FPT Software

ANDROID TRAINING

LESSON 5

Version 0.1

Agenda

- Toast
- Dialog
  - AlertDialog
  - ProgressDialog
  - DatePickerDialog
  - TimePickerDialog
- Dealing with thread
- AsyncTask

Toast

- A toast notification is a message that pops up on the surface of the window. It only fills the amount of space required for the message and the user's current activity remains visible and interactive.
- The notification automatically fades in and out, and does not accept interaction events.

Toast

- instantiate a Toast object with one of the makeText() methods.
- display the toast notification with show()

```
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;

Toast toast = Toast.makeText(context, text, duration);
toast.show();
```

Toast

- **Positioning your Toast**
  - change this position with the setGravity(int, int, int) method. This accepts three parameters: a Gravity constant, an x-position offset, and a y-position offset.

```
toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

Toast

- **Creating a Custom Toast View**
  - To create a custom layout, define a View layout, in XML or in your application code, and pass the root View object to the setView(View) method.

```
LayoutInflater li = getLayoutInflater();
View v = li.inflate(R.layout.toast_layout,
    (ViewGroup)findViewById(R.id.my_toast_layout_root));
Toast t = new Toast(getApplicationContext());
t.setView(v);
t.setDuration(Toast.LENGTH_LONG);
t.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
t.show();
```

Dialog

- Dialog is usually a small window that appears in front of the current Activity. The underlying Activity loses focus and the dialog accepts all user interaction.
- Dialogs are normally used for notifications that should interupt the user and to perform short tasks that directly relate to the application in progress (such as a progress bar or a login prompt).
- The Dialog class is the base class for creating dialogs. However, you typically should not instantiate a Dialog directly. Instead, you should use one of the following subclasses:
  - AlertDialog
  - ProgressDialog
  - DatePickerDialog
  - TimePickerDialog

Dialog

- Activities provide a facility to manage the creation, saving and restoring of dialogs.
  - onCreateDialog(int)
  - onPrepareDialog(int, Dialog)
  - showDialog(int)
  - dismissDialog(int)
- If you decide to create a dialog outside of the onCreateDialog() method, it will not be attached to an Activity. You can, however, attach it to an Activity with setOwnerActivity(Activity).

AlertDialog

- Use it for dialogs that use any of the following features:
  - A title
  - A text message
  - One, two, or three buttons
  - A list of selectable items (with optional checkboxes or radio buttons)
- To create an AlertDialog, use the AlertDialog.Builder subclass.
- Get a Builder with AlertDialog.Builder(Context) and then use the class's public methods to define all of the AlertDialog properties.
- Retrieve the AlertDialog object with create().

ProgressDialog

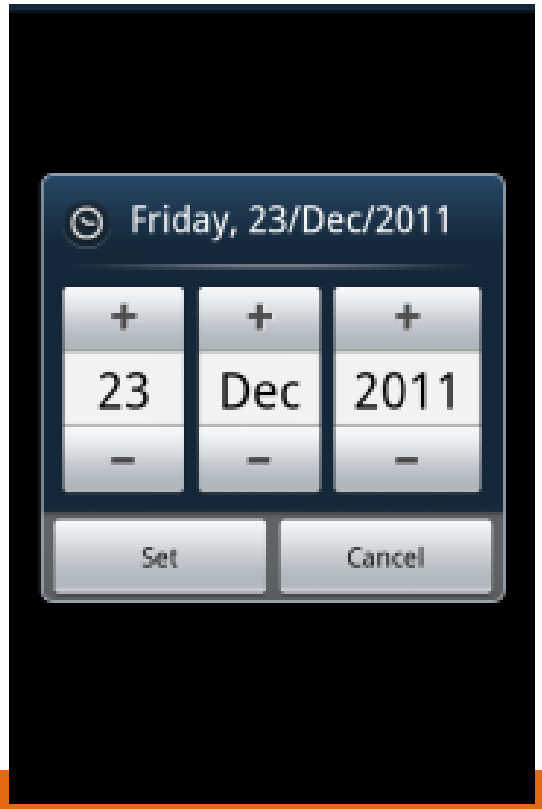
- Display a progress animation in the form of a spinning wheel, for a task with progress that's undefined, or a progress bar, for a task that has a defined progression.
- The dialog can also provide buttons, such as one to cancel a download.

