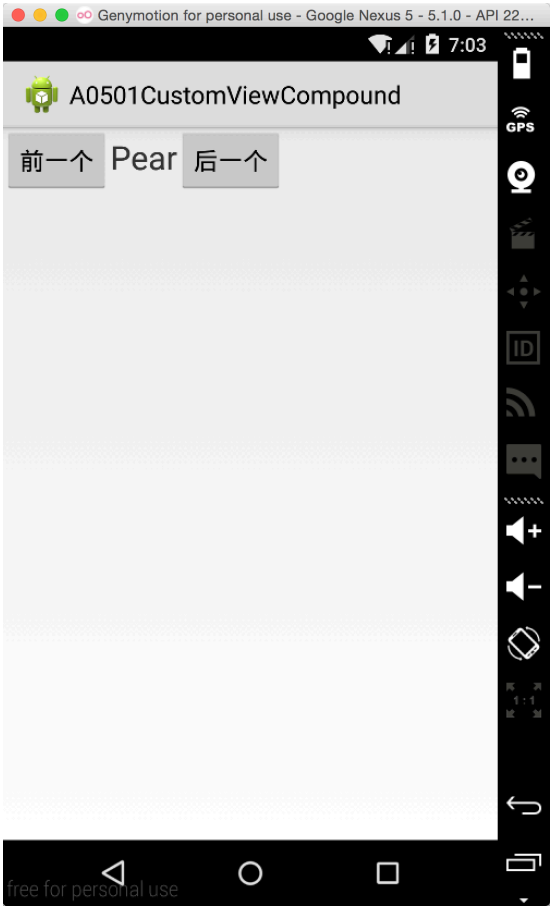
定义复杂 CompoundView

1.1 知识解析

如果需要进一步的定制，例如对其行为进行定制，则需要对其进行类定义。

1.2 功能演示



1.3 实战操作

具体步骤：

1. 在 layout 中，根据需要定义将需要的组件定义在一起，根节点为<merge>，例如：

```
<merge
xmlns:android="http://schemas.android.com/apk/res/android" >

    <Button
        android:id="@+id/compound_view_previous"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/prev" />

    <TextView
        android:id="@+id/compound_view_current_value"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="24sp" />

    <Button
        android:id="@+id/compound_view_next"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/next" />

</merge>
```

2. 为这个 merge layout 定义一个类，根据需要让它继承一个 ViewGroup（例如 LinearLayout、FrameLayout、TableLayout、GridLayout、GridView、ListView），根据自己的需求，定义其行为等，如果有必要，可以覆盖 **void onFinishInflate() 方法，这个回调方法将会在这个 merge layout 文件 inflate 结束后被调用**

```
public class CompoundView extends LinearLayout {
    private Button mPreviousButton;
    private Button mNextButton;

    public CompoundView(Context context) {
        super(context);
        // TODO Auto-generated constructor stub
        initializeViews(context);
    }

    public CompoundView(Context context, AttributeSet attrs) {
        super(context, attrs);
        // TODO Auto-generated constructor stub
        initializeViews(context);
    }

    /**
     * Inflates the views in the layout.
     *
     * @param context
     *            the current context for the view.
     */
    private void initializeViews(Context context) {
        LayoutInflater inflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        inflater.inflate(R.layout.compound_view, this);
    }

    @Override
    protected void onFinishInflate() {

        // 在 layout 被 inflate 后，给按钮加上监听器
```

```
        super.onFinishInflate();

        mPreviousButton = (Button) this.findViewById(R.id.compound_view_previous);
        // 当“前一个”按钮被点击时候，选择列表中的上一个需要现实的内容
        mPreviousButton.setOnClickListener(new OnClickListener() {
            public void onClick(View view) {
                if (mSelectedIndex > 0) {
                    int newSelectedIndex = mSelectedIndex - 1;
                    setSelectedIndex(newSelectedIndex);
                }
            }
        });

        mNextButton = (Button) this.findViewById(R.id.compound_view_next);
        // 当“next”按钮被点击时候，选择列表中的下一个需要现实的内容
        mNextButton.setOnClickListener(new OnClickListener() {
            public void onClick(View view) {
                if (mSpinnerValues != null && mSelectedIndex < mSpinnerValues.length - 1) {
                    int newSelectedIndex = mSelectedIndex + 1;
                    setSelectedIndex(newSelectedIndex);
                }
            }
        });

        // 默认设置为第一个元素显示
        setSelectedIndex(0);
    }
```

```
    private CharSequence[] mSpinnerValues = null;
    private int mSelectedIndex = -1;

    /**
     * 为 CompoundView 设置数据集合
     *
     * @param values
     *            需要设置的数据集合
     */
    public void setValues(CharSequence[] values) {
        mSpinnerValues = values;

        // 为 CompundView 设置数据集后，让第一个数据被选中显示
        setSelectedIndex(0);
    }

    /**
     * 设置这个组件选中的 index，并处理其对应的 TextView 的显示行为
     *
     * @param index
     *            the index of the value to select.
     */
    public void setSelectedIndex(int index) {
        // 如果没有给 CompoundView 的 text view 设置对应的数据集合，
则不做任何设置
        if (mSpinnerValues == null || mSpinnerValues.length == 0)
            return;

        // 如果索引不正确，则不做任何设置
        if (index < 0 || index >= mSpinnerValues.length)
            return;
```

```
        // 设置当前索引并显示其值
        mSelectedIndex = index;
        TextView currentValue;
        currentValue                    =                    (TextView)
this.findViewById(R.id.compound_view_current_value);
        currentValue.setText(mSpinnerValues[index]);

        // 如果目前显示的是第一个 item，则隐藏“previous”按钮
        if (mSelectedIndex == 0)
            mPreviousButton.setVisibility(INVISIBLE);
        else
            mPreviousButton.setVisibility(VISIBLE);

        // 如果目前显示的是最后一个 item，则隐藏“next”按钮
        if (mSelectedIndex == mSpinnerValues.length - 1)
            mNextButton.setVisibility(INVISIBLE);
        else
            mNextButton.setVisibility(VISIBLE);
    }

    /**
     * 获取当前显示的内容
     *
     * @return 显示的内容的值
     */
    public CharSequence getSelectedValue() {
        if (mSpinnerValues == null || mSpinnerValues.length == 0)
            return "";

        if    (mSelectedIndex    <    0    ||    mSelectedIndex    >=
mSpinnerValues.length)
```

```
            return "";

        return mSpinnerValues[mSelectedIndex];
    }

    /**
     * 当前显示的内容对应的 index
     *
     * @return 当前显示的内容的索引
     */
    public int getSelectedIndex() {
        return mSelectedIndex;
    }
}
```

3. 在需要用到这个组建的 layout 文件中，使用下面的方法调用：

```
    <com.exam.a0501customviewcompound.CompoundView
        android:id="@+id/compoundView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >
    </com.exam.a0501customviewcompound.CompoundView>
```

1.4 职业素质

如果需要进一步的定制，例如对其行为进行定制，则需要对其进行类定义。

可以实现一般复杂性的布局。