Android Course Day 5

Lucio Cossio

Luis Mazoni

© Copyright 2014 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.

Course Agenda

Day 1

- **Development environment and tools**
- **Android project structure**
  - Source, tests , resources, manifest
- **Activity**
  - Lifecycle
  - Layout interaction
  - ActionBar
- **Practice**

Day 2

- **Review Day 1**
- **Resources**
  - String
  - Dimensions
  - Layout
- **Views**
  - TextView, Edit Text, Button
  - String resources
  - View listeners
- **Animation**
- **Practice**

Day 3

- **Review Day 2**
- **Intents**
  - Explicit
    - Extras
  - Implicit
    - Intent Filters
- **Android Manifest**
  - Overview
  - Add activities
- **Notifications**
- **Practice**

Day 4

- **Review Day 3**
- **Fragments**
  - Lifecycle
  - Fragment Manager
  - Arguments
- **Broadcast Receiver**
  - **Otto**
- **Async Task**
- **Practice**

Day 5

- **Review Day 4**
- **ListView**
  - Adapter
  - View Holder
- **Service**
- **Storage**
  - SharedPreferen ces
  - SQLite
- **Practice**

© Copyright 2014 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.

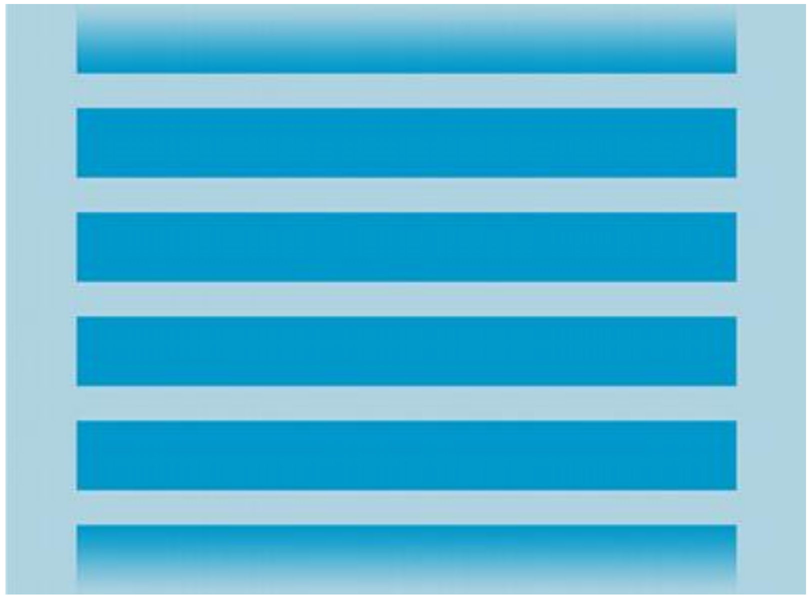
Review day 4

- Fragments
- Broadcast Receivers
  - Otto
- Async Tasks

© Copyright 2014 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.

ListView

- ListView is a view group that displays a list of scrollable items. The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database query and converts each item result into a view that's placed into the list.



© Copyright 2014 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.

Adapter - ArrayAdapter

- Use this adapter when your data source is an array. By default, ArrayAdapter creates a view for each array item by calling toString() on each item and placing the contents in a TextView.

