第 8 章

使用布局与组件创建用户界面

8

本章，在为CriminalIntent应用添加crime记录时间及处理状态的过程中，我们将学习到更多有关布局和组件的知识。

8.1 升级 Crime 类

打开Crime.java文件，新增两个实例变量。Date变量表示crime发生的时间，Boolean变量表示crime是否已得到处理，如代码清单8-1所示。

代码清单8-1 添加更多变量到Crime（Crime.java）

```
public class Crime {
    private UUID mId;
    private String mTitle;
    private Date mDate;
    private boolean mSolved;

    public Crime() {
        mId = UUID.randomUUID();
        mDate = new Date();
    }

   ...
}
```

使用默认的Date构造方法初始化Data变量，设置mDate变量值为当前日期。该日期将作为crime默认的发生时间。

接下来，为新增变量生成getter与setter方法（选择Source → Generate Getters and Setters...菜单项），如代码清单8-2所示。

代码清单8-2 已产生的getter与setter方法（Crime.java）

```
public class Crime {
   ...

    public void setTitle(String title) {
```

```
            mTitle = title;
        }


    public Date getDate() {
        return mDate;
    }
    public void setDate(Date date) {
        mDate = date;
    }

    public boolean isSolved() {
        return mSolved;
    }
    public void setSolved(boolean solved) {
        mSolved = solved;
    }
}
```

接下来，使用新组件更新fragment_crime.xml文件中的布局，然后在CrimeFragment.java文件中生成并使用这些组件。

8.2 更新布局

本章结束时，完成后的`CrimeFragment`视图应如图8-1所示。

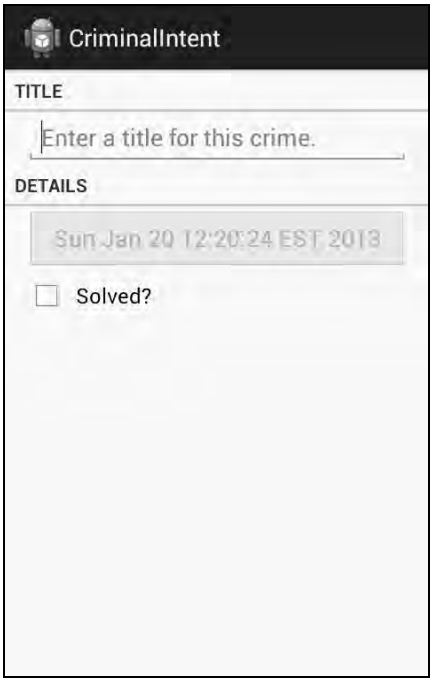

图8-1 CriminalIntent应用界面（本章完成部分）

要得到图8-1所示的用户界面，还需为CrimeFragment的布局添加四个组件，即两个TextView组件、一个Button组件以及一个CheckBox组件。

打开fragment_crime.xml文件，如代码清单8-3所示添加四个组件的定义。可能会出现缺少字符串资源的错误提示，我们稍后会创建这些字符串资源。

代码清单8-3　添加新组件（fragment_crime.xml）

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:orientation="vertical"
  >
  <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/crime_title_label"
    style="?android:listSeparatorTextViewStyle"
    />
  <EditText android:id="@+id/crime_title"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:hint="@string/crime_title_hint"
    />
  <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/crime_details_label"
    style="?android:listSeparatorTextViewStyle"
    />
  <Button android:id="@+id/crime_date"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    />
  <CheckBox android:id="@+id/crime_solved"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:text="@string/crime_solved_label"
    />
</LinearLayout>
```

注意，Button组件定义中没有android:text属性。该按钮将用于显示Crime的发生日期，显示的日期文字内容将通过代码的方式进行设置。

为什么要在Button上显示日期呢？这是在为应用的后续开发做准备。目前，crime的发生日期默认为当前日期且不可更改。第12章，我们将启用按钮组件，通过单击按钮弹出DatePicker组件以供用户设置自定义日期。

布局定义中还有一些新的知识点可供探讨，如style及margin属性。不过首先还是先把添加

了新组件的CriminalIntent运行起来吧。

回到res/values/strings.xml文件中，添加必需的字符串资源，如代码清单8-4所示。

代码清单8-4 添加字符串资源（strings.xml）

