Data Persistence

Persisting Data

- Persisting data is an important topic in application development, as users typically expect to reuse data in the future.
- For Android, there are primarily three basic ways of persisting data:
  - A lightweight mechanism known as shared preferences to save small chunks of data
  - Traditional file systems
  - A relational database management system through the support of SQLite databases

Saving And Loading User Preferences

- Android provides the SharedPreferences object to help you save simple application data.
- For example, your application may have an option that enables users to specify the font size of the text displayed in your application.
- In this case, your application needs to remember the size set by the user so that the next time he or she uses the application again, it can set the size appropriately.

Saving And Loading User Preferences

- In order to do so, you have several options :

1. You can save the data to a file, but you have to perform some file management routines, such as writing the data to file.
2. if you have several pieces of information to save, such as text size, font name, preferred background color, and so on, then the task of writing to a file becomes more onerous.

Saving And Loading User Preferences

- In order to do so, you have several options :

2. Use a database, but saving simple data to a database is overkill, both from a developer's point of view and in terms of the application's run-time performance.
3. Using the SharedPreferences object, however, you save the data you want through the use of name/value p airs — specify a name for the data you want to save, and then both it and its value will be saved automatically to an XML file for you.

Accessing Preferences Using an Activity

- In the following Try It Out, you learn how to use the SharedPreferences object to store application data.
- You will also learn how the stored application data can be modified directly by the user through a special type of activity provided by the Android OS.

Accessing Preferences Using an Activity

- Using Android Studio, create an Android project and name it UsingPreferences.
- Create a new subfolder in the res folder and name it xml. In this newly created folder, add a fi le and name it myapppreferences.xml

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory android:title="Category 1">
        <CheckBoxPreference
            android:title="Checkbox"
            android:defaultValue="false"
            android:summary="True or False"
            android:key="checkboxPref" />
        </PreferenceCategory>
    <PreferenceCategory android:title="Category 2">
        <EditTextPreference
            android:summary="Enter a string"
            android:defaultValue="[Enter a string here]"
            android:title="Edit Text"
            android:key="editTextPref" />
        <RingtonePreference
            android:summary="Select a ringtone"
            android:title="Ringtones"
            android:key="ringtonePref" />
        <PreferenceScreen
            android:title="Second Preference Screen"
```

Accessing Preferences Using

```
            android:summary=
                "Click here to go to the second Preference Screen"
            android:key="secondPrefScreenPref" >
            <EditTextPreference
                android:summary="Enter a string"
                android:title="Edit Text (second Screen)"
                android:key="secondEditTextPref" />
        </PreferenceScreen>
    </PreferenceCategory>
</PreferenceScreen>
```

Accessing Preferences Using an Activity

In the preceding snippet, you created the following:

- Two preference categories for grouping different types of preferences
- Two checkbox preferences with keys named checkboxPref and secondEditTextPref
- A ringtone preference with a key named ringtonePref
- A preference screen to contain additional preferences

The android:key attribute specifies the key that you can programmatically reference in your code to set or retrieve the value of that particular preference.

The android:key attribute specifies the key that you can

Accessing Preferences Using an Activity

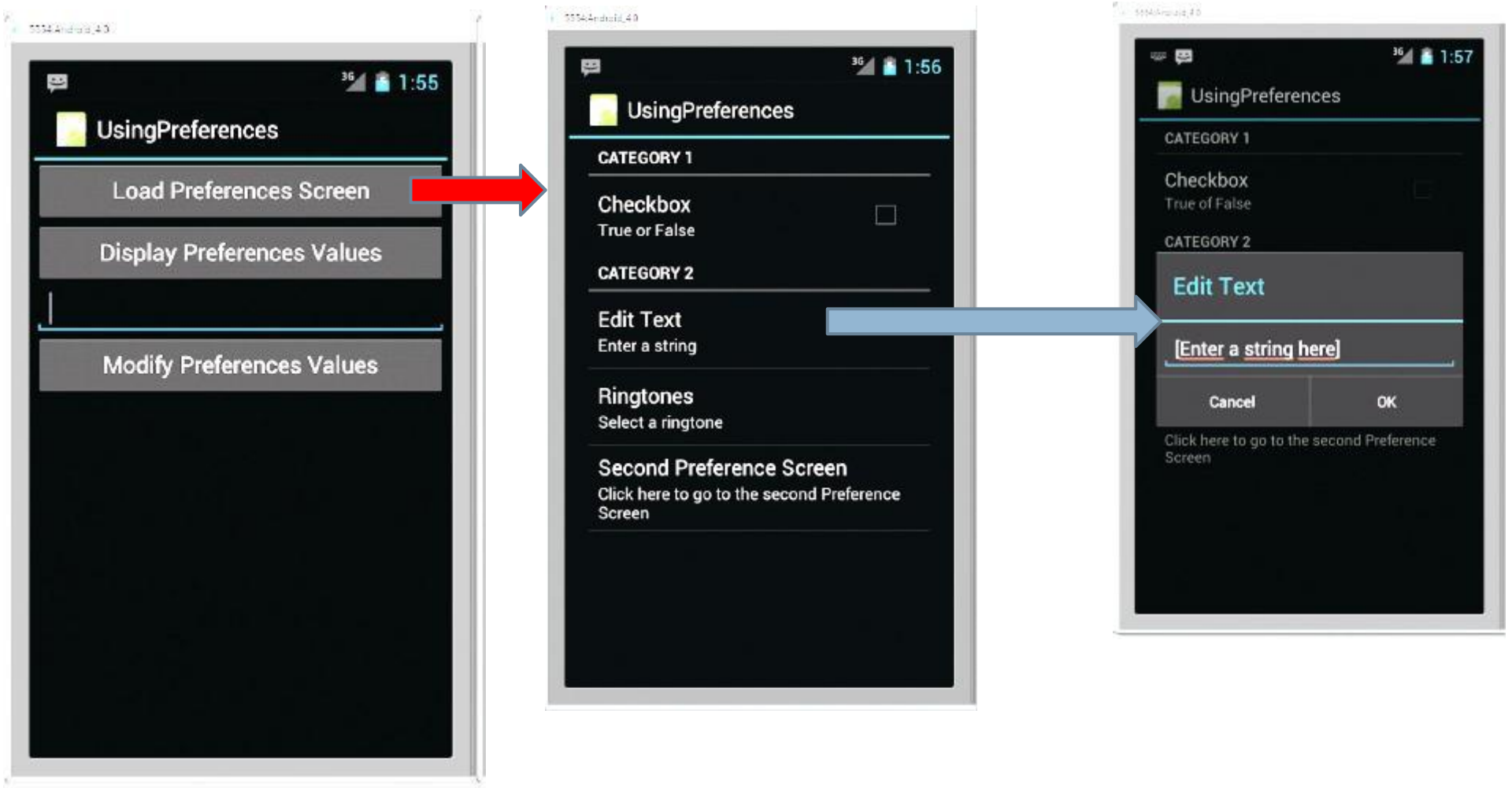
- To get the OS to display all these preferences for users to edit, you create an activity that extends the PreferenceActivity base class,
- and then call the addPreferencesFromResource() method to load the XML file containing the preferences:

Accessing Preferences Using an Activity

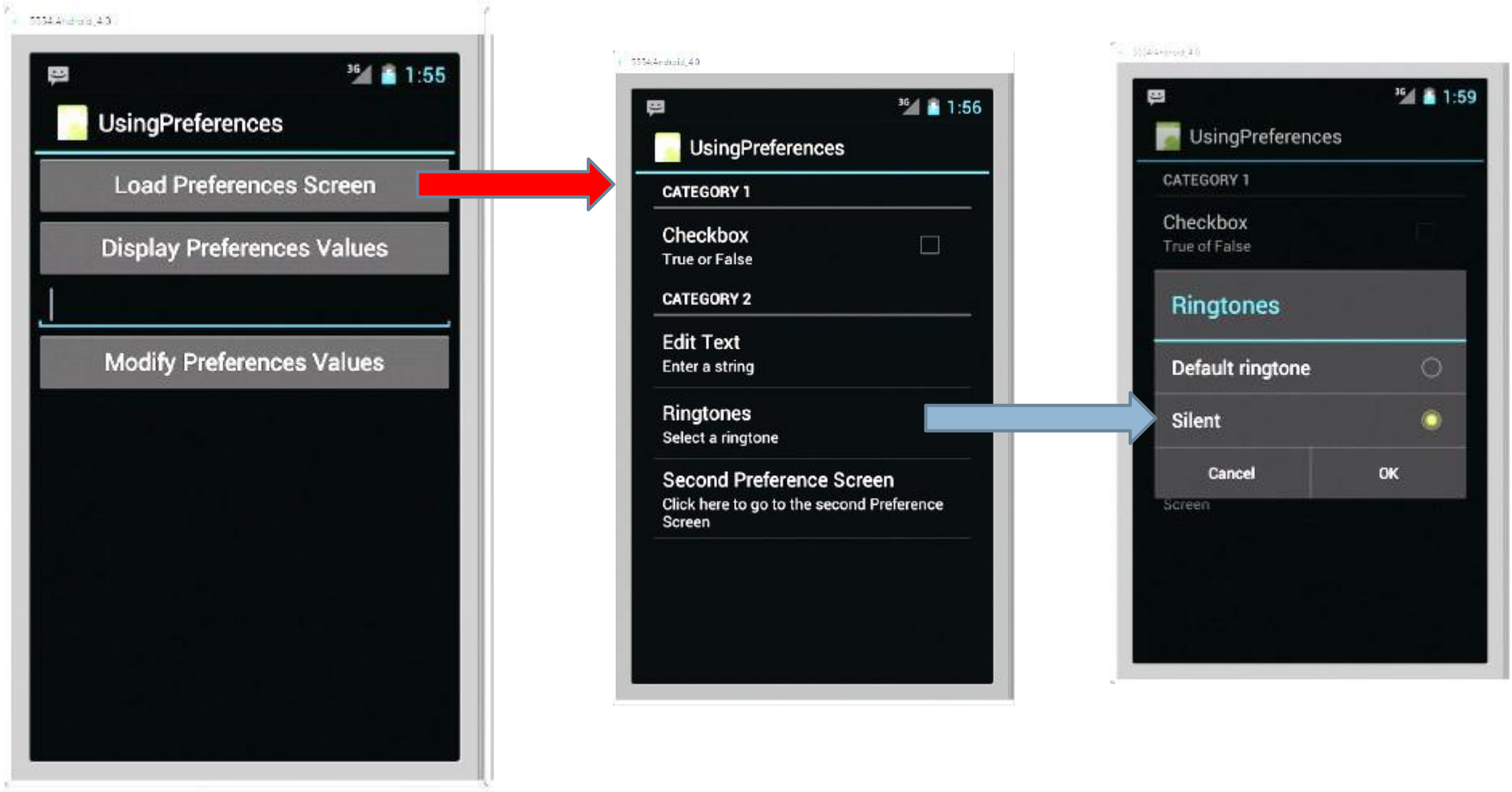
```
import android.os.Bundle;
import android.preference.PreferenceActivity;

public class AppPreferenceActivity extends PreferenceActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //---load the preferences from an XML file---
        addPreferencesFromResource(R.xml.myapppreferences);
    }
}
```

Accessing Preferences Using an Activity



Accessing Preferences Using an Activity



Accessing Preferences Using an Activity

- To get the OS to display all these preferences for users to edit, you create an activity that extends the PreferenceActivity base class,
- and then call the addPreferencesFromResource() method to load the XML fi le containing the preferences:
- All the changes made to the preferences are automatically persisted to an XML file in the shared preferences folder of the application.

Accessing Preferences Using an Activity

```
public class UsingPreferencesActivity extends Activity {

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

    public void onClickLoad(View view) {
        Intent i = new Intent("net.learn2develop.AppPreferenceActivity");
        startActivity(i);
    }
}
```

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

<Button
    android:id="@+id/btnPreferences"
    android:text="Load Preferences Screen"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:onClick="onClickLoad"/>

<Button
    android:id="@+id/btnDisplayValues"
    android:text="Display Preferences Values"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:onClick="onClickDisplay"/>

<EditText
    android:id="@+id/txtString"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />

<Button
    android:id="@+id/btnModifyValues"
    android:text="Modify Preferences Values"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:onClick="onClickModify"/>

</LinearLayout>
```

Programmatically Retrieving and Modifying the Preferences Values

- In the previous section, you saw how the PreferenceActivity class both enables developers to easily create preferences and enables users to modify them during runtime.
- To make use of these preferences in your application, you use the SharedPreferences class.

Programmatically Retrieving and Modifying the Preferences Values

```
public void onClickDisplay(View view) {

    SharedPreferences appPrefs =
            getSharedPreferences("net.learn2develop.UsingPreferences_preferences",
                    MODE_PRIVATE);
    DisplayText(appPrefs.getString("editTextPref", ""));
}

public void onClickModify(View view) {
    SharedPreferences appPrefs =
            getSharedPreferences("net.learn2develop.UsingPreferences_preferences",
                    MODE_PRIVATE);
    SharedPreferences.Editor prefsEditor = appPrefs.edit();
    prefsEditor.putString("editTextPref",
            ((EditText) findViewById(R.id.txtString)).getText().toString());
    prefsEditor.commit();
}

private void DisplayText(String str) {
    Toast.makeText(getBaseContext(), str, Toast.LENGTH_LONG).show();
}
}
```

Programmatically Retrieving and Modifying the Preferences Values

- In the onClickDisplay() method, you first used the getSharedPreferences() method to obtain an instance of the SharedPreferences class.
- You do so by specifying the name of the XML file in this case it is:

“com.smd.sharedpreferences.preferences” using the format: <PackageName>.preferences.

Programmatically Retrieving and Modifying the Preferences Values

- To retrieve a string preference, you used the getString() method .
- passing it the key to the preference that you want to retrieve .

```
appPrefs.getString("editTextPref", "")
```

The MODE_PRIVATE constant indicates that the preference file can only be opened by the application that created it.

Programmatically Retrieving and Modifying the Preferences Values

- In the onClickModify() method, you created a SharedPreferences.Editor object through the edit() method of the SharedPreferences object.
- SharedPreferences.Editor prefsEditor = appPrefs.edit();
- To change the value of a string preference, use the putString() method.
- To save the changes to the preferences file, use the commit() method: