Accelerating Information Technology Innovation

http://aiti.mit.edu

Lecture 5: More Views and Data Binding

AITI Nigeria Summer 2012

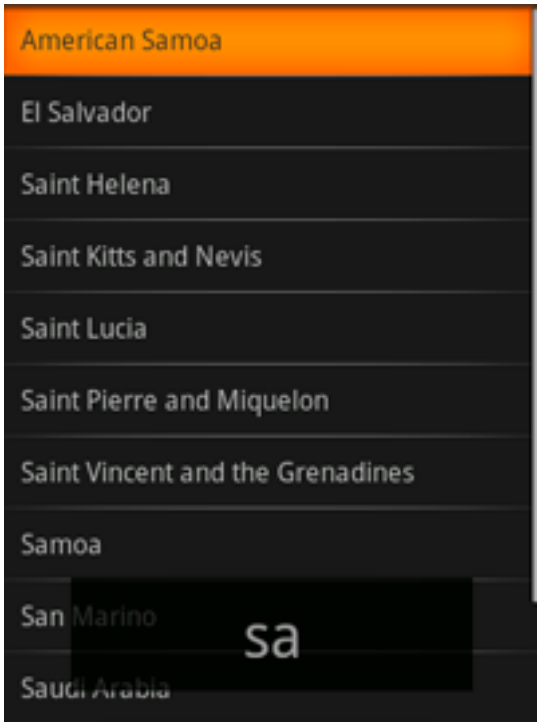
University of Lagos.

MISTI

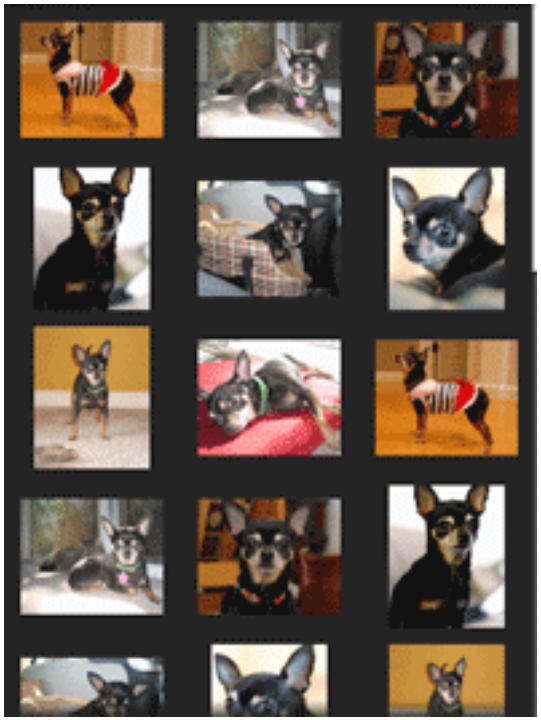
Agenda

- Views we haven’t yet learned
- The importance of data binding
- How to implement it?
  - An example with Lists

Views we haven't yet learned

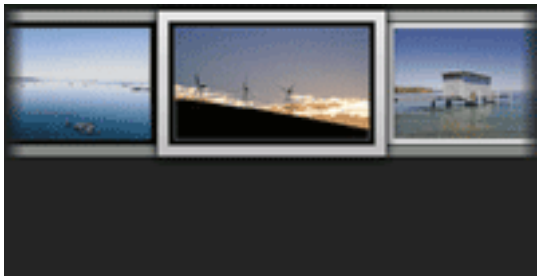


ListView



GridView

Spinner (drop-down)



