Phát triển ứng dụng Smartphone

Tài liệu lưu hành nội bộ

Đây là tài liệu tham khảo sử dụng trong môn học Lập trình ứng dụng Smartphone – Android được tổng hợp, biên soạn từ nhiều nguồn bởi các thành viên của Nhóm nghiên cứu và ứng dụng công nghệ A106-Đại học Hoa Sen.

Phát triển ứng dụng Smartphone

Phần 02: Giao diện người dùng

Lê Đức Huy

Email: leduchuy89vn@gmail.com

Mục lục

Tại thời điểm này, bạn có thể thấy rằng thành phần cơ bản của một ứng dụng android chính là **Activity**. **Activity** là thành phần tối quan trọng của bất kỳ một ứng dụng Android nào. Thuật ngữ Activity chỉ một việc mà người dùng có thể thực hiện trong một ứng dụng Android. Do gần như mọi activity đều tương tác với người dùng nên lớp Activity đảm nhận việc tạo ra một cửa sổ (**window**) để người lập trình đặt lên đó một giao diện người dùng bằng phương thức **setContentView(View)**. Ta có thể xem Activity là một cửa sổ ứng dụng. Một activity có thể mang nhiều dạng khác nhau: một cửa sổ toàn màn hình (full screen window), một cửa sổ floating (với windowsIsFloating) hay nằm lồng bên trong một activity khác (với **ActivityGroup**). Cửa sổ ứng dụng mà Activity tạo ra có thể chứa các **widgets** như button, lable, text boxex ... Thông thường, để xây dựng giao diện người dùng ta sử dụng file xml trong forder res/layout để định nghĩa các đối tượng đồ họa muốn thể hiện lên giao diện người dùng cùng với các thuộc tính tương ứng quyết định cách thức thể hiện đối tượng đồ họa đó trên cửa sổ ứng dụng. Dưới đây là một ví dụ file xml định nghĩa các đối tượng đồ họa thể hiện giao diện người dùng của ứng dụng:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayout
     xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent">
     <TextView
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="@string/hello"/>
</LinearLayout>
```

Khi chương trình thực thi, giao diện đồ họa được nạp từ file xml trong phương thức **onCreate()** bằng cách sử dụng phương thức **setContentView()** của **Activity** để chỉ định file xml mong muốn.

Ví dụ:

```
@Override
publicvoid onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
}
```

Khi mã code của chương trình được biên dịch, mỗi phần tử trong file xml được biên dịch thành những đối tượng đồ họa tương ứng cùng với các thuộc tính được cung cấp kèm theo các phần tử trong thẻ xml. Android sau đó sẽ tạo ra các thể hiện của các đối tượng đồ họa người dùng cung cấp cho Activity khi nó được nạp.

Việc xây dựng một giao diện người dùng bằng cách sử dụng file xml rất đơn giản, tuy nhiên sẽ tốn rất nhiều thời gian nếu nhà lập trình cần phải xây dựng giao diện đồ họa người dùng một các linh

hoạt trong lúc thực thi (Ví dụ: Khi xây dựng game). Tuy nhiên các nhà phát triển ứng dụng vẫn có thể xây dựng giao diện đồ họa người dùng bằng cách sử dụng câu lệnh.

Views and ViewGroups

Một **Activity** chứa các đối tượng **View** và các đối tượng **ViewGroup**. Một đối tượng View là một đối tượng đồ hoạ dùng hiển thị nội dung lên màn hình. Ví dụ: Button, lablel, text boxes … Một **View** được dẫn suất từ các class: **android.view.View**.Một hoặc nhiều View có thể được nhóm lại cùng nhau bên trong một **ViewGroup**. Một **ViewGroup** cung cấp khả năng cho phép bố trí các đối tượng đồ hoạ chứa trong nó một cách phù hợp. Có thể sử dụng các ViewGroup để bố trị các đối tượng đồ hoạ theo dòng, theo cột, theo dạng bảng hay theo sự chỉ định toạ độ cụ thể. Mỗi **ViewGroup** được dẫn xuất từ class **android.view.ViewGroup**. Android hỗ trợ các **ViewGroup** sau đây:

LinearLayout

AbsoluteLayout

TableLayout

RelativeLayout

FrameLayout

ScrollView

Dưới đây là chuỗi các bài học sẽ nói rõ về từng loại **ViewGroup** một cách chi tiết hơn. Chú ý rằng, trong các bài thực hành dưới đây thường xuyên lồng các loại **ViewGroup** với nhau để xây dựng một giao diện người dùng.

1 ViewGroup

Tạo 1 Project đơn giản

Trong hướng dẫn này, chúng ta sẽ tạo một project android với các thông số như sau:

Project name: UIExample

Build target: Android 2.3.3.

Application name: UI Example

Package name: android.uiexample

Create Activity: main

1.1 LinearLayout

LinearLayout là loại ViewGroup cho phép sắp xếp các đối tượng **View** (chứa trong nó) trong một cột đơn hay một hàng đơn. Để thấy cách một **LinearLauout** làm việc như thế nào, hãy bổ sung vào file main.xml trong forder layout như sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
      <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/hello"
/>
</LinearLayout>
```

Trong file **main.xml** này, ta quan sát thấy rằng phần tử gốc là **<LinearLayout>** và nó có chứa một thành phần **<TextView>**. Phần tử **<LinearLayout>** đó sẽ xác định cách hiển thị của các đối tượng View chứa trong nó thông qua các thuộc tính.

View và **ViewGroup** có một tập các thuộc tính rất hay sử dụng được liệt kê qua bảng sau:

Thuộc tính	Đặc tả
layout_width	Xác định chiều rộng của một View hoặc ViewGroup
layout_height	Xác định chiều cao của một View hoặc ViewGroup
layout_marginTop	Xác định phần mở rộng phía bên trên của View hoặc ViewGroup
layout_marginBottom	Xác định phần mở rộng phía bên dưới của View hoặc ViewGroup
layout_marginLeft	Xác định phần mở rộng phía bên trái của View hoặc ViewGroup
layout_marginRight	Xác định phần mở rộng phía bên phải của View hoặc ViewGroup
layout_gravity	Xác định cách đặt các đối tượng View con
layout_weight	Xác định phần kích thước thể hiện của View hoặc ViewGroup (Sử dụng với thuộc tính layout_width hoặc layout_height được gán giá trị "**fill_parent**" "**wrap_content**" hoặc giá trị cố định: 100px)
layout_x	Xác định toạ độ x của the View hoặc ViewGroup
layout_y	Xác định toạ độ y của the View hoặc ViewGroup

Chú ý rằng một vài các thuộc tính trên chỉ được sử dụng khi một View được định nghĩa rõ ràng trong một ViewGroup cụ thể.

Ví dụ:

- Hai thuộc tính **layout_weight** và **layout_gravity** chỉ được sử dụng nếu View nằm trong LinearLayout hoặc là trong TableLayout.

Xem xét phần code xml sau:

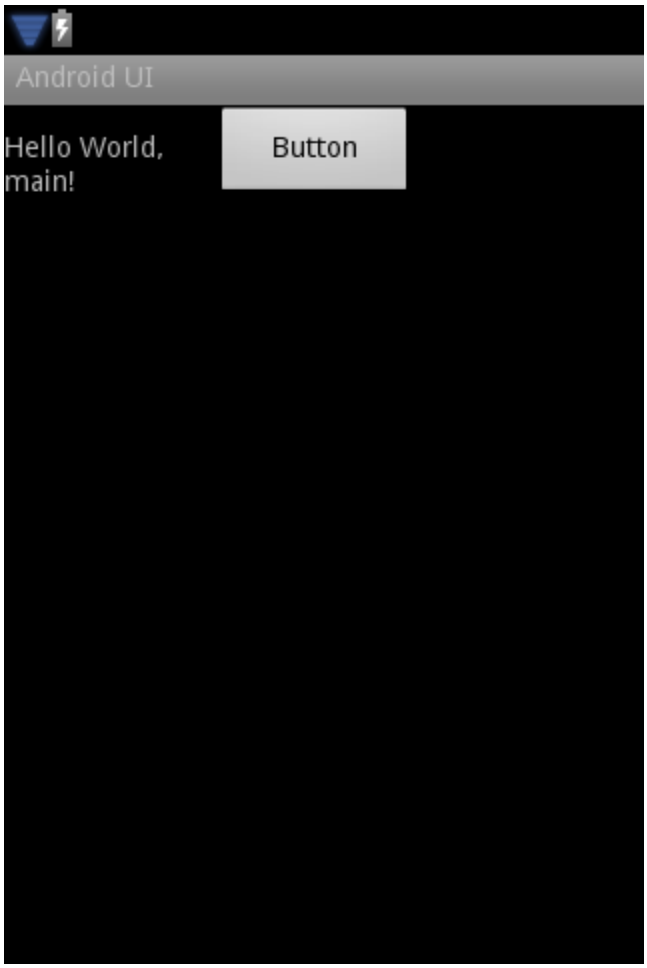
```
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
     android:text="@string/hello"/>
```

- Phần tử TextView ở trên có chiều rộng bằng với chiều rộng của phần tử cha của nó (***fill_parent***- Trong trường hợp này nó bằng với chiều rộng màn hình). Chiều cao trong trường hợp này, khi bạn khai báo là ***wrap_content*** thì chiều cao của nó sẽ là chiều cao cần thiết mà nội dụng của **TextView** dùng để hiển thị đoạn text.

Còn trong trường hợp dưới đây, chiều rộng của TextView được đặt là 105 pixels.

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayout
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     xmlns:android="http://schemas.android.com/apk/res/android">
     <TextView
     android:layout_width="105px"
          android:layout_height="wrap_content"
     android:text="@string/hello">
     </TextView>
     <Button
          android:layout_width="100px"
          android:layout_height="wrap_content"
          android:text="Button">
     </Button>
</LinearLayout>
```

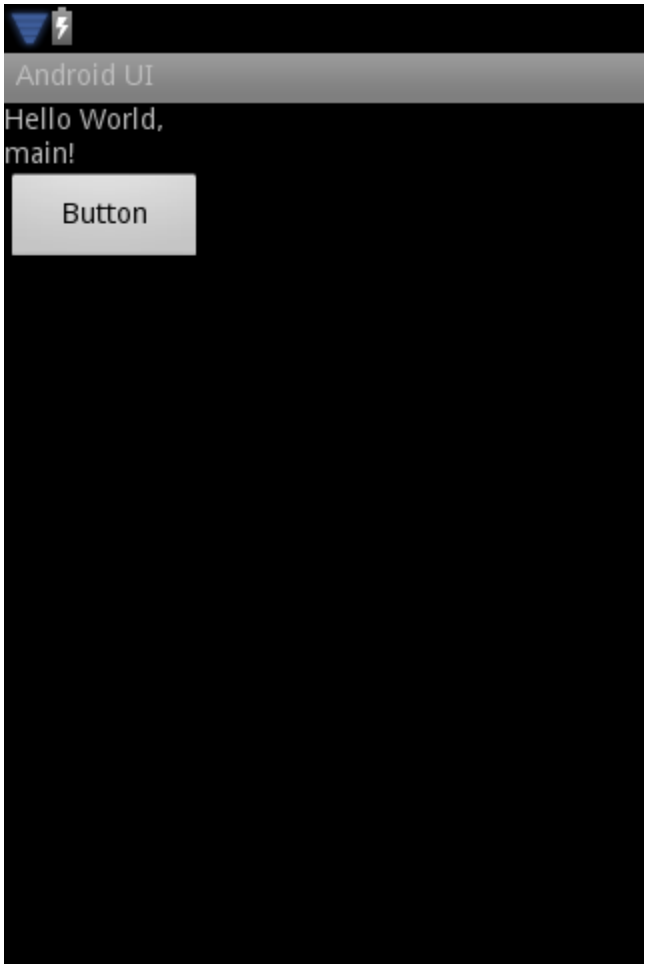
Khi phần code giao diện trên được nạp lên Activity sẽ được giao diện như sau:



Mặc định của **LinearLayout** sẽ sắp xếp các đối tượng đồ hoạ (View) trong nó theo chiều ngang, nếu bạn muốn thay đổi nó thành chiều dọc bạn có thể sửa phần code phần tử gốc **LinearLayout** lại như sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayout
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:orientation="vertical"
      xmlns:android="http://schemas.android.com/apk/res/android">
      <TextView
            android:layout_width="105px"
            android:layout_height="wrap_content"
            android:text="@string/hello">
      </TextView>
      <Button
            android:layout_width="100px"
            android:layout_height="wrap_content"
            android:text="Button">
      </Button>
```

```
</LinearLayout>
```



Bằng cách thay đổi thuộc tính **android:orientation="vertical"** ta đã thay đổi cách **LinearLayout** sắp xếp các đối tượng đồ hoạ bên trong nó. Thuộc tính **android:orientation** có thể gán hai giá trị tương ứng sau:

- **vertical**: Dùng để xác định các đối tượng đồ hoạ chứa trong **LinearLayout** được sắp xếp theo chiều dọc.
- **horizontal**: Dùng để xác định các đối tượng đồ hoạ chứa trong **LinearLayout** được sắp xếp theo chiều ngang.

Giá trị của thuộc tính kích thước một đối tượng đồ hoạ trong Android được phân thành hai loại:

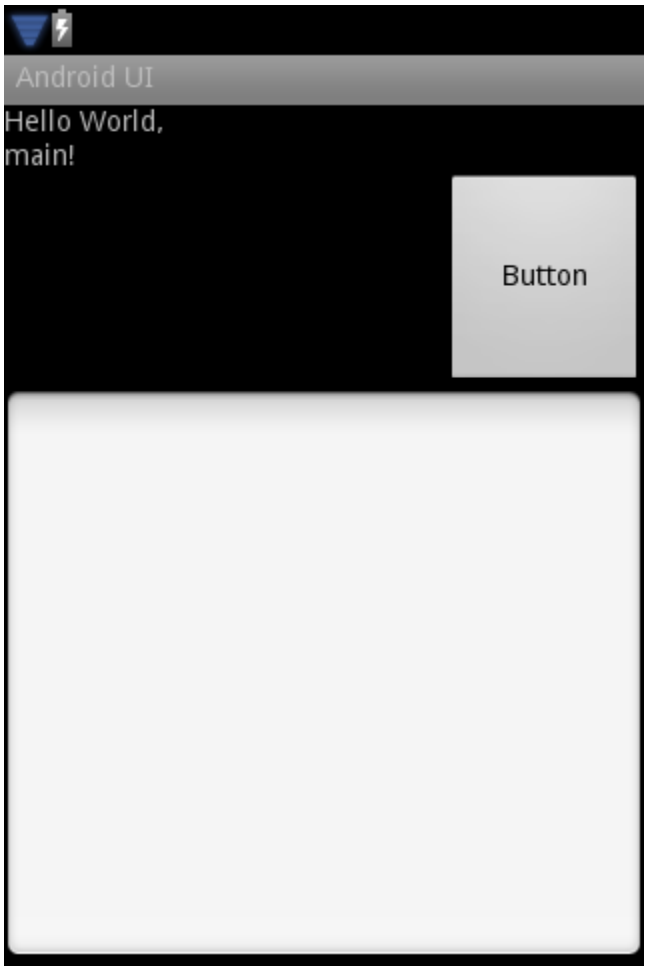
- Giá trị cố định (Xác định cụ thể giá trị kích thước. Ví dụ: 150px).
- Giá trị linh hoạt. Giá trị này phụ thuộc vào một trong hai yếu tố:
    - Nội dung mà một đối tượng đồ hoạ muốn hiện thị.Với View thì nội dung này là đoạn text,…. Với ViewGroup là các phần tử con mà nó chứa.

- Phần không gian mà đối tượng cha còn trống.

