Trường Đại học Bách Khoa Hà Nội

Hanoi University of Science and Technology

Chapter 4. Graphical User Interfaces

Hanoi University of Science and Technology

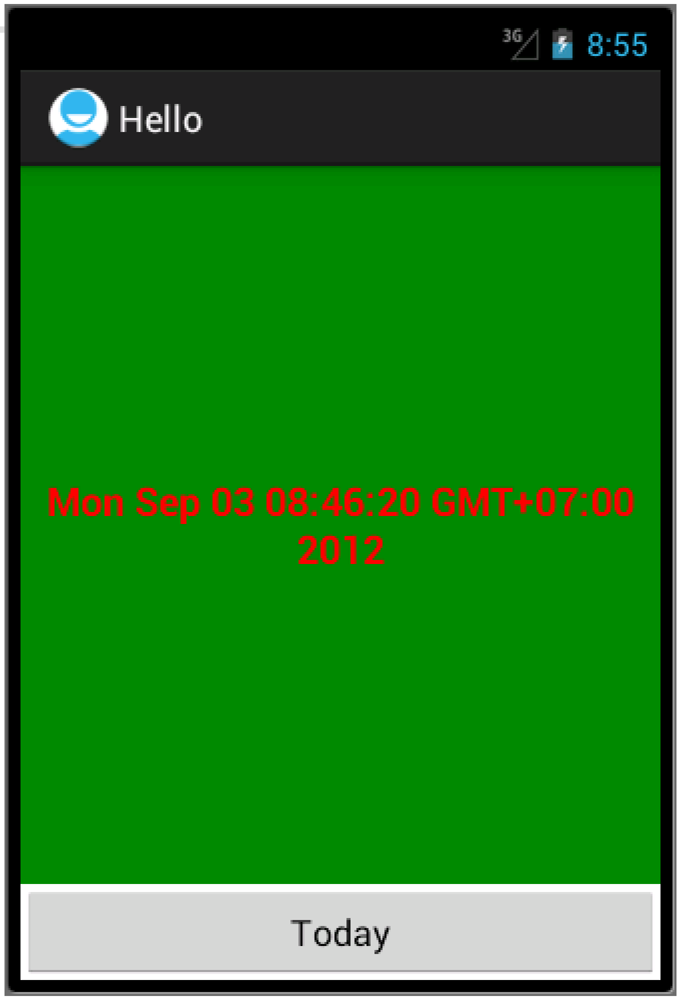
Nguyen Hong Quang
2/9/2012

Content

- 4.1a. Layout
  - LinearLayout
  - RelativeLayout
  - TableLayout
  - ScrollView
- 4.1b. UI Events Handling
  - Listener Introduction
  - onClickListener
  - onKeyListener, onFocusChangeListener, onLongClickListener, onTouchListener

Example

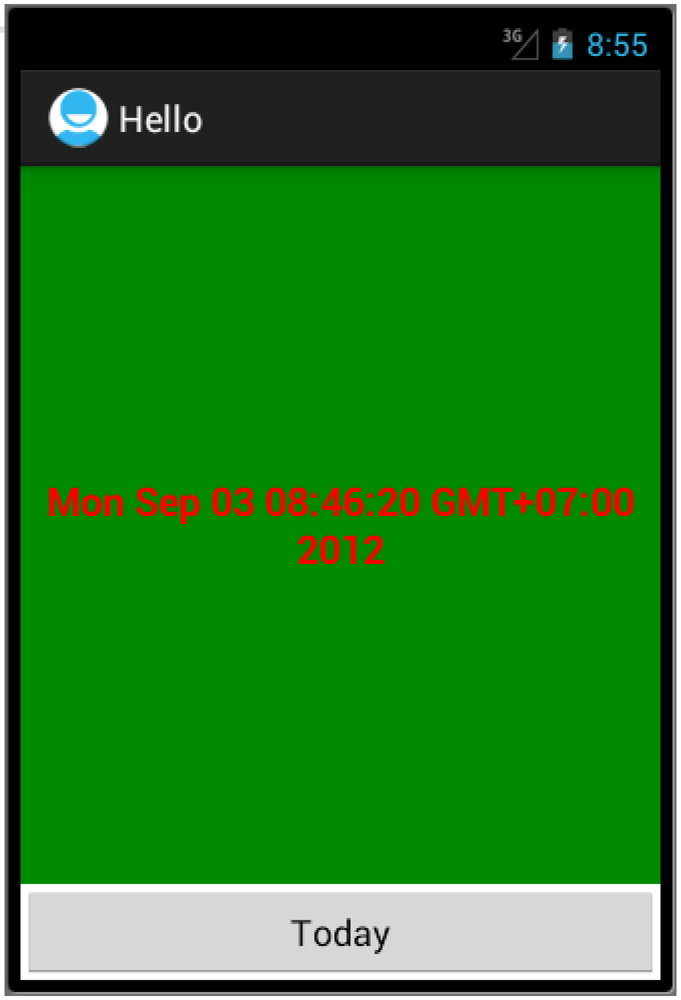
- 2 widgets: TextView and Button
- Pressing button : show today in TextView



