

*Lesson 14*


# Android Persistency: Preferences

Victor Matos  
Cleveland State University

Notes are based on:  
 The Busy Coder's Guide to Android Development  
 by Mark L. Murphy  
 Copyright © 2008-2009 CommonsWare, LLC.  
 ISBN: 978-0-9816780-0-9  
 &  
 Android Developers  
<http://developer.android.com/index.html>

Portions of this page are reproduced from work created and [shared by Google](#) and used according to terms described in the [Creative Commons 3.0 Attribution License](#).

# Android Data Storage



Android offers a number of options for the storage of your datasets:

<b>Shared Preferences</b>	Good for private small datasets in key-value pairs.
<b>Internal Storage</b>	Store private data on the device memory.
<b>External Storage</b>	Store public data on the shared external storage.
<b>SQLite Databases</b>	Store structured data in a public database.
<b>Network Connection</b>	Store data on the web with your own network server.
<b>Content Provider</b>	Shared repository globally shared by all apps.

REFERENCE: <http://developer.android.com/guide/topics/data/data-storage.html>

2

## Android Data Storage

Android uses a particular data sharing scheme:



*On Android, all application data held in the device's private memory area is **private** to that application*

**Note:**

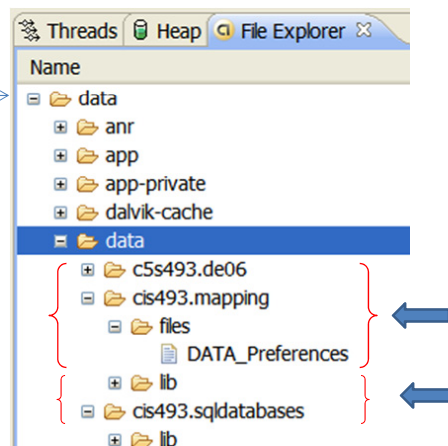
Remember private memory is usually smaller and faster than external storage (SDcards).

<http://developer.android.com/guide/topics/data/data-storage.html>

3

## Android Data Storage

*On Android, all internal application data objects are (including files) private to that application.*

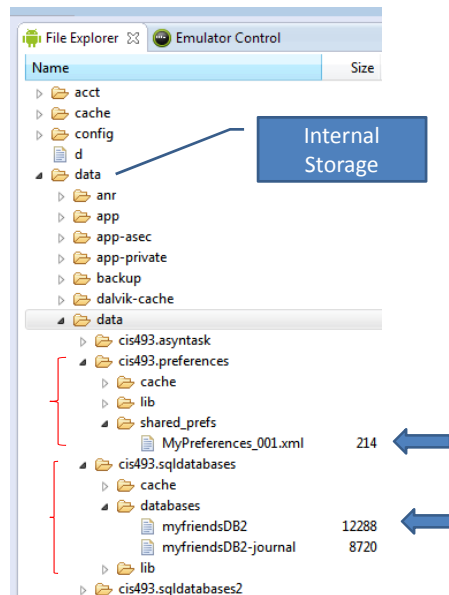


<http://developer.android.com/guide/topics/data/data-storage.html>

4

## Android Data Storage

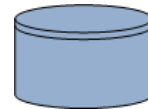
*On Android, all internal application data objects are (including files) private to that application.*



<http://developer.android.com/guide/topics/data/data-storage.html>

5

## Android Data Storage



### Content Providers

- Provide a data-interface for non-SQL developers.
- Users can invoke API methods to retrieve and update data.
- Android uses content providers for global data objects, such as
  - *image,*
  - *audio,*
  - *video files and*
  - *personal contact information.*

6

# Preferences

KEY	VALUE

## Preferences

is an Android *lightweight* mechanism to store and retrieve **<Key-Value>** pairs of primitive data types (also called *Maps*, and *Associative Arrays*).

*PREFERENCES* are typically used to keep state information and shared data among several activities of an application.

In each entry of the form **<key-value>** the *key* is a string and the *value* must be a primitive data type.

Preferences are similar to Bundles however they are **persistent** while Bundles are not.



7

# Preferences

KEY	VALUE

## Using Preferences API calls

You have three API choices to pick a Preference:

1. **getPreferences()** retrieve a specific *private* preferences file.
2. **getSharedPreferences()** retrieve an *application-level* preferences.

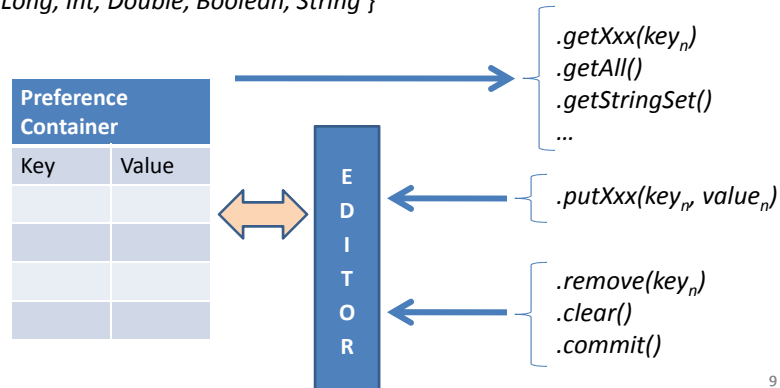
8

## Preferences

### Using Preferences API calls

All of the **getXXX** Preference methods return a Preference object whose contents can be manipulated by an *editor* that allows *putXXX...* and *getXXX...* commands to place data in and out of the Preference container.

$Xxx = \{ Long, Int, Double, Boolean, String \}$



9

## Preferences

### Example 1

1. In this example a persistent *SharedPreferences* object is created at the end of an activity lifecycle. It contains some *formatting* specifications made by the user to define aspects of the graphical interface.
2. When re-executed, it finds the saved *Preference* and uses its persistent data to reproduce the UI according to the specifications previously given by the user.

#### Warning

Make sure you test from a 'fresh' configuration. If necessary use DDMS tool and *delete* existing Preferences held in the application's name-space.

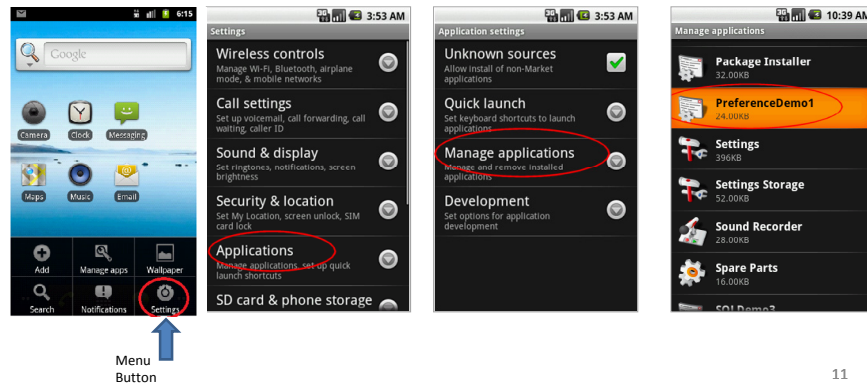
10

# Preferences

## Example 1

### Warning

Make sure you test from a 'fresh' configuration. Next images illustrate the process of removing existing traces of an application from the phone's system area using device's Application Manager

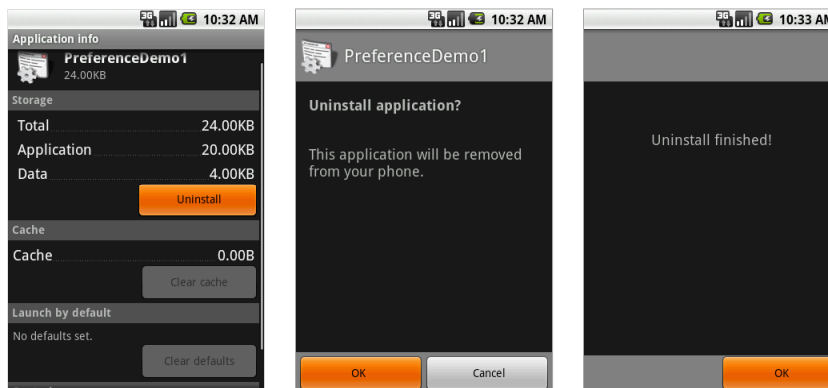


# Preferences

## Example 1. cont.

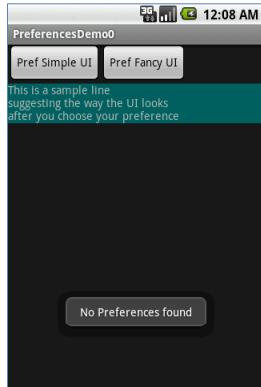
### Warning

Make sure you test from a 'fresh' configuration. Next images illustrate the process of removing existing traces of an application from the phone's system area using device's Application Manager

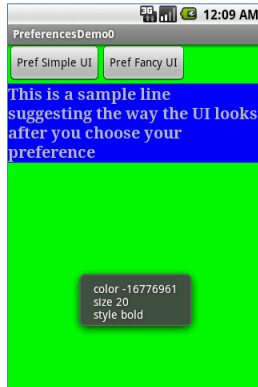


# Preferences

**Example 1:** Saving/Retrieving a SharedPreferences Object holding UI user choices.



Initial UI with no choices made/save yet.



Images of the choices made by the user regarding the looks of the UI. The 'green screen' corresponds to the fancy layout, the 'grey screen' is the simple choice. Data is saved into the SharedPreferences object: *myPreferences\_001*.

# Preferences

**Example 1:** Saving/Retrieving a SharedPreferences Object

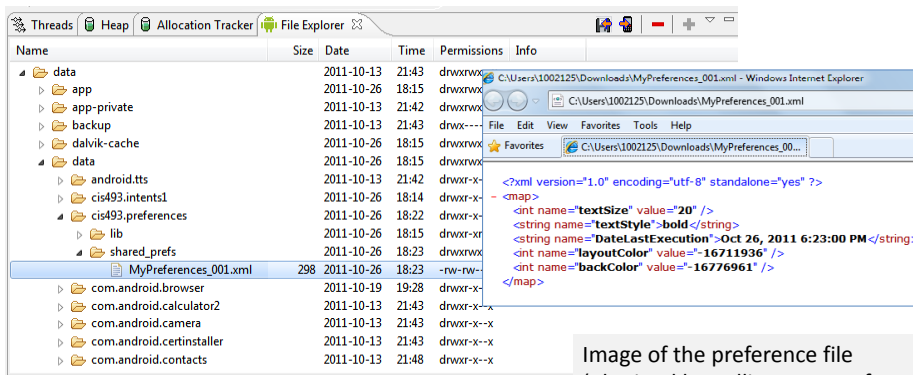


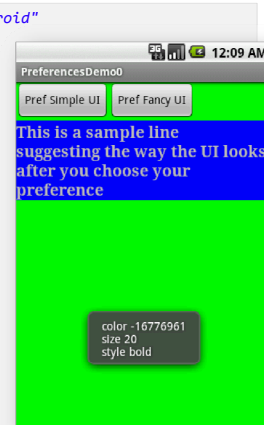
Image of the preference file (obtained by pulling a copy of the file out of the device).

Using DDMS to explore the Device's memory map. Observe the choices made by the user are saved in the *data/data/Shared\_prefs/* folder as an XML file.

# Preferences

## Example 1: Saving/Retrieving a SharedPreferences Object

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/LinLayout1Vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <LinearLayout
        android:id="@+id/LinLayout2Horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <Button
            android:id="@+id/btnPrefSimple"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Pref Simple UI" />
        <Button
            android:id="@+id/btnPrefFancy"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Pref Fancy UI" />
    </LinearLayout>
    <TextView
        android:id="@+id/txtCaption1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff066666"
        android:text="This is some sample text " />
</LinearLayout>
```



15

# Preferences

## Example1: Saving/Retrieving a SharedPreferences Object

```
package cis493.preferences;
import ...

public class PreferenceDemo0 extends Activity implements OnClickListener {
    Button btnSimplePref;
    Button btnFancyPref;
    TextView txtCaption1;
    Boolean fancyPrefChosen = false;
    View myLayout1Vertical;

    final int mode = Activity.MODE_PRIVATE;
    final String MYPREFS = "MyPreferences_001";

    // create a reference to the shared preferences object
    SharedPreferences mySharedPreferences;

    // obtain an editor to add data to my SharedPreferences object
    SharedPreferences.Editor myEditor;
```

File creation modes:  
MODE\_APPEND  
MODE\_

16



# Preferences

## Example1: Saving/Retrieving a SharedPreferences Object

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    myLayout1Vertical = (View) findViewById(R.id.linLayout1Vertical);
    txtCaption1 = (TextView) findViewById(R.id.txtCaption1);
    txtCaption1.setText("This is a sample line \n"
        + "suggesting the way the UI looks \n"
        + "after you choose your preference");

    // create a reference & editor for the shared preferences object
    mySharedPreferences = getSharedPreferences(MYPREFS, 0);
    myEditor = mySharedPreferences.edit();
    // has a Preferences file been already created?
    if (mySharedPreferences != null
        && mySharedPreferences.contains("backColor")) {
        // object and key found, show all saved values
        applySavedPreferences();
    } else {
        Toast.makeText(getApplicationContext(),
            "No Preferences found", 1).show();
    }
    btnSimplePref = (Button) findViewById(R.id.btnPrefSimple);
    btnSimplePref.setOnClickListener(this);
    btnFancyPref = (Button) findViewById(R.id.btnPrefFancy);
    btnFancyPref.setOnClickListener(this);
} // onCreate

```

# Preferences

## Example1: Saving/Retrieving a SharedPreferences Object

```

@Override
public void onClick(View v) {
    // clear all previous selections
    myEditor.clear();

    // what button has been clicked?
    if (v.getId() == btnSimplePref.getId()) {
        myEditor.putInt("backColor", Color.BLACK); // black background
        myEditor.putInt("textSize", 12); // humble small font
    } else { // case btnFancyPref
        myEditor.putInt("backColor", Color.BLUE); // fancy blue
        myEditor.putInt("textSize", 20); // fancy