHelper.PermissionGrantListener() {
            @Override
            public void onGranted() {
                // 1.1 权限授予
                doFetch();
            }

            @Override
            public void onDenied(boolean shouldToSetting) {
                // 1.2 权限拒绝
                if (shouldToSetting) {
                    PermissionHelper.showSettingDialog(activity,
context.getString(R.string.permission_camera_dialog_content), new
PermissionPurposeDialog.ResultCallback() {
                        @Override
                        public void onOk() {
                            callback.onFail();
                        }

                        @Override
                        public void onCancel() {
                            callback.onFail();
                        }
                    });
                } else {
                    Logger.showToastShort(context,
R.string.permission_camera_denied_hint);
                    callback.onFail();
                }
            }
        });
    }

    private void doFetch() {
        try {
            File imageFile = new File(MediaUtil.getImagesDir(), UUID.randomUUID()
+ ".jpg");
            Uri imageUri = MediaUtil.getUriForFile(context, imageFile);
            tempCameraFilePath = imageFile.getAbsolutePath();

            final Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
            intent.putExtra(MediaStore.EXTRA_OUTPUT, imageUri);
            MediaUtil.grantPermissionForFileShareIntent(intent, true, true);

            // 判断存在系统拍照组件
            List<ResolveInfo> activities =
context.getPackageManager().queryIntentActivities(intent,
PackageManager.MATCH_DEFAULT_ONLY);
            if (activities.size() <= 0) {
                Logger.showToastShort(context,
R.string.session_dont_take_photo);
                return;
            }

            ActivityResult.with(context).start(new ActivityIntentLauncher() {
                @Override
                public Intent createIntent() {
```

```
                    return intent;
                }

                @Override
                public void onActivityResult(int resultCode, Intent data) {
                    // 返回
                    onResult(resultCode, data);
                }
            });
        } catch (Exception e) {
            tempCameraFilePath = null;
            Logger.showToastShort(context, R.string.session_take_photo_fail);
        }
    }

    /**
     * 处理结果
     */
    private void onResult(int resultCode, Intent data) {
        if (Activity.RESULT_OK != resultCode) {
            callback.onFail();
            return;
        }
        if (null == tempCameraFilePath) {
            return;
        }
        new AsyncQueryHandler<ImageEntity>() {
            @Override
            public ImageEntity queryInThread() {
                // 压缩图片
                return CompressUtil.compress(context, tempCameraFilePath,
param.getUploadMaxSize());
            }

            @Override
            public void complete(ImageEntity entity) {
                if (null != entity) {
                    // 裁剪图片
                    crop(entity);
                } else {
                    callback.onFail();
                }
            }
        }.execute();
    }
}
```

7.选择图片SelectImageAction

```
package cn.kidyn.qdmedical160.media.image;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.net.Uri;
```

```
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;

import com.ny.jiuyi160_doctor.common.ui.ViewHelper;
import com.ny.mqttuikit.media.image.util.ImageUtil;
import com.nykj.onresult.ActivityIntentLauncher;
import com.nykj.onresult.ActivityResult;

import java.util.ArrayList;
import java.util.List;

import cn.kidyn.qdmedical160.R;
import cn.kidyn.qdmedical160.activity.picture.PictureSelectorActivity;
import cn.kidyn.qdmedical160.activity.picture.SelectMediaType;
import cn.kidyn.qdmedical160.activity.picture.bean.Picture;
import cn.kidyn.qdmedical160.database.base.AsyncQueryHandler;
import cn.kidyn.qdmedical160.entity.session.ImageEntity;
import cn.kidyn.qdmedical160.media.util.CompressUtil;
import cn.kidyn.qdmedical160.nybase.util.Logger;
import cn.kidyn.qdmedical160.nybase.util.PermissionHelper;
import cn.kidyn.qdmedical160.nybase.view.dialog.PermissionPurposeDialog;
import cn.kidyn.qdmedical160.util.Pictureutil;

import static android.app.Activity.RESULT_OK;

/**
 * 选择图片行为
 * <p>
 * Created by pengxr on 2020/9/10.
 */
public class SelectImageAction extends BaseFetchImageAction {

    public SelectImageAction(@Nullable FetchImageParam param) {
        super(param);
    }

    @Override
    public void fetchImage(@NonNull final Context context, @NonNull final
Listener callback) {
        super.fetchImage(context, callback);
        // 1. 权限
        final Activity activity = ViewHelper.getWrappedActivity(context);
        if (null == activity) {
            return;
        }

        PermissionHelper.getInstance().requestStoragePermission(activity, new
PermissionHelper.PermissionGrantListener() {
            @Override
            public void onGranted() {
                // 1.1 权限授予
                doFetch();
            }

            @Override
            public void onDenied(boolean shouldToSetting) {
                // 1.2 权限拒绝
                if (shouldToSetting) {
```

```
                    PermissionHelper.showSettingDialog(activity,
context.getString(R.string.permission_storage_dialog_content), new
PermissionPurposeDialog.ResultCallback() {

                        @Override
                        public void onOk() {
                            callback.onFail();
                        }

                        @Override
                        public void onCancel() {
                            callback.onFail();
                        }
                    });
                } else {
                    Logger.showToastShort(context,
context.getString(R.string.permission_storage_denied_hint));
                    callback.onFail();
                }
            }
        });
    }

    private void doFetch() {
        ActivityResult.with(context).start(new ActivityIntentLauncher() {
            @Override
            public Intent createIntent() {
                return PictureSelectorActivity.getIntent(context,
SelectMediaType.TYPE_IMAGE, param.getSelectImageCount(), param.isOriginMode(),
param.getGalleryBtnName());
            }

            @Override
            public void onActivityResult(int resultCode, Intent data) {
                // 返回
                onResult(resultCode, data);
            }
        });
    }

    private void onResult(int resultCode, final Intent data) {
        if (RESULT_OK != resultCode) {
            callback.onFail();
            return;
        }
        // 是否选中原图按钮
        final boolean isOrigin = null != data.getExtras() &&
data.getExtras().getBoolean(PictureSelectorActivity.EXTRA_SELECT_ORIGIN, false);
        // 结果集
        final ArrayList<Picture> pictureUrls = (ArrayList<Picture>)
data.getSerializableExtra(PictureSelectorActivity.EXTRA_SELECT_PICTURES);

        new AsyncQueryHandler<List<ImageEntity>>() {
            @Override
            public List<ImageEntity> queryInThread() {
                // 上传多张图片

                ArrayList<ImageEntity> imageEntities = new ArrayList<>();
```

```
                for (Picture picture : pictureUrls) {
                    String filePath = Pictureutil.getPath(context,
Uri.parse(picture.uri));
                    if (isOrigin) {
                        // 原图不压缩
                        int[] bound = ImageUtil.decodeBound(filePath);
                        ImageEntity entity = new ImageEntity(filePath, bound[0],
bound[1]);
                        imageEntities.add(entity);
                    } else {
                        // 压缩
                        ImageEntity imageEntity = CompressUtil.compress(context,
filePath, param.getUploadMaxSize());
                        if (null != imageEntity) {
                            imageEntities.add(imageEntity);
                        }
                    }
                }
                return imageEntities;
            }

            @Override
            public void complete(List<ImageEntity> entities) {
                if (null == entities || entities.isEmpty()) {
                    callback.onFail();
                } else {
                    if (param.isCrop()) {
                        if (entities.size() > 1) {
                            throw new IllegalStateException("只允许裁剪一张图片");
                        }
                        // 裁剪
                        crop(entities.get(0));
                    } else {
                        callback.onSuccess(entities, isOrigin);
                    }
                }
            }
        }.execute();
    }
}
```

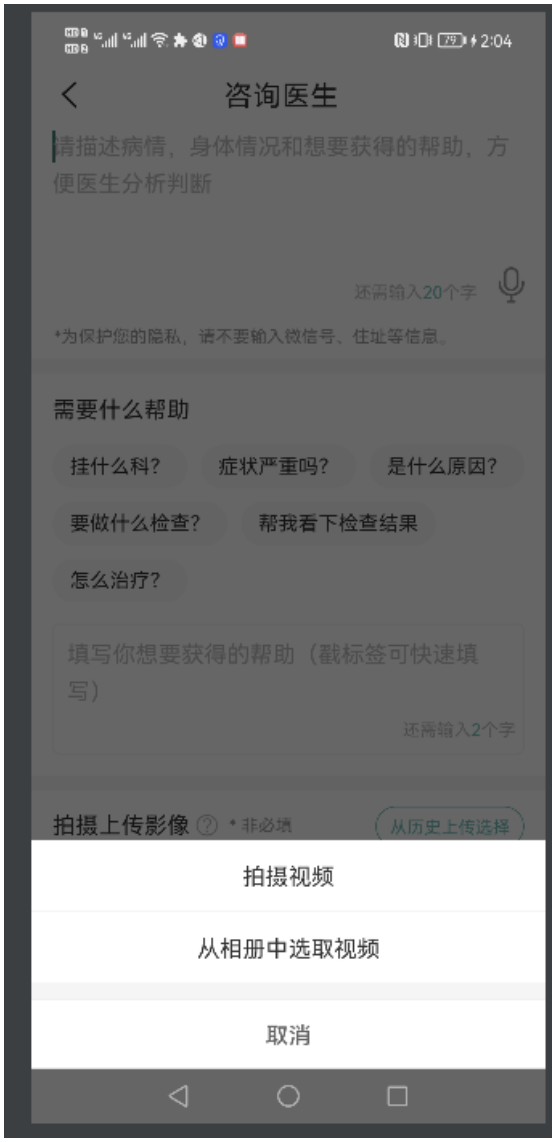
7.1 AsyncQueryHandler单线程执行器

```
package cn.kidyn.qdmedical160.database.base;

import android.os.Handler;
import android.os.Looper;

import java.util.concurrent.Executor;

import cn.kidyn.qdmedical160.QDApplicationContext;