ProgressDialog

- To show the progression with an animated progress bar:
  - Initialize the ProgressDialog with the class constructor, ProgressDialog(Context).
  - Set the progress style to "STYLE_HORIZONTAL" with setProgressStyle(int) and set any other properties, such as the message.
  - When you're ready to show the dialog, call show() or return the ProgressDialog from the onCreateDialog(int) callback.
  - You can increment the amount of progress displayed in the bar by calling either setProgress(int) with a value for the total percentage completed so far or incrementProgressBy(int) with an incremental value to add to the total percentage completed so far.

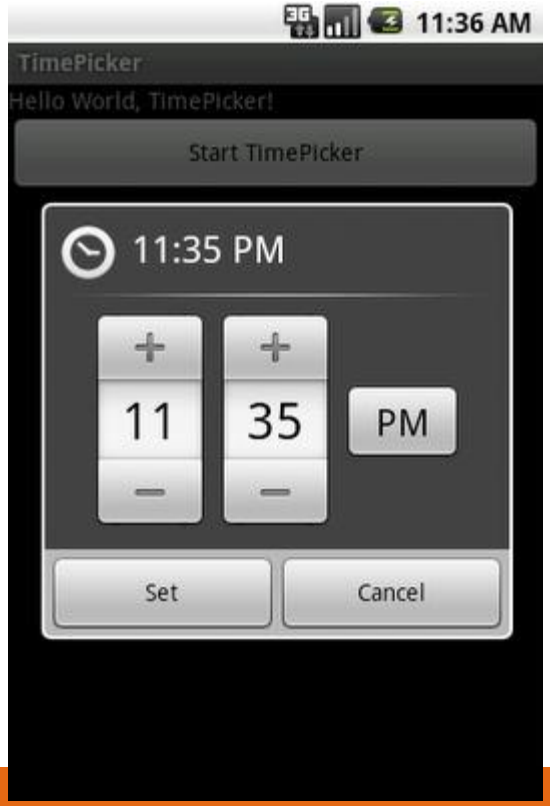
DatePickerDialog

- A dialog that prompts the user for the Date using a DatePicker.



TimePickerDialog

- A dialog that prompts the user for the time of day using a TimePicker.



Dealing with thread

- When an application is launched, the system creates a thread of execution for the application, called "main." This thread is very important because it is in charge of dispatching events to the appropriate user interface widgets, including drawing events.
- Interacts with components from the Android UI toolkit
- The main thread is also sometimes called the UI thread.

Dealing with thread

- The Andoid UI toolkit is *not* thread-safe. So, you must not manipulate your UI from a worker thread—you must do all manipulation to your user interface from the UI thread. Thus, there are simply two rules to Android's single thread model:
  - Do not block the UI thread
  - Do not access the Android UI toolkit from outside the UI thread

Dealing with thread

- Android offers several ways to access the UI thread from other threads:
  - Activity.runOnUiThread(Runnable)
  - View.post(Runnable)
  - View.postDelayed(Runnable, long)
  - Handler

AsyncTask

- **AsyncTask** simplifies the creation of long-running tasks that need to communicate with the user interface.
- The goal of AsyncTask is to take care of thread management for you

AsyncTask

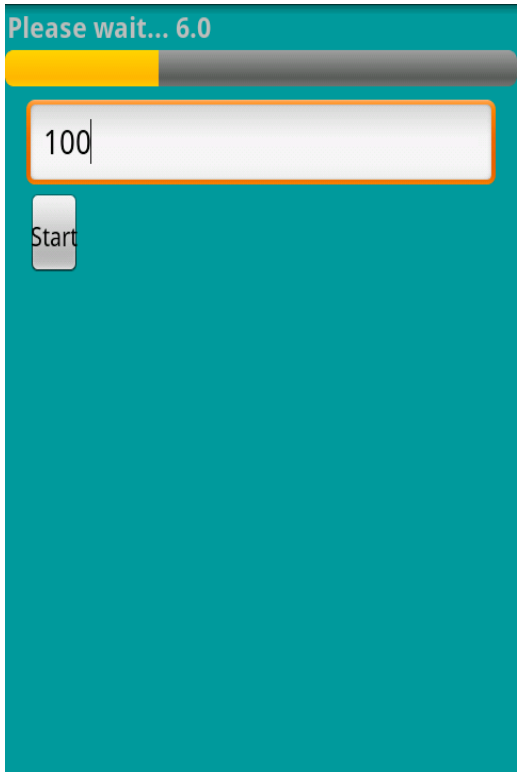
- The method doInBackground() executes automatically on a worker thread
- onPreExecute(), onPostExecute() and onProgressUpdate() are all invoked on the UI thread
- The value returned by doInBackground() is sent to onPostExecute()
- You can call publishProgress() at anytime in doInBackground() to execute onProgressUpdate() on the UI thread
- You can cancel the task at any time, from any thread

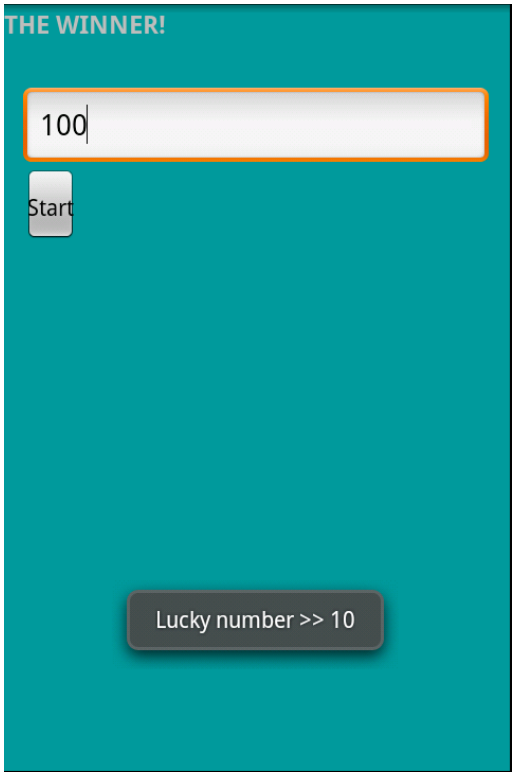
AsyncTask

- **AsynTask<String, void, String>**
- Things to note here are:
  - 1st String represents **Params** i.e the type of parameter **doInBackground** method will accept and also represents the type of parameter **execute** method will accept.
  - 2nd void represent **Progress** i.e the parameter type for **onProgressUpdate** method
  - 3rd String represents **Result** i.e the type of parameter accepted by **onPostExecute** method

Homework

- Create a simple “lucky draw” game





Homework

- Start button will start a background thread
- The background thread runs in 10 s -> update time left and seekbar on GUI
- After 10s, the background thread random 1 number in range on Textbox, alerts message to user

Thank you!