Mon Sep 03 08:46:20 GMT+07:00 2012

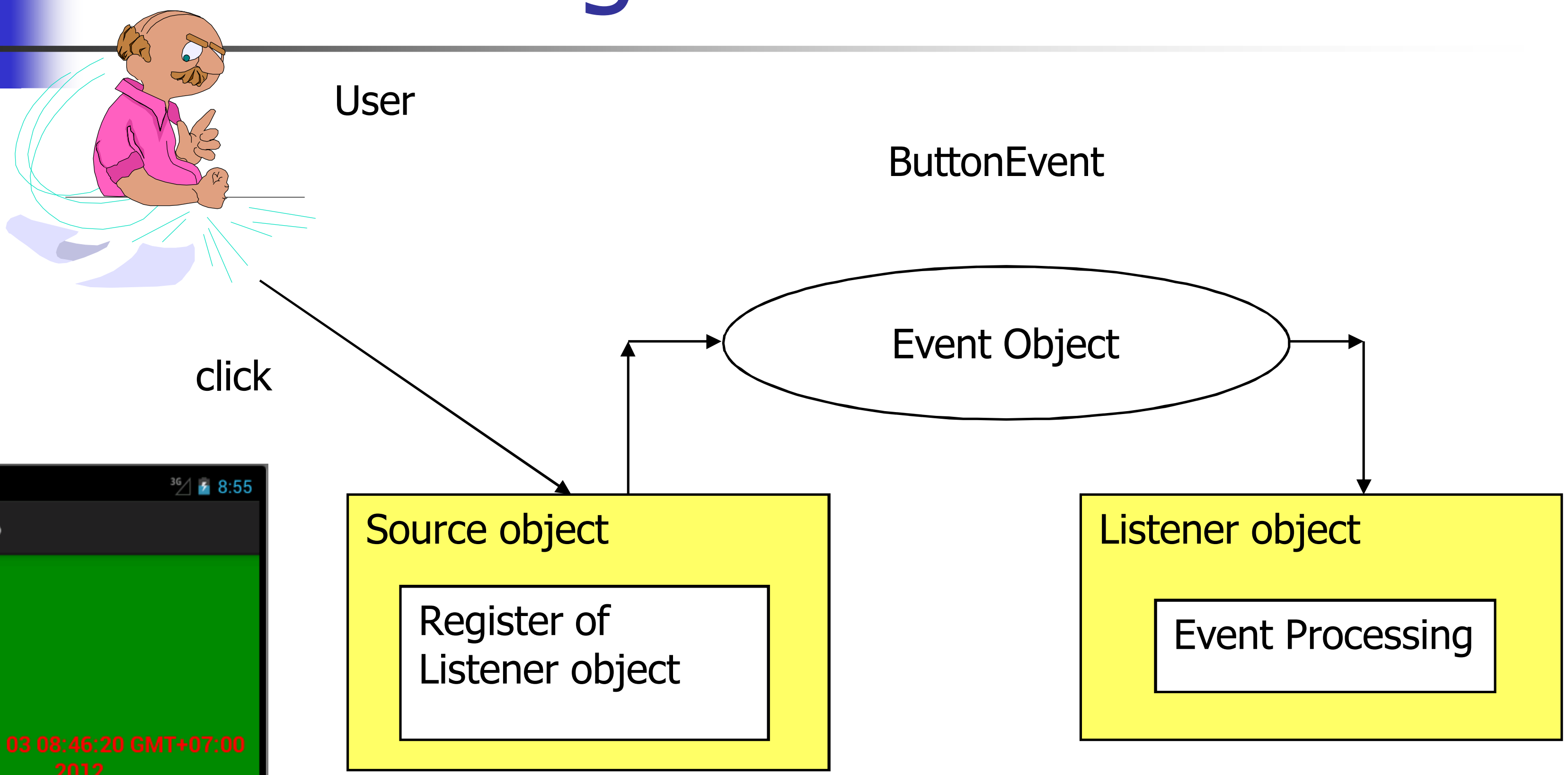
Main.xml

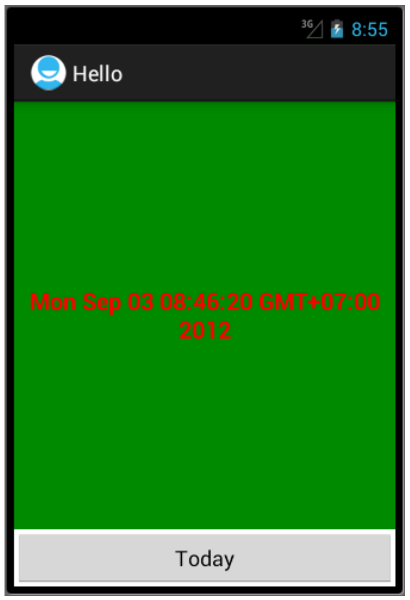
```
<RelativeLayout>
     <TextView
            android:id="@+id/text"
            android:layout_above="@+id/button"
    />
    <Button
       android:id="@id/button"
     android:layout_alignParentBottom="true"
    />
</RelativeLayout>
```



Mon Sep 03 08:46:20 GMT+07:00 2012

Processing of GUI events





```
public void onClick(View v) {
    TextView name=(TextView)findViewById(R.id.text);
    name.setText(new Date().toString());
}
```

Creating Listener object

- Listener object must implements corresponding interface.
- Example:

```
class ButtonListener implements View.OnClickListener
{
    public void onClick(View v) {}
}
```

Source object registers listener object

- Using method setABCListener with ABC events.
- Set on source object.

Method 1
Creating new Listener class

```
class ButtonListener implements View.OnClickListener {
    @Override
    public void onClick(View v) {
    // TODO Auto-generated method stub
    TextView name=(TextView)findViewById(R.id.text);
    name.setText(new Date().toString());
    }
}
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button button = (Button)findViewById(R.id.button);
```

Method 1
Creating new Listener class

```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button button = (Button)findViewById(R.id.button);
        ButtonListener btProcess = new ButtonListener();
        button.setOnClickListener(btProcess);
}
```

Method 2. Use of an anonymous inner class

```
private View.OnClickListener onSave = new View.OnClickListener() {
    public void onClick(View v) {
        TextView name=(TextView)findViewById(R.id.text);
        name.setText(new Date().toString());
    }
};
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(onSave);
}
```

Method 2. Use of an anonymous inner class

```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(
            new View.OnClickListener() {
                  public void onClick(View v) {
                     TextView name=(TextView)findViewById(R.id.text);
                      name.setText(new Date().toString());
                  }
            }