Gallery

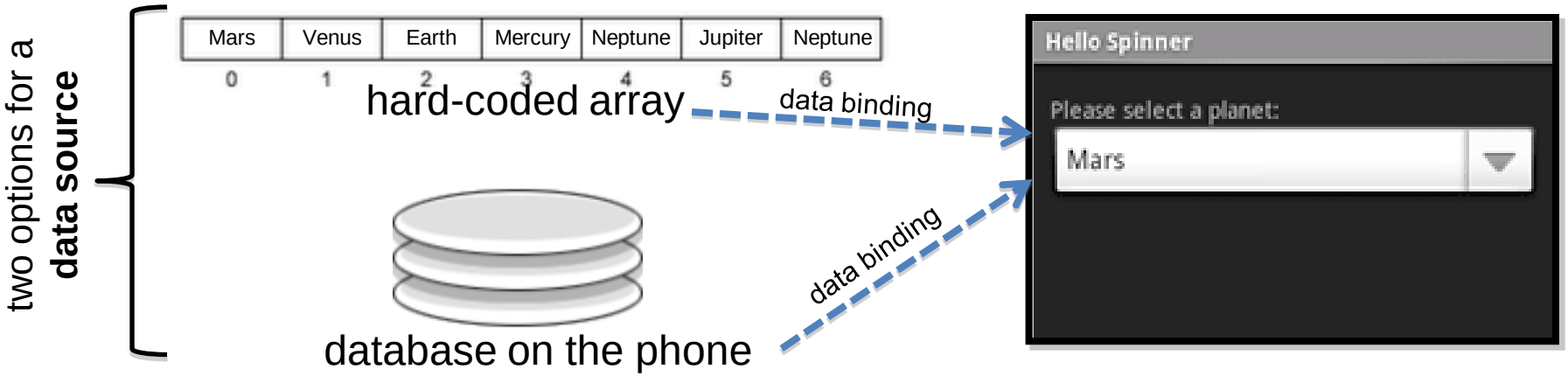
What do all of these have in common?

What do all those views have in common?

- All of them store and display **multiple** items!
  - A ListView displays items in a 1-D vertical, scrollable list
  - A GridView displays items in a 2-D, scrollable grid
  - A Spinner displays items in 1-D vertical *drop-down* component.
  - A Gallery displays items in a 1-D, horizontal, scrollable list.
- How do we provide multiple items to these views?
  - Use Data Binding

Importance of Data Binding

- **Data Binding**: the process of connecting views that display multiple items to a **data source**
- Any modifications to the data source will be reflected on the view immediately and automatically.



Possible Data Sources

- Data can be fetched from multiple sources:
  - Hard-coded arrays, defined in code
  - XML Resource Files
  - Databases on the phone
  - Content Providers / Content Resolvers (e.g. to populate a ListView with all the contacts on your phone)

Example: ListView with an array data source

- Step 1: Create a class of type ListActivitiy (as opposed to Activity)
- Step 2: Create the array data source. Two ways of doing this:
  - Hard-code the array in the ListActivity Class :

```
static final String[] THE_BIG_FIVE = new String[] {
    "Lion",
    "Leapord",
    "Rhino",
    "Elephant",
    "Buffalo"
};
```

  - Define the array in as an XML resource. Add the <string-array> to **`res/values/strings.xml`**

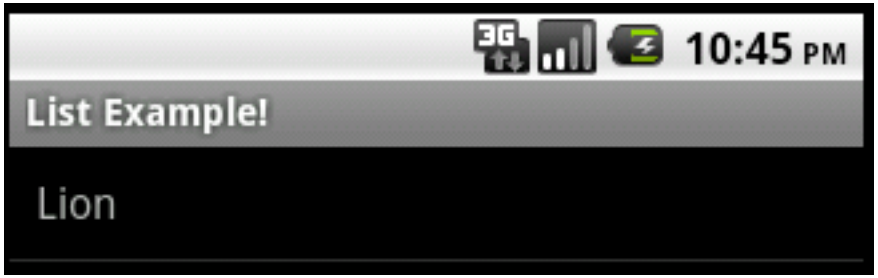
```
<resources>
    <string-array name="animals_array">
        <item>Lion</item>
        <item>Leapord</item>
        <item>Rhino</item>
        <item>Elephant</item>
        <item>Buffalo</item>
    </string-array>
</resources>
```

ListView Example, continued…

- Step 3: Create an XML layout file that will define how each cell or item in the ListView will look. Call this file "list_item.xml" and add to **`res/layout/`**

