Drag and Drop

Tushar B. Kute,
http://tusharkute.com

Drag and Drop

- Android drag/drop framework allows your users to move data from one View to another View in the current layout using a graphical drag and drop gesture. As of API 11 drag and drop of view onto other views or view groups is supported.
- The framework includes following three important components to support drag & drop functionality –
  - Drag event class.
  - Drag listeners.
  - Helper methods and classes.

Drag and Drop Process

- There are basically four steps or states in the drag and drop process –
- Started – This event occurs when you start dragging an item in a layout, your application calls startDrag() method to tell the system to start a drag. The arguments inside startDrag() method provide the data to be dragged, metadata for this data, and a callback for drawing the drag shadow.
- Continuing – The user continues the drag. System sends ACTION_DRAG_ENTERED action followed by ACTION_DRAG_LOCATION action to the registered drag event listener for the View where dragging point enters. The listener may choose to alter its View object's appearance in response to the event or can react by highlighting its View.

Drag and Drop Process

- Dropped – The user releases the dragged item within the bounding box of a View. The system sends the View object's listener a drag event with action type ACTION_DROP.
- Ended – Just after the action type ACTION_DROP, the system sends out a drag event with action type ACTION_DRAG_ENDED to indicate that the drag operation is over.

The DragEvent class

- The DragEvent represents an event that is sent out by the system at various times during a drag and drop operation.
- This class provides few Constants and important methods which we use during Drag/Drop process.

The DragEvent constants

- ACTION_DRAG_STARTED
  - Signals the start of a drag and drop operation.
- ACTION_DRAG_ENTERED
  - Signals to a View that the drag point has entered the bounding box of the View.
- ACTION_DRAG_LOCATION
  - Sent to a View after ACTION_DRAG_ENTERED if the drag shadow is still within the View object's bounding box.
- ACTION_DRAG_EXITED
  - Signals that the user has moved the drag shadow outside the bounding box of the View.
- ACTION_DROP
  - Signals to a View that the user has released the drag shadow, and the drag point is within the bounding box of the View.
- ACTION_DRAG_ENDED
  - Signals to a View that the drag and drop operation has concluded.

The DragEvent methods

- int getAction()
  - Inspect the action value of this event..
- ClipData getClipData()
  - Returns the ClipData object sent to the system as part of the call to startDrag().
- ClipDescription getClipDescription()
  - Returns the ClipDescription object contained in the ClipData.
- boolean getResult()
  - Returns an indication of the result of the drag and drop operation.
- float getX()
  - Gets the X coordinate of the drag point.
- float getY()
  - Gets the Y coordinate of the drag point.
- String toString()
  - Returns a string representation of this DragEvent object.

Listening DragEvent

- If you want any of your views within a Layout should respond Drag event then your view either implements View.OnDragListener or setup onDragEvent(DragEvent) callback method.
- When the system calls the method or listener, it passes to them a DragEvent object explained above. You can have both a listener and a callback method for View object. If this occurs, the system first calls the listener and then defined callback as long as listener returns true.
- The combination of the onDragEvent(DragEvent) method and View.OnDragListener is analogous to the combination of the onTouchEvent() and View.OnTouchListener used with touch events in old versions of Android.

Starting DragEvent

- You start with creating a ClipData and ClipData.Item for the data being moved. As part of the ClipData object, supply metadata that is stored in a ClipDescription object within the ClipData. For a drag and drop operation that does not represent data movement, you may want to use null instead of an actual object.
- Next either you can extend extend View.DragShadowBuilder to create a drag shadow for dragging the view or simply you can use View.DragShadowBuilder(View) to create a default drag shadow that's the same size as the View argument passed to it, with the touch point centered in the drag shadow.

Example

- DragDrop.java

Thank you

Web Resources
http://mitu.co.in
http://tusharkute.com

Blogs
http://digitallocha.blogspot.in
http://kyamputar.blogspot.in

tushar@tusharkute.com