Chapter 7: Reveal!
Displaying Pictures in a Gallery
Objectives

In this chapter, you learn to:

• Create an Android project using a Gallery control
• Add a Gallery to display a horizontal list of images
• Reference images through an array
• Create an ImageAdapter class
• Code an OnItemClickListener
• Display a custom toast message
• Define a Context resource
• Understand the use of constructors
Objectives (continued)

• Return a value from a method
• Determine the length of an array
• Assign an ImageView control using setImageResource
• Change the scale and layout size of the Gallery
Gallery View

- A gallery is a center-locked horizontally scrolling list
  - Adds Visual Appeal
  - Clean, Professional Effect
  - Flip Photos with Fingers
  - Tap for full-size

Figure 7-2 Snow Leopard image selected in the Gallery
• **Steps to complete the App:**
  1. Add a Gallery control to the emulator.
  2. Add XML code for an ImageView control not linked to a particular image.
  3. Place six images in a drawable folder.
  4. Define an array to hold the image files.
  5. Instantiate the Gallery and ImageView controls.
  6. Create an ImageAdapter class.
  7. Display a custom toast message.
  8. Display the selected image.
  9. Customize the ImageAdapter class.
  10. Define the layout using the getView() method.
Adding a Gallery Control

– A **view** container is a rectangular area on the screen that displays an image or text object
– The **Gallery** view container displays a horizontal list of objects with the center item displaying the current image
– Photos can be sized as thumbnail images or full-screen images
– Photos can be stored in the drawable folders, in a phone’s storage, or on a Web site such as Picassa
Adding a Gallery Control (continued)

- Adding the ImageView Control and Image Files
  - Images can be dragged onto the emulator.
  - Select from the list of images in the drawable folders

Figure 7-4 Gallery control
Adding a Gallery Control (continued)

- Adding the ImageView Control and Image Files (continued)

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">

    <Gallery
        android:id="@+id/gallery1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <ImageView
        android:id="@+id/imgAnimal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Figure 7-5 ImageView XML code
Adding a Gallery Control (continued)

- Creating an Array for the Images

![Image of Android project structure]

Six image files placed in drawable-hdpi folder

<table>
<thead>
<tr>
<th>Element of Array</th>
<th>Image File</th>
</tr>
</thead>
<tbody>
<tr>
<td>Animals[0]</td>
<td>elephant.png</td>
</tr>
<tr>
<td>Animals[1]</td>
<td>gorilla.png</td>
</tr>
<tr>
<td>Animals[2]</td>
<td>leopard.png</td>
</tr>
<tr>
<td>Animals[3]</td>
<td>monkey.png</td>
</tr>
<tr>
<td>Animals[4]</td>
<td>panda.png</td>
</tr>
<tr>
<td>Animals[5]</td>
<td>redpanda.png</td>
</tr>
</tbody>
</table>

Table 7-1 Animals array

Figure 7-6 Images copied
Creating a Gallery Control (continued)

- Creating an Array for the Images (continued)
  - Images must be placed in the drawable folder
    - Must be referenced in the code
    - Must be assigned to an array

```java
Integer[] Animals = { R.drawable.elephant,
                      R.drawable.gorilla,
                      R.drawable.leopard,
                      R.drawable.monkey,
                      R.drawable.panda,
                      R.drawable.redpanda };
```
Creating a Gallery Control (continued)

Figure 7-8 ImageView referenced
Instantiating the Gallery and ImageView Controls

Figure 7-9 Gallery control is instantiated

Figure 7-10 ImageView control is instantiated
Using a setAdapter with an Image Adapter

- A `setAdapter` provides a data model for the gallery layout
- Code Syntax:
  
  ```java
  ga.setAdapter(new ImageAdapter(this));
  ```
- ImageAdapter must be instantiated
- ImageAdapter class must be added to extend the custom BaseAdapter class
Using a `setAdapter` with an Image Adapter (continued)

```java
package net.androidbootcamp.endangeredspecies;

import android.app.Activity;

public class Main extends Activity {
    Integer[] Animals = { R.drawable.elephant, R.drawable.gorilla, R.drawable.leopard, R.drawable.monkey, R.drawable.panda, R.drawable.redpanda);
    ImageView imageView;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Gallery ga = (Gallery)findViewById(R.id.gallery1);
        imageView = (ImageView)findViewById(R.id.imgAnimal);
        ga.setAdapter(new ImageAdapter(this));
    }
}
```

Figure 7-11 Instance of the ImageAdapter class
Using a `setAdapter` with an Image Adapter (continued)

```java
package set.androidonchamp.endangerespecter;

public class Main extends Activity {

    // Image[] animals = { R.drawable.elephant, R.drawable.gorilla, R.drawable.zebra, 
    // R.drawable.monkey, R.drawable.panda, R.drawable.narwhal };

    // ImageView imageView;

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

        animals = (animal) findViewById(R.id.androidAnimals);
        imageView = (ImageView) findViewById(R.id.imageView);

        ImageAdapter newImageAdapter = new ImageAdapter(this);
        imageView.setAdapter(newImageAdapter);
    }

    public class ImageAdapter extends BaseAdapter {

        @Override
        public void setListAdapter(List<?> list) {
            // Android Auto-generated constructor stub
        }

        @Override
        public int getCount() {
            // Android Auto-generated method stub
            return 0;
        }

        @Override
        public Object getItem(int index) {
            // Android Auto-generated method stub
            return null;
        }

        @Override
        public long getItemId(int position) {
            // Android Auto-generated method stub
            return 0;
        }

        @Override
        public View getView(int position, View convertView, ViewGroup parent) {
            // Android Auto-generated method stub
            return null;
        }
    }
}
```