/**
* @author fanjh
* @date 2017/7/11 15:33
```

```
* @description 在单一子线程中执行CRUD，然后回调在主线程
**/
public abstract class AsyncQueryHandler<T> {
    private static Executor sSingleExecutor =
QDApplicationContext.getSingleExecutors();
    private static Handler sMainHandler = new Handler(Looper.getMainLooper());

    public void execute(){
        sSingleExecutor.execute(new Runnable() {
            @Override
            public void run() {
                final T result = queryInThread();
                sMainHandler.post(new Runnable() {
                    @Override
                    public void run() {
                        complete(result);
                    }
                });
            }
        });
    }

    public abstract T queryInThread();
    public abstract void complete(T t);

}
```

2.选择视频

1.UI



2.源码

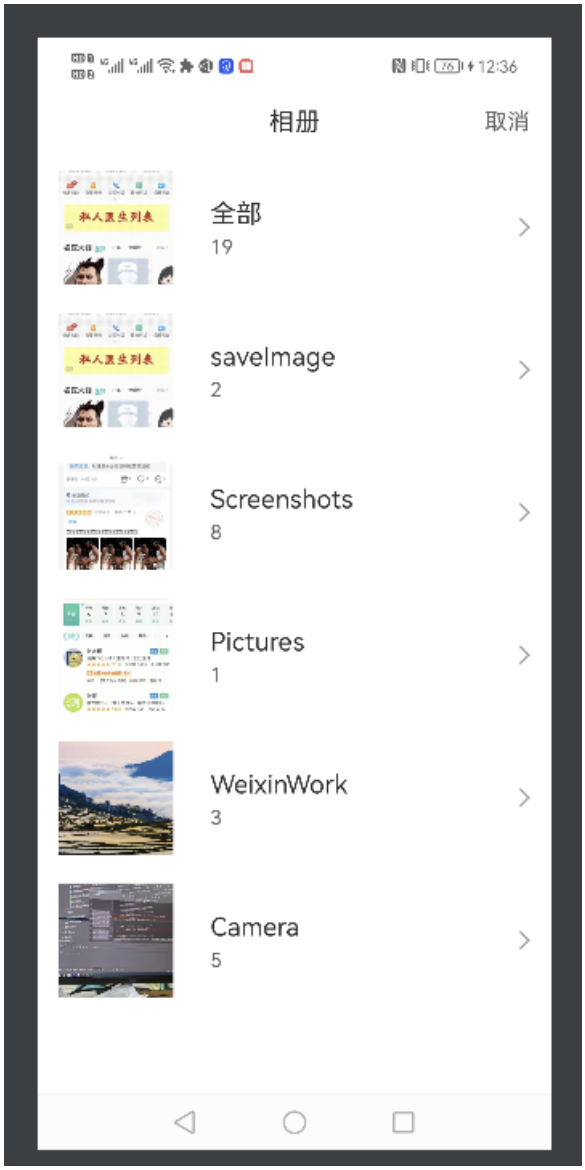
选择视频与拍摄视频的源码与 拍照差不多，有兴趣的可以去项目中查看【这里以 拍照为主体介绍整个结构源码】

```
/**
 * 获取视频途径选择：拍摄或相册
 */
public static void fetchVideo(final FragmentActivity activity, @Nullable
FetchVideoParam param, @NonNull final IFetchVideo.Listener callback) {
    CaptureVideoAction captureAction = new CaptureVideoAction(param);
    SelectVideoAction selectAction = new SelectVideoAction(param);

    VideoBean captureButton = new VideoBean(R.string.capture_video,
captureAction);
    VideoBean selectButton = new VideoBean(R.string.select_video_from_album,
selectAction);

    List<VideoBean> buttons = Arrays.asList(captureButton, selectButton);

    CommonBottomSheetFragment<VideoBean> dialog =
CommonBottomSheetFragment.newInstance(buttons, -1, false);
    dialog.setInjector(new CommonBottomSheetFragment.BindViewInjector<VideoBean>
() {
        @Override
```

```
        public void inject(TextView tv, VideoBean item) {
            tv.setText(item.textRes);
        }
    });

    dialog.setListener(new
CommonBottomSheetFragment.OnItemSelectListener<VideoBean>() {
        @Override
        public void onSelect(DialogFragment dialog, VideoBean object) {
            object.action.fetchVideo(activity, callback);
        }
    });

    dialog.show(activity);
}
```

3.选择图片或者视频页面PictureSelectorActivity

1.UI结构



2.页面结构分析

```
<cn.kidyn.qdmedical160.nybase.view.viewpager.TransformViewPager
    android:id="@+id/vp_content"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    app:canTouchScroll="false" />
```

整体结构：去除上面下面的按钮，中间的实际是 一个ViewPager，然后分两个页面，一个是 选择图片九宫格页面，一个是 相册缩略图页面，【选择图片九宫格页面】点击返回按钮实际是 切换到 【 相册缩略图页面】，默认 进入 【选择图片九宫格页面】

页面设置如下：

```
// 九宫格页面
private void initPictureView(BaseSelectorSpec<Picture> selectorSpec) {
        int padding = ViewCommonUtils.dipToPx(mContext, 1);
        int total = getResources().getDisplayMetrics().widthPixels;
        int imageSize = (total - (SPAN_COUNT + 1) * padding) / SPAN_COUNT;
        basePictureAdapter = new PictureSelectorAdapter(mContext, imageSize,
maxSelectCount);
        if (null != selectorSpec) {
```