Trong **LinearLayout** bạn có thể áp dụng các thuộc tính **layout_weight** và **layout_gravity** như sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayout
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical">
     <TextView
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="@string/hello"/>
     <Button
     android:layout_width="100px"
     android:layout_height="wrap_content"
     android:text="Button"
     android:layout_gravity="right"
     android:layout_weight="0.2"/>
     <EditText
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:textSize="18sp"
     android:layout_weight="0.8"/>
</LinearLayout>
```

Hình sau sẽ hiển thị **Button** được gán vào bên phải của phần tử cha của nó (Trong trường hợp này là **LinearLayout**) sử dụng thuộc tính **layout_gravity**, và sử dụng thuộc tính **layout_weight** cho các **Button**, **EditText** (tổng giá trị của **layout_weight** bằng 1). Sau khi nạp phần giao diện trên vào Activity ta sẽ được giao diện như sau:



Phân tích phần giao diện bên trên ta sẽ thấy rằng:

- Thẻ gốc của toàn bộ giao diện là **LinearLayout** với hai thuộc tính **android:layout_width="fill_parent"**và **android:layout_height="fill_parent"**xác định chiều rộng và chiều cao **LinearLayout** sẽ chiếm toàn chiều rộng và chiều cao của đối tượng chứa nó. Nhưng do thẻ **LinearLayout** này là thẻ gốc (Thẻ ngoài cùng) nên nó sẽ được đặt lên màn hình điện thoại khi ứng dụng được mở lên. Vì vậy nó sẽ chiếm toàn bộ chiều rộng và chiều cao của màn hình điện thoại. Đây là một trường hợp giá trị kích thước của một đối tượng đồ hoạ là linh hoạt và phụ thuộc vào không gian còn trống của đối tượng cha (Trong trường hợp này đối tượng cha là màn hình điện thoại, do chưa chứa gì hết nên nó được xem là rỗng).
- Thẻ **TextView** có thuộc tính **android:layout_width="fill_parent"** và **android:layout_height="wrap_content"** xác định kính thước chiều ngang của **TextView** sẽ chiếm toàn bộ khoảng trống còn lại đối tượng chứa nó (**LinearLayout**), và chiều cao của **TextView** sẽ phụ thuộc vào nội dung nó cần hiển thị (**TextView** chiếm một khoảng chiều cao bằng với chiều cao cần để hiển thị đoạn text mà nó đang giữ).

- Thẻ **Button** có thuộc tính **android:layout_width=“100px”**xác định kích thước chiều rộng của nó là 100px. Đây là một trường hợp mà giá trị kích thước của một đối tượng đồ hoạ là cố định (Chiều rộng = 100px).
- Thẻ **Button** có thuộc tính **android:layout_gravity="right"**xác định rằng đối tượng **Button** khi được nạp lên giao diện sẽ canh lề bên phải so với đối tượng chứa nó.

Bảng sau liệt kê các giá trị có thể gán cho thuộc tính **android:layout_gravity**:

Giá trị	Mô tả
top	Đặt đối tượng đồ hoạ lên đỉnh trên cùng của ViewGroup chứa nó. Thuộc tính này không thay đổi giá trị kích thước.
bottom	Đặt đối tượng đồ hoạ xuống dưới cùng của ViewGroup chứa nó. Thuộc tính này không thay đổi giá trị kích thước.
left	Đặt đối tượng đồ hoạ bên trái của ViewGroup chứa nó. Thuộc tính này không thay đổi giá trị kích thước.
right	Đặt đối tượng đồ hoạ bên phải của ViewGroup chứa nó. Thuộc tính này không thay đổi giá trị kích thước.
center_vertical	Đặt đối tượng đồ hoạ canh giữa theo chiều dọc của ViewGroup chứa nó. Thuộc tính này không thay đổi giá trị kích thước.
fill_vertical	Tăng kích thước của của đối tượng đồ hoạ tràn đầy chiều cao ViewGroup chứa nó.
center_horizontal	Đặt đối tượng đồ hoạ canh giữa theo chiều ngang của ViewGroup chứa nó. Thuộc tính này không thay đổi giá trị kích thước.
fill_horizontal	Tăng kích thước của của đối tượng đồ hoạ tràn đầy chiều rộng ViewGroup chứa nó.
center	Đặt đối tượng đồ hoạ ở chính giữa của ViewGroup chứa nó. Thuộc tính này không thay đổi giá trị kích thước.
fill	Tăng kích thước của của đối tượng đồ hoạ tràn đầy kích thước ViewGroup chứa nó.
clip_vertical	Additional option that can be set to have the top and/or bottom edges of the child clipped to its container's bounds. The clip will be based on the vertical gravity: a top gravity will clip the bottom edge, a bottom gravity will clip the top edge, and neither will clip both edges.
clip_horizontal	Additional option that can be set to have the left and/or right edges of the child clipped to its container's bounds. The clip will be based on the horizontal gravity: a left gravity will clip the right edge, a right gravity

Lưu ý: Thuộc tính android:layout_gravity có thể gán nhiều hơn một thuộc tính.

VD: **android:layout_gravity:“top|left”**

Cả hai thẻ **Button** và **EditText** đều có thuộc tính **android:layout_height=“wrap_content”**và **android:layout_weight** với các giá trị tương ứng: **Button**: 0.2 **EditText**: 0.8.Nếu thuộc tính kích thước của một đối tượng đồ hoạ (**android:layout_height** và **android:layout_width**) xác định là **wrap_content** thì kích thước này sẽ vừa đủ để hiển thị nội dung của nó (Với View thì nội dung này là đoạn text,…. Với ViewGroup là các phần tử con mà nó chứa). Tuy nhiên nếu sử dụng thêm thuộc tính **android:layout_weight** thì kích thước của các đối tượng đồ hoạ này sẽ phụ thuộc vào giá trị của thuộc tính này.Để giải thích về cách sử dụng thuộc tính **android:layout_weight** ta xem xét lại ví dụ trên như sau:

- Ta có một thẻ gốc **LinearLayout** chứa ba đối tượng **TextView**, **Button**, **EditText**. Cả ba đối tượng này được sắp xếp theo chiều dọc từ trên xuống.
- Chiều cao của **TextView** là **“wrap_content”**nghĩa là nó sẽ có một độ cao vừa đủ để hiển thị một đoạn text cố định (“Hello Android”-Ta sẽ bàn về khái niệm @string/hello ở phần sau của giáo trình).
- Sau khi xác định được chiều cao của **TextView** thì ta sẽ còn lại một khoảng trống dùng để sắp xếp **Button** và **EditText**. Ta gọi khoảng trống này là H. Chiều cao của hai đối tượng này là “**wrap_content**” và giá trị **android:layout_weight**lần lượt là 0.2 và 0.8.Lúc này chiều cao của **Button** sẽ là 0.2*H và chiều cao của **EditText** là 0.8*H.

Ta cần lưu ý rằng tổng **android:layout_weight**của các đối tượng trong cùng một **ViewGroup**luôn là 1.0 (**android:layout_weight** của **Button** + **android:layout_weight** của**EditText** = 1.0)

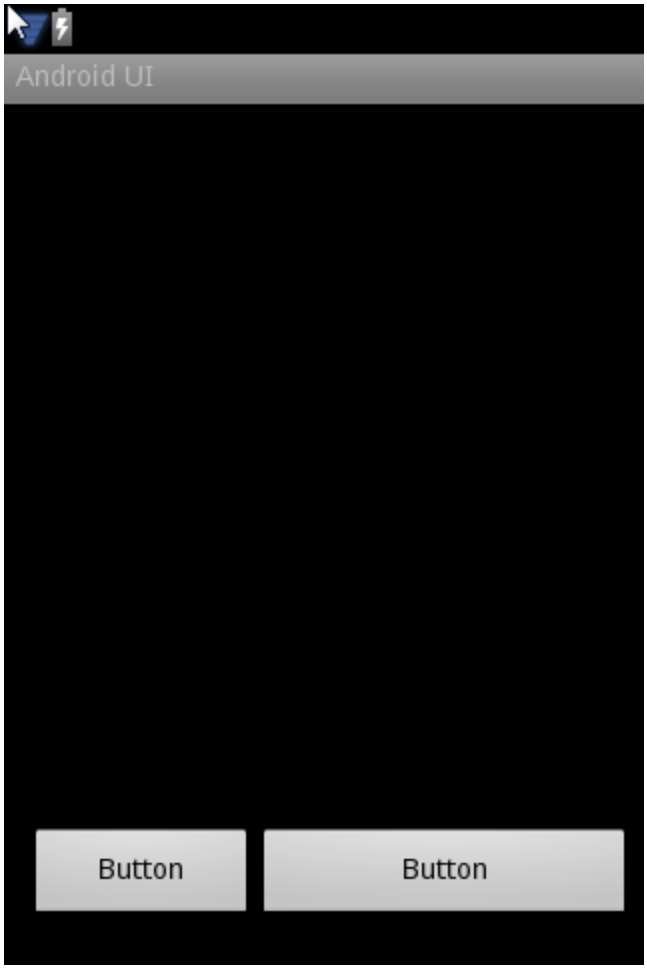
1.2 AbsoluteLayout

AbsoluteLayout cho bạn chỉ định chính xác vị trí của các thành phần con, xem xét ví dụ sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<AbsoluteLayout
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      xmlns:android="http://schemas.android.com/apk/res/android">
      <Button
            android:layout_width="188px"
      android:layout_height="wrap_content"
      android:text="Button"
      android:layout_x="126px"
      android:layout_y="361px"
      />
<Button
```

```
      android:layout_width="113px"
      android:layout_height="wrap_content"
      android:text="Button"
      android:layout_x="12px"
      android:layout_y="361px"
/>
</AbsoluteLayout>
```

Khi nạp phần code xml trên lên Activity thì ta sẽ được giao diện như sau:



Ở ví dụ trên xuất hiện thêm hai thuộc tính là:

- **android:layout_x**: Xác định toạ độ x của đối tượng trong toạ độ Oxy.
- **android:layout_y**: Xác định toạ độ x của đối tượng trong toạ độ Oxy.

Lưu ý: Gốc toạ được xác định được xác định là điểm trên cùng ở góc trái của **ViewGroup**gần nhất chứa nó.

1.3 TableLayout

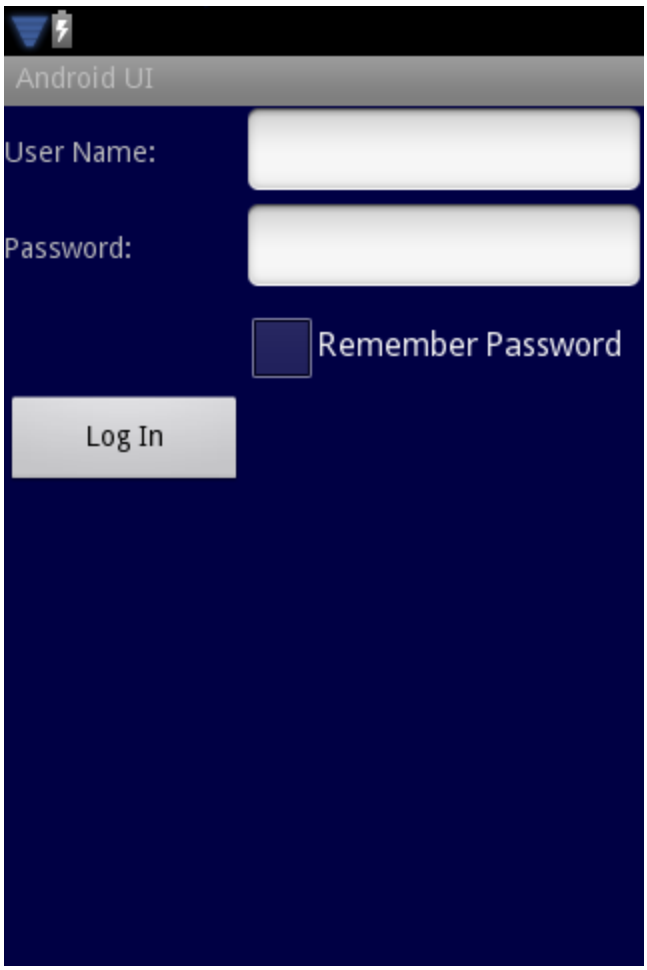
TableLayout dùng để tổ chức các đối tượng **View**dưới dạng một bảng gồm nhiều hàng, nhiều cột. Mỗi dòng được thể hiện bằng một thẻ <**TableRow**>. Trong đó mỗi dòng có thể chứa một hoặc nhiều đối tượng View. Mỗi đối tượng View đặt trên một dòng sẽ tạo thành một ô trong giao diện bảng do TableLayout tạo ra. Chiều rộng của mỗi cột được xác định bằng chiều rộng lớn nhất của các ô nằm trên cùng một cột.

Ví dụ:

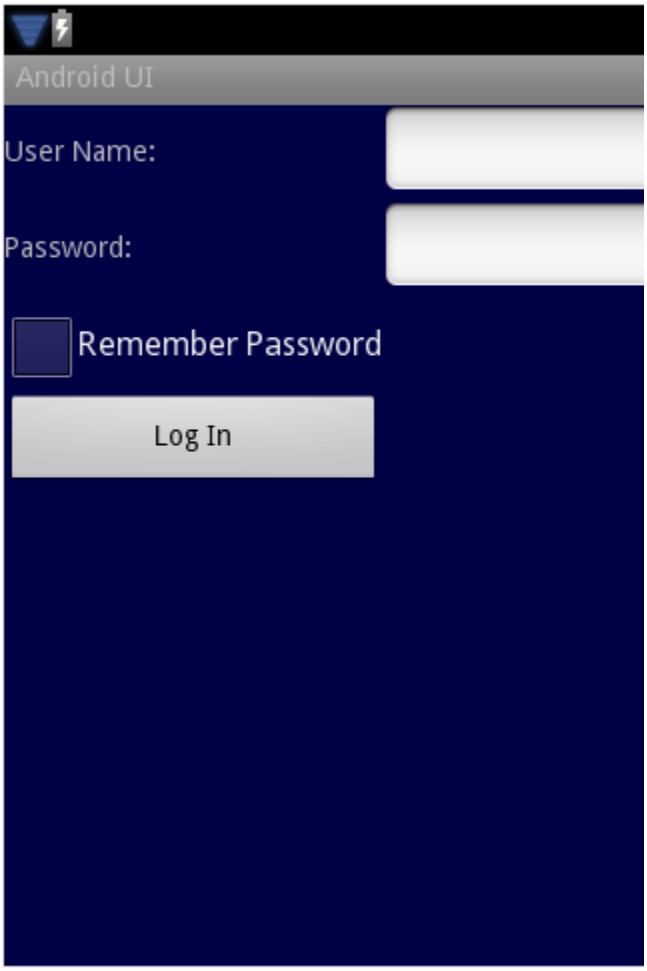
```
<?xmlversion="1.0"encoding="utf-8"?>
<TableLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_height="fill_parent"
      android:layout_width="fill_parent"
      android:background="#000044">
      <TableRow>
            <TextView
            android:text="User Name:"
            android:width="120px"
      />
            <EditText
            android:id="@+id/txtUserName"
            android:width="200px"/>
      </TableRow>
      <TableRow>
            <TextView
            android:text="Password:"
      />
            <EditText
            android:id="@+id/txtPassword"
            android:password="true"
      />
      </TableRow>
      <TableRow>
            <TextView/>
            <CheckBox
                        android:id="@+id/chkRememberPassword"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Remember Password"
      />
      </TableRow>
      <TableRow>
            <Button
            android:id="@+id/buttonSignIn"
            android:text="Log In"/>
```

```
    </TableRow>
</TableLayout>
```

Khi nạp phần code xml trên lên Activity thì ta sẽ được giao diện như sau:



Ở ví dụ trên, giao diện đồ hoạ được tổ chức dưới dạng một bàng gồm có bốn dòng với hai cột. Ô ở ngay dưới ô chứa **TextView** Password có một thẻ **<TextView/>**rỗng. Nếu không đặt thẻ này, thì **CheckBox** Remember Password sẽ xuất hiện ngay bên dưới **TextView** Password như hình sau:



1.4 RelativeLayout

RelativeLayout cho phép bạn chỉ định mối quan hệ giữa các đối tượng **View/ViewGroup**. Ta xét ví dụ sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<RelativeLayout
      android:id="@+id/RLayout"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      xmlns:android="http://schemas.android.com/apk/res/android"
>
      <TextView
            android:id="@+id/lblComments"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Comments"
            android:layout_alignParentTop="true"
            android:layout_alignParentLeft="true"
```

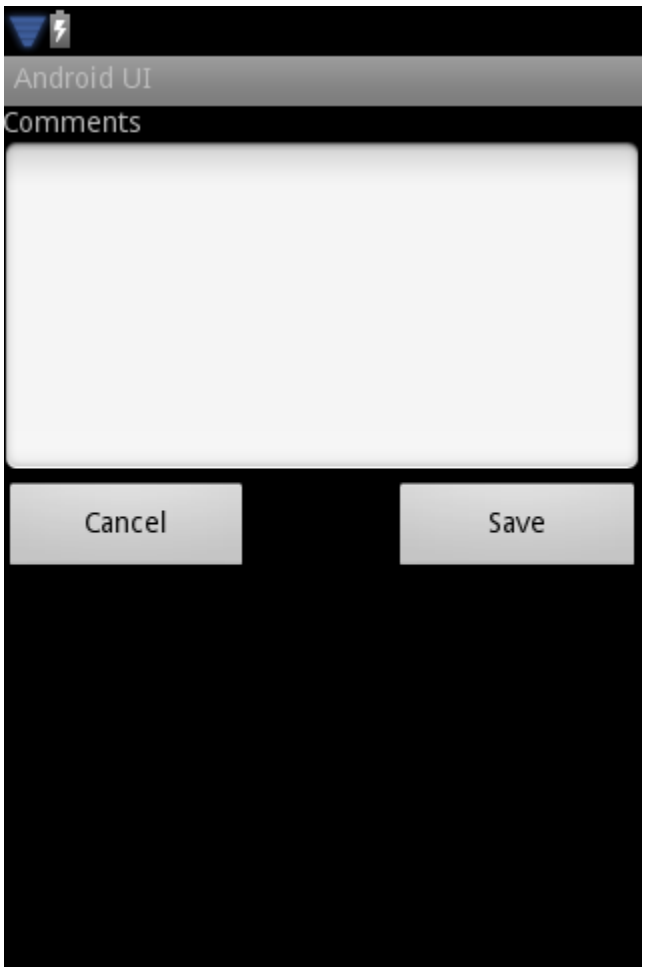
```
        />
        <EditText
              android:id="@+id/txtComments"
              android:layout_width="fill_parent"
              android:layout_height="170px"
              android:textSize="18sp"
              android:layout_alignLeft="@+id/lblComments"
              android:layout_below="@+id/lblComments"
              android:layout_centerHorizontal="true"
        />
        <Button
              android:id="@+id/btnSave"
              android:layout_width="125px"
              android:layout_height="wrap_content"
              android:text="Save"
              android:layout_below="@+id/txtComments"
              android:layout_alignRight="@+id/txtComments"
        />
        <Button
              android:id="@+id/btnCancel"
              android:layout_width="124px"
              android:layout_height="wrap_content"
              android:text="Cancel"
              android:layout_below="@+id/txtComments"
              android:layout_alignLeft="@+id/txtComments"
        />
    </RelativeLayout>
```

Chú ý rằng mỗi View được nhúng bên trong **RelateLayout** có các thuộc tính cho phép xác định mối quan hệ giữa chúng với RelativeLayout chứa nó hoặc các View, ViewGroup khác. Một số thuộc tính như là:

- **layout_alignParentTop:** Thuộc tính này có hai giá trị true/false. Nếu là true thì đối tượng đồ hoạ này sẽ nằm sát trên cùng của RelativeLayout chứa nó.
- **layout_alignParentBottom:** Thuộc tính này có hai giá trị true/false. Nếu là true thì đối tượng đồ hoạ này sẽ nằm sát dưới cùng của RelativeLayout chứa nó.
- **layout_alignParentLeft:**Thuộc tính này có hai giá trị true/false. Nếu là true thì đối tượng đồ hoạ này sẽ nằm sát bên trái của RelativeLayout chứa nó.
- **layout_alignParentRight:** Thuộc tính này có hai giá trị true/false. Nếu là true thì đối tượng đồ hoạ này sẽ nằm sát bên phải của RelativeLayout chứa nó.
- **layout_alignLeft:** Thuộc tính này xác định id của đối tượng View hoặc ViewGroup mà nó sẽ nằm ở bên trái của đối tượng đó.

- **layout_alignRight:** Thuộc tính này xác định id của đối tượng View hoặc ViewGroup mà nó sẽ nằm ở bên phải của đối tượng đó.
- **layout_below:** Thuộc tính này xác định id của đối tượng View hoặc ViewGroup mà nó sẽ nằm ở bên dưới của đối tượng đó.
- **layout_above:** Thuộc tính này xác định id của đối tượng View hoặc ViewGroup mà nó sẽ nằm ở bên trên đối tượng đó.
- **layout_centerInParent:** Thuộc tính này có hai giá trị true/false. Nếu là true thì đối tượng đồ hoạ này sẽ nằm ở chính giữa của RelativeLayout chứa nó.
- **layout_centerHorizontal:**Thuộc tính này có hai giá trị true/false. Nếu là true thì đối tượng đồ hoạ này sẽ canh giữa theo chiều ngang so với RelativeLayout chứa nó.
- **layout_centerVertical:** Thuộc tính này có hai giá trị true/false. Nếu là true thì đối tượng đồ hoạ này sẽ canh giữa theo chiều dọc so với RelativeLayout chứa nó.

Khi nạp phần code xml trên lên Activity thì ta sẽ được giao diện như sau:



1.5 FrameLayout

FrameLayout là ViewGroup đặc biệt có thể sử dụng để hiển thị một View đơn. Tất cả các đối tượng View được đặt trong FrameLayout sẽ luôn ở trên cùng bên trái của FrameLayout.

Xem xét đoạn code xml sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<AbsoluteLayout
      android:id="@+id/widget68"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      xmlns:android="http://schemas.android.com/apk/res/android">
      <FrameLayout
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_x="40px"
      android:layout_y="35px">
      <ImageView
      android:src="@drawable/android_logo"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
/>
      </FrameLayout>
</AbsoluteLayout>
```