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dp"
    android:textSize="16sp" >
</TextView>
```

Note: This XML code means that each list item will essentially be a TextView, i.e. a simple text label. If we wanted each list item to also show an icon, we would need to modify this xml file to also include an ImageIcon and a layout of some sort.



Each list item, e.g. "Lion", is simply a TextView

ListView Example, continued…

- Step 4: Now, establish the data binding in the onCreate() method of the ListActivity class
  - If data source is a hard-coded array, use the following:

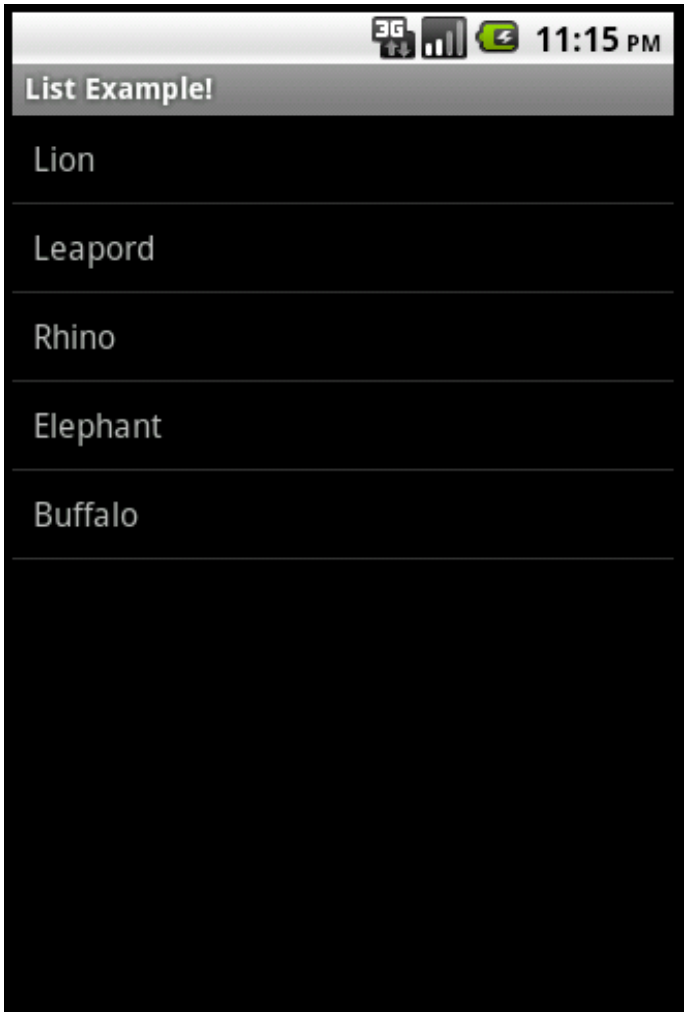
```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);

  //method 1
  setListAdapter(new ArrayAdapter<String>(this, R.layout.listitem_view, THE_BIG_FIVE));

  ListView lv = getListView();
  lv.setTextFilterEnabled(true);

  lv.setOnItemClickListener(new OnItemClickListener() {
    public void onItemClick(AdapterView<?> parent, View view,
        int position, long id) {
      // Handle list item click and do something here

    }
  });
}
```

ListView Example, continued…

- Step 4 contd…
  - However, If data source is defined in an XML resource file, use the following:

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);

  //method 2
  String[] students = getResources().getStringArray(R.array.students_array);
  setListAdapter(new ArrayAdapter<String>(this, R.layout.listitem_view, students));

  ListView lv = getListView();
  lv.setTextFilterEnabled(true);

  lv.setOnItemClickListener(new OnItemClickListener() {
    public void onItemClick(AdapterView<?> parent, View view,
        int position, long id) {
      // Handle list item click and do something here

    }
  });
}
```

The End Result!



One more thing ..

www.stackoverflow.com

IS YOUR BEST FRIEND!!