```
            basePictureAdapter.setSelectorSpec(selectorSpec);
            changeSelectState(basePictureAdapter.getSelectedCount());
        }
        basePictureAdapter.setOnPictureClickListener(this);
        basePictureAdapter.setOnPictureSelectCallback(this);
        rvPicture = new RecyclerView(mContext);
        rvPicture.setLayoutParams(new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.MATCH_PARENT));

rvPicture.setBackgroundColor(getResources().getColor(R.color.consultation_bg));
        rvPicture.setLayoutManager(new GridLayoutManager(mContext, SPAN_COUNT));
        rvPicture.addItemDecoration(new PictureItemDecoration(padding,
SPAN_COUNT));
        rvPicture.setHasFixedSize(true);
        rvPicture.setAdapter(basePictureAdapter);
    }

// 相册实体页面
    private void initAlbumView() {
        albumAdapter = new AlbumAdapter();
        albumAdapter.setOnItemClickListener(this);

        rvAlbum = new RecyclerView(mContext);
        rvAlbum.setLayoutParams(new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.MATCH_PARENT));
        rvAlbum.setBackgroundColor(getResources().getColor(R.color.white));
        rvAlbum.setLayoutManager(new LinearLayoutManager(mContext));
        rvAlbum.setAdapter(albumAdapter);

    }



vpContent.setAdapter(new PagerAdapter() {
    @Override
    public Object instantiateItem(ViewGroup container, int position) {
        View view = null;
        switch (position) {
            case PAGE_ALBUM:
                view = rvAlbum;
                break;
            case PAGE_PICTURE:
                view = rvPicture;
                break;
            default:
                break;
        }
        container.removeView(view);
        container.addView(view);
        return view;
    }

    @Override
    public int getCount() {
        return 2;
    }
```

```
    @Override
    public void destroyItem(ViewGroup container, int position, Object object) {
        super.destroyItem(container, position, object);
        container.removeView((View) object);
    }

    @Override
    public boolean isViewFromObject(View view, Object object) {
        return view == object;
    }
});
vpContent.setCurrentItem(1);
vpContent.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {
    @Override
    public void onPageScrolled(int position, float positionOffset, int
positionOffsetPixels) {

    }

    @Override
    public void onPageSelected(int position) {
        switch (position) {
            case PAGE_ALBUM:
                tvSelectAlbum.setText(getString(R.string.album));
                btnTopBack.setVisibility(View.GONE);
                rlFinishLayout.setVisibility(View.GONE);
                break;
            case PAGE_PICTURE:
                rlFinishLayout.setVisibility(View.VISIBLE);
                btnTopBack.setVisibility(View.VISIBLE);
                break;
            default:
                break;
        }
    }

    @Override
    public void onPageScrollStateChanged(int state) {

    }
});
```

3.数据集加载

```
switch (mode) {
    case SelectMediaType.TYPE_IMAGE_VIDEO:
    case SelectMediaType.TYPE_IMAGE:
        // 选择图片

        tvSelectAlbum.setText(R.string.select_picture);

        localAlbumLoader = new LocalAlbumLoader(this, this, new
ImageAlbumFetcher(this));
```

```
        localPictureLoader = new LocalPictureLoader<>(this, this, new
PictureMediaCursorParser());
        break;
    case SelectMediaType.TYPE_VIDEO:
        // 选择视频

        tvSelectAlbum.setText(R.string.select_video);

        localAlbumLoader = new LocalAlbumLoader(this, this, new
VideoAlbumFetcher(this));
        localPictureLoader = new LocalPictureLoader<>(this, this, new
VideoMediaCursorParser());
        break;
}

开启加载：
 localPictureLoader.start(currentAlbumID, 0, PAGE_COUNT);
 localAlbumLoader.start(getLoaderManager());

 数据加载回调：
 /**
     * 追加一页数据，加载下一页
     *
     * @param pictures 获得的相册图片列表
     */
    @Override
    public void onPictureReady(ArrayList<Picture> pictures) {
        basePictureAdapter.add(pictures);
        if (pictures.size() == PAGE_COUNT) {
            // 分页加载
            localPictureLoader.start(currentAlbumID, pictures.get(pictures.size()
- 1)._id, PAGE_COUNT);
        }
    }

    /**
     * 更新相册列表
     */
    @Override
    public void onAlbumReady(List<Album> albums) {
        albumAdapter.update(albums);
    }
```

这边 相册加载是利用的是 android原生提供的CursorLoader进行实现的【CursorLoader实现了AsyncTaskLoader】，实际加载器通过外部进行设置【加载图片相册，加载视频相册】，而九宫格图片加载采用的是 自己继承【AsyncTaskLoader进行实现，图片或者视频数据库查询代码都是公共的，所以查询路径以及解析是全部查询出来后解析为对象是通过外部设置】

下面的代码比较简单，主要是 学习 加载以及封装思想

1.LocalAlbumLoader相册加载器