Ở đây ta có một FrameLayout bên trong một AbsoluteLayout. Bên trong FrameLayout, bạn nhúng một ImageView. Sau khi nạp phần code xml trên lên Activity bạn sẽ được giao diện như sau:

Nếu ta thêm một View khác (Một Button) bên trong FrameLayout, View mới thêm vô sẽ nằm đè lên View trước đó:

```
<?xmlversion="1.0"encoding="utf-8"?>
<AbsoluteLayout
android:id="@+id/widget68"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<FrameLayout
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_x="40px"
android:layout_y="35px"
>
<ImageView
android:src="@drawable/android_logo"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

```
/>
<Button
android:layout_width="124px"
android:layout_height="wrap_content"
android:text="Print Picture"
/>
</FrameLayout>
</AbsoluteLayout>
```

Sau khi nạp lên Activity ta sẽ được giao diện như sau:

Bạn có thể thêm nhiều View vào FrameLayout, nhưng mỗi view đó sẽ chồng lên đỉnh của View trước đó và nằm sát góc trên cùng bên trái của FrameLayout chứa nó.

1.6 ScrollView

Một **ScrollView** là một trường hợp đặc biệt của **FrameLayout**, nó cho phép người sử dụng cuộn qua một danh sách các đối tượng View hoặc ViewGroup chiếm giữ không gian hiển thị lớn hơn so với màn hình điện thoại hỗ trợ. **ScrollView** chỉ có thể chứa duy nhất một đối tượng View hoặc ViewGroup (Thường là **LinearLayout**).

Chú ý: Không sử dụng **ListView** cùng với **ScrollView**. **ListView** được thiết kế để hiển thị một danh sách các đối tượng có liên hệ với nhau (VD: Danh sách contact) và được tối ưu hoá để phù hợp với những danh sách lớn.

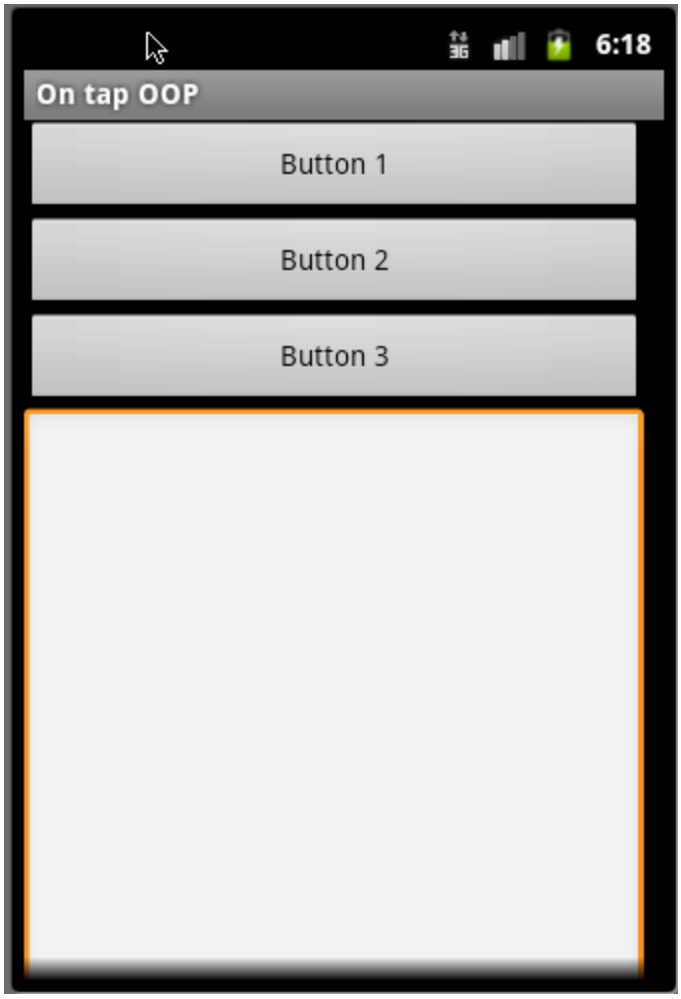
Ví dụ sau minh họa **ScrollView** chứa một **LinearLayout**:

Thực thi đoạn code trên bạn sẽ được kết quả như sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<ScrollView
android:id="@+id/widget54"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<LinearLayout
android:layout_width="310px"
android:layout_height="wrap_content"
android:orientation="vertical"
>
<Button
android:id="@+id/button1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Button 1"
/>
<Button
android:id="@+id/button2"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Button 2"
/>
<Button
android:id="@+id/button3"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Button 3"
/>
<EditText
android:id="@+id/txt"
android:layout_width="fill_parent"
android:layout_height="300px"
/>
<Button
android:id="@+id/button4"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
```

```
android:text="Button 4"
/>
<Button
android:id="@+id/button5"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Button 5"
/>
</LinearLayout>
</ScrollView>
```

Sau khi nạp phần code xml trên lên Activity ta sẽ được giao diện như sau:



Giao diện trên cho phép ta dùng ngón tay vuốt lên màn hình theo chiều từ dưới lên.

2 View

Đơn vị căn bản của giao diện đồ hoạ trên Android chính là View. Một View đại diện cho một widget, trong bài học này bạn sẽ được học về sự khác nhau giữa các View mà bạn có thể sử dụng nó trong quá trình phát triển ứng dụng android.

Đây là một danh mục các loại View mà chúng ta sẽ đề cập trong bài này:

- Basic View: Đây là những đối tượng View thường xuyên sử dụng như TextView, EditText, Button.
- Picker View: Đây là những đối tượng View cho phép người dùng ứng dụng lựa chọn một option, ngày tháng, thơi gian này đó. Loại View này thường hiển thị dưới dạng một danh sách các lựa chọn như: contact, date time…
- Display View: Đây là loại đối tượng View dùng để hiển thị hình ảnh như: Gallery và ImageSwitcher.
- Menu: Đây là loại đối tượng View dùng để tạo giao diện menu lựa chọn
- Additional View: những đối tượng View như AnalogClock và DigitalClock.

Tạo một project mới với các thông số như sau:

Project name: UIExample

Build target: Android 2.3.3.

Application name: UI Example

Package name: android.uiexample

Create Activity: main

2.1 Basic Views

Trong phần này, ta sẽ xem xét các View cơ bản trong android mà cho phép hiển thị một đoạn văn bản cũng như thực hiện một số các lựa chọn cơ bản. Ta sẽ tiến hành tìm hiểu về các View sau đây:

- TextView
- EditText
- Button
- ImageButton
- CheckBox

- ToggleButton
- RadioButton
- RadioGroup
- TextView

Khi bạn tạo một project mới trong android, eclipse sẽ tạo một file main.xml trong thư mục layout (res/layout) trong đó nó chứa một TextView như sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/a
ndroid"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/hello"
/>
</LinearLayout>
```

TextView được sử dụng để hiển thị một đoạn văn bản. Đây là một View cơ bản thường được sử dụng trong phát triển ứng dụng Android. Nếu cần cho phép người sử dụng chỉnh sửa văn bản đang hiển thị thì có thể sử dụng một subclass của TextView là EditText. Đây là một loại View mà chúng ta sẽ xem xét trong phần sau. Bên cạnh TextView, còn có một số View cơ bản thường được sử dụng trong cách ứng dụng Android như: Button, ImageButton, EditText, CheckBox, ToggleButton, RadioButton và RadioButton.

Đầu tiên chúng ta thêm một file mới trong thư mục res/layout và đặt cho nó một cái tên là: basic_views.xml và thêm vào nội dung như sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/a
ndroid"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<Buttonandroid:id="@+id/btnSave"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Save"
/>
```

```
<Buttonandroid:id="@+id/btnOpen"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Open"
/>
<ImageButtonandroid:id="@+id/btnImg1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:src="@drawable/icon"
/>
<EditTextandroid:id="@+id/txtName"
     android:text="EditText"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
/>
<CheckBoxandroid:id="@+id/chkAutosave"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Autosave"
/>
<CheckBoxandroid:id="@+id/star"
style="?android:attr/starStyle"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>

<RadioGroupandroid:id="@+id/rdbGp1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:orientation="vertical"
>
<RadioButtonandroid:id="@+id/rdb1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Option 1"
/>
<RadioButtonandroid:id="@+id/rdb2"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Option 2"
/>
</RadioGroup>

<ToggleButtonandroid:id="@+id/toggle1"
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
/>

</LinearLayout>
```

Chú ý rằng bạn sử dụng thuộc tính id để định nghĩa cho mỗi View. Id của View phải có định dạng "@+id/tenView".

Trên đây chúng ta đã sử dụng các View sau đây:

Button: Nút nhấn.

ImageButton:Nút nhấn có hỗ trợ hiển thị hình ảnh trên nút nhấn.

EditText: Đây là một class con của TextView, bổ sung thêm khả năng cho phép chỉnh sửa đoạn text đang hiển thị trên nó.

CheckBox: Một loại nút nhấn đặc biệt có hai trạng thái: Checked hoặc Unchecked.

RadioGroup and RadioButton:RadioButton cũng là loại nút nhấn có hai trạng thái: Checked và Unchecked. Một RadioButton khi đã check chọn thì ko thể uncheck. RadioGroup được sử dụng để nhóm một hoặc nhiều đối tượng RadioButton. Bằng cách sử gom nhóm các RadioButton vào một RadioGroup cho phép hạn chế chỉ được check chọn một RadioButton trong cùng một group.

ToggleButton: Thể hiện hai trạng thái: Check hoặc Uncheck có sử dụng đèn báo (light indicator).

Để xử lý các sự kiện thông thường của các View kể trên, chúng ta tạo thêm một class trong thư mục src với tên: BasicViewsExampleActivity.java. Sau đó thay đổi nội dung bằng bằng phần mã sau:

```
package android.uiexample;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.RadioGroup;
import android.widget.Toast;
import android.widget.ToggleButton;
import android.widget.RadioGroup.OnCheckedChangeListener;

publicclass BasicViewsExampleActivityextends Activity
{
@Override
publicvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.basic_views);

//---Button view---
        Button btnOpen = (Button) findViewById(R.id.btnOpen);
        btnOpen.setOnClickListener(new View.OnClickListener() {
publicvoid onClick(View v) {
                displayToast("You have clicked the Open button");
            }
        });

        Button btnSave = (Button) findViewById(R.id.btnSave);
        btnSave.setOnClickListener(new View.OnClickListener()
        {
publicvoid onClick(View v) {
displayToast ("You have clicked the Save button");
            }
        });

//---CheckBox---
        CheckBox checkBox = (CheckBox)
findViewById(R.id.chkAutosave);
        checkBox.setOnClickListener(new View.OnClickListener()
        {
publicvoid onClick(View v) {
if (((CheckBox)v).isChecked())
displayToast ("CheckBox is checked");
else
displayToast ("CheckBox is unchecked");
            }
        });

//---RadioButton---
        RadioGroup radioGroup = (RadioGroup)
findViewById(R.id.rdbGp1);
        radioGroup.setOnCheckedChangeListener(new
OnCheckedChangeListener()
        {
publicvoid onCheckedChanged(RadioGroup group, int checkedId) {
//---displays the ID of the RadioButton that is checked---
                displayToast(Integer.toString(checkedId));
            }
        });

//---ToggleButton---
```

```
        ToggleButton toggleButton = (ToggleButton)
findViewById(R.id.toggle1);
        toggleButton.setOnClickListener(new
View.OnClickListener()
        {
publicvoid onClick(View v) {
if (((ToggleButton)v).isChecked())
displayToast ("Toggle button is On");
else
     displayToast ("Toggle button is Off");
            }
        });
    }

privatevoid displayToast(String msg)
    {
         Toast.makeText(getBaseContext(), msg,
                Toast.LENGTH_SHORT).show();
    }
}
```

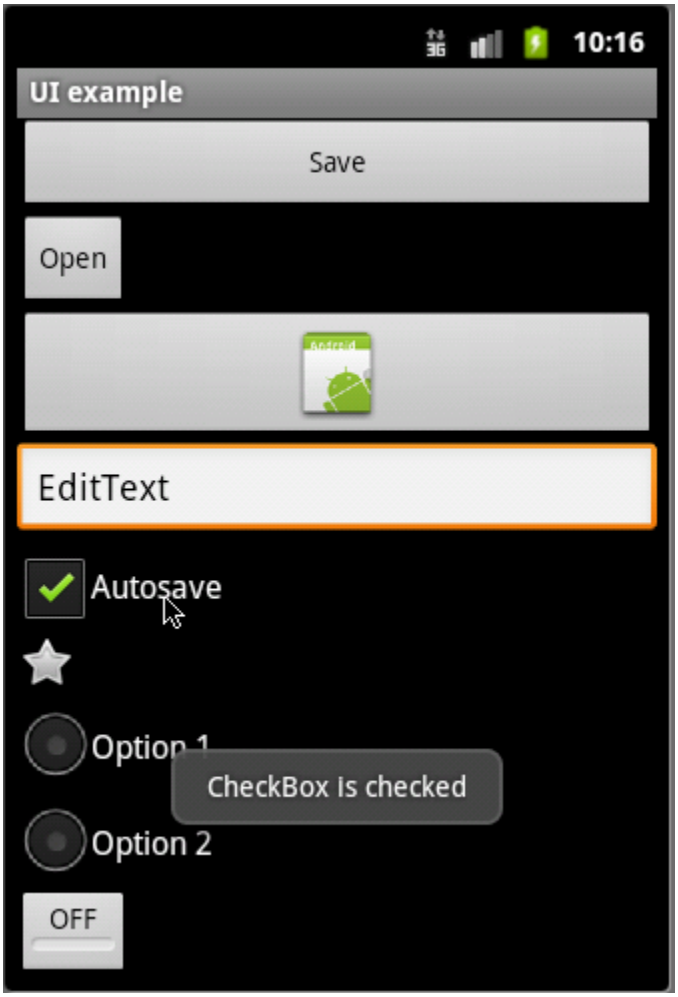
Sau đó thay đổi file AndroidMainifest.xml để đăng kí một Activity mới tên là BasicViewsExampleActivity như sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<manifest
     xmlns:android="http://schemas.android.com/apk/res/android"
package="android.uiexample"
android:versionCode="1"
android:versionName="1.0">

<application
          android:icon="@drawable/icon"
          android:label="@string/app_name">
     <activityandroid:name=".main"
android:label="@string/app_name">
     <intent-filter>
     <actionandroid:name="android.intent.action.MAIN"/>
     <categoryandroid:name="android.intent.category.LAUNCHER"/>
     </intent-filter>
     </activity>
          <activity
               android:name=".BasicViewsExampleActivity"
          android:label="@string/app_name"/>
</application>
</manifest>
```

Thêm các dòng mã sau vào file main.java để hiển thị activity BasicViewsExampleActivity:

```
package android.uiexample;

import android.app.Activity;
importandroid.content.Intent;
import android.os.Bundle;

publicclass main extends Activity {
/** Called when the activity is first created. */
@Override
publicvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

Intent    intent    =    new  Intent(this,
BasicViewsExampleActivity.class);
        startActivity(intent);

    }
}
```

Thực thi chương trình và được kết quả như sau:



Ta sẽ xem xét lại phần code hiện thực BasicViewsExampleActivity.java để xem cách các loại đối tượng View hoạt động.

Trước tiên BasicViewsExampleActivity là một class extends class Activity của Android. BasicViewsExampleActivity sẽ tạo thành một cửa sổ Activity khi nó được nạp lên bởi chương trình. Ta sẽ xem xét phương thức **public void onCreate(Bundle savedInstanceState)**để làm rõ cách hoạt động của Button, ImageButton, EditText, CheckBox, ToggleButton, RadioButton và RadioButton.

Phương thức **public void onCreate(Bundle savedInstanceState)** như ta đã biết là phương thức đầu tiên sẽ chạy khi một Activity được nạp lên màn hình điện thoại. Trên phương thức này ta sẽ gọi phương thức **setContentView(R.layout.basic_views);** để thông báo cho Activity hiện tại nạp tất cả các đối tượng đồ hoạ được khai báo trong file basic_views.xml (Được đặt trong thư mục res/layout/basic_views.xml). Sau khi nạp tất cả các đối tượng đồ hoạ từ file xml lên giao diện của Activity ta tiến hành tạo các liên kết đến các đối tượng đồ hoạ mà ta cần thay đổi thuộc tính hoặc tác động các sự kiện lên nó. Để tạo liên kết đến đối tượng Button có id là btnOpen đã được khai báo trong file xml ta sử dụng đoạn mã lệnh sau đây:

```
Button btnOpen = (Button) findViewById(R.id.btnOpen);
btnOpen.setOnClickListener(new View.OnClickListener() {
     publicvoid onClick(View v) {
          displayToast("You have clicked the Open button");
}
});
```

Đầu tiên ta sử dụng phương thức **findViewById(R.id.btnOpen);** để tìm về đối tượng Button có id là btnOpen được khai báo trong file xml. Phương thức **findViewById(R.id.btnOpen);** sẽ trả về một đối tượng kiểu View nên ta cần ép kiểu đối tượng được trả về thành kiểu Button và gán cho đối tượng btnOpen kiểu Button. Từ đây ta có thể gọi các phương thức để thay đổi các thuộc tính của đối tượng Button nằm trên giao diện đồ hoạ của Activity.

Tiếp đến ta gọi phương thức **btnOpen.setOnClickListener()** để gán cho đối tượng btnOpen một đối tượng dùng để lắng nghe sự kiện người nhấn (click) lên đối tượng btnOpen. Một đối tượng muốn xử lý được sự kiện người dùng nhấn lên Button thì đối tượng đó phải cài đặt (implement) lại interface OnClickListener. Interface này chỉ có duy nhất một phương thức là **public void onClick(View v)**. Đây là phương thức sẽ được gọi khi người dùng nhấn lên đối tượng btnOpen.

Ở ví dụ này ta sẽ không xây dựng một class riêng biệt để xử lý sự kiện người dùng nhấn lên đối tượng btnOpen mà sẽ sử dụng inner class ngay trong BasicViewsExampleActivity như phần code sau:

```
btnOpen.setOnClickListener(new View.OnClickListener() {
     publicvoid onClick(View v) {
          displayToast("You have clicked the Open button");
     }
});
```

Đây là một cách khai báo tắt một inner class và ghi đè (override) lại phương thức là **public void onClick(View v)** của interface **OnClickListener**. Trên phương thức onClick này ta đơn giản gọi phương thức **displayToast()** được cài đặt trong class BasicViewsExampleActivity. Phương thức này sẽ hiển thị đoạn text "You have clicked the Open button" bằng cách sử dụng đối tượng Toast (Ta sẽ tạm không quan tâm đến đối tượng Toast trong mục này của giáo trình).

Bằng cách này khi người dùng nhấn lên đối tượng btnOpen thì hệ thống sẽ gọi phương thức onClick kể trên. Tham số View v đi kèm được truyền theo phương thức onClick chính là đối tượng mà người dùng mới tương tác (Trong trường hợp này cũng chính là đối tượng btnOpen). Ta có thể sử dụng phương thức v.getId() để lấy về id của đối tượng được gởi kèm.

Với đối tượng ImageButton cũng tương tự như Button nhưng có hỗ trợ thêm thuộc tính android:src="@drawable/icon" để gán hình nền cho ImageButton.

Lưu ý rằng tất cả các hình ảnh sử dụng trên giao diện của ứng dụng android cần phải đặt trong thư mục res/drawable và sử dụng cú pháp "@drawable/tenHinhAnh" để sử dụng hình ảnh trên giao diện ứng dụng Android (VD: Gán hình nền cho ImageButton).

Ta tiếp tục xem xét đối tượng CheckBox ở phần code sau:

```
CheckBox checkBox = (CheckBox) findViewById(R.id.chkAutosave);
checkBox.setOnClickListener(new View.OnClickListener()
{
     publicvoid onClick(View v) {
     if (((CheckBox)v).isChecked())
          displayToast ("CheckBox is checked");
     else
          displayToast ("CheckBox is unchecked");
     }
});
```

Dễ dàng nhận thấy dòng lệnh **CheckBox checkBox = (CheckBox) findViewById(R.id.chkAutosave);** sẽ tạo liên kết đối đối tượng CheckBox có id là chkAutosave . Sau đó ta cũng tiếp tục gán đối tượng lắng nghe sự kiện người dùng nhấn lên CheckBox như của ví dụ về Button ở trên. Trên hàm onClick của inner class được sử dụng để lắng nghe sự kiện người dùng nhấn lên CheckBox kể trên sẽ ép kiểu đối tượng v thành kiểu CheckBox và gọi phương thức getChecked() để kiểm tra CheckBox hiện tại đang ở trạng thái checked hay unchecked và hiển thị đoạn text tương ứng.

Cách sử dụng và thuộc tính của ToggleButton cũng tương tự như CheckBox, tuy nhiên ToggleButton có bổ sung thêm đèn báo (light indicator) để làm nổi bật ToggleButton.

Tiếp đến ta sẽ xem xét cách sử dụng RadioButton và RadioGroup. RadioButton cũng tương tự như CheckBox tuy nhiên khi gom nhóm các RadioButton và đặt vào cùng một RadioGroup thì tại một thời điểm ta chỉ có thể chọn duy nhất một RadioButton trong cùng một nhóm.

Ví dụ: Khi check chọn RadioButton có id là rdb1 thì rdb2 tự chuyển thành trạng thái unchecked. Nếu chọn lại rdb2 thì rdb1 tự động chuyển thành trạng thái unchecked.

Ta xem xét phần code xml trích từ basic_views.xml sau:

```
<RadioGroupandroid:id="@+id/rdbGp1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:orientation="vertical"
>
<RadioButtonandroid:id="@+id/rdb1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Option 1"
/>
<RadioButtonandroid:id="@+id/rdb2"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Option 2"
/>
```

```
</RadioGroup>
```

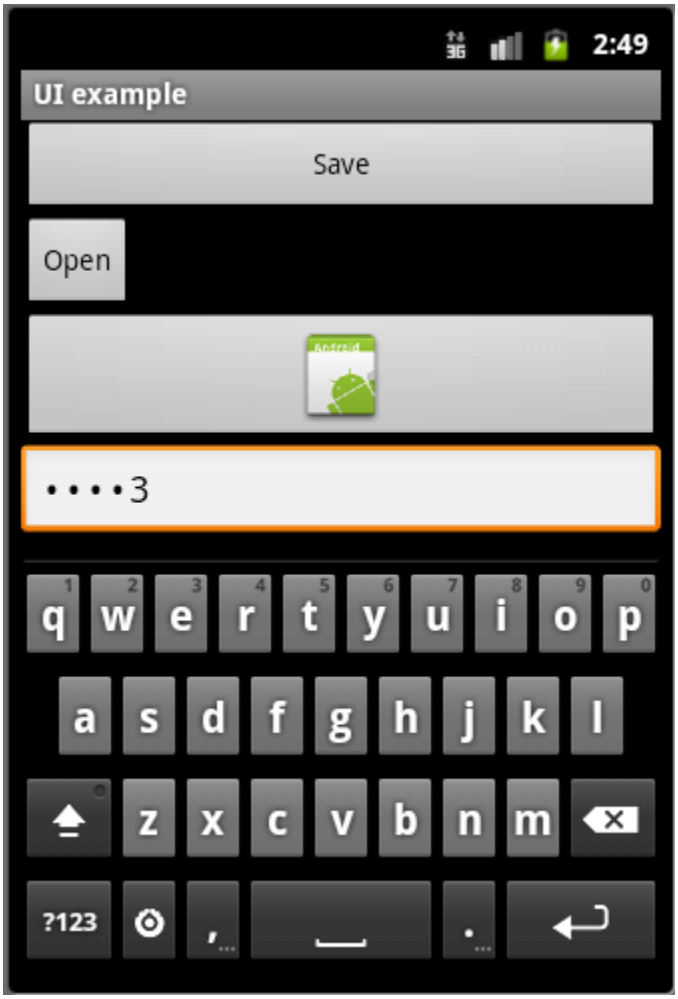
Để làm rỏ cách hoạt động của RadioGroup, ta xem xét phần mã sau đây được trích từ BasicViewsExampleActivity:

```
RadioGroup radioGroup = (RadioGroup) findViewById(R.id.rdbGp1);
radioGroup.setOnCheckedChangeListener(new
OnCheckedChangeListener(){
     publicvoid onCheckedChanged(RadioGroup group, int
checkedId) {
//---displays the ID of the RadioButton that is checked---
     displayToast(Integer.toString(checkedId));
     }
     });
```

Dòng lệnh RadioGroup radioGroup = (RadioGroup) findViewById(R.id.rdbGp1); sẽ tạo liên kết đến RadioGroup có id là rdbGp1. Sau đó gọi phương thức radioGroup.setOnCheckedChangeListener() để gán cho radioGroup đối tượng sẽ xử lý sự kiện khi mà lựa chọn của các RadioButton trong RadioGroup rdbGp1 thay đổi. Đối tượng này cũng là một khai báo inner class tắt. Trong đó cài đặt lại phương thức **public void onCheckedChanged(RadioGroup group, int checkedId)**. Đây là phương thức mà hệ thống sẽ gọi khi sự lựa chọn của các RadioButton trên group thay đổi. Phương thức **onCheckedChanged** đơn giản là gọi phương thức **displayToast** để in ra id của **RadioButton** hiện tại trong group đang ở trạng thái checked.

Đối tượng đồ hoạ tiếp theo chúng ta sẽ nghiên cứu là EditText. Đây là một sub class của TextView. Ngoài chức năng hiển thị một đoạn văn bản như TextView ta còn có thể chỉnh sửa đoạn text trên EditText. Đoạn text mà EditText hiển thị được xác định bằng thuộc tính android:text và có thể truy xuất hoặc thay đổi bằng cách gọi phương thức EditText.getText() và EditText.setText(). Ngoài ra nếu bạn mong muốn EditText hiển thị một đoạn text dưới dạng các kí tự đặc biệt nhằm che dấu nội dung của đoạn text thì ta thay đổi thuộc tính android:password=“true” như sau:

```
<EditTextandroid:id="@+id/txtName"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
     android:password="true"/>
```



2.1.1 ProgressBar View

ProcessBar là kiểu đối tượng View cung cấp một thông tin phản hồi trực quan về một tác vụ đang thực thi dưới nền.

VD: Việc download dữ liệu từ một trang web cần phải thể hiện phần trăm dữ liệu đã download được.

Sử dụng activity đã tạo như bài trước, thêm một số thành phần vào basicviews.xml:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/a
ndroid"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
```

```
<ProgressBarandroid:id="@+id/progressbar"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>

     <!--Những view cũ -->

</LinearLayout>
```

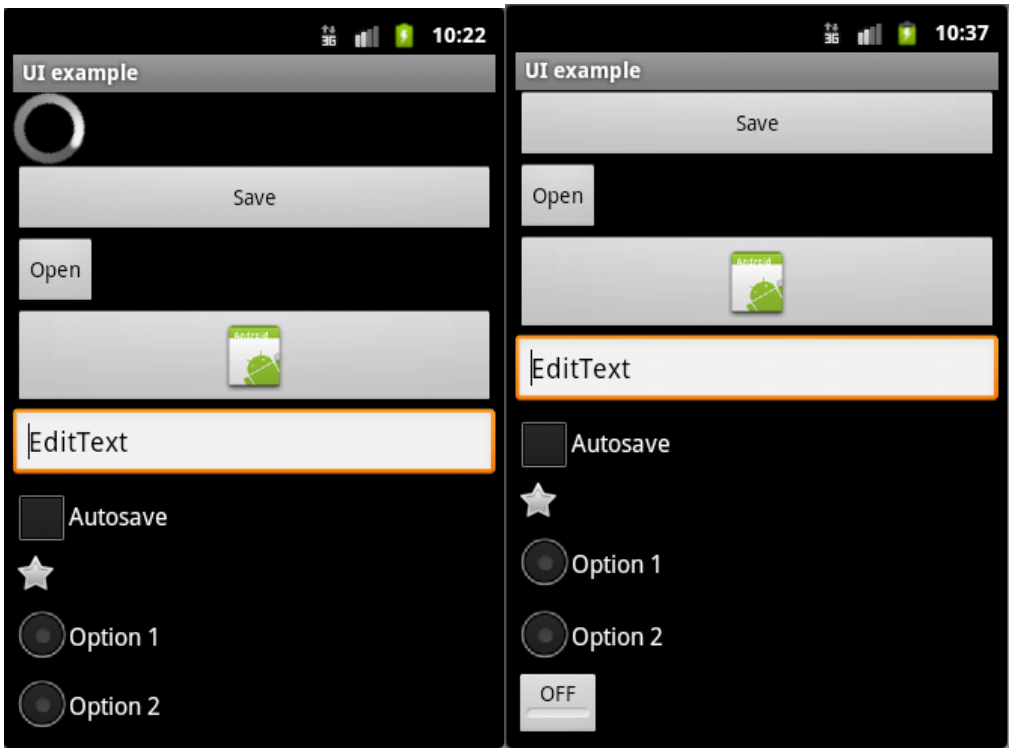
Mặc định ProcessBar sẽ để ở chế độ Indeterminate - Ở chế độ này nó sẽ thể hiện hiệu ứng xoay tròn vô tận. Chế độ này rất thích hợp cho những tác vụ không xác định rõ được khi nào nó sẽ hoàn tất.

Ví dụ: Gửi dữ liệu đến một webservice nào đó và chờ đợi sự hồi đáp của server.

Phần code dưới đây chỉ ra cách sử dụng luồng để thực thi một tác vụ và cập nhật lại trạng thái của ProcessBar sau khi tác vụ hoàn tất:

```
package android.uiexample;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.ProgressBar;
import android.widget.RadioGroup;
import android.widget.Toast;
import android.widget.ToggleButton;
import android.widget.RadioGroup.OnCheckedChangeListener;

publicclass BasicViewsExampleActivity  extends Activity
{

     Private static int progress = 0;
     private ProgressBar progressBar;
     private int progressStatus = 0;
     private Handler handler = new Handler();

     @Override
     publicvoid onCreate(Bundle savedInstanceState)
     {
          super.onCreate(savedInstanceState);
     setContentView(R.layout.basic_views);

     progressBar = (ProgressBar) findViewById(R.id.progressbar);
```

```
      //---do some work in background thread---
      Thread updateProcessBarThread = new Thread(new Runnable()
      {
      publicvoid run()
      {
      //---do some work here---
      while (progressStatus< 10)
      {
            progressStatus = doSomeWork();
      }

      //---hides the progress bar---
      handler.post(new Runnable()
      {
      publicvoid run()
      {
            progressBar.setVisibility(View.GONE);
      }
      });

      }

//---do some long lasting work here---
privateint doSomeWork()
                  {
      try {
//---simulate doing some work---
                        Thread.sleep(500);
      } catch (InterruptedException e)
      {
            e.printStackTrace();
      }
      return ++progress;
                  }

         });
      updateProcessBarThread =.start();
     }
}
```



Phần code phía trên sử dụng một luồng tên là **updateProcessBarThread**để tăng giá trị của biến int processStatus từ 1 lên 10. Tất cả công việc kể trên được thực thi trong hàm run() của Thread với nội dung sau:

```
publicvoid run()
{
      //---do some work here---
while (progressStatus< 10)
            {
                  progressStatus = doSomeWork();
            }

            //---hides the progress bar---
            handler.post(new Runnable()
            {
                  publicvoid run()
                  {
                        progressBar.setVisibility(View.GONE);
                  }
            });

}
```

Trong đó Thread sẽ thực hiện một vòng lặp while thực thi liên tục khi processStatus<10. Vòng lặp while này sẽ gọi phương thức doSomeWork():

```
privateint doSomeWork()
{
	try {
		//---simulate doing some work---
		Thread.sleep(500);
	} catch (InterruptedException e)
	{
		e.printStackTrace();
	}
	return ++progress;
}
```

Phương thức doSomeWork sẽ cho luồng **updateProcessBarThread** hiện tại tạm dừng 500ms và sau đó tăng process lên một đơn vị và trả về. Giá trị trả về sẽ gán cho biến processStatus. Khi processStatus >=10 chương trình sẽ gọi phương thức handler.post() để post một đối tượng implement interface Runable(). Interface Runable chỉ có duy nhất một phương thức run. Do**handle** là đối tượng được khởi tạo ở phần khai báo các thuộc tính của BasicViewsExampleActivity nên là một đối tượng Handle gắn với thead mà activity BasicViewsExampleActivity đang thực thi. Do vậy khi ta post một đối tượng Runable thì đối tượng này sẽ được đưa vào message queue và thực thi trên thread của BasicViewsExampleActivity. Điều này nghe chừng như làm rắc rối thêm phần code của chúng ta tuy nhiên nó là cần thiết bởi vì lệnh progressBar.setVisibility(View.GONE); không thể thực thi trên thread **updateProcessBarThread**. Tất cả các câu lệnh liên quan đến thay đổi thuộc tính của các đối tượng đồ hoạ chỉ có thể thực thi trên UIThread. Chính vì vậy ta phải gởi một đối tượng Runable đến thread của BasicViewsExampleActivity để thực thi việc thay đổi ProcessBar.

Nếu bạn không muốn sử dụng ProcessBar với chế độ Indeterminate, bạn có thể chuyển chế độ ProcessBar về chế độ hiển thị dưới dạng một thanh ngang thể hiện phần trăm của tác vụ.