```
<resources>
  <string name="app_name">CriminalIntent</string>
  <string name="title_activity_crime">CrimeActivity</string>
  <string name="crime_title_hint">Enter a title for this crime.</string>
  <string name="crime_title_label">Title</string>
  <string name="crime_details_label">Details</string>
  <string name="crime_solved_label">Solved?</string>
</resources>
```

保存修改过的文件，检查确认无拼写错误发生。

8.3 生成并使用组件

CheckBox需显示Crime是否已得到处理。用户勾选清除CheckBox时，Crime的mSolved变量的状态值也需得到相应的更新。

目前，新增Button要做的就是显示Crime类中mDate变量的日期值。

在CrimeFragment.java中，新增两个实例变量，如代码清单8-5所示。

代码清单8-5 添加组件实例变量（CrimeFragment.java）

```
public class CrimeFragment extends Fragment {
    private Crime mCrime;
    private EditText mTitleField;
    private Button mDateButton;
    private CheckBox mSolvedCheckBox;

    @Override
    public void onCreate(Bundle savedInstanceState) {
    ...
```

使用类包组织导入功能，完成CheckBox相关类包的导入。

接下来，在onCreateView(...)方法中，引用新添加的按钮，如代码清单8-6所示设置它的文字属性值为crime的日期，然后暂时禁用灰掉它。

代码清单8-6 设置Button上的文字显示（CrimeFragment.java）

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup parent,
        Bundle savedInstanceState) {
    View v = inflater.inflate(R.layout.fragment_crime, parent, false);

    ...

    mTitleField.addTextChangedListener(new TextWatcher() {
        ...
    });
```

```
        mDateButton = (Button)v.findViewById(R.id.crime_date);
        mDateButton.setText(mCrime.getDate().toString());
        mDateButton.setEnabled(false);

        return v;
    }
```

禁用按钮可以保证它不响应用户的单击事件。按钮禁用后，它的外观样式也会发生改变，以此表明它已处于禁用状态。等到第12章我们设置监听器时，会再次启动该按钮。

下面要处理的是CheckBox组件，在代码中引用它并设置监听器用于更新Crime的mSolved变量值，如代码清单8-7所示。

代码清单8-7　侦听CheckBox状态的变化（CrimeFragment.java）

```
        ...
        mDateButton = (Button)v.findViewById(R.id.crime_date);
        mDateButton.setText(mCrime.getDate().toString());
        mDateButton.setEnabled(false);

        mSolvedCheckBox = (CheckBox)v.findViewById(R.id.crime_solved);
        mSolvedCheckBox.setOnCheckedChangeListener(new OnCheckedChangeListener() {
            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
                // Set the crime's solved property
                mCrime.setSolved(isChecked);
            }
        });

        return v;
    }
```

引入OnCheckedChangeListener接口时，Eclipse将提供分别定义在CompoundButton以及RadioGroup两个类中的接口以供选择。选择CompoundButton接口，因为CheckBox是CompoundButton的子类。

如使用了代码自动补全功能，则可能会在onCheckedChanged(...)方法的代码上方，看到@Overrides注解，而该注解在代码清单8-7中是不存在的。忽略此处差异，OnCheckedChange-Listener接口中的方法不需要@Overrides注解。

运行CriminalIntent应用。尝试勾选清除CheckBox状态，欣赏一下用于显示日期的禁用Button吧。

8.4　深入探讨 XML 布局属性

本小节，我们一起回顾一下fragment_crime.xml文件中添加的一些属性定义，并解答可能一直困扰你的组件与属性相关问题。

8.4.1　样式、主题及主题属性

样式（style）是XML资源文件，含有用来描述组件行为和外观的属性定义。例如，下列样式资源就是用来配置组件，使其显示的文字大小大于正常值的一段代码。

```
<style name="BigTextStyle">
  <item name="android:textSize">20sp</item>
  <item name="android:layout_margin">3dp</item>
</style>
```