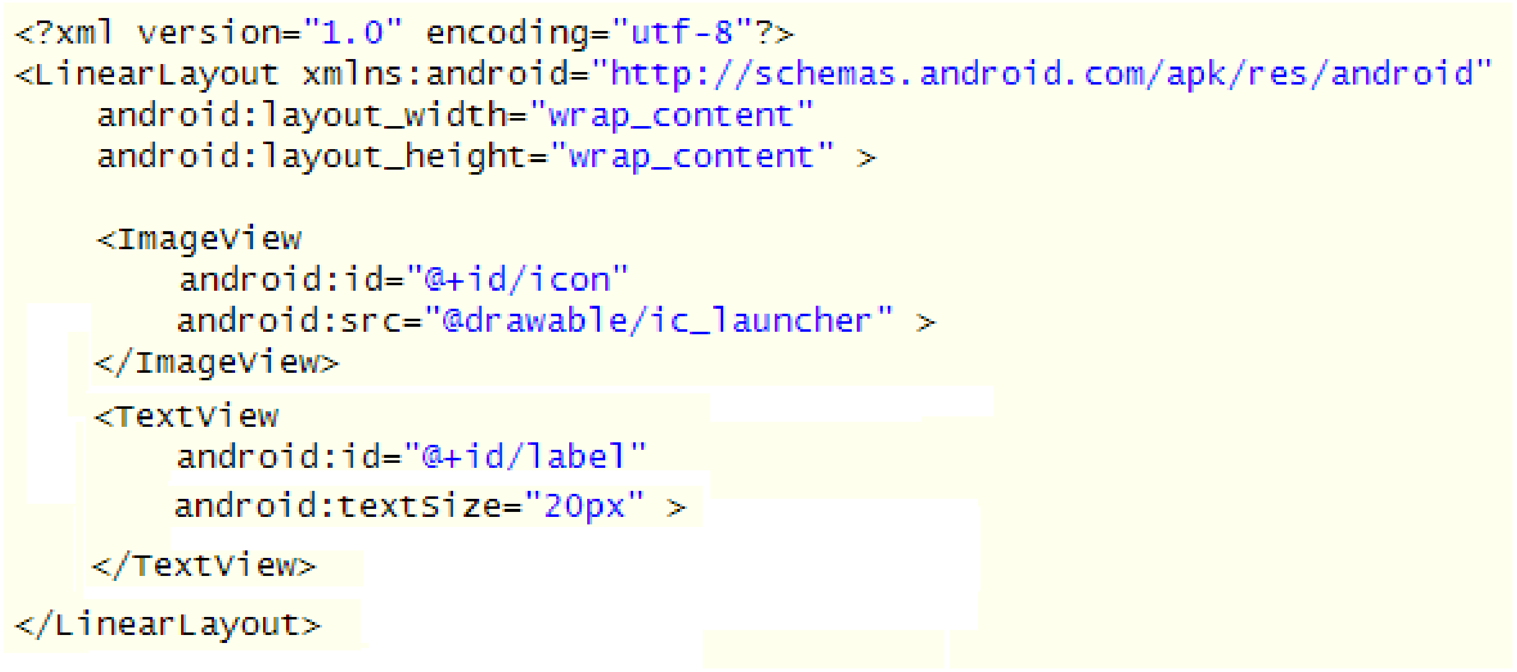
```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, myStringArray);
```

```
ListView listView = (ListView) findViewById(R.id.listview);
listView.setAdapter(adapter);
```

© Copyright 2014 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.