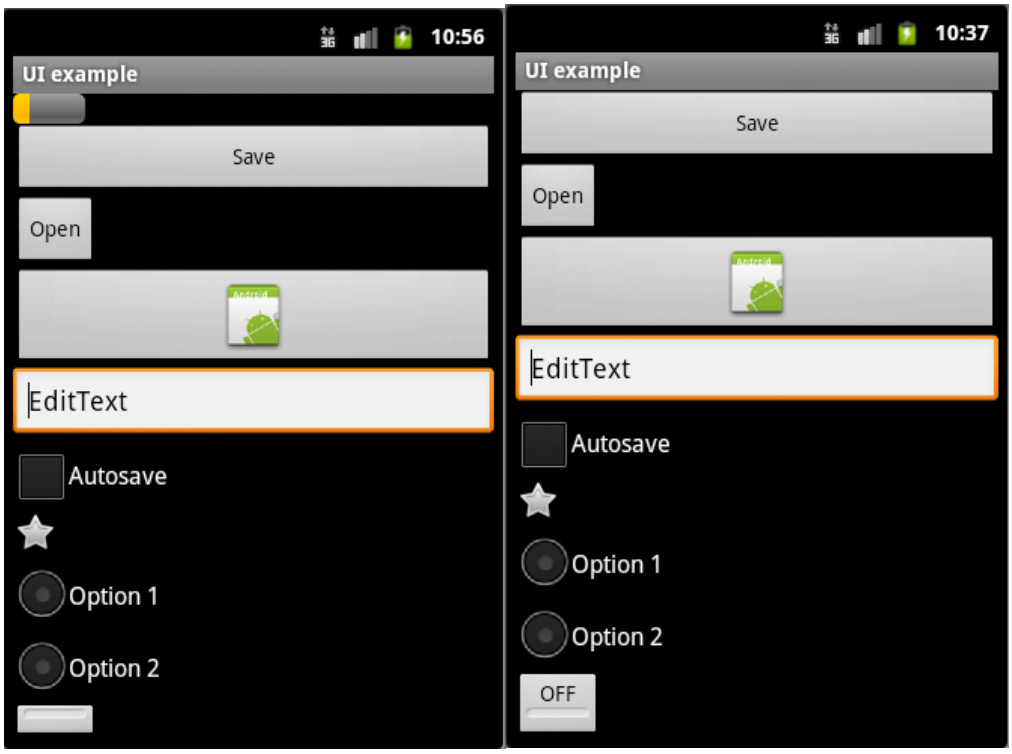
```
<ProgressBarandroid:id="@+id/progressbar"
	android:layout_width="wrap_content"
	android:layout_height="wrap_content"
	style="?android:attr/progressBarStyleHorizontal"/>
```

Phần code dưới dây sử dụng một luồng để tăng phần trăm công việc từ 1 đến 100:

```
package android.uiexample;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
```

```
import android.widget.ProgressBar;
import android.widget.RadioGroup;
import android.widget.Toast;
import android.widget.ToggleButton;
import android.widget.RadioGroup.OnCheckedChangeListener;

publicclass BasicViewsExampleActivity  extends Activity
{

Private static int progress = 0;
private ProgressBar progressBar;
private int progressStatus = 0;
private Handler handler = new Handler();

@Override
publicvoid onCreate(Bundle savedInstanceState)
    {
super.onCreate(savedInstanceState);
setContentView(R.layout.basic_views);
progressBar = (ProgressBar) findViewById(R.id.progressbar);

//---do some work in background thread---
Thread updateProcessBarThread = new Thread(new Runnable()
        {
     publicvoid run()
            {
while (progressStatus< 100)
                {
progressStatus = doSomeWork();

//---Update the progress bar---
handler.post(new Runnable()
                    {
publicvoid run() {
progressBar.setProgress(progressStatus);
                        }
                    });

                }
//---hides the progress bar---
handler.post(new Runnable()
                {
publicvoid run() {
//---0 - VISIBLE; 4 - INVISIBLE; 8 - GONE---
progressBar.setVisibility(8);
```

```
                    }
                });
            }

//---do some long lasting work here---
privateint doSomeWork()
            {
try {
//---simulate doing some work---
                    Thread.sleep(500);
                } catch (InterruptedException e)
                {
                    e.printStackTrace();
                }
return ++progress;
            }

        });
     updateProcessBarThread.start();
    }
}
```



Mỗi lần tăng một đơn vị phần trăm công việc ta lại update ProcessBar bằng phương thức progressBar.setProgress(progressStatus);. Phần trăm công việc sẽ thể hiện tương ứng trên thanh ProcessBar như trên hình.

2.1.2 AutoComplete TextView

AutoCompletelà một loại View tương tự với EditText (Trên thực tế AutoComplete là một subclass của EditText). Ngoài những chức năng tương tự như EditText nó còn có khả năng đưa ra một danh sách các gợi ý tương tự như phần text mà người dùng nhập vào. Thêm một file autocomplete.xml vào thư mục res/layout với nội dung như sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayout
     xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent">
     <AutoCompleteTextViewandroid:id="@+id/txtCountries"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"/>
</LinearLayout>
```

Thêm một file AutoCompleteExampleActivity.java với nội dung như sau:

```
package android.uiexample;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;

publicclass AutoCompleteExampleActivity extends Activity{

     String[] presidents =
    {
"Dwight D. Eisenhower",
"John F. Kennedy",
"Lyndon B. Johnson",
"Richard Nixon",
"Gerald Ford",
"Jimmy Carter",
"Ronald Reagan",
"George H. W. Bush",
"Bill Clinton",
"George W. Bush",
```

```
"Barack Obama"
    };

@Override
publicvoid onCreate(Bundle savedInstanceState)
    {
super.onCreate(savedInstanceState);
        setContentView(R.layout.autocomplete);

        ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,
            android.R.layout.simple_dropdown_item_1line,
presidents);

        AutoCompleteTextView textView = (AutoCompleteTextView)
                findViewById(R.id.txtCountries);
        textView.setThreshold(3);
        textView.setAdapter(adapter);
    }

}
```

Chú ý rằng danh sách được đưa ra được chứa trong một đối tượng ArrayAdapter. Phương thức setThreshold đặt số nhỏ nhất của các kí tự mà user phải nhập trước khi đưa ra các text trong ArrayAdapter.

Thêm những dòng sau vào file Manifest.xml:

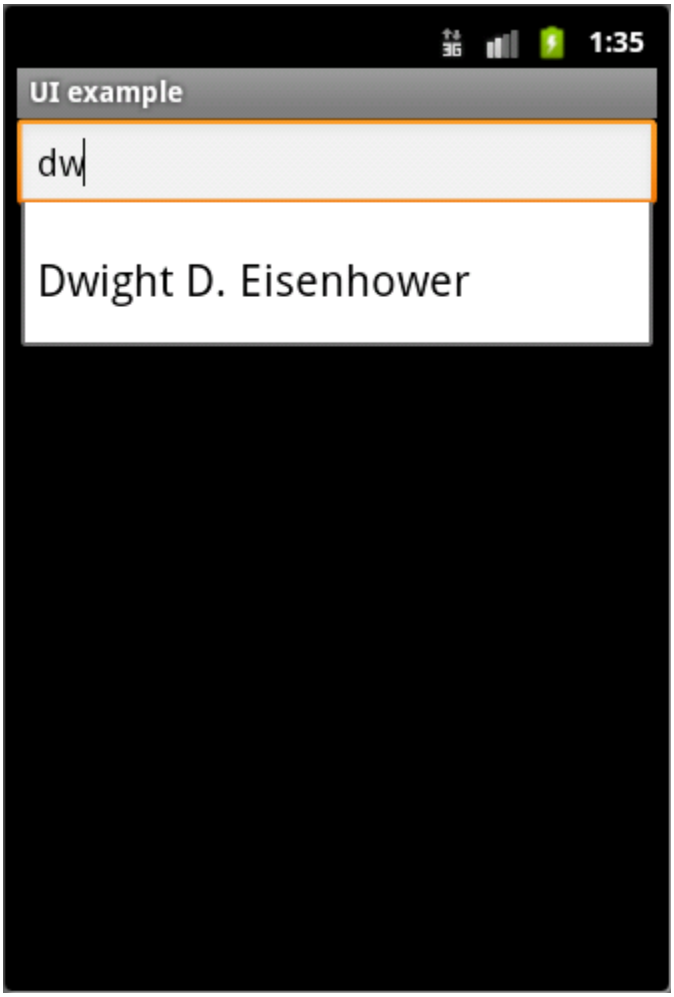
```
<?xmlversion="1.0"encoding="utf-8"?>
<manifest
     xmlns:android="http://schemas.android.com/apk/res/android"
package="android.uiexample"
android:versionCode="1"
android:versionName="1.0">

     <application
          android:icon="@drawable/icon"
          android:label="@string/app_name">
     <activityandroid:name=".main"
               android:label="@string/app_name">
     <intent-filter>
     <actionandroid:name="android.intent.action.MAIN"/>
     <categoryandroid:name="android.intent.category.LAUNCHER"/>
     </intent-filter>
     </activity>
```

```
            <activity
                  android:name=".BasicViewsExampleActivity"
            android:label="@string/app_name"/>
      <activity
                  android:name=".AutoCompleteExampleActivity"
            android:label="@string/app_name"/>
</application>
</manifest>
```

Bổ sung vào file activity chính khi khởi chạy ứng dụng như sau:

```
package android.uiexample;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

publicclass main extends Activity {
/** Called when the activity is first created. */
@Override
publicvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Intent intent    =    new  Intent(this,
AutoCompleteExampleActivity.class);
        startActivity(intent);
    }

}
```

Thực thi chương trình và nhập thử một đoạn text sẽ nhận được kết quả như sau:


1:35
UI example
dw
Dwight D. Eisenhower

2.2 PickerView

2.2.1 TimePicker

TimePicker là View cho phép người sử dụng chọn thời gian của một ngày, trong cả hai chế độ 24h hoặc chế độ AM, PM.

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/
android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<TimePicker
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
</LinearLayout>
```

Thêm một class DateTimePickerExampleActivity.java với nội dung như sau:

```
package android.uiexample;

import android.app.Activity;
import android.os.Bundle;
importandroid.widget.TimePicker;

publicclass DateTimePickerExampleActivity extends Activity{

    @Override
    protectedvoid onCreate(Bundle savedInstanceState)
    {
    super.onCreate(savedInstanceState);
        setContentView(R.layout.datetimepicker);
    }

}
```

Bổ sung trong file AndroidManifest.xml đăng ký một Activity mới :

```
<?xmlversion="1.0"encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
package="android.uiexample"
android:versionCode="1"
android:versionName="1.0">
```

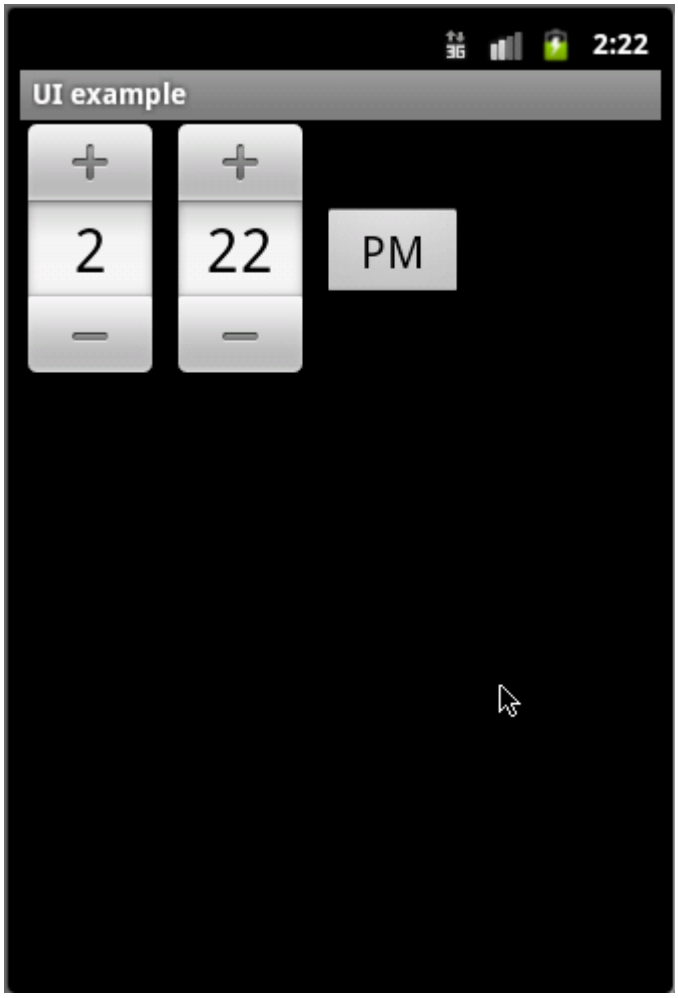
```
      <application
            android:icon="@drawable/icon"android:label="@string/a
            pp_name">
      <activity
                  android:name=".main"
android:label="@string/app_name">
      <intent-filter>
      <actionandroid:name="android.intent.action.MAIN"/>
      <categoryandroid:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
      </activity>
            <activity
                  android:name=".BasicViewsExampleActivity"
            android:label="@string/app_name"/>
      <activity
                  android:name=".AutoCompleteExampleActivity"
            android:label="@string/app_name"/>
      <activity
                  android:name=".DateTimePickerExampleActivity"
                  android:label="@string/app_name"/>
      </application>
</manifest>
```

Bổ sung vào file main.java để khởi chạy activity DateTimePickerExampleActivity như sau:

```
package android.uiexample;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

publicclass main extends Activity {
/** Called when the activity is first created. */
@Override
publicvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Intent intent    =    new  Intent(this,
DateTimePickerExampleActivity.class);
        startActivity(intent);
    }
```

```
}
```

Thực thi chương trình ta sẽ nhận được kết quả như sau:



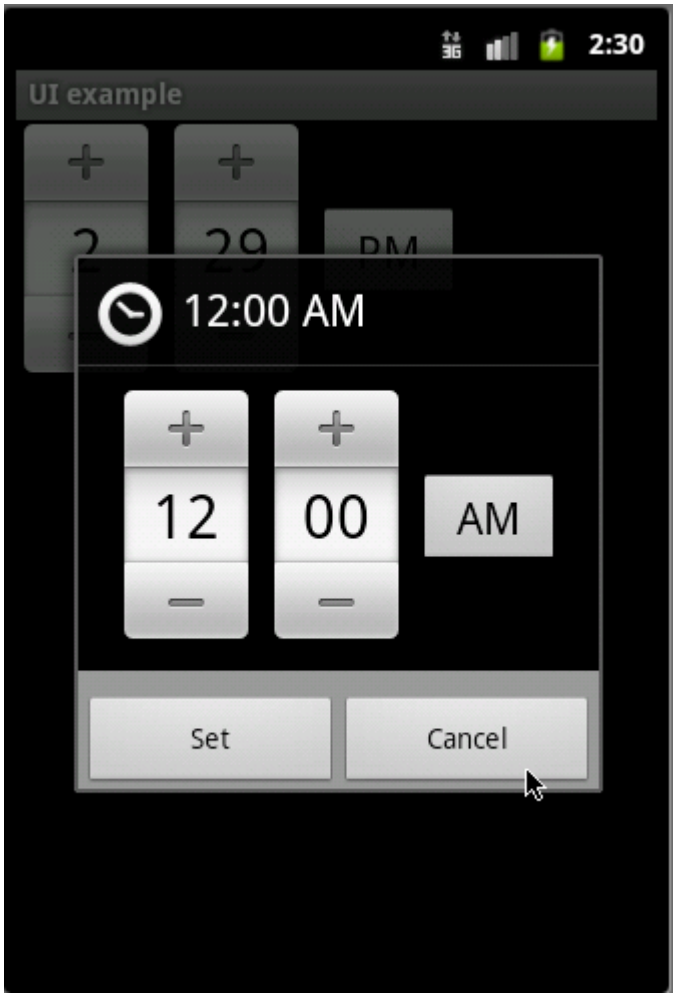
Để lấy thời gian trên TimPicker ta cần tạo liên kết đến nó và gọi getCurrentHour() và getCurrentMinute() để lấy giờ và phút trên TimePicker.

Ngoài ra ta có thể hiển thị TimePicker trên một Dialog Window. Bổ sung vào DateTimePickerExampleActivity.java đoạn mã như sau :

```
package android.uiexample;

import android.app.Activity;
import android.app.Dialog;
import android.app.TimePickerDialog;
import android.os.Bundle;
import android.widget.TimePicker;
```

```
import android.widget.Toast;

publicclass DateTimePickerExampleActivity extends Activity{

      int hour, minute;
      static final int TIME_DIALOG_ID = 0;

      @Override
      protectedvoid onCreate(Bundle savedInstanceState)
      {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.datetimepicker);
      showDialog(TIME_DIALOG_ID);
      }

      @Override
      protected Dialog onCreateDialog(int id)
      {
      switch (id) {
      caseTIME_DIALOG_ID:
      return new TimePickerDialog(this, mTimeSetListener, hour,
minute,false);
      }
      Return null;
      }

      private TimePickerDialog.OnTimeSetListener
mTimeSetListener =new TimePickerDialog.OnTimeSetListener()
     {
      publicvoid onTimeSet(TimePicker view, int hourOfDay, int
minuteOfHour)
      {
      hour = hourOfDay;
      minute = minuteOfHour;
      Toast.makeText(getBaseContext(),
"You have selected : " + hour + ":" + minute,
Toast.LENGTH_SHORT).show();
      }
      };
}
```

Khi thực thi chương trình ta sẽ được kết quả như sau:



2.2.2 DatePicker

Giống như **TimePicker** View, **DatePicker** cho phép người dùng chọn date. Thay đổi file datetimepicker.xml thành nội dung như sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/
android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>

<DatePicker
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
```

```
</LinearLayout>
```

Thay đổi nội dung của file DateTimePickerExampleActivity như sau:

```
@Override
protectedvoid onCreate(Bundle savedInstanceState)
{
      super.onCreate(savedInstanceState);
      setContentView(R.layout.datetimepicker);
      //showDialog(TIME_DIALOG_ID);
}
```

Khi thực thi chương trình ta sẽ được kết quả như sau:

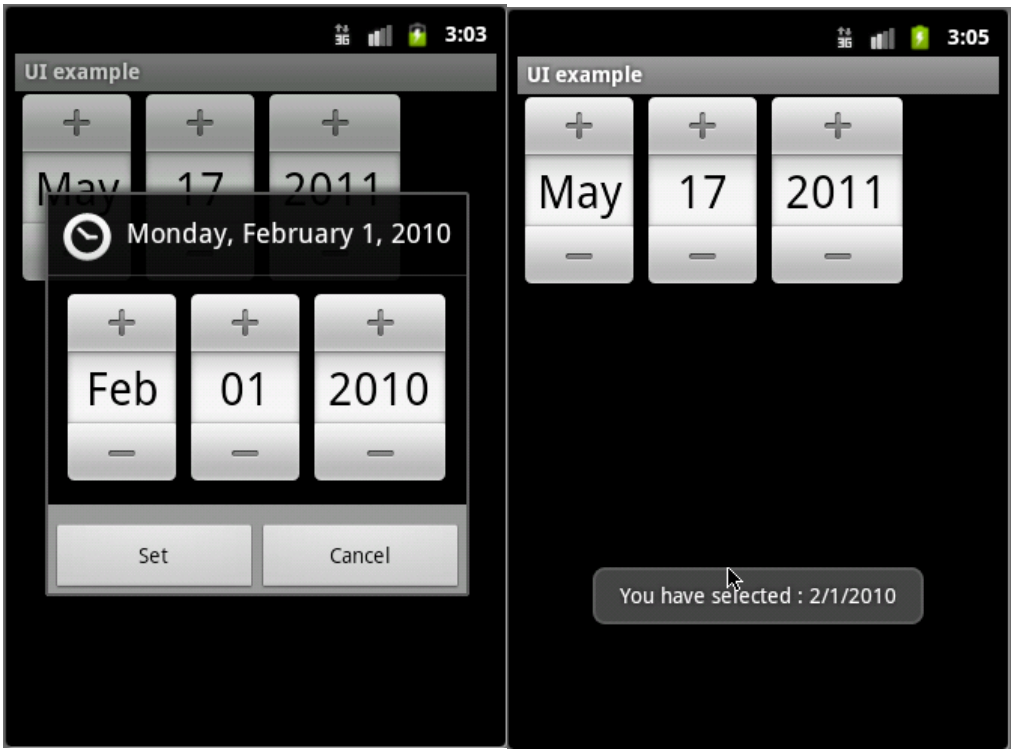
Hiển Thị DatePicker View trong một Dialog Window

Bạn cũng có thể hiển thị DatePicker View trong một Dialog . sau đó bổ sung vào file DatePickerExampleActivity.java đoạn mã như sau:

```
package android.uiexample;

import android.app.Activity;
import android.app.DatePickerDialog;
import android.app.Dialog;
import android.app.TimePickerDialog;
import android.os.Bundle;
import android.widget.DatePicker;
import android.widget.TimePicker;
import android.widget.Toast;

publicclass DateTimePickerExampleActivity extends Activity{

int hour, minute;
int Year=2010, month=1, day=1;

static final int TIME_DIALOG_ID = 0;
static final int DATE_DIALOG_ID = 1;

@Override
Protected void onCreate(Bundle savedInstanceState)
    {
super.onCreate(savedInstanceState);
        setContentView(R.layout.datetimepicker);
showDialog(DATE_DIALOG_ID);
    }

@Override
protected Dialog onCreateDialog(int id)
    {
switch (id) {
case TIME_DIALOG_ID:
return new TimePickerDialog(
this, mTimeSetListener, hour, minute, false);

case DATE_DIALOG_ID:
return new DatePickerDialog(
this, mDateSetListener, Year, month, day);
        }
Return null;
    }
```

```
private TimePickerDialog.OnTimeSetListener mTimeSetListener =
new TimePickerDialog.OnTimeSetListener()
    {

          @Override
          publicvoid onTimeSet(TimePicker view, int hourOfDay,
int minute) {
               // TODO Auto-generated method stub

          }
//...
    };

private DatePickerDialog.OnDateSetListener mDateSetListener =
new DatePickerDialog.OnDateSetListener()
    {
publicvoid onDateSet(DatePicker view, int year, int
monthOfYear,
int dayOfMonth)
        {
Year = year;
month = monthOfYear;
day = dayOfMonth;
            Toast.makeText(getBaseContext(),
"You have selected : " + (month + 1) +
"/" + day + "/" + Year,
                    Toast.LENGTH_SHORT).show();
        }
    };

}
```

Thực thi chương trình ta sẽ được kết quả như sau:



2.2.3 List Views

ListView hiển thị một danh sách các phần tử khác nhau trên một giao diện cho phép cuộn theo chiều dọc. Để tìm hiểu cách hoạt động của ListView hãy tạo thêm một file listview.xml trong thư mục res/layout như sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      >
      <ListViewandroid:id="@+id/android:list"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"/>
</LinearLayout>
```

Sau đó thêm một class mới trong src, và đặt tên cho nó là ListViewExampleActivity.javavới nội dung như sau:

```
package android.uiexample;
```

```
import android.app.Activity;
import android.app.ListActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

publicclass ListViewExampleActivity extends ListActivity{

     String[] presidents = {
"Dwight D. Eisenhower",
"John F. Kennedy",
"Lyndon B. Johnson",
"Richard Nixon",
"Gerald Ford",
"Jimmy Carter",
"Ronald Reagan",
"George H. W. Bush",
"Bill Clinton",
"George W. Bush",
"Barack Obama"
    };

     @Override
     publicvoid onCreate(Bundle savedInstanceState)
     {
          super.onCreate(savedInstanceState);
     setContentView(R.layout.listview);
     setListAdapter(new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1, presidents));
     }

     publicvoid onListItemClick(ListView parent, View v,int
position, long id)
     {
     Toast.makeText(this, "You have selected " +
presidents[position], Toast.LENGTH_SHORT).show();
     }

}
```

Chú ý rằng ListViewExampleActivity kế thừa từ ListActivity, và không quên bổ sung vào file AndroidManifest.xml để đăng kí một Activity mới:

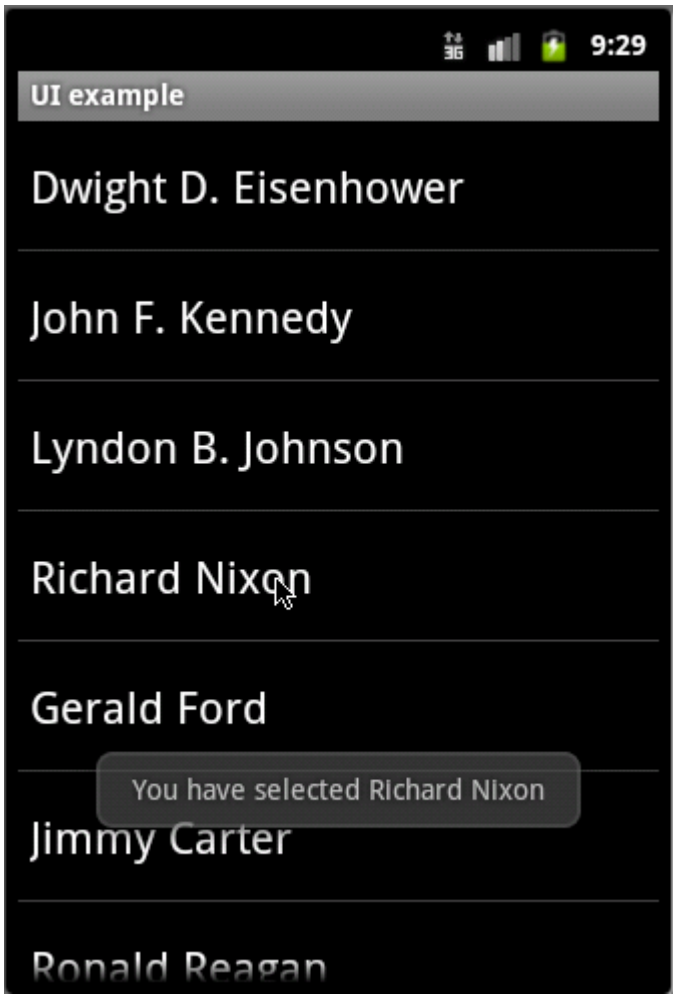
```
<?xmlversion="1.0"encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/andr
oid"
package="android.uiexample"
android:versionCode="1"
android:versionName="1.0">
     <applicationandroid:icon="@drawable/icon"android:label="@s
tring/app_name">
     <activityandroid:name=".main"
     android:label="@string/app_name">
     <intent-filter>
     <actionandroid:name="android.intent.action.MAIN"/>
     <categoryandroid:name="android.intent.category.LAUNCHER"/>
     </intent-filter>
     </activity>
          <activityandroid:name=".BasicViewsExampleActivity"
          android:label="@string/app_name"/>
     <activityandroid:name=".AutoCompleteExampleActivity"
          android:label="@string/app_name"/>
     <activityandroid:name=".DateTimePickerExampleActivity"
               android:label="@string/app_name"/>
          <activityandroid:name=".ListViewExample"
               android:label="@string/app_name"/>
</application>
</manifest>
```

Cuối cùng chúng ta chỉnh sửa file main.java để khởi chạy ListViewExampleActivity:

```
package android.uiexample;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

publicclass main extends Activity {
/** Called when the activity is first created. */
@Override
publicvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
```

```
        Intent intent    =    new  Intent(this,
ListViewExampleActivity.class);
        startActivity(intent);
    }

     }
```

Thực thi chương trình ta sẽ được kết quả như sau:



2.2.4 Spinner View

Spinner View là loại View được dùng để hiển thị một danh mục cho phép người dùng lựa chọn. Để làm rỏ cách hoạt động của Spinner ta thêm file spinner.xml trong thư mục res/layout với nội dung như sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/
android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<Spinner
android:id="@+id/spinner1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:drawSelectorOnTop="true"/>
</LinearLayout>
```

Thêm một class mới trong src và đặt tên là SpinnerExampleActivity.java với nội dung như sau :

```
package android.uiexample;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;

publicclass SpinnerExampleActivity extends Activity {

     String[] presidents = {
"Dwight D. Eisenhower",
"John F. Kennedy",
"Lyndon B. Johnson",
"Richard Nixon",
"Gerald Ford",
"Jimmy Carter",
"Ronald Reagan",
"George H. W. Bush",
"Bill Clinton",
"George W. Bush",
"Barack Obama"
    };

    Spinner s1;
```

```
      @Override
      publicvoid onCreate(Bundle savedInstanceState)
      {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.spinner);

      s1 = (Spinner) findViewById(R.id.spinner1);

      ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,
android.R.layout.simple_spinner_item, presidents);

            s1.setAdapter(adapter);
      s1.setOnItemSelectedListener(newOnItemSelectedListener(){
            @Override
      publicvoid onItemSelected(AdapterView<?> arg0, View arg1,
int arg2, long arg3)
      {
      int index = s1.getSelectedItemPosition();
      Toast.makeText(getBaseContext(),"You have selected item :
" + presidents[index], Toast.LENGTH_SHORT).show();
      }

      publicvoid onNothingSelected(AdapterView<?> arg0)
                  {

                  }
      });
      }

}
```

Bổ sung file AndroidManifest.xml để đăng kí Activity mới:

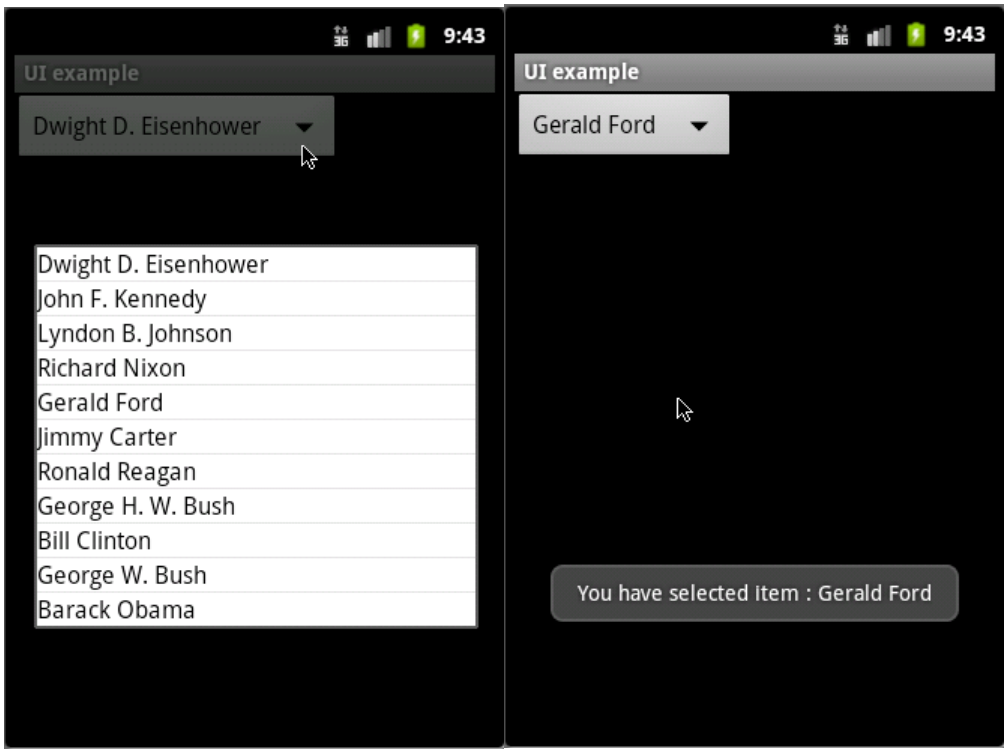
```
<?xmlversion="1.0"encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/andr
oid"
package="android.uiexample"
android:versionCode="1"
android:versionName="1.0">
<applicationandroid:icon="@drawable/icon"android:label="@string
/app_name">
<activityandroid:name=".main"
android:label="@string/app_name">
<intent-filter>
```

```
<actionandroid:name="android.intent.action.MAIN"/>
<categoryandroid:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
          <activityandroid:name=".BasicViewsExampleActivity"
     android:label="@string/app_name"/>
<activityandroid:name=".AutoCompleteExampleActivity"
     android:label="@string/app_name"/>
<activityandroid:name=".DateTimePickerExampleActivity"
               android:label="@string/app_name"/>
          <activityandroid:name=".ListViewExampleActivity"
               android:label="@string/app_name"/>
          <activityandroid:name=".SpinnerExampleActivity"
               android:label="@string/app_name"/>

</application>
</manifest>
```

Chỉnh sửa file main.java như sau:

```
package android.uiexample;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

publicclass main extends Activity {
/** Called when the activity is first created. */
@Override
publicvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Intent intent    =    new  Intent(this,
SpinnerExampleActivity.class);
        startActivity(intent);
    }

}
```

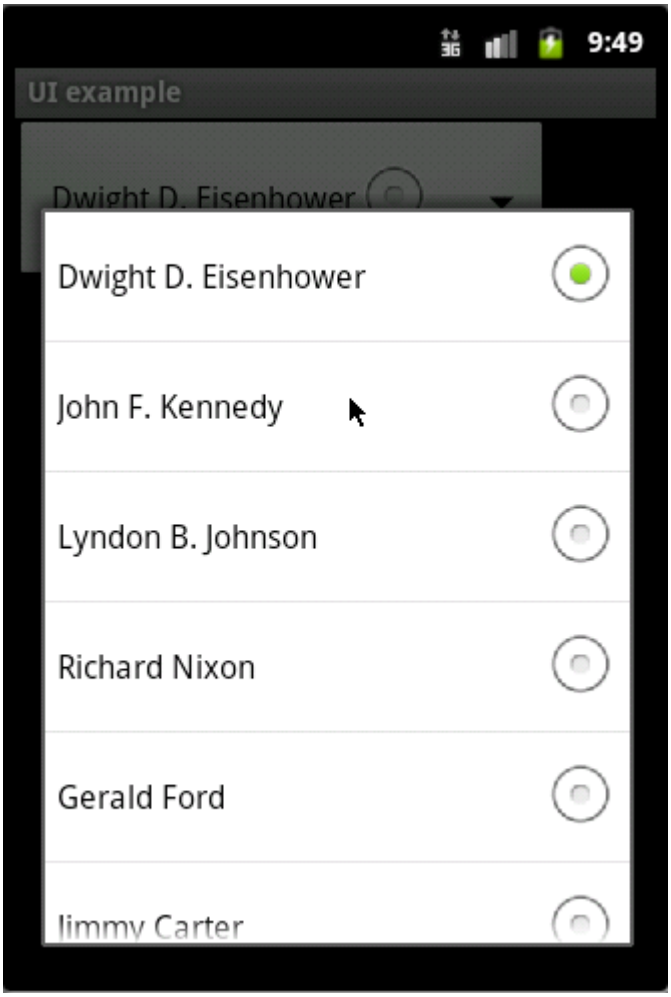
Thực thi chương trình và được kết quả như sau:



Thay vì hiển thị các mục dưới dạng một danh sách đơn như hình trên thì bạn còn có thể hiển thị chúng bằng cách sử dụng RadioButton. Để làm việc này chỉ cần chỉnh sửa lại tham số thứ 2 trong hàm dựng ArrayAdapter:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_spinner_dropdown_item, presidents);
```

Thực thi chương trình và nhận kết quả như sau:



2.3 Display Views

2.3.1 Gallery and ImageView

Gallery là một loại View hiển thị các mục dưới dạng một danh sách cuộn ngang và cố định phần tử ở trung tâm. Hình dưới đây là một Gallery được sử dụng trên Android Market:

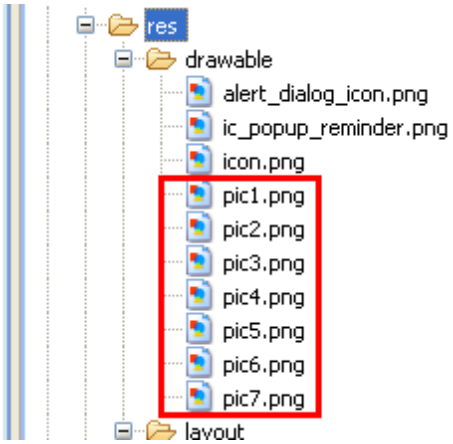
Để tìm hiểu cách hoạt động của Gallery, ta tạo thêm một file displayview.xml trong thư mục res/layout với nội dung như sau:

Thêm một file mới trong res/values đặt tên là attrs.xml với nội dung như sau :

```
<?xmlversion="1.0"encoding="utf-8"?>
<resources>
<declare-styleablename="Gallery1">
<attrname="android:galleryItemBackground"/>
</declare-styleable>
</resources>
```

Gallery1 được sử dụng để áp dụng hiển thị các images trong Gallery và mỗi image sẽ có một border bao quanh nó, hình minh họa :

Thêm một số image trong thư mục res/drawable như sau:



Thêm một classDisplayViewsExampleActivity.javatrong src/android/uiexample với nội dung như sau:

```
package android.uiexample;

import android.app.Activity;
import android.content.Context;
import android.content.res.TypedArray;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageView;
import android.widget.Toast;

publicclassDisplayViewsExampleActivityextends Activity{
     //---the images to display---
    Integer[] imageIDs = {
```

```
                R.drawable.pic1,
                R.drawable.pic2,
                R.drawable.pic3,
                R.drawable.pic4,
                R.drawable.pic5,
                R.drawable.pic6,
                R.drawable.pic7
    };

@Override
publicvoid onCreate(Bundle savedInstanceState)
    {
super.onCreate(savedInstanceState);
        setContentView(R.layout.displayview);

        Gallery gallery = (Gallery)
findViewById(R.id.gallery1);

        gallery.setAdapter(new ImageAdapter(this));
        gallery.setOnItemClickListener(new
OnItemClickListener()
        {
publicvoid onItemClick(AdapterView parent,
            View v, int position, long id)
            {
                Toast.makeText(getBaseContext(),
"pic" + (position + 1) + " selected",
                        Toast.LENGTH_SHORT).show();
            }
        });
    }

publicclass ImageAdapter extends BaseAdapter
    {
private Context context;
privateintitemBackground;

public ImageAdapter(Context c)
        {
context = c;
//---setting the style---
            TypedArray a =
obtainStyledAttributes(R.styleable.Gallery1);
itemBackground = a.getResourceId(
```

```
R.styleable.Gallery1_android_galleryItemBackground, 0);
            a.recycle();
        }

//---returns the number of images---
publicint getCount() {
returnimageIDs.length;
        }

//---returns the ID of an item---
public Object getItem(int position) {
return position;
        }

publiclong getItemId(int position) {
return position;
        }

//---returns an ImageView view---
public View getView(int position, View convertView, ViewGroup
parent) {
            ImageView imageView = new ImageView(context);
            imageView.setImageResource(imageIDs[position]);
            imageView.setScaleType(ImageView.ScaleType.FIT_XY);
            imageView.setLayoutParams(new
Gallery.LayoutParams(150, 120));
            imageView.setBackgroundResource(itemBackground);
return imageView;
        }
    }

}
```

Trong ví dụ trên ta tạo một class ImageAdapter (Class này kế thừa BaseAdapter) dùng để kết nối một loạt các đối tượng ImageView đến đối tượng Gallery. Những đối tượng ImageView này sẽ được dùng để hiện thị hình ảnh trong Gallery. ImageAdapter sẽ tạo cung cấp hai hàm căn bản sau:

- public int getCount(): Trả về số lượng ImageView có trong Gallery.
- public View getView(int position, View convertView, ViewGroup parent): Trả về đối tượng ImageView sẽ hiển thị tại một vị trí xác định.