         );
}
```

Method 3. Android fashion

- Define some method on your Activity that holds the button that takes a single View parameter, has a void return value, and is public
- In your layout XML, on the Button element, include the android:onClick attribute with the name of the method you defined in the previous step

Example

- Have a method on Activity

```
public void **someMethod(View theButton) {**
   // do something useful here
}
```

- Use this XML declaration for the Button itself, including android:onClick

```
<Button
   android:onClick="someMethod"
     ...
/>
```

Details

```
public void btProcess(View theButton) {
    TextView name=(TextView)findViewById(R.id.text);
    name.setText(new Date().toString());
}

<Button
     android:onClick="btProcess"
     ...
/>
```

onKeyListener

- Interface definition for a callback to be invoked when a hardware key event is dispatched to this view.
- public abstract boolean onKey(View v, int keyCode, KeyEvent event)
- Parameters:
  - V: The view the key has been dispatched to.
  - keyCode: The code for the physical key that was pressed
  - Event: The KeyEvent object containing full information about the event.
- Returns
  - True if the listener has consumed the event, false otherwise.

onFocusChangeListener

- Called when the focus state of a view has changed.
- public abstract void onFocusChange (View v, boolean hasFocus)
- Parameters
  - v: The view whose state has changed.
  - hasFocus: The new focus state of v.

onLongClickListener

- Called when a view has been clicked and held.
- public abstract boolean onLongClick (View v)
- Parameters
  - v: The view that was clicked and held.
- Returns
  - True: if the callback consumed the long click, false otherwise.

onTouchListener

- Called when a touch event is dispatched to a view. This allows listeners to get a chance to respond before the target view.
- **public abstract boolean onTouch (View v, MotionEvent event)**
- **Parameters**
  - v: The view the touch event has been dispatched to.
  - vvent: The MotionEvent object containing full information about the event.
- **Returns**
  - True: if the listener has consumed the event, false otherwise.

End of Lecture

Q&A