Custom Adapter

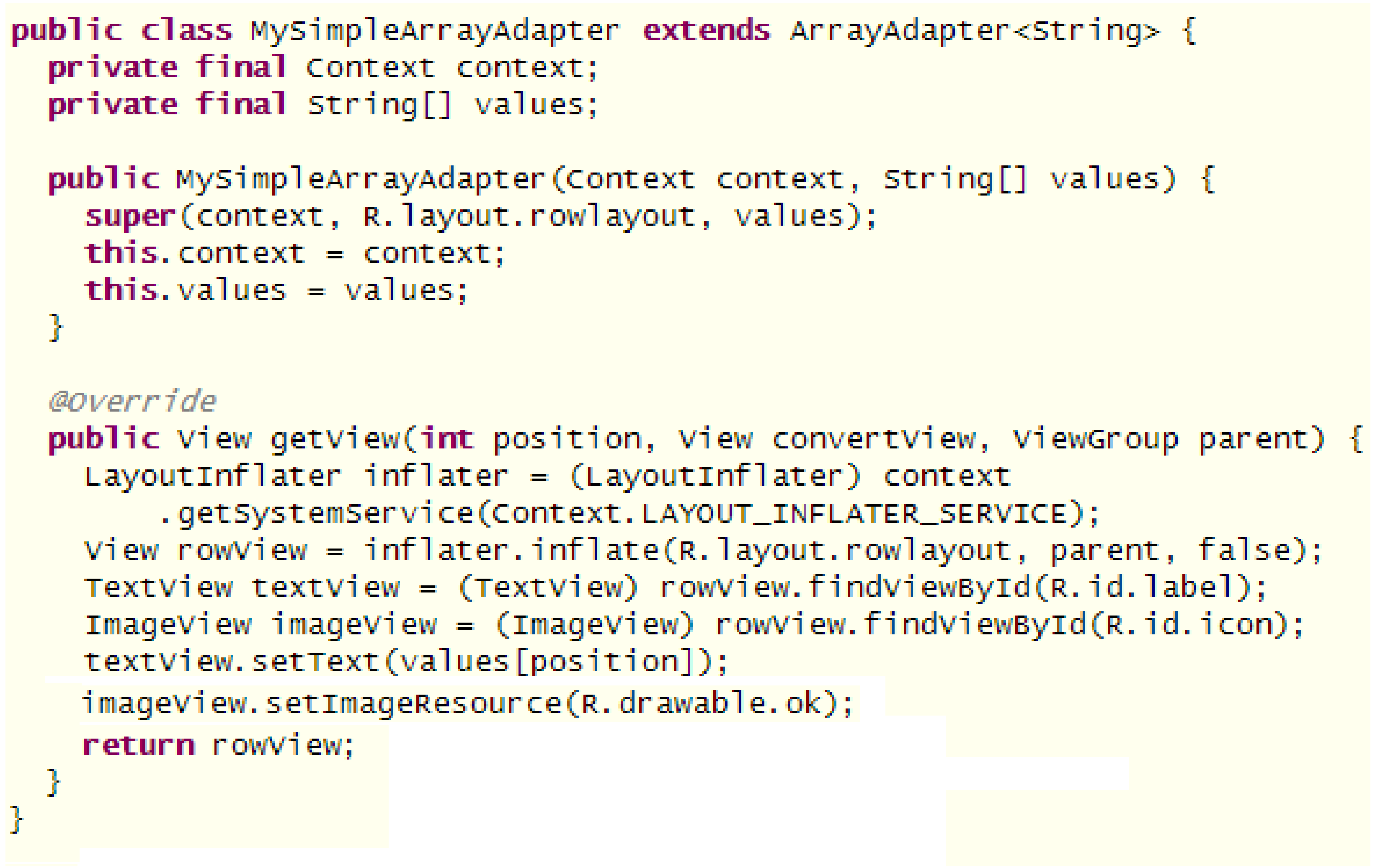
- We can extend an arrayadapter and use a custom layout
- Layout:



```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >

    <ImageView
        android:id="@+id/icon"
        android:src="@drawable/ic_launcher" >
    </ImageView>
    <TextView
        android:id="@+id/label"
        android:textSize="20px" >
    </TextView>
</LinearLayout>
```

© Copyright 2014 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.

Custom Adapter

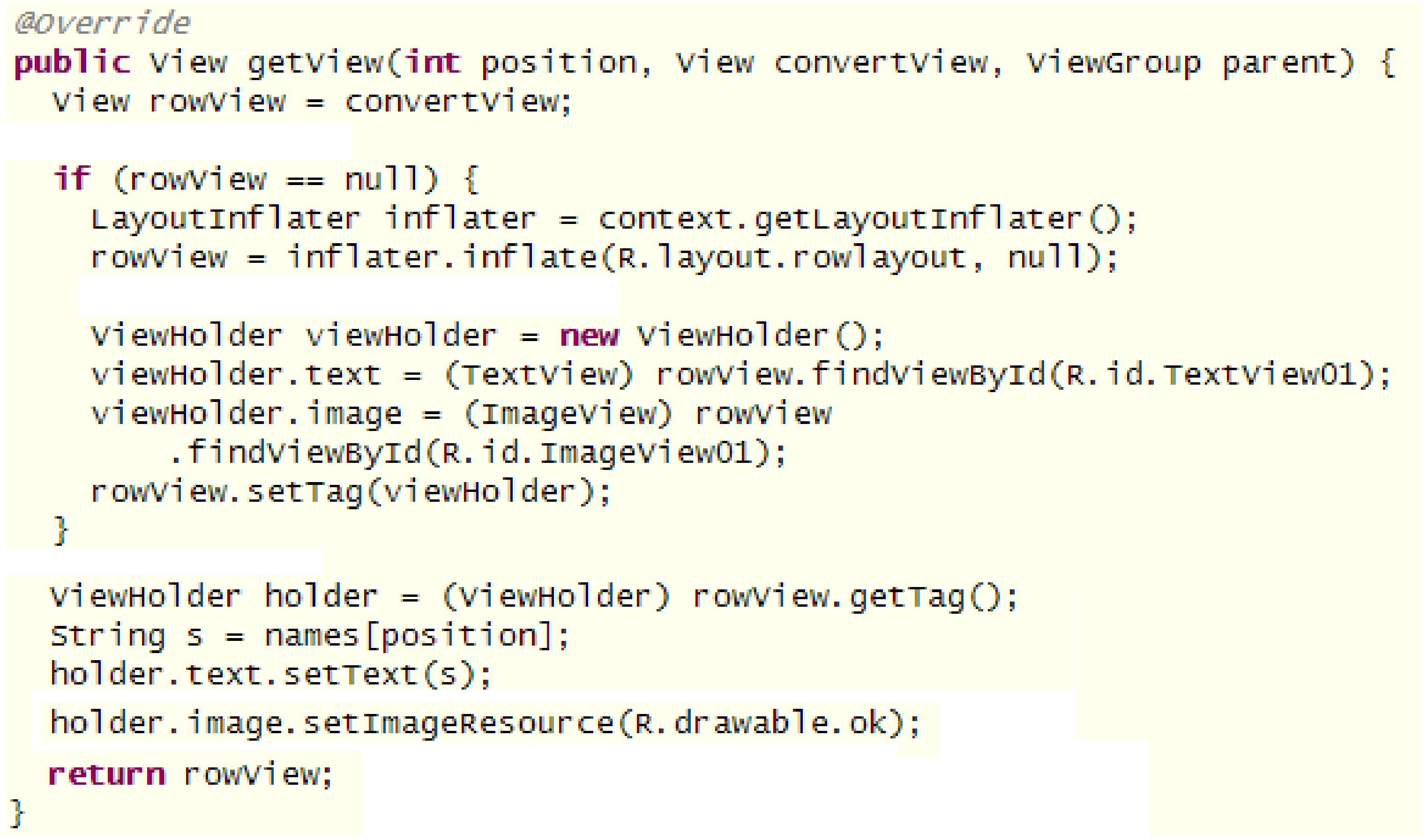


```java
public class MySimpleArrayAdapter extends ArrayAdapter<String> {
  private final Context context;
  private final String[] values;

  public MySimpleArrayAdapter(Context context, String[] values) {
    super(context, R.layout.rowlayout, values);
    this.context = context;
    this.values = values;
  }

  @Override
  public View getView(int position, View convertView, ViewGroup parent) {
    LayoutInflater inflater = (LayoutInflater) context
        .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    View rowView = inflater.inflate(R.layout.rowlayout, parent, false);
    TextView textView = (TextView) rowView.findViewById(R.id.label);
    ImageView imageView = (ImageView) rowView.findViewById(R.id.icon);
    textView.setText(values[position]);
    imageView.setImageResource(R.drawable.ok);
    return rowView;
  }
}
```

© Copyright 2014 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.

View Holder

• To improve performance on a ListView you should use the View Holder pattern.



```java
@Override
public View getView(int position, View convertView, ViewGroup parent) {
  View rowView = convertView;

  if (rowView == null) {
    LayoutInflater inflater = context.getLayoutInflater();
    rowView = inflater.inflate(R.layout.rowlayout, null);

    ViewHolder viewHolder = new ViewHolder();
    viewHolder.text = (TextView) rowView.findViewById(R.id.TextView01);
    viewHolder.image = (ImageView) rowView
        .findViewById(R.id.ImageView01);
    rowView.setTag(viewHolder);
  }

  ViewHolder holder = (ViewHolder) rowView.getTag();
  String s = names[position];
  holder.text.setText(s);
  holder.image.setImageResource(R.drawable.ok);
  return rowView;
}
```

© Copyright 2014 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.

Practice: Movie search

Services

- A Service is an application component that can perform **long-running operations in the background** and does not provide a user interface. Another application component can start a service and it will continue to run in the background even if the user switches to another application.
- Additionally, a component can bind to a service to interact with it and even perform interprocess communication (IPC).
- For example, a service might handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background.

© Copyright 2014 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.

Services

A service can essentially take two forms:

- **Started**: a service is "started" when an application component (such as an activity) starts it by calling startService(). **Once started, a service can run in the background indefinitely, even if the component that started it is destroyed**. Usually, a started service performs a single operation and does not return a result to the caller. For example, it might download or upload a file over the network. When the operation is done, the service should stop itself.

© Copyright 2014 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.

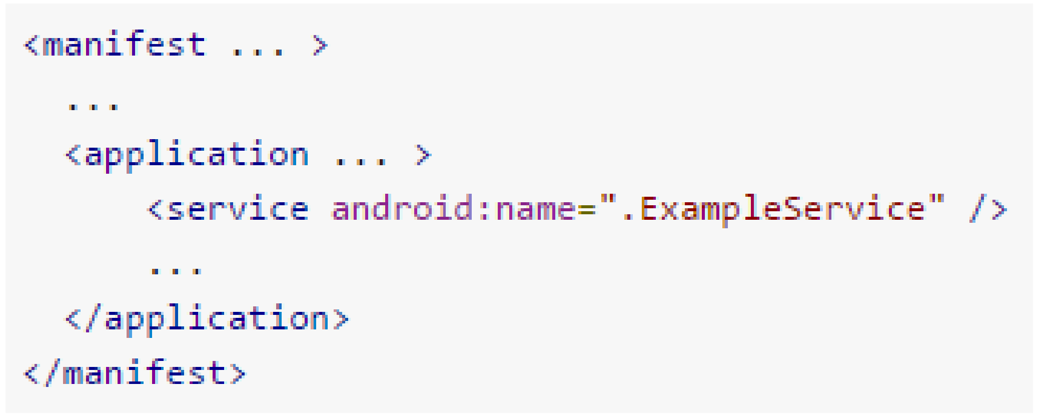
Services

- **Bound:** A service is "bound" when an application component binds to it by calling bindService(). A bound service offers a client-server interface that **allows components to interact with the service**, send requests, get results, and even do so across processes with interprocess communication (IPC). **A bound service runs only as long as another application component is bound to it.** Multiple components can bind to the service at once, but when all of them unbind, the service is destroyed (unless the service was also started by startService()).

© Copyright 2014 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.

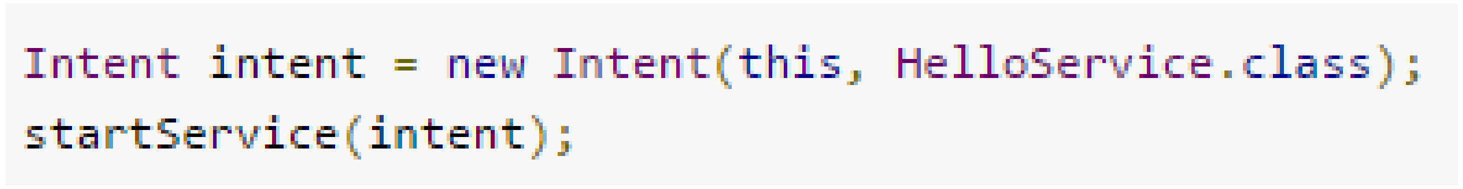
Services

- A service should be declared in the manifest:



```
<manifest ... >
  ...
  <application ... >
      <service android:name=".ExampleService" />
      ...
  </application>
</manifest>
```

- Starting a service:



```
Intent intent = new Intent(this, HelloService.class);
startService(intent);
```

© Copyright 2014 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.

Services

```java
public class HelloIntentService extends IntentService {
  public HelloIntentService() {
      super("HelloIntentService");
  }

  /**
   * The IntentService calls this method from the default worker thread with
   * the intent that started the service. When this method returns, IntentService
   * stops the service, as appropriate.
   */
  @Override
  protected void onHandleIntent(Intent intent) {
      // Normally we would do some work here, like download a file.
      // For our sample, we just sleep for 5 seconds.
      long endTime = System.currentTimeMillis() + 5*1000;
      while (System.currentTimeMillis() < endTime) {
          synchronized (this) {
              try {
                  wait(endTime - System.currentTimeMillis());
              } catch (Exception e) {
              }
          }
      }
  }
}
```

© Copyright 2014 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.

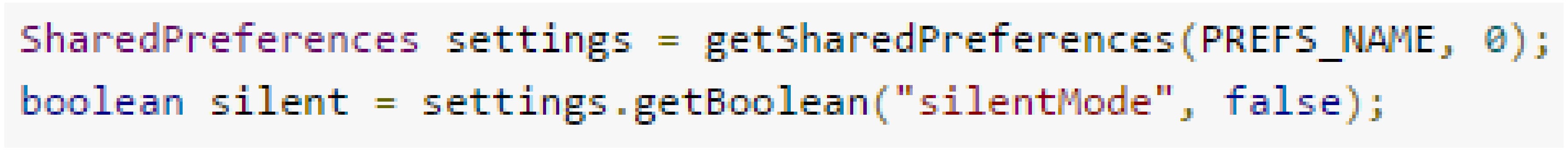
Storage - options

- Shared Preferences
- Internal Storage
- External Storage
- SQLite Databases
- Network Conneciton

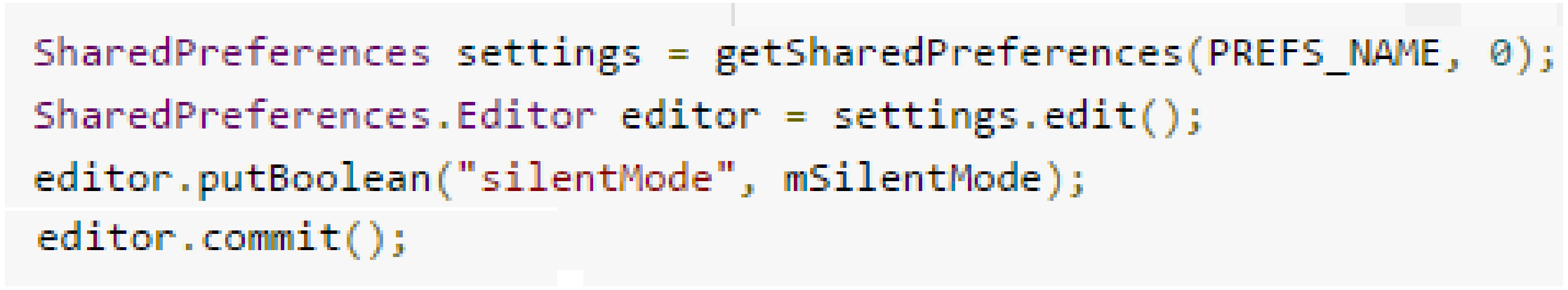
© Copyright 2014 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.

SharedPreferences

- The SharedPreferences class provides a general framework that allows you to save and retrieve persistent key-value pairs of primitive data types. You can use SharedPreferences to save any primitive data: booleans, floats, ints, longs, and strings. This data will persist across user sessions (even if your application is killed).



```
SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
boolean silent = settings.getBoolean("silentMode", false);
```

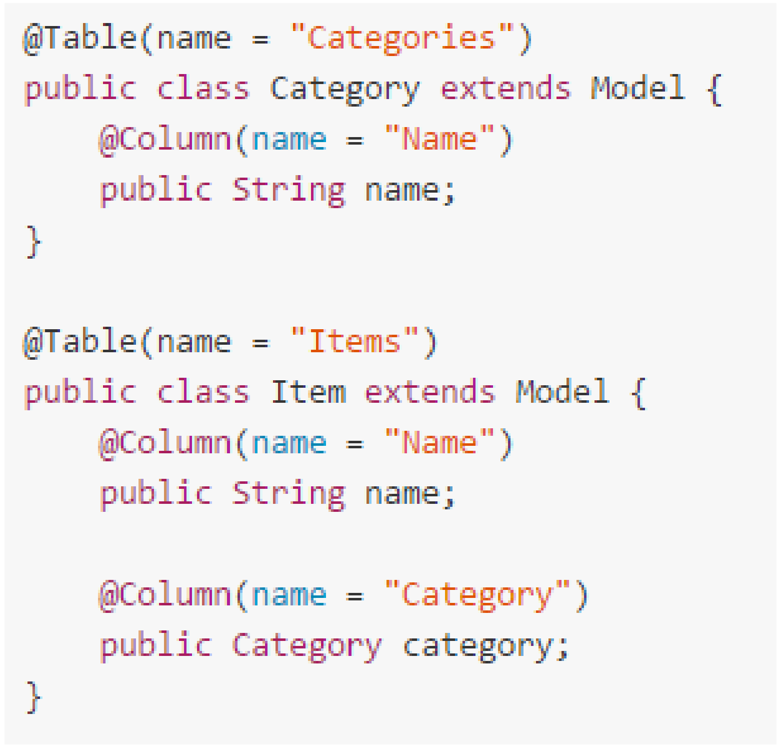


```
SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
SharedPreferences.Editor editor = settings.edit();
editor.putBoolean("silentMode", mSilentMode);
editor.commit();
```

© Copyright 2014 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.

Sqlite – ActiveAndroid

- ActiveAndroid is an active record style ORM.
- Define your model:



```
@Table(name = "Categories")
public class Category extends Model {
    @Column(name = "Name")
    public String name;
}

@Table(name = "Items")
public class Item extends Model {
    @Column(name = "Name")
    public String name;

    @Column(name = "Category")
    public Category category;
}
```

© Copyright 2014 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.

Sqlite – ActiveAndroid

- Save Models:

```
Category restaurants = new Category();
restaurants.name = "Restaurants";
restaurants.save();
Item item = new Item();
item.category = restaurants;
item.name = "Outback Steakhouse";
item.save();
```

- Query Models

```
public static List<Item> getAll(Category category) {
    return new Select()
        .from(Item.class)
        .where("Category = ?", category.getId())
        .orderBy("Name ASC")
        .execute();
}
```

© Copyright 2014 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.

Practice:
Save movie favorites

Thank you

Lucio Cossio

Luis Mazoni

© Copyright 2014 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.