--------------UI绘制流程-----------------

一、从setContentView(R.layout.activity\_main);入手了解UI的绘制起始过程

1.Activity.java

public void setContentView(@LayoutRes int layoutResID) {

getWindow().setContentView(layoutResID);//①

initWindowDecorActionBar();

}

2.getWindow()拿到的是Window的实现类PhoneWindow

PhoneWindow源码:

com.android.internal.policy

@Override

public void setContentView(int layoutResID) {

// Note: FEATURE\_CONTENT\_TRANSITIONS may be set in the process of installing the window

// decor, when theme attributes and the like are crystalized. Do not check the feature

// before this happens.

if (mContentParent == null) {

installDecor();//②

}

……

mLayoutInflater.inflate(layoutResID, mContentParent);//⑥

}

private void installDecor() {

if (mDecor == null) {

mDecor = generateDecor();//③生成一个DecorView(继承的FrameLayout)

mDecor.setDescendantFocusability(ViewGroup.FOCUS\_AFTER\_DESCENDANTS);

mDecor.setIsRootNamespace(true);

if (!mInvalidatePanelMenuPosted && mInvalidatePanelMenuFeatures != 0) {

mDecor.postOnAnimation(mInvalidatePanelMenuRunnable);

}

}

if (mContentParent == null) {

mContentParent = generateLayout(mDecor);//④

}

protected ViewGroup generateLayout(DecorView decor) {//⑤

View in = mLayoutInflater.inflate(layoutResource, null);

decor.addView(in, new ViewGroup.LayoutParams(MATCH\_PARENT, MATCH\_PARENT));

mContentRoot = (ViewGroup) in;

}

首先根据设置的主题样式来设置DecorView的风格，比如说有没有titlebar之类的，接着为DecorView添加子View，而这里的子View则是上面提到的mContentParent，如果上面设置了FEATURE\_NO\_ACTIONBAR，那么DecorView就只有mContentParent一个子View，

DecorView是顶级View，内部有titlebar和contentParent两个子元素，contentParent的id是content，而我们设置的main.xml布局则是contentParent里面的一个子元素

通过setContentView方法，创建了DecorView和加载了我们提供的布局，但是这时，我们的View还是不可见的，因为我们仅仅是加载了布局，并没有对View进行任何的测量、布局、绘制工作。在View进行测量流程之前，还要进行一个步骤，那就是把DecorView添加至window中，然后经过一系列过程触发ViewRootImpl#performTraversals方法

**ViewRootImpl#setView**方法，并把DecorView作为参数传递进去，在这个方法内部，会通过跨进程的方式向WMS（WindowManagerService）发起一个调用，从而将DecorView最终添加到Window上，在这个过程中，ViewRootImpl、DecorView和WMS会彼此关联，至于详细过程这里不展开来说了。  
最后通过WMS调用**ViewRootImpl#performTraverals**方法开始View的测量、布局、绘制流程

DecorView是视图的顶级View，我们添加的布局文件是它的一个子布局，而ViewRootImpl则负责渲染视图，它调用了一个performTraveals方法使得ViewTree开始三大工作流程，然后使得View展现在我们面前

里面主要执行了三个方法，分别是performMeasure、performLayout、performDraw这三个方法，在这三个方法内部又会分别调用measure、layout、draw这三个方法来进行不同的流程。

三、measure、layout、draw的三个执行流程

View.java类

measure：测量，测量自己有多大，如果是ViewGroup的话会同时测量里面的子控件的大小

layout：摆放里面的子控件bounds(left,top,right,bottom)

draw：绘制 (直接继承了view一般都会重写onDraw)

ViewGroup.java

看View.java类的源码：

1.view的requestLayout()方法开始，递归地不断往上找父容器，最终找到DecorView

2.执行了DecorView的ViewRootImp类的performTranversal()方法 （ViewRootImp类：是PhoneWindow和DecorView的桥梁）

3.performTranversal(){

// Ask host how big it wants to be

performMeasure(childWidthMeasureSpec, childHeightMeasureSpec);

performLayout(lp, desiredWindowWidth, desiredWindowHeight);

performDraw();

}

1. private void performMeasure(int childWidthMeasureSpec, int childHeightMeasureSpec) {

Trace.traceBegin(Trace.TRACE\_TAG\_VIEW, "measure");

try {

mView.measure(childWidthMeasureSpec, childHeightMeasureSpec);

} finally {

Trace.traceEnd(Trace.TRACE\_TAG\_VIEW);

}

}

------------------------ViewGroup.java总结：-----------------------

一、measure的过程

如何去合理的测量一颗View树？

如果ViewGroup和View都是直接指定的宽高，我还要测量吗？

正是因为谷歌设计的自适应尺寸机制(比如Match\_parent,wrap\_content),造成了宽高不确定，所以就需要进程测量measure过程。

measure过程会遍历整颗View树，然后依次测量每一个View的真实的尺寸。(树的遍历--先序遍历)