```
package cn.kidyn.qdmedical160.activity.picture.album;

import android.app.LoaderManager;
import android.content.Context;
```

```
import android.content.Loader;
import android.database.Cursor;
import android.os.AsyncTask;
import android.os.Bundle;
import androidx.annotation.NonNull;

import java.lang.ref.WeakReference;
import java.util.ArrayList;
import java.util.List;

import cn.kidyn.qdmedical160.QDApplicationContext;
import cn.kidyn.qdmedical160.R;


/**
 * 本地相册加载器
 * <p>
 * Created by fanjh on 2017/12/19.
 **/
public class LocalAlbumLoader implements LoaderManager.LoaderCallbacks<Cursor> {

    /**
     * 后台运行的阈值
     */
    private static final int BACKGROUND_LIMIT = 500;
    /**
     * Loader的ID
     */
    private static final int LOADER_ID = 10001;

    private WeakReference<Context> weak;
    private WeakReference<Callback> callbackWeak;
    private IAlbumFetcher albumFetcher;

    public LocalAlbumLoader(Context context, Callback callback, IAlbumFetcher
albumFetcher) {
        this.weak = new WeakReference<>(context);
        this.callbackWeak = new WeakReference<>(callback);
        this.albumFetcher = albumFetcher;
    }

    public void start(LoaderManager manager) {
        if (null == manager) {
            return;
        }
        try {
            manager.restartLoader(LOADER_ID, null, this);
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    public void destroy(LoaderManager manager) {
        if (null == manager) {
            return;
        }
        try {
            manager.destroyLoader(LOADER_ID);
```

```
            weak.clear();
            callbackWeak.clear();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    @Override
    public Loader<Cursor> onCreateLoader(int id, Bundle args) {
        return albumFetcher;
    }

    @Override
    public void onLoadFinished(Loader<Cursor> loader, Cursor data) {
        if (null == data || null == weak.get()) {
            return;
        }
        int count = data.getCount();
        if (count > BACKGROUND_LIMIT) {
            new AsyncTask<Cursor, Integer, List<Album>>() {
                @Override
                protected List<Album> doInBackground(Cursor... params) {
                    return parseCursor(params[0]);
                }

                @Override
                protected void onPostExecute(List<Album> alba) {
                    super.onPostExecute(alba);
                    if (null != weak.get() && null != callbackWeak.get()) {
                        callbackWeak.get().onAlbumReady(alba);
                    }
                }
            }.execute(data);
        } else if (null != callbackWeak.get()) {
            callbackWeak.get().onAlbumReady(parseCursor(data));
        }
    }

    private List<Album> parseCursor(@NonNull Cursor cursor) {
        List<Album> alba = new ArrayList<>();
        int totalCount = 0;
        while (cursor.moveToNext()) {
            Album album = albumFetcher.parse(cursor);
            alba.add(album);
            totalCount += album.count;
        }
        Album allAlbum = new Album();
        allAlbum.displayName =
QDApplicationContext.getInstance().getString(R.string.all);
        allAlbum.count = totalCount;
        allAlbum.uri = alba.size() > 0 ? alba.get(0).uri : null;
        allAlbum.bucketID = null;
        alba.add(0, allAlbum);
        return alba;
    }

    @Override
    public void onLoaderReset(Loader<Cursor> loader) {
```

```
        if (null != weak.get() && null != callbackWeak.get()) {
            callbackWeak.get().onReset();
        }
    }

    public interface Callback {
        /**
         * 相册准备完成
         *
         * @param albums 获得的相册列表
         */
        void onAlbumReady(List<Album> albums);

        /**
         * 当前需要释放资源
         */
        void onReset();
    }
}
```

#1.Album相册实体

```
package cn.kidyn.qdmedical160.activity.picture.album;

import android.content.ContentUris;
import android.database.Cursor;
import android.net.Uri;
import android.provider.BaseColumns;
import android.provider.MediaStore;


/**
 * 相册实体（图片&视频）
 *
 * Created by pengxr on 2020/9/17.
 **/
public class Album {

    public String bucketID;
    public long _id;
    public String displayName;
    public long count;
    public Uri uri;

    /**
     * 解析图片相册实体
     */
    public static Album parseFromImageCursor(Cursor cursor) {
        Album album = new Album();
        album.bucketID =
cursor.getString(cursor.getColumnIndex(MediaStore.Images.ImageColumns.BUCKET_ID)
);
        album._id = cursor.getLong(cursor.getColumnIndex(BaseColumns._ID));
        album.displayName =
cursor.getString(cursor.getColumnIndex(MediaStore.Images.ImageColumns.BUCKET_DIS
PLAY_NAME));
```

```
        album.count =
cursor.getLong(cursor.getColumnIndex(IAlbumFetcher.COLUMN_COUNT));
        album.uri =
ContentUris.withAppendedId(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
album._id);
        return album;
    }

    /**
     * 解析视频相册实体
     */
    public static Album parseFromVideoCursor(Cursor cursor) {
        Album album = new Album();
        album.bucketID =
cursor.getString(cursor.getColumnIndex(MediaStore.Video.VideoColumns.BUCKET_ID))
;
        album._id = cursor.getLong(cursor.getColumnIndex(BaseColumns._ID));
        album.displayName =
cursor.getString(cursor.getColumnIndex(MediaStore.Video.VideoColumns.BUCKET_DISP
LAY_NAME));
        album.count =
cursor.getLong(cursor.getColumnIndex(IAlbumFetcher.COLUMN_COUNT));
        album.uri =
ContentUris.withAppendedId(MediaStore.Video.Media.EXTERNAL_CONTENT_URI,
album._id);
        return album;
    }