我们可以创建自己的样式文件（创建方法请参见第24章）。将属性定义添加并保存在res/values/目录下的样式文件中，然后在布局文件中以@style/my_own_style（样式文件名）的形式引用它们。

再来看一看fragment_crime.xml文件中的两个TextView组件，每个组件都有一个引用Android自带样式文件的style属性。该预定义样式来自于应用的主题，使得屏幕上的TextView组件看起来是以列表样式分隔开的。主题是各种样式的集合。从结构上来说，主题本身也是一种样式资源，只不过它的属性指向了其他样式资源。

Android自带了一些供应用使用的预定义平台主题。例如，在创建CriminalIntent应用时，我们就接受了向导使用Holo Light with Dark Action Bar作为应用主题的建议。

使用主题属性引用，可将预定义的应用主题样式添加给指定组件。例如，在fragment_crime.xml文件中，样式属性值?android:listSeparatorTextViewStyle的使用就是一个很好的例子。

使用主题属性引用，相当于告知Android运行资源管理器：“在应用主题里找到名为listSeparatorTextViewStyle的属性。该属性指向其他样式资源，请将其资源的值放在这里”。

所有Android主题都包括名为listSeparatorTextViewStyle的属性。不过，基于主题的整体观感，它们的定义稍有不同。使用主题属性引用，可以确保TextView组件在应用中拥有正确一致的界面观感。

第24章，我们将学习到更多有关样式及主题的使用知识。

8.4.2 dp、sp以及屏幕像素密度

在fragment_crime.xml文件中，我们以dp为单位来指定边距属性值。dp单位在之前的布局文件中已经出现过了，下面我们来具体学习一下它。

有时需为视图属性指定大小尺寸值（通常以像素为单位，但有时也用点、毫米或英寸）。最常见的属性有：

- 文字大小（text size），指设备上显示的文字像素高度；
- 边距（margin），指定视图组件间的距离；
- 内边距（padding），指定视图外边框与其内容间的距离。

Android使用drawable-ldpi、drawable-mdpi以及drawable-hdpi三个目录下的图像文件自动适配不同像素密度的屏幕。假如图像完成了自动适配，但边距无法缩放适配，又或者用户配置了大于默认值的文字大小，会发生什么情况呢?

为解决这些问题，Android提供了密度无关的尺寸单位（density-independent dimension units）。使用这种单位，可在不同屏幕密度的设备上获得同样大小的尺寸。无需麻烦的转换计算，应用运行时，Android会自动将这种单位转换成像素单位。图8-2展示了这种尺寸单位在Textview上的应用。