Figure 7-12 ImageAdapter class
Coding the OnItemClickListener

- Recall that the OnItemClickListener awaits user interaction within the Gallery Control
- The `onItemClick` method is the event the listener responds to
- ListView, GridView and Gallery enable the Android device to monitor for click events
- Code Syntax:

```java
ga.setOnItemClickListener(new OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) { }
});
```
Coding the OnItemClickListener (continued)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdapterView&lt;-&gt; arg0</td>
<td>The AdapterView records “where” the user actually touched the screen in the argument variable arg0. In other words, if the app has more than one View control, the AdapterView determines if the user touched this Gallery control or another control in the application.</td>
</tr>
<tr>
<td>View arg1</td>
<td>The View parameter is the specific View within the item that the user touched. This is the View provided by the adapter.</td>
</tr>
<tr>
<td>int arg2</td>
<td>This is one of the most important portions of this statement in the chapter project. The arg2 argument is an integer value that holds the position of the View in the adapter. For example, if the user taps the gorilla picture, the integer value of 2 is stored in arg2 because the gorilla picture is the second image in the Animals array.</td>
</tr>
<tr>
<td>long arg3</td>
<td>The Gallery control is displayed across one row of the Android device. The argument arg3 determines the row id of the item that was selected by the user. This is especially useful for a GridView control that has multiple rows in the layout.</td>
</tr>
</tbody>
</table>

Table 7-2 Arguments in the onItemClick method
Coding the OnItemClickListener (continued)

Figure 7-13 Gallery OnItemClickListener
Coding the OnItemClickListener (continued)

Figure 7-14 The OnItemClick method
Coding a Custom Toast Notification

– Recall that toast notifications provide feedback to the user
– Previous toast notification code:
  Toast.makeText(Main.this, "A typical Toast message", Toast.LENGTH_SHORT).show();
– Since notification above would be in the onItemClick method, it is not used in Main Activity, so the reference to Main.this creates an error
• New toast notification code:

```java
Toast.makeText(getBaseContext(), "You have selected picture " + (arg2 + 1) + " of the endangered species", Toast.LENGTH_SHORT).show();
```

Figure 7-15 Toast message displayed when the user selects the gorilla image
Displaying the Selected Image

– When the user selects a picture, a toast message appears and the ImageView control displays the selected image

```java
imageView.setImageResource(Animals[arg2]);
```

Figure 7-17 ImageView control displays the selected Gallery picture
Customizing the ImageAdapter Class

- Data sources from the array must be connected to the Gallery control

• **Defining the Context of the ImageAdapter Class**
  - Constructors are used to initialize the instance variables of an object

```java
public class ImageAdapter extends BaseAdapter{
    private Context context;
    public ImageAdapter(Context c){
        // TODO Auto-generated constructor stub
        context=c;
    }

    @Override
    public int getCount() {
        // TODO Auto-generated method stub
        return 0;
    }
}
```

Figure 7-19 ImageAdapter constructor
Calculating the Length of an Array

- A Java **method** is a series of statements that perform some repeated task
- ImageAdapter class includes methods called:
  - `getCount()` which determines how many pictures to display
  - `length()` which returns the number of pictures in the array

```java
public int getCount() {
    // TODO Auto-generated constructor stub
    return Animals.length;
}
```
Coding the getView Method

- getView method uses Context to create a new ImageView instance to temporarily hold each image displayed in the Gallery

```java
public View getView(int arg0, View arg1, ViewGroup arg2) {
    // TODO Auto-generated method stub
    ImageView pic = new ImageView(context);
    pic.setImageResource(Animals[arg0]);
    pic.setScaleType(ImageView.ScaleType.FIT_XY);
    pic.setLayoutParams(new Gallery.LayoutParams(200, 175));
    return pic;
}
```

- The returned pic is a scaled, resized image, ready to display in the Gallery
Scaling keeps or changes the aspect ratio of the image to the bounds of the ImageView.

**Table 7-3 Popular ScaleType options**

<table>
<thead>
<tr>
<th>ScaleType option</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ImageView.ScaleType.CENTER</code></td>
<td>This option centers the image within the View type, but does not change the aspect ratio (no scaling).</td>
</tr>
<tr>
<td><code>ImageView.ScaleType.CENTER_CROP</code></td>
<td>This option centers the image within the View type and scales the image uniformly, maintaining the same aspect ratio.</td>
</tr>
<tr>
<td><code>ImageView.ScaleType.FIT_XY</code></td>
<td>This option scales the image to fit the View type. The aspect ratio is changed to fit within the control.</td>
</tr>
</tbody>
</table>
Completed code

Figure 7-25 Completed code of Main.java
Completed code (continued)

Figure 7-25 Completed code of Main.java
Summary

- A View container is a rectangular area of the screen that displays an image or text object
- A Gallery layout displays a horizontal list of objects
- Users can scroll the Gallery list and select an object
- XML code needed in mail.xml to display an image in the ImageView control
- Array variables can store multiple images and assign them to the Gallery control
Summary (continued)

• A setAdapter provides a data model for the Gallery layout
• The OnItemClickListener waits for user interaction in a Gallery control
• Including a toast notification displays a message indicating which image is selected in the gallery control
• Must use getBaseContext() method instead of main.this.
Summary (continued)

• Use setImageResource() method to insert an ImageView control
• Use the Context class to load and access resources for the application
• Use getCount() to determine how many pictures to display in the Gallery and length() to determine the number of elements in the Gallery
• getCount() returns an integer
• getView() created a new ImageView instance to hold each images displayed in the Gallery