MeasureSpec：测量规格

int 32位：010111100011100

拿前面两位当做mode，后面30位当做值。

1.mode：

1） EXACTLY: 精确的。比如给了一个确定的值 100dp

2) AT\_MOST: 根据父容器当前的大小，结合你指定的尺寸参考值来考虑你应该是多大尺寸，需要计算（Match\_parent,wrap\_content就是属于这种）

3) UNSPECIFIED: 最多的意思。根据当前的情况，结合你制定的尺寸参考值来考虑，在不超过父容器给你限定的只存的前提下，来测量你的一个恰好的内容尺寸。

用的比较少，一般见于ScrollView，ListView（大小不确定，同时大小还是变的。会通过多次测量才能真正决定好宽高。）

2.value：宽高的值。

经过大量测量以后，最终确定了自己的宽高，需要调用：setMeasuredDimension(w,h)

写自定义控件的时候，我们要去获得自己的宽高来进行一些计算，必须先经过measure，才能获得到宽高---不是getWidth(),而是getMeasuredWidth()

也就是当我们重写onMeasure的时候，我们需要在里面调用child.measure()才能获取child的宽高。

从规格当中获取mode和value：

final int widthMode = MeasureSpec.getMode(widthMeasureSpec);

final int heightMode = MeasureSpec.getMode(heightMeasureSpec);

int widthSize = MeasureSpec.getSize(widthMeasureSpec);

int heightSize = MeasureSpec.getSize(heightMeasureSpec);

反过来将mode和value合成一个规格呢：

MeasureSpec.makeMeasureSpec(resultSize, resultMode);

ViewGroup:

设计它的目的是什么？

1）作为容器处理焦点问题。

2）作为容器处理事件分发问题；

3）控制容器添加View的流程：addView()，removeView()

4）抽出了一些容器的公共的工具方法：measureChildren，measureChild,measureChildWidthMargins方法。

-------------------重点：-----------------------

玩自定义控件的时候，需要进行测量measure，如何做好这件事？

两种情况：

1.继承自View的子类

只需要重写onMeasure测量好自己的宽高就可以了。

最终调用setMeasuredDimension()保存好自己的测量宽高。

套路：

int mode = MeasureSpec.getMode(widthMeasureSpec);

int Size = MeasureSpec.getSize(widthMeasureSpec);

int viewSize = 0;

switch(mode){

case MeasureSpec.EXACTLY:

viewSize = size;//当前view的尺寸就为父容器的尺寸

break;

case MeasureSpec.AT\_MOST:

viewSize = Math.min(size, getContentSize());//当前view的尺寸就为内容尺寸和费容器尺寸当中的最小值。

break;

case MeasureSpec.UNSPECIFIED:

viewSize = getContentSize();//内容有多大，久设置多大尺寸。

break;

default:

break;

}

//setMeasuredDimension(width, height);

setMeasuredDimension(size);

2.继承自ViewGroup的子类：

不但需要重写onMeasure测量自己，还要测量子控件的规格大小。

可以直接使用ViewGroup的工具方法来测量里面的子控件，也可以自己来实现这一套子控件的测量（比如：RelativeLayout）

套路：

//1.测量自己的尺寸

ViewGroup.onMeasure();

//1.1 为每一个child计算测量规格信息(MeasureSpec)

ViewGroup.getChildMeasureSpec();

//1.2 将上面测量后的结果，传给每一个子View，子view测量自己的尺寸

child.measure();

//1.3 子View测量完，ViewGroup就可以拿到这个子View的测量后的尺寸了

child.getChildMeasuredSize();//child.getMeasuredWidth()和child.getMeasuredHeight()

//1.4ViewGroup自己就可以根据自身的情况(Padding等等),来计算自己的尺寸

ViewGroup.calculateSelfSize();

//2.保存自己的尺寸

ViewGroup.setMeasuredDimension(size);

二、layout的过程

三、draw的过程

作业：如何让一个ScrollView里面的ListView全部展开？

有一种解决办法就是继承ListView，重写onMeasure方法：

public void onMeasure(){

int expandSpec = MeasureSpec.makeMeasureSpec(Integer.MAX\_VALUE >> 2, MeasureSpec.AT\_MOST);

super.onMeasure(widthMeasureSpec, expandSpec);

}

为什么要这么做？1.设置mode为 MeasureSpec.AT\_MOST？2.value为Integer.MAX\_VALUE >> 2？

MeasureSpec .AT\_MOST 在布局中指定了warp\_content

MeasureSpec.EXACTLY 在布局中指定了固定的大小 100dp match\_parent fill\_parent

MeasureSpec.UNSPECIFIED 尽可能的大 , 很少用得到

===========================

ScrollView + listVIew 会显示不全问题：

ScrollView 在测量子布局的时候会用UNSPECIFIED，listVIew 在给自己指定宽高的时候调用setMeasuredDimension方法，进入到了UNISPECIFIED模式