Trong ví dụ trên có sử dụng phương thức setOnItemClickListener() để gán đối tượng lắng nghe sự kiện người dùng nhấn lên mỗi ImageView trong Gallery. Đây cũng là một khai báo inner class. Khi

người dùng nhấn lên một ImageView trong Gallery thì hệ thống sẽ tự gọi phương thức**public void onItemClick(AdapterView parent, View v, int position, long id)**

Bổ sung trong file AndroidManifest.xml với nội dung như sau:

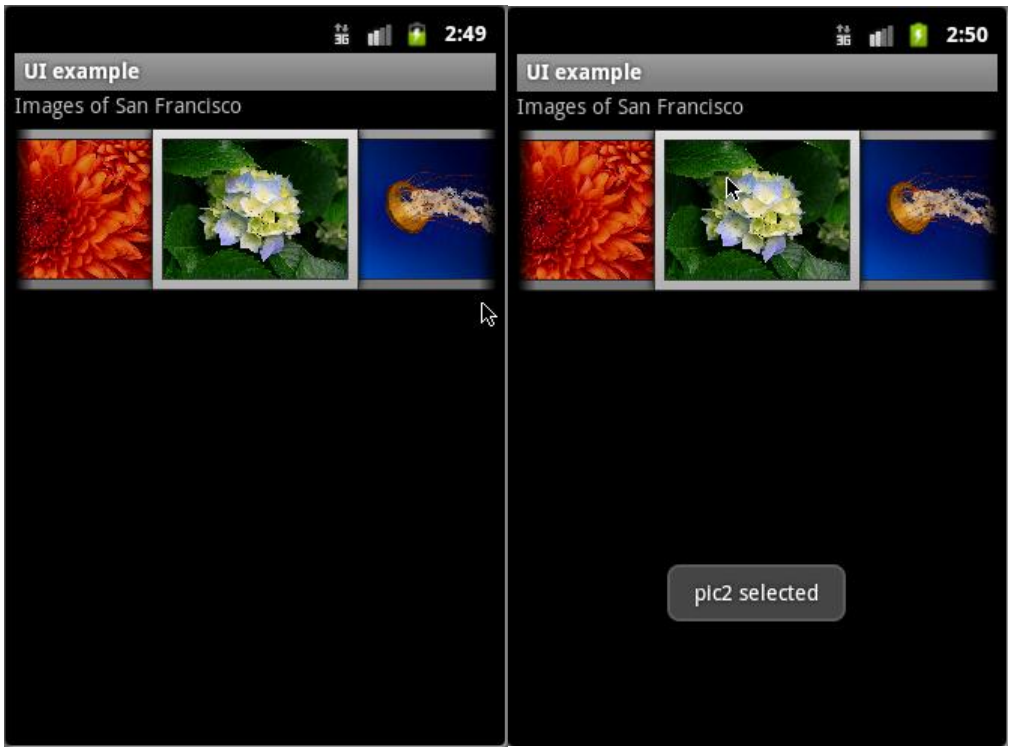
```
<?xmlversion="1.0"encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/andr
oid"
package="android.uiexample"
android:versionCode="1"
android:versionName="1.0">
     <application
          android:icon="@drawable/icon"android:label="@string/a
          pp_name">
     <activityandroid:name=".main"
               android:label="@string/app_name">
     <intent-filter>
     <actionandroid:name="android.intent.action.MAIN"/>
     <categoryandroid:name="android.intent.category.LAUNCHER"/>
     </intent-filter>
     </activity>
          <activity
               android:name=".BasicViewsExampleActivity"
          android:label="@string/app_name"/>
     <activityandroid:name=".AutoCompleteExampleActivity"
          android:label="@string/app_name"/>
     <activityandroid:name=".DateTimePickerExampleActivity"
               android:label="@string/app_name"/>
          <activityandroid:name=".ListViewExampleActivity"
               android:label="@string/app_name"/>
          <activityandroid:name=".SpinnerExampleActivity"
               android:label="@string/app_name"/>
          <activityandroid:name=".DisplayViewsExampleActivity"
android:label="@string/app_name"/>
</application>
</manifest>
```

và bổ sung trong file main.java như sau:

```
package android.uiexample;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
```

```
import android.widget.Button;

publicclass main extends Activity {
/** Called when the activity is first created. */
@Override
publicvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Intent intent    =    new  Intent(this,
DisplayViewsExampleActivity.class);
        startActivity(intent);
    }

}
```

Thực thi chương trình và nhận kết quả như sau:



Thay đổi phương thức onItemClick để hiển thị hình ảnh khi nhấn vào một item với nội dung như sau:

```
publicvoid onItemClick(AdapterView parent, View v, int
position, long id)
{
```

```
        ImageView imageView = (ImageView)
findViewById(R.id.image1);
        imageView.setImageResource(imageIDs[position]);

}
```

2.3.2 ImageSwitcher

Phần trước chúng ta đã học cách sử dụng một Gallery để hiển thị một loạt ImageView dưới dạng hình ảnh thu nhỏ. Khi người dùng nhấn lên một ImageView thì nó sẽ hiển thị lên ImageView bên dưới. Do kiểu hiển thị này rất phổ biến nên Android đã hỗ trợ sẵn một loại View tên là ImageSwitcher cho cách hiển thị hình ảnh như trên.

Tiến hành thay đổi nội dung của file displayview.xml nhưsau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/re
s/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#ff000000"
>

<ImageSwitcher
android:id="@+id/switcher1"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:layout_alignParentLeft="true"
android:layout_alignParentRight="true"
android:layout_alignParentBottom="true"
/>
<Gallery
android:id="@+id/gallery1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
/>

</RelativeLayout>
```

Thay đổi nội dung của file DisplayViewsExampleActivity.java như sau:

```
package android.uiexample;

import android.app.Activity;
```

```
import android.content.Context;
import android.content.res.TypedArray;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewGroup.LayoutParams;
import android.view.animation.AnimationUtils;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageSwitcher;
import android.widget.ImageView;
importandroid.widget.Toast;
import android.widget.ViewSwitcher.ViewFactory;

publicclass DisplayViewsExampleActivity extends Activity
implements ViewFactory{
     //---the images to display---
    Integer[] imageIDs = {
            R.drawable.pic1,
            R.drawable.pic2,
            R.drawable.pic3,
            R.drawable.pic4,
            R.drawable.pic5,
            R.drawable.pic6,
            R.drawable.pic7
    };

private ImageSwitcher imageSwitcher;

@Override
publicvoid onCreate(Bundle savedInstanceState)
    {
super.onCreate(savedInstanceState);
        setContentView(R.layout.displayview);

imageSwitcher = (ImageSwitcher) findViewById(R.id.switcher1);
imageSwitcher.setFactory(this);
imageSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
                android.R.anim.fade_in));
imageSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this
,
                android.R.anim.fade_out));
```

```
        Gallery gallery = (Gallery)
findViewById(R.id.gallery1);
        gallery.setAdapter(new ImageAdapter(this));
        gallery.setOnItemClickListener(new
OnItemClickListener()
        {
publicvoid onItemClick(AdapterView parent,
            View v, int position, long id)
            {
imageSwitcher.setImageResource(imageIDs[position]);
            }
        });
    }

public View makeView()
    {
        ImageView imageView = new ImageView(this);
        imageView.setBackgroundColor(0xFF000000);
        imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);
        imageView.setLayoutParams(new
                ImageSwitcher.LayoutParams(
                        LayoutParams.FILL_PARENT,
                        LayoutParams.FILL_PARENT));
return imageView;
    }

publicclass ImageAdapter extends BaseAdapter
    {
private Context context;
privateintitemBackground;

public ImageAdapter(Context c)
        {
context = c;

//---setting the style---
            TypedArray a =
obtainStyledAttributes(R.styleable.Gallery1);
itemBackground = a.getResourceId(

R.styleable.Gallery1_android_galleryItemBackground, 0);
            a.recycle();
        }

//---returns the number of images---
```

```
        publicint getCount()
                {
        returnimageIDs.length;
                }

        //---returns the ID of an item---
        public Object getItem(int position)
                {
        return position;
                }

        publiclong getItemId(int position)
                {
        return position;
                }

        //---returns an ImageView view---
        public View getView(int position, View convertView, ViewGroup
        parent)
                {
                    ImageView imageView = new ImageView(context);
                    imageView.setImageResource(imageIDs[position]);
                    imageView.setScaleType(ImageView.ScaleType.FIT_XY);
                    imageView.setLayoutParams(new
        Gallery.LayoutParams(150, 120));
                    imageView.setBackgroundResource(itemBackground);
        return imageView;
                }
           }

}
```

Khi thực thi chương trình ta sẽ được kết quả như sau:


4:32
UI example

2.3.3 GridView

The GridView view shows items in two-dimensional scrolling grid. You can use the GridView view together with ImageView views to display a series of images.

Thay đổi file displayview.xml với nội dung như sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<GridViewxmlns:android="http://schemas.android.com/apk/res/andr
oid"
android:id="@+id/gridview"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:numColumns="auto_fit"
android:verticalSpacing="10dp"
android:horizontalSpacing="10dp"
android:columnWidth="90dp"
android:stretchMode="columnWidth"
android:gravity="center"
/>
```

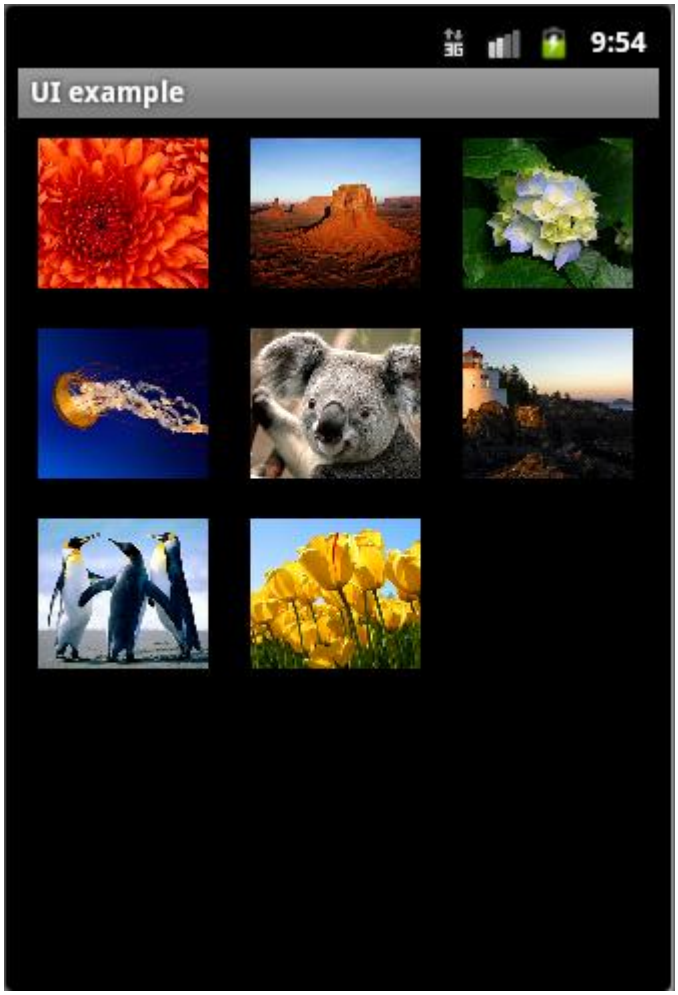
Thay đổi nội dung file DispalyViewsExampleActivity.java như sau:

```
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;
import android.widget.Toast;
```

```
publicclass DisplayViewsExampleActivity extends Activity{
//---the images to display---
    Integer[] imageIDs = {
            R.drawable.pic1,
            R.drawable.pic2,
            R.drawable.pic3,
            R.drawable.pic4,
            R.drawable.pic5,
            R.drawable.pic6,
            R.drawable.pic7,
            R.drawable.pic8
    };

@Override
publicvoid onCreate(Bundle savedInstanceState)
    {
super.onCreate(savedInstanceState);
        setContentView(R.layout.displayview);

        GridView gridView = (GridView)
findViewById(R.id.gridview);
        gridView.setAdapter(new ImageAdapter(this));

        gridView.setOnItemClickListener(new
OnItemClickListener()
        {
publicvoid onItemClick(AdapterView parent,
            View v, int position, long id)
            {
                Toast.makeText(getBaseContext(),
"pic" + (position + 1) + " selected",
                        Toast.LENGTH_SHORT).show();
            }
        });
    }

publicclass ImageAdapter extends BaseAdapter
    {
private Context context;

public ImageAdapter(Context c)
        {
context = c;
        }
```

```
//---returns the number of images---
publicint getCount() {
returnimageIDs.length;
        }

//---returns the ID of an item---
public Object getItem(int position) {
return position;
        }

publiclong getItemId(int position) {
return position;
        }

//---returns an ImageView view---
public View getView(int position, View convertView, ViewGroup
parent)
        {
ImageView imageView;
if (convertView == null) {
                imageView = new ImageView(context);
                imageView.setLayoutParams(new
GridView.LayoutParams(85, 85));

imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
                imageView.setPadding(5, 5, 5, 5);
            } else {
                imageView = (ImageView) convertView;
            }
            imageView.setImageResource(imageIDs[position]);
return imageView;
        }
    }

}
```

Thực thi chương trình ta sẽ được kết quả như sau:



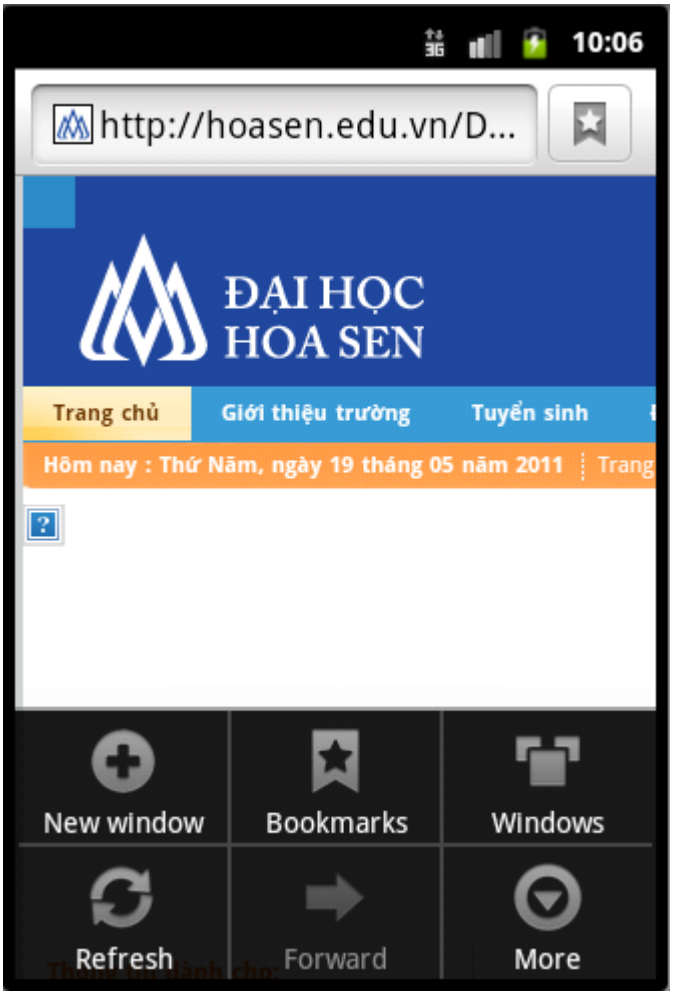
2.4 Menus

Menus là một lựa chọn rất hữu dụng để hiển thị thêm các tùy chọn mà nó không trực tiếp hiển thị trên giao diện chính của ứng dụng. Có hai loại menu trên Android:

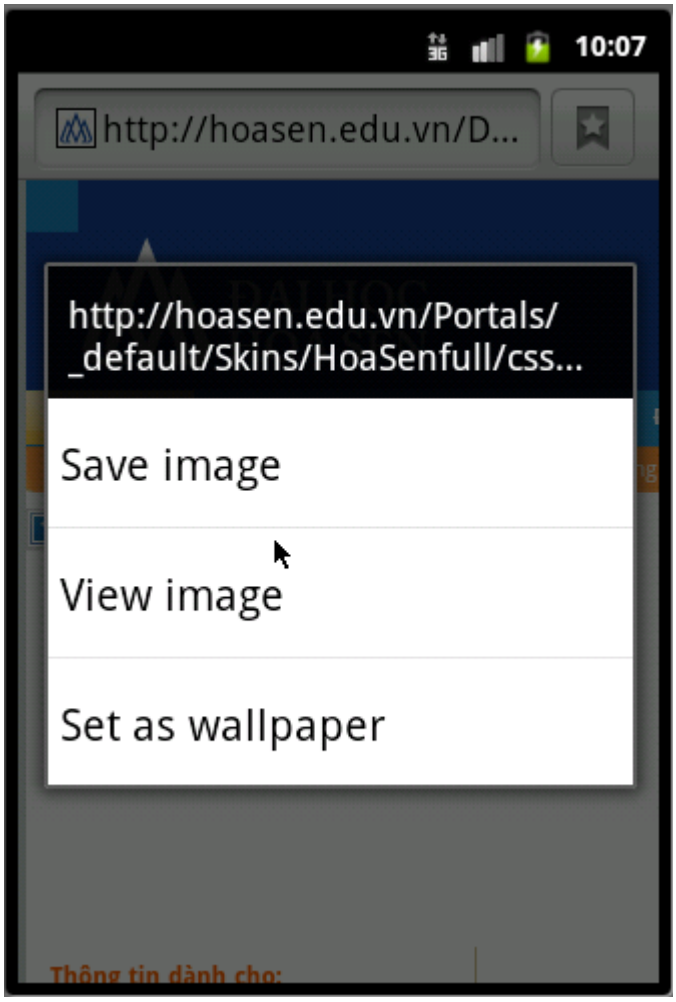
Context Menu: Hiển thị những thông tin liên quan đến một đối tượng View cụ thể. Trên Android, để kích hoạt ContextMenu người dùng cần nhấn-và-giữ một đối tượng View(Button, ImageView…).

