Gallery

Gallery is a layout widget used to display items in a horizontally scrolling list and positions the current selection at the center of the view.

In this tutorial, you'll create a gallery of photos and then display a toast message each time a gallery item is selected.

1. Start a new project named *HelloGallery*.
2. Find some photos you'd like to use, or use these sample images. Save the images into the project's `res/drawable/` directory.
3. Open the `res/layout/main.xml` file and insert the following:

```
<?xml version="1.0" encoding="utf-8"?>
<Gallery xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gallery"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
/>
```

4. Open the `HelloGallery.java` file and insert the following code for the `onCreate()` method:

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

    Gallery gallery = (Gallery) findViewById(R.id.gallery);
    gallery.setAdapter(new ImageAdapter(this));

    gallery.setOnItemClickListener(new OnItemClickListener() {
        public void onItemClick(AdapterView parent, View v, int position, long
id) {
            Toast.makeText(HelloGallery.this, "" + position,
Toast.LENGTH_SHORT).show();
        }
    });
}
```

This starts by setting the `main.xml` layout as the content view and then capturing the `Gallery` from the layout with `findViewById(int)`. A custom `BaseAdapter` called `ImageAdapter` is instantiated and applied to the `Gallery` with `setAdapter()`. (The `ImageAdapter` class is defined next.) Then an anonymous `AdapterView.OnItemClickListener` is instantiated. The `onItemClick(AdapterView, View, int, long)` callback method receives the `AdapterView` where the click occurred, the specific `View` that received the click, the position of the `View` clicked (zero-based), and the row ID of the item clicked (if applicable). In this example, all that's needed is the position of the click to show a `Toast` message that says the position of the item, using `makeText(Context, CharSequence, int)` and `show()` (in a real world scenario, this ID could be used to get the full sized image for some other task).

5. Create a new XML file in the `res/values/` directory named `attrs.xml`. Insert the following:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="HelloGallery">
```

```
        <attr name="android:galleryItemBackground" />
    </declare-styleable>
</resources>
```

This is a custom styleable resource that can be applied to a layout. In this case, it will be applied to the individual items placed into the `Gallery` widget. The `<attr>` element defines a specific attribute for the styleable, and in this case, it refers to an existing platform attribute, `galleryItemBackground`, which defines a border styling for gallery items. In the next step, you'll see how this attribute is referenced and then later applied to each item in the gallery.

6. Go back to the `HelloGallery.java` file. After the `onCreate(Bundle)` method, define the custom `ImageAdapter` class:

```
public class ImageAdapter extends BaseAdapter {
    int mGalleryItemBackground;
    private Context mContext;

    private Integer[] mImageIds = {
            R.drawable.sample_1,
            R.drawable.sample_2,
            R.drawable.sample_3,
            R.drawable.sample_4,
            R.drawable.sample_5,
            R.drawable.sample_6,
            R.drawable.sample_7
    };

    public ImageAdapter(Context c) {
        mContext = c;
        TypedArray attr =
mContext.obtainStyledAttributes(R.styleable.HelloGallery);
        mGalleryItemBackground = attr.getResourceId(
                R.styleable.HelloGallery_android_galleryItemBackground, 0);
        attr.recycle();
    }

    public int getCount() {
        return mImageIds.length;
    }

    public Object getItem(int position) {
        return position;
    }

    public long getItemId(int position) {
        return position;
    }

    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageView = new ImageView(mContext);

        imageView.setImageResource(mImageIds[position]);
        imageView.setLayoutParams(new Gallery.LayoutParams(150, 100));
        imageView.setScaleType(ImageView.ScaleType.FIT_XY);
        imageView.setBackgroundResource(mGalleryItemBackground);

        return imageView;
    }
}
```

First, there are a few member variables, including an array of IDs that reference the images saved in the drawable resources directory (`res/drawable/`).

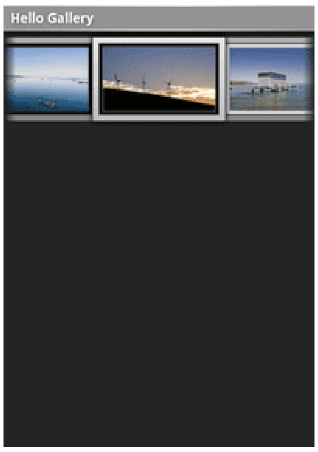
Next is the class constructor, where the `Context` for an `ImageAdapter` instance is defined and the styleable resource defined in the last step is acquired and saved to a local field. At the end of the constructor, `recycle()` is called on the `TypedArray` so it can be re-used by the system.

The methods `getCount()`, `getItem(int)`, and `getItemId(int)` are methods that must be implemented for simple queries on the `Adapter`. The `method does the work to apply an image to an {@link android.widget.ImageView` that will be embedded in the `Gallery`. In this method, the member `Context` is used to create a new `ImageView`. The `ImageView` is prepared by applying an image from the local array of drawable resources, setting the `Gallery.LayoutParams` height and width for the image, setting the scale to fit the `ImageView` dimensions, and then finally setting the background to use the styleable attribute acquired in the constructor.

See `ImageView.ScaleType` for other image scaling options.

7. Run the application.

You should see something like this:



References

- `BaseAdapter`
- `Gallery`
- `ImageView`
- `AdapterView.OnItemClickListener`

← Back to Hello, Views ↑ Go to top