    @Override
    public String toString() {
        return displayName;
    }
}
```

#2.IAlbumFetcher

```
package cn.kidyn.qdmedical160.activity.picture.album;

import android.content.Context;
import android.database.Cursor;
import android.net.Uri;
import android.provider.BaseColumns;
import android.provider.MediaStore;
import androidx.annotation.NonNull;
import android.content.CursorLoader;

/**
 * 相册获取器
 *
 * Created by pengxr on 2020/9/14.
 */
public abstract class IAlbumFetcher extends CursorLoader {

    public static final String COLUMN_COUNT = "count";

    private static final String[] PROJECTION = {
            MediaStore.Images.ImageColumns.BUCKET_ID,
```

```
            MediaStore.Images.ImageColumns.BUCKET_DISPLAY_NAME,
            BaseColumns._ID,
            "COUNT(*) AS " + COLUMN_COUNT};

    private static final String BUCKET_GROUP_BY = "1) GROUP BY 1,(2";

    private static final String BUCKET_ORDER_BY = "MAX(" +
MediaStore.Images.ImageColumns.DATE_TAKEN + ") DESC";

    public IAlbumFetcher(Context context) {
        super(context);
        setUri(getUri());
        setProjection(PROJECTION);
        setSelection(BUCKET_GROUP_BY);
        setSortOrder(BUCKET_ORDER_BY);
    }

    public abstract Album parse(Cursor cursor);

    @NonNull
    public abstract Uri getUri();
}
```

#3.ImageAlbumFetcher 图片相册加载器

```
package cn.kidyn.qdmedical160.activity.picture.album;

import android.content.Context;
import android.database.Cursor;
import android.net.Uri;
import android.provider.MediaStore;
import androidx.annotation.NonNull;

/**
 * 图片相册获取器
 *
 * Created by pengxr on 2020/9/14.
 */
public class ImageAlbumFetcher extends IAlbumFetcher {

    public ImageAlbumFetcher(Context context) {
        super(context);
    }

    @Override
    public Album parse(Cursor cursor) {
        return Album.parseFromImageCursor(cursor);
    }

    @NonNull
    @Override
    public Uri getUri() {
        return MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
    }
}
```

#4.视频相册加载器

```
package cn.kidyn.qdmedical160.activity.picture.album;

import android.content.Context;
import android.database.Cursor;
import android.net.Uri;
import android.provider.MediaStore;
import androidx.annotation.NonNull;

/**
 * Created by pengxr on 2020/9/14.
 */
public class VideoAlbumFetcher extends IAlbumFetcher {

    public VideoAlbumFetcher(Context context) {
        super(context);
    }

    @Override
    public Album parse(Cursor cursor) {
        return Album.parseFromVideoCursor(cursor);
    }

    @NonNull
    @Override
    public Uri getUri() {
        return MediaStore.Video.Media.EXTERNAL_CONTENT_URI;
    }
}
```

2.LocalPictureLoader本地图片文件加载器

#1.BasePicture

```
public abstract class BasePicture {

    public abstract String getImageUrl();

    /**
     * 是否为视频
     */
    public abstract boolean isVideo();

    /**
     * 描述
     */
    public abstract String getDesc();
}
```

#2.Picture 媒体实体（图片&视频）

```
package cn.kidyn.qdmedical160.activity.picture.bean;

import android.content.ContentUris;
import android.content.Context;
import android.database.Cursor;
import android.provider.BaseColumns;
import android.provider.MediaStore;

import com.netease.nim.avchatkit.common.util.TimeUtil;

import java.io.Serializable;

import cn.kidyn.qdmedical160.entity.BasePicture;
import cn.kidyn.qdmedical160.util.config.ConstantConfig;

/**
 * 媒体实体（图片&视频）
 * <p>
 * Created by pengxr on 2020/9/14.
 **/
public class Picture extends BasePicture implements Serializable {

    private static final long serialVersionUID = 8153100252603332898L;

    public long _id;
    public String mimeType;
    public long size;
    public String uri; // 转来转去的?

    public int duration; // 秒
    public String thumbPath; // 缩略图

    /**
     * 解析图片实体
     */
    public static Picture parseFromImageCursor(Context context, Cursor cursor) {
        Picture picture = new Picture();
        picture._id =
cursor.getLong(cursor.getColumnIndexOrThrow(MediaStore.Images.Media._ID));
        picture.mimeType =
cursor.getString(cursor.getColumnIndexOrThrow(MediaStore.Images.Media.MIME_TYPE)
);
        picture.size =
cursor.getLong(cursor.getColumnIndexOrThrow(MediaStore.Images.Media.SIZE));
        picture.uri =
ContentUris.withAppendedId(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
picture._id).toString();
        return picture;
    }

    /**
     * 解析视频实体
     */
    public static Picture parseFromVideoCursor(Context context, Cursor cursor) {
        Picture picture = new Picture();
```