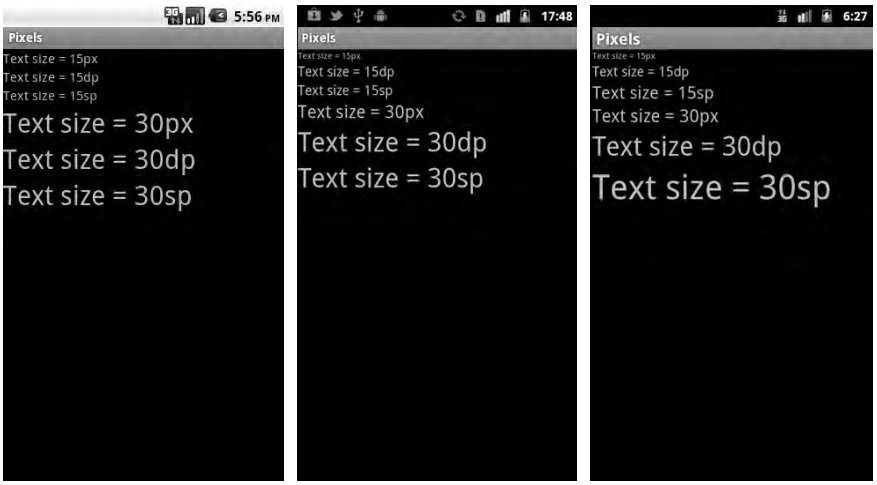

图8-2　应用在Textview上的密度无关尺寸单位（左：MDPI；中：HDPI；右：HDPI+大字体）

- dp

  英文density-independent pixel的缩写形式，意为密度无关像素。在设置边距、内边距或任何不打算按像素值指定尺寸的情况下，通常都使用dp这种单位。如设备屏幕密度较高，密度无关像素会相应扩展至整个屏幕。1dp单位在设备屏幕上总是等于1/160英寸。使用dp的好处是，无论屏幕密度如何，总能获得同样的尺寸。

- sp

  英文scale-independent pixel的缩写形式，意为缩放无关像素。它是一种与密度无关的像素，这种像素会受用户字体偏好设置的影响。我们通常会使用sp来设置屏幕上的字体大小。

- pt、mm、in

  类似于dp的缩放单位。允许以点（1/72英寸）、毫米或英寸为单位指定用户界面尺寸。但在实际开发中不建议使用这些单位，因为并非所有设备都能按照这些单位进行正确的尺寸缩放配置。

在本书及实际开发中，我们几乎只会用到dp和sp两种单位。Android在运行时会自动将它们的值转换为像素单位。

8.4.3　Android开发设计原则

注意，如代码清单8-3所示，我们用16dp单位值设定边距尺寸。该单位值的设定遵循了Android的“48dp调和”设计原则。访问网址http://developer.android.com/design/index.html，可查看Android所有的开发设计原则。

现代Android应用都应严格遵循这些开发设计原则。不过，Android这些设计原则严重依赖于SDK较新版本的功能。而旧版本设备往往无法获得或实现这些功能。虽然有些设计原则可通过使

用支持库获得支持，但多数情况下，我们必须依靠第三方库的使用，如ActionBarSherlock库等，第16章中我们会详细介绍它。

8.4.4 布局参数

我们可能已经注意到，有些属性名称以layout_开头，如android:layout_marginLeft，而其他属性名称则不是，如android:text。

名称不以layout_开头的属性作用于组件。组件生成时，会调用某个方法按照属性及属性值进行自我配置。

名称以layout_开头的属性则作用于组件的父组件。我们将这些属性统称为*布局参数*。它们会告知父布局如何在内部安排自己的子元素。

即使布局对象（如LinearLayout）是布局的根元素，它仍然是一个带有布局参数的子组件。在fragment_crime.xml文件中定义LinearLayout时，我们赋予了它两个属性，即android:layout_width和android:layout_height。LinearLayout生成时，它的父布局会使用这两个属性。这里，CrimeActivity内容视图里的FrameLayout会使用LinearLayout的布局参数。

8.4.5 边距与内边距

在fragment_crime.xml文件中，组件已经有了边距与内边距属性。开发新手有时分不清这两个属性。既然我们已经明白了什么是布局参数，那么二者的区别也就显而易见了。边距属性是布局参数，决定了组件间的距离。假设一个组件对外界毫无所知，边距必须对该组件的父组件负责。

而内边距并非布局参数。属性android:padding告诉组件：在绘制组件自身时，要比所含内容大多少。例如，在不改变文字大小的情况下，想把日期按钮变大一些，可将以下属性添加给Button，如代码清单8-8所示，保存布局文件，然后重新运行应用。

代码清单8-8 内边距属性的实际应用（fragment_crime.xml）

```
<Button android:id="@+id/crime_date"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_marginLeft="16dp"
  android:layout_marginRight="16dp"
  android:padding="80dp"
  />
```