Options Menu: Hiển thị những thông tin liên quan đến Activity hiện tại. Trong Android, để kích hoạt Options Menu người dùng cần nhấn nút Menu trên thiết bị.

Hình sau là một ví dụ của một Options Menu trên ứng dụng browser. Dạng menu này sẽ hiển thị khi người dùng nhấn phím Menu trên thiết bị:



Hình tiếp theo là một Context Menu. Để kích hoạt Context Menu, người sử dụng nhấn-và-giữ một đối tượng View trên màn hình hoặc chọn một đối tượng View và nhấn phím trung tâm (center-button) trên thiết bị.



Để thấy rỏ cách thức sử dụng menu trên Android, ta tiến hành thêm một file menu.xml trong thư mục res/layout với nội dung như sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/
android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>

<Buttonandroid:id="@+id/btn1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Hello, world!"
/>
```

```
</LinearLayout>
```

Bổ sung file AndroidManifest.xml với nội dung như sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/andr
oid"
package="android.uiexample"
android:versionCode="1"
android:versionName="1.0">
<applicationandroid:icon="@drawable/icon"android:label="@string
/app_name">
<activityandroid:name=".main"
android:label="@string/app_name">
<intent-filter>
<actionandroid:name="android.intent.action.MAIN"/>
<categoryandroid:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
          <activityandroid:name=".BasicViewsExampleActivity"
     android:label="@string/app_name"/>
<activityandroid:name=".AutoCompleteExampleActivity"
     android:label="@string/app_name"/>
<activityandroid:name=".DateTimePickerExampleActivity"
               android:label="@string/app_name"/>
          <activityandroid:name=".ListViewExampleActivity"
               android:label="@string/app_name"/>
          <activityandroid:name=".SpinnerExampleActivity"
               android:label="@string/app_name"/>
          <activityandroid:name=".DisplayViewsExampleActivity"
android:label="@string/app_name"/>
<activityandroid:name=".MenuExampleActivity"
               android:label="@string/app_name" />
</application>
</manifest>
```

Tiến hành tạo mới một file MenuExampleActivity.java và tạo 02 phương thức hỗ trợ createMenu và menuChoice với nội dung như sau:

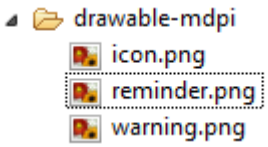
```
package android.uiexample;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
```

```
import android.view.MenuItem;
import android.view.View.OnCreateContextMenuListener;
import android.widget.Button;
import android.widget.Toast;

publicclass MenuExampleActivity extends Activity implements
OnCreateContextMenuListener{

     @Override
     publicvoid onCreate(Bundle savedInstanceState)
    {
super.onCreate(savedInstanceState);

        setContentView(R.layout.menu);
    }

privatevoidcreateMenu(Menu menu)
    {
        menu.setQwertyMode(true);
        MenuItem mnu1 = menu.add(0, 0, 0, "Item 1");
        {
            mnu1.setAlphabeticShortcut('a');
            mnu1.setIcon(R.drawable.icon);
        }
        MenuItem mnu2 = menu.add(0, 1, 1, "Item 2");
        {
            mnu2.setAlphabeticShortcut('b');
            mnu2.setIcon(R.drawable.icon);
        }
        MenuItem mnu3 = menu.add(0, 2, 2, "Item 3");
        {
            mnu3.setAlphabeticShortcut('c');
            mnu3.setIcon(R.drawable.icon);
        }
        MenuItem mnu4 = menu.add(0, 3, 3, "Item 4");
        {
            mnu4.setAlphabeticShortcut('d');
        }
        menu.add(0, 3, 3, "Item 5");
        menu.add(0, 3, 3, "Item 6");
        menu.add(0, 3, 3, "Item 7");
    }

privatebooleanmenuChoice(MenuItem item)
    {
```

```
switch (item.getItemId()) {
case 0:
            Toast.makeText(this, "You clicked on Item 1",
                Toast.LENGTH_LONG).show();
returntrue;
case 1:
            Toast.makeText(this, "You clicked on Item 2",
                Toast.LENGTH_LONG).show();
returntrue;
case 2:
            Toast.makeText(this, "You clicked on Item 3",
                Toast.LENGTH_LONG).show();
returntrue;
case 3:
            Toast.makeText(this, "You clicked on Item 4",
                Toast.LENGTH_LONG).show();
returntrue;
case 4:
            Toast.makeText(this, "You clicked on Item 5",
                Toast.LENGTH_LONG).show();
returntrue;
case 5:
            Toast.makeText(this, "You clicked on Item 6",
                Toast.LENGTH_LONG).show();
returntrue;
case 6:
            Toast.makeText(this, "You clicked on Item 7",
                Toast.LENGTH_LONG).show();
returntrue;
        }
returnfalse;
    }

}
```

Phương thức createMenu() các tác dụng tạo một danh sách các menu item. Phương thức **publicMenuItemadd(int groupId, int itemId, int order, CharSequence title)**sẽ thêm một menu item và menu hiện tại với các thông số xác định các thuộc tính của menu item được thêm vào. Phương thức **public MenuItemsetAlphabeticShortcut(char alphaChar)**sẽ gán phím tắt bàn phím để mở nhanh một menu item. Phương thức **public MenuItemsetIcon(int iconRes)**sẽ gán một hình ảnh dùng làm biểu tưởng cho menu item mới thêm vào.

Tiếp theo thêm hai file hình**reminder.png** và **warning.png**trong thư mục res/drawable như sau:



2.4.1.1 Option menu

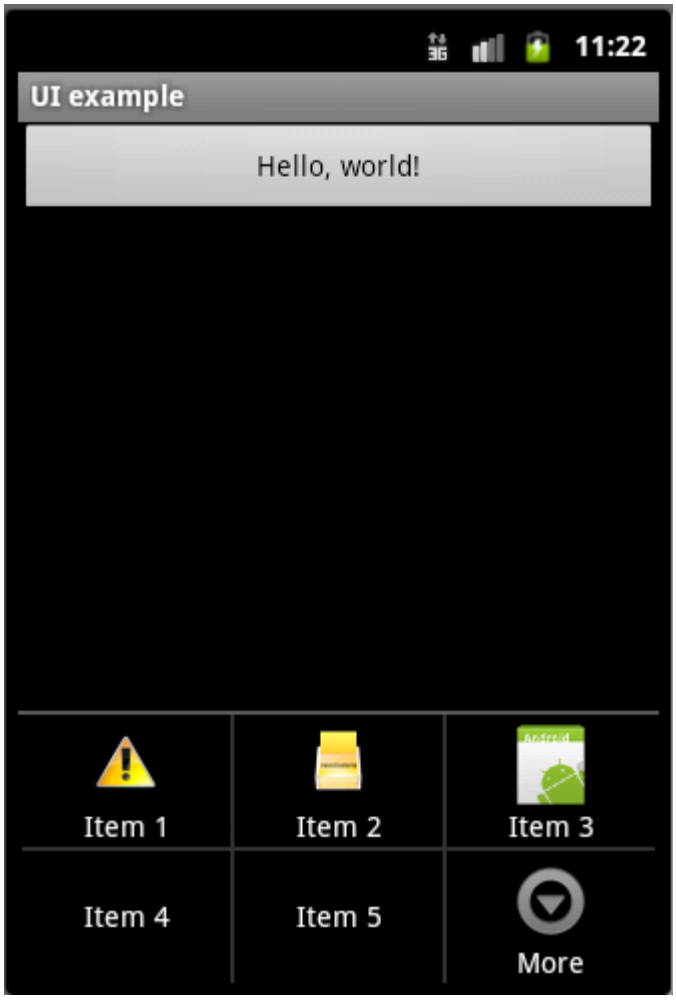
Để hiện thịOption menu trên Activity, ta cần orverride hai phương thức onCreateOptionsMenu() và onOptionsItemSelected(). Phương thức onCreateOptionsMenu() được gọi khi người dùng nhấn lên phím Menu trên thiết bị. Tại phương thức này ta sẽ gọi phương thức createMenu() ta đã khai báo ở trên để khởi tạo menu cho Option menu. Phương thức onOptionsItemSelected() sẽ được gọi khi người dùng nhấn chọn một menu item trên Option menu. Tại phương thức này ta sẽ tiến hành gọi phương thức menuChoice() ta đã định nghĩa ở trên để xử lý sự kiện người dùng nhấn chọn menu item.

Thay đổi file MenuExampleActivity.java với nội dung như sau:

```
package android.uiexample;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View.OnCreateContextMenuListener;
import android.widget.Button;
import android.widget.Toast;

publicclass MenuExampleActivity extends Activity implements
OnCreateContextMenuListener{

     @Override
     publicvoid onCreate(Bundle savedInstanceState)
    {
super.onCreate(savedInstanceState);

        setContentView(R.layout.menu);
    }

privatevoid createMenu(Menu menu)
    {
          // Xem mục trên
    }

privateboolean menuChoice(MenuItem item)
```

```
    {
// Xem mục trên
}

@Override
publicboolean onCreateOptionsMenu(Menu menu) {
super.onCreateOptionsMenu(menu);
        createMenu(menu);
returntrue;
    }

@Override
publicboolean onOptionsItemSelected(MenuItem item)
    {
return menuChoice(item);
    }

}
```

Thực thi chương trình, nhấn nút Menu trên thiết bị để nhận được kết quả như sau:



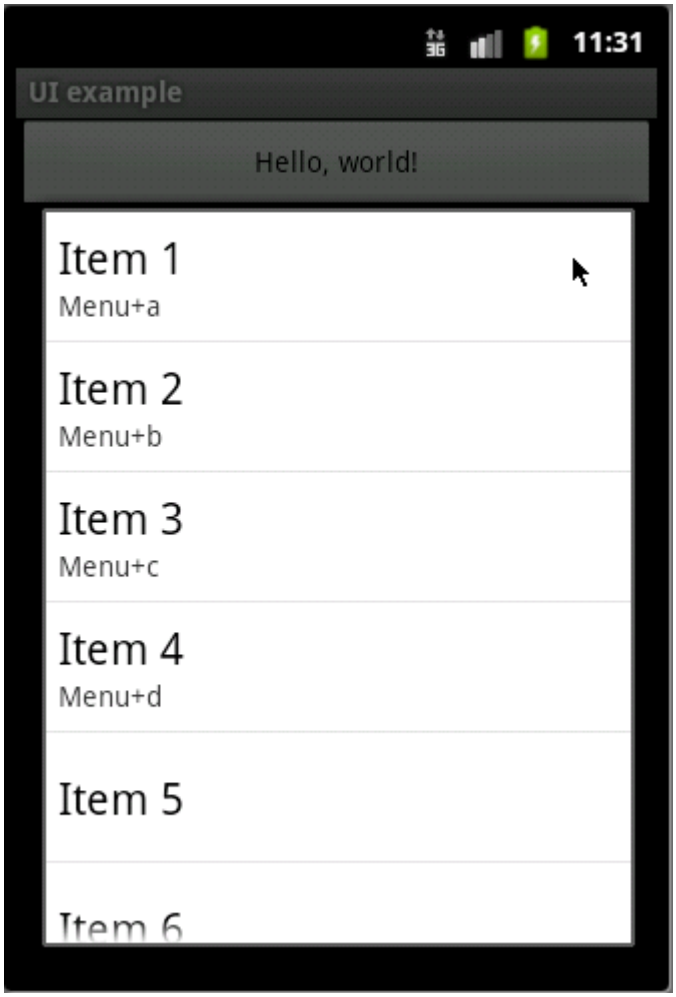
2.4.1.2 Context menu

Nếu muốn gắn một Menu lên một đối tượng View ta cần gọi phương thức setOnCreateContextMenuListener()để gán đối tượng nhận nhiệm vụ tạo ContextMenu. Thay đổi file MenuExampleActivity với nội dung như sau:

```
package android.uiexample;

import android.app.Activity;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuInfo;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnCreateContextMenuListener;
```

```
import android.widget.Button;
import android.widget.Toast;

publicclass MenuExampleActivity extends Activity implements
OnCreateContextMenuListener{

     @Override
     publicvoid onCreate(Bundle savedInstanceState)
    {
super.onCreate(savedInstanceState);

        setContentView(R.layout.menu);

        Button btn = (Button) findViewById(R.id.btn1);
        btn.setOnCreateContextMenuListener(this);
    }

privatevoidcreateMenu(Menu menu)
    {
     // Xem phần trên

    }

privateboolean menuChoice(MenuItem item)
    {
          // Xem phần trên
    }

@Override
publicvoid onCreateContextMenu(ContextMenu menu, View view,
ContextMenuInfo menuInfo)
    {
super.onCreateContextMenu(menu, view, menuInfo);
createMenu(menu);
    }

@Override
publicboolean onContextItemSelected(MenuItem item)
    {
return menuChoice(item);
    }

}
```

Thực thi chương trình để nhận được kết quả như sau:



2.5 Additional View

2.5.1 AnalogClock và DigitalClock

Thay đổi file main.xml với nội dung như sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/
android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>

<AnalogClockandroid:id="@+id/clock1"
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
/>
     <DigitalClockandroid:id="@+id/clock2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
</LinearLayout>
```

Thay đổi nội dung file main.java với nội dung như sau:

```
package android.uiexample;

import android.app.Activity;
import android.os.Bundle;

publicclass main extends Activity {
/** Called when the activity is first created. */
@Override
publicvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

}
```

Thực thi chương trình để nhận được kết quả như sau:

2.5.2 WebView

WebView là một loại View cho phép hiển thị một nội dung HTML trên giao diện của ứng dụng. Để tìm hiểu về cách hoạt động của WebView ta tiến hành thêm một file webview.xml vào thư mục res/layout với nội dung như sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/
android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>

<WebViewandroid:id="@+id/webview1"
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
android:text="Hello, world!"
/>

</LinearLayout>
```

Thêm một file WebViewExampleActivity.java trong thư mục src/android/uiexample với nội dung như sau:

```
package android.uiexample;

import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebView;

publicclass WebViewExampleActivity extends Activity{
     /** Called when the activity is first created. */
@Override
publicvoid onCreate(Bundle savedInstanceState)
    {
super.onCreate(savedInstanceState);
        setContentView(R.layout.webview);

        WebView wv = (WebView) findViewById(R.id.webview1);
        wv.loadUrl("http://www.hoasen.edu.vn");
    }
}
```

Đăng kí mới một Activity trong file AndroidManifest.xml và gán quyền truy cập Internet cho ứng dụng:

```
<?xmlversion="1.0"encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/andr
oid"
package="android.uiexample"
android:versionCode="1"
android:versionName="1.0">
<applicationandroid:icon="@drawable/icon"android:label="@string
/app_name">
<activityandroid:name=".main"
android:label="@string/app_name">
<intent-filter>
<actionandroid:name="android.intent.action.MAIN"/>
<categoryandroid:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
```

```
            <activityandroid:name=".BasicViewsExampleActivity"
        android:label="@string/app_name"/>
    <activityandroid:name=".AutoCompleteExampleActivity"
        android:label="@string/app_name"/>
    <activityandroid:name=".DateTimePickerExampleActivity"
                android:label="@string/app_name"/>
            <activityandroid:name=".ListViewExampleActivity"
                android:label="@string/app_name"/>
            <activityandroid:name=".SpinnerExampleActivity"
                android:label="@string/app_name"/>
            <activityandroid:name=".DisplayViewsExampleActivity"
    android:label="@string/app_name" />
    <activityandroid:name=".MenuExampleActivity"
                android:label="@string/app_name"/>
    <activityandroid:name=".WebViewExampleActivity"
                android:label="@string/app_name"/>
        </application>
        <uses-
permissionandroid:name="android.permission.INTERNET">
    </uses-permission>
    </manifest>
```

Thay đổi nội dung của file main.java để bật Activity WebViewExampleActivity:

```
package android.uiexample;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;

publicclass main extends Activity {
/** Called when the activity is first created. */
@Override
publicvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Intent intent    =    new  Intent(this,
WebViewExampleActivity.class);
        startActivity(intent);
    }

}
```

Thực thi chương trình để nhận được kết quả như sau:

Ngoài cách hiển thị nội dung HTML lấy từ internet về ta cũng có thể hiển thị nội dung HTML bằng cách tạo ra chuỗi có nội dung như sau:

```
WebView wv = (WebView) findViewById(R.id.webview1);

final String mimeType = "text/html";
final String encoding = "UTF-8";
String html = "<H1>Đây là một trang web đơn giản</H1><body>" +
"<p>Bạn có thể tạo ra nội dung HTML bằng cách này</p>";
```

Và nạp lên WebView như sau:

```
wv.loadDataWithBaseURL("", html, mimeType, encoding, "");
```

Hơn thế nữa ta cũng có thể soạn một file HTML tên là Index.html và để trong thư mục assets của project Android với nội dung như sau:

```
<h1>Đây là một file HTML đơn giản</h1>
```

```
<body>
<p>Chào mừng bạn đến với thế giới Android</p>
<imgsrc='http://www.google.com/logos/Logo_60wht.gif'/>
</body>
```

và nạp lên WebView:

```
WebView wv = (WebView) findViewById(R.id.webview1);
wv.loadUrl("file:///android_asset/Index.html");
```