```
        picture._id =
cursor.getLong(cursor.getColumnIndexOrThrow(BaseColumns._ID));
        picture.mimeType =
cursor.getString(cursor.getColumnIndexOrThrow(MediaStore.MediaColumns.MIME_TYPE)
);
        picture.duration = (int)
(cursor.getLong(cursor.getColumnIndexOrThrow(MediaStore.Video.VideoColumns.DURAT
ION)) / 1000L);
        picture.duration = Math.max(picture.duration, 1);
        picture.size =
cursor.getLong(cursor.getColumnIndexOrThrow(MediaStore.MediaColumns.SIZE));
        picture.uri =
ContentUris.withAppendedId(MediaStore.Video.Media.EXTERNAL_CONTENT_URI,
picture._id).toString();
//        picture.thumbPath = getThumbnailPath(context, picture._id); // 回调时再
查找 @see SelectVideoAction
        return picture;
    }

    @Override
    public boolean equals(Object o) {
        if (!(o instanceof Picture)) {
            return false;
        }
        Picture temp = (Picture) o;
        return temp._id == _id;
    }

    @Override
    public int hashCode() {
        int hashCode = 1;
        hashCode = 31 * hashCode + (int) (_id ^ (_id >>> 32));
        return hashCode;
    }

    @Override
    public String getImageUrl() {
        return null != uri ? uri : ConstantConfig.IS_NULL;
    }

    public boolean isGif() {
        return "image/gif".equals(mimeType);
    }

    /**
     * 是否为视频
     */
    @Override
    public boolean isVideo() {
        return null != mimeType && mimeType.startsWith("video/");
    }

    @Override
    public String getDesc() {
        if (!isVideo()) {
            return "";
        } else {
            return TimeUtil.secToTimeShort(duration);
```

```
        }
    }
}
```

#3.LocalPictureLoader

```
package cn.kidyn.qdmedical160.activity.picture.loader;

import android.content.Context;
import android.content.Loader;

import java.lang.ref.WeakReference;
import java.util.ArrayList;
import java.util.List;

/**
 * 本地图片加载器
 * <p>
 * Created by fanjh on 2017/12/19.
 **/
public class LocalPictureLoader<T> {

    private WeakReference<Context> weak;
    private WeakReference<Callback<T>> callbackWeak;
    private String currentAlbumID;
    private MediaCursorParser<T> mediaCursorParser;
    private LocalPictureCursorLoader<T> loader;

    public LocalPictureLoader(Context context, Callback<T> callback,
MediaCursorParser<T> mediaCursorParser) {
        this.weak = new WeakReference<>(context);
        this.callbackWeak = new WeakReference<>(callback);
        this.mediaCursorParser = mediaCursorParser;
    }

    public void start(String albumID, long firstPictureID, int pageSize) {
        if (weak.get() == null) {
            return;
        }
        try {
            currentAlbumID = albumID;
            if (null == loader) {
                loader = LocalPictureCursorLoader.getInstance(weak.get(),
mediaCursorParser);
                loader.registerListener(0, new
Loader.OnLoadCompleteListener<List<T>>() {
                    @Override
                    public void onLoadComplete(Loader<List<T>> loader, List<T>
data) {
                        if (null == data || null == weak.get()) {
                            return;
                        }
                        if (null != callbackWeak.get()) {
                            callbackWeak.get().onPictureReady((ArrayList<T>)
data);
                        }
                    }
```

```
                });
            }
            loader.startLoad(currentAlbumID, firstPictureID, pageSize);
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    public void destroy() {
        try {
            weak.clear();
            callbackWeak.clear();
            currentAlbumID = null;
            loader = null;
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    public interface Callback<T> {
        /**
         * 相册图片准备完成
         *
         * @param pictures 获得的相册图片列表
         */
        void onPictureReady(ArrayList<T> pictures);

    }

    public String getCurrentAlbumID() {
        return currentAlbumID;
    }
}
```

#4.LocalPictureCursorLoader

```
package cn.kidyn.qdmedical160.activity.picture.loader;

import android.content.AsyncTaskLoader;
import android.content.Context;
import android.database.Cursor;
import android.provider.MediaStore;
import android.text.TextUtils;

import java.util.ArrayList;
import java.util.List;

/**
 * @author fanjh
 * @date 2017/12/19 18:07
 * @description 本地媒体图片游标读取者
 * @note 目前是分段加载，并没有监听本地数据库的变化，不需要那么实时
 * 比方说 20->20->20->20相对于60ms来说虽然总时长变高了，但是第一页加载快，体验更好
 **/
public class LocalPictureCursorLoader<T> extends AsyncTaskLoader<List<T>> {
```

```
    private static final String ORDER_BY = MediaStore.Images.Media._ID + "
DESC";

    private String currentAlbumID;
    private long firstID;
    private int pageSize;

    private MediaCursorParser<T> mediaCursorParser;

    private LocalPictureCursorLoader(Context context, MediaCursorParser<T>
mediaCursorParser) {
        super(context);
        this.mediaCursorParser = mediaCursorParser;
    }

    public void startLoad(String currentAlbumID, long firstID, int pageSize) {
        this.currentAlbumID = currentAlbumID;
        this.firstID = firstID;
        this.pageSize = pageSize;
        forceLoad();
    }

    @Override
    public List<T> loadInBackground() {
        final List<T> data = new ArrayList<>();
        String selection = null;
        String[] selectArgs = null;
        if (!TextUtils.isEmpty(currentAlbumID)) {
            if (firstID <= 0) {
                selection = MediaStore.Images.Media.BUCKET_ID + " = ?";
                selectArgs = new String[]{currentAlbumID};
            } else {
                selection = MediaStore.Images.Media.BUCKET_ID + " = ? and " +
MediaStore.Images.Media._ID + " < ?";
                selectArgs = new String[]{currentAlbumID, firstID + ""};
            }
        } else if (firstID > 0) {
            selection = MediaStore.Images.Media._ID + " < ?";
            selectArgs = new String[]{firstID + ""};
        }

        String orderBy = null;
        if (pageSize > 0) {
            orderBy = ORDER_BY + " LIMIT " + pageSize;
        } else {
            orderBy = ORDER_BY;
        }
        Cursor cursor = getContext().getContentResolver()
                .query(mediaCursorParser.getUri(), mediaCursorParser.getIndex(),
selection, selectArgs, orderBy);

        if (cursor == null) {
            return data;
        }

        try {
            while (cursor.moveToNext()) {
                data.add(mediaCursorParser.parseCursor(getContext(), cursor));
```

```
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        } finally {
            cursor.close();
        }
        return data;
    }

    public static <T> LocalPictureCursorLoader getInstance(Context context,
MediaCursorParser<T> mediaCursorParser) {
        return new LocalPictureCursorLoader<T>(context, mediaCursorParser);
    }

}
```

#5.查询路径字段解析MediaCursorParser

```
package cn.kidyn.qdmedical160.activity.picture.loader;

import android.content.Context;
import android.database.Cursor;
import android.net.Uri;

/**
* @author fanjh
* @date 2017/12/26 18:08
* @description 从本地数据库中获取图片cursor后的处理
**/
public interface MediaCursorParser<T> {
    /**
     * 用于将当前游标转换为指定的对象
     * @param cursor 当前查询后获得的游标
     * @return 对象
     */
    T parseCursor(Context context, Cursor cursor);

    /**
     * 指定当前要查询的列
     * 比方说{MediaStore.Images.Media._ID, MediaStore.Images.Media.DISPLAY_NAME,
     MediaStore.Images.Media.MIME_TYPE, MediaStore.Images.Media.SIZE}
     * @return 当前要查询的列
     */
    String[] getIndex();

    Uri getUri();
}
```

#6.PictureMediaCursorParser 与 VideoMediaCursorParser

```
package cn.kidyn.qdmedical160.activity.picture.loader;

import android.content.Context;
import android.database.Cursor;
import android.net.Uri;
```

```
import android.provider.MediaStore;

import cn.kidyn.qdmedical160.activity.picture.bean.Picture;

/**
 * Created by fanjh on 2017/12/27
 **/
public class PictureMediaCursorParser implements MediaCursorParser<Picture> {

    private static final String[] PROJECTION = {
            MediaStore.Images.Media._ID,
            MediaStore.Images.Media.MIME_TYPE,
            MediaStore.Images.Media.SIZE};

    @Override
    public Picture parseCursor(Context context, Cursor cursor) {
        return Picture.parseFromImageCursor(context, cursor);
    }

    @Override
    public String[] getIndex() {
        return PROJECTION;
    }

    @Override
    public Uri getUri() {
        return MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
    }
}
```

```
package cn.kidyn.qdmedical160.activity.picture.loader;

import android.content.Context;
import android.database.Cursor;
import android.net.Uri;
import android.provider.MediaStore;

import cn.kidyn.qdmedical160.activity.picture.bean.Picture;

/**
 * Created by pengxr on 2020/9/14.
 */
public class VideoMediaCursorParser implements MediaCursorParser<Picture> {

    private static final String[] PROJECTION = {
            MediaStore.Video.Media._ID,
            MediaStore.Video.Media.MIME_TYPE,
            MediaStore.Video.Media.DURATION,
            MediaStore.Video.Thumbnails.DATA,
            MediaStore.Video.Media.SIZE};

    @Override
    public Picture parseCursor(Context context, Cursor cursor) {
        return Picture.parseFromVideoCursor(context, cursor);
    }
```

```
    @Override
    public String[] getIndex() {
        return PROJECTION;
    }

    @Override
    public Uri getUri() {
        return MediaStore.Video.Media.EXTERNAL_CONTENT_URI;
    }
}
```

九.上拉加载下拉刷新

1.统计

1.cn.kidyn.qdmedical160.nybase.view.easypulllayout.base.EasyPullLayout 以前的老代码使用较多，支持上下左右滑动，添加头部比较兼容，但控制上比较复杂
2.com.nykj.pulllayout.PullLayout 新代码使用较多，可以单独支持头部view与顶部view，目前一般用法是支持头部下拉刷新view，底部上拉加载利用MqttPagerMultiTypeAdapter，这时候PullLayout#canDownToUp设置为false