完成界面如图8-3所示。

很可惜，大按钮虽方便，但在继续学习前，我们还是应该删除这个属性。

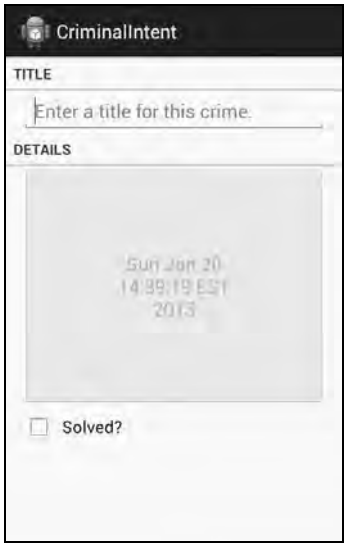

图8-3　实话实说，我喜欢大按钮

8.5　使用图形布局工具

目前为止，布局都是通过手工输入XML的方式创建的。本小节，我们将开始学习使用图形布局工具，并为`CrimeFragment`创建一个水平模式下使用的布局。

设备旋转时，大多数内置布局类，如`LinearLayout`，都会自动拉伸和重新调整自己和自己的子类。不过，有时默认的调整并不能充分利用全部用户界面空间。

运行CriminalIntent应用，然后旋转设备查看水平方位下的`CrimeFragment`布局，如图8-4所示。

图8-4　水平模式下的`CrimeFragment`

可以看到，显示日期的按钮变成了一个长条。水平模式下，按钮如果能与check box并排放置的话会更美观些。要实现这个效果，需创建res/layout-land目录（在包浏览器中，右键单击res目录，选择New → Folder菜单项）。然后将fragment_crime.xml文件复制至res/layout-land目录下。

下面我们使用图形布局工具进行一些调整。如果已经在编辑器中打开了res/layout/fragment_crime.xml文件，请先关闭它。现在打开res/layout-land/fragment_crime.xml文件并切换至图形布局标签页。

图形布局工具的中间区域是布局的界面预览视图。左边是组件面板视图。该视图包含了所有我们希望用到的按类别组织的组件。如图8-5所示。

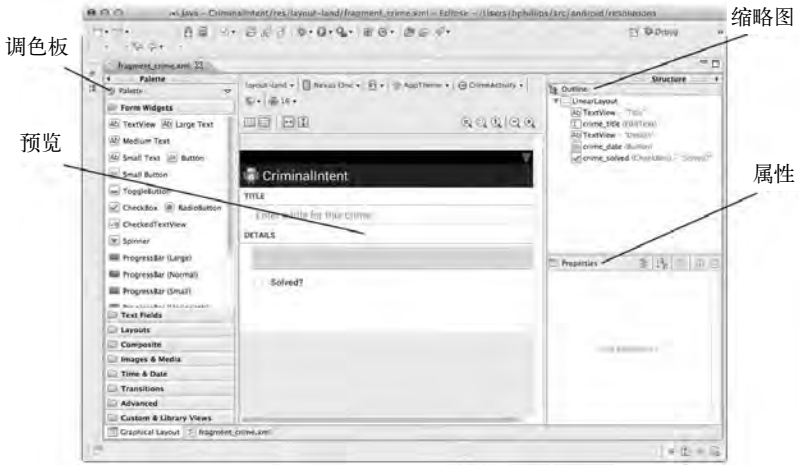

图8-5 图形布局工具中的视图

框架视图位于预览视图的右边。框架显示了组件是如何在布局中组织的。

框架视图下是属性视图。在此视图中，我们可以查看并编辑框架视图中选中的组件属性。

现在，参照图8-6，看看要对布局做哪些调整。

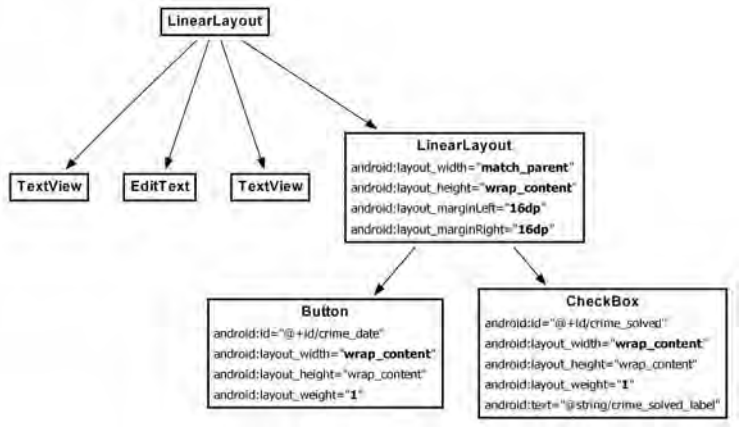

图8-6 CrimeFragment的水平模式布局

需要的调整可分为四部分：

- 新增一个LinearLayout组件；

- 编辑LinearLayout组件的属性；
- 将Button以及CheckBox组件设置为LinearLayout的子组件；
- 更新Button和CheckBox组件的布局参数。

8.5.1　添加新组件

在组件面板中选中目标组件，然后将其拖曳到框架视图中，即可完成组件的添加。单击组织面板的*布局*类别，选中水平的LinearLayout并将其拖曳到框架视图中。把LinearLayout放置在根LinearLayout上，将其新增为根LinearLayout的直接子组件，如图8-7所示。

图8-7　添加到fragment_crime.xml中的LinearLayout

也可以直接把组件从组件面板中拖曳到预览界面中，从而完成组件的添加。但由于布局组件通常是空的或者被其他视图所遮挡，所以要想获得所需的组件层级结构，很难判断到底该把组件放在预览视图的哪个部分。因此，拖曳组件到框架视图中是一种更为容易的方式。

8.5.2　属性视图中编辑组件属性

选择框架视图中新添加的LinearLayout后，属性视图中会显示出它的属性。依次展开*布局参数*以及*边距*类别。

我们需要调整LinearLayout的边距来匹配其他组件。选中Left右边栏位，输入16dp；选中“Right”右边栏位，同样输入16dp，如图8-8所示。

图8-8　在属性视图中设置边距属性

（在有些平台上，尝试在相关栏位进行输入时会弹出一个窗口。该弹出窗口是某已知缺陷的临时解决办法。但糟糕的是，在弹出窗口依然无法进行输入。因此，如碰到这种情况，请保存布局文件，切换到XML代码模式，然后将左右边距属性值从EditText复制到LinearLayout。）

保存布局文件，选中预览界面底部的fragment_crime.xml标签切换到XML模式。应该可以看到带有刚才新增边距属性的LinearLayout元素。

8.5.3 在框架视图中重新组织组件

接下来我们将Button及CheckBox调整为新增LinearLayout的子组件。返回到图形布局工具，在框架视图中，选中Button后将其拖曳至LinearLayout上。

从框架视图可以看出，Button现在是新增LinearLayout的一个子组件。对CheckBox进行同样的操作，如图8-9所示。

图8-9 Button及CheckBox现在是新增LinearLayout的子组件

如果子组件排列顺序不合适，可在框架视图中通过拖曳重新安排顺序。当然，也可以直接删除框架视图布局中的组件。但要当心，删除组件也会连带删除它的子组件。

回到预览界面，CheckBox貌似不见了。这是因为Button遮挡住了它。LinearLayout考虑到了它第一个子组件（Button）的宽度属性（match_parent），并赋予了它全部空间，以至于CheckBox没有了立身之地，如图8-10所示。

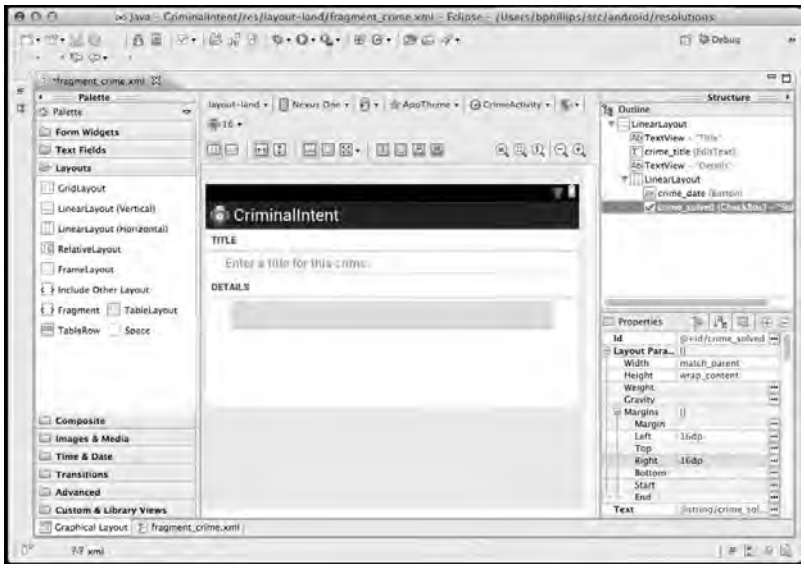

图8-10 Button子组件遮住了CheckBox组件

通过调整子组件的布局参数，可让其他子组件获得LinearLayout的平等对待。

8.5.4　更新子组件的布局参数

首先，在框架视图中选中日期按钮。在属性视图里，单击当前宽度值栏位，在弹出的下拉框中选择wrap_content，如图8-11所示。

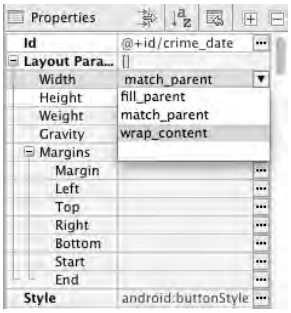

图8-11　调整Button的宽度属性值为wrap_content

接下来，删除按钮左右16dp的边距值。既然按钮已被放置在LinearLayout里面了，因此也就不再需要边距值了。

最后，在*布局参数*区找到Weight栏位，设置Weight值为1。该栏位在XML文件中对应的属性是android:layout_weight，如图8-6所示。

在框架视图里选中CheckBox组件，参照Button进行同样的属性调整：设置Width值为wrap_content，Weight值为1，边距值为空值。

完成后，查看预览界面确认两个组件都能正确显现。然后保存文件，并返回XML文件确认已做的修改。代码清单8-9展示了完成后的XML代码。

代码清单8-9　图形工具创建的布局XML（layout-land/fragment_crime.xml）

```
...

<TextView
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:text="@string/crime_details_label"
  style="?android:listSeparatorTextViewStyle"
  />
<LinearLayout
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_marginLeft="16dp"
  android:layout_marginRight="16dp" >
    <Button
      android:id="@+id/crime_date"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_weight="1" />
    <CheckBox
```

```
            android:id="@+id/crime_solved"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/crime_solved_label" />
    </LinearLayout>
</LinearLayout>
```

8.5.5 android:layout_weight属性的工作原理

android:layout_weight属性告知LinearLayout如何进行子组件的布置安排。我们已经为两个组件设置了同样的值，但这并不意味它们在屏幕上占据着同样的宽度。在决定子组件视图的宽度时，LinearLayout使用的是layout_width与layout_weight参数的混合值。

LinearLayout是分两个步骤来设置视图宽度的。

第一步，LinearLayout查看layout_width属性值(竖直方位则查看layout_height属性值)。Button和CheckBox组件的layout_width属性值都设置为wrap_content，因此它们获得的空间大小仅够绘制自身，如图8-12所示。

（在预览界面，很难看出layout_weight是如何工作的，因为按钮显示内容不是布局的一部分。图8-12展示了按钮组件在已经显示了日期的情况下，LinearLayout布局的显示效果）

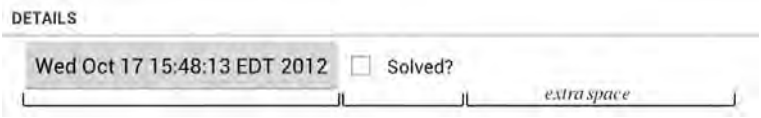

图8-12 第一步：基于layout_width属性值分配的空间大小

第二步，LinearLayout依据layout_weight属性值进行额外的空间分配，如图8-13所示。

图8-13 第二步：基于1：1 layout_weight属性值分配的额外空间

在布局中，Button和CheckBox组件拥有相同的layout_weight属性值，因此它们均分了同样大小的额外空间。如将Button组件的weight值设置为2，那么它将获得2/3的额外空间，CheckBox组件则获得剩余的1/3，如图8-14所示。

图8-14 基于2：1 layout_weight属性值不等比分配的额外空间

weight设置值也可以是浮点数。对于weight设置值，开发者有着各自的使用习惯。在fragment_ crime.xml中，我们使用的是一种cocktail recipe式的weight设置风格。另一种常见的设定方式是各组件属性值加起来等于1.0或100。这样，上个例子中按钮组件的weight值则应该是0.66或66。

如想让LinearLayout分配完全相同的宽度给各自的视图，该如何处理呢？很简单，只需设置各组件的layout_width属性值为0dp以避开第一步的空间分配就可以了，这样LinearLayout就会只考虑使用layout_weight属性值来完成所需的空间分配了，如图8-15所示。

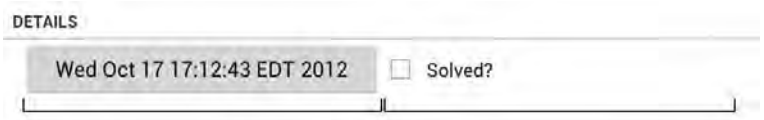

图8-15　如layout_width="0dp"，则只考虑layout_weight属性值

8.5.6　图形布局工具使用总结

图形布局工具非常有用。随着ADT各个版本的发布，Android一直在努力改进这一工具。然而，图形布局工具却仍时不时地存在着一些问题，如运行慢和有缺陷等。有时可能不得不回到XML的编辑方式。我们可在调整布局的两种模式间不断切换。为避免切换时元素的丢失，请记得先保存文件。

本书应用开发的布局创建，可选择使用图形布局工具。后续章节中如需创建布局，我们都会提供如图8-6所示的示意图。在创建布局时，究竟是使用XML方式、图形布局工具还是两种方式同时使用，一切请自行决定。

8.5.7　组件ID与多种布局

除组件摆放位置不一样外，CriminalIntent应用创建的两个布局并没有太大的差别。但有时，设备处于不同方向时使用的布局会有很大差异。如发生这样的情况，应在保证组件已确实存在后，再在代码中引用它们。

如果一个组件只存在于一个布局上，则需先在代码中进行空值检查，确认当前方向的组件存在后，再调用相关方法：

```
    Button landscapeOnlyButton = (Button)v.findViewById(R.id.landscapeOnlyButton);
    if (landscapeOnlyButton != null) {
        // Set it up
}
```

最后，请记住，定义在水平或竖直布局文件里的同一组件必须具有同样的android:id属性，这样代码才能引用到它。

8.6 挑战练习：日期格式化

与其说Date对象是一个常见的日期，不如说它是一个时间戳。调用Date对象的toString()方法，可获得一个时间戳。因此，CriminalIntent应用的按钮上显示的也是一个时间戳。尽管时间戳有利于文档记录，但在按钮上显示人们习惯看到的日期应该会更好，如“Oct 12, 2012”。使用android.text.format.DateFormat类的实例可实现此目标。为正确使用它，请先查阅Android文档库中有关该类的相关参考页。

使用DateFormat类中的方法，可获得日期的常见格式。或者也可以实现自己的字符串格式化。最后我们来挑战一个更有难度的任务：创建一个包含星期的格式字符串，如“Tuesday, Oct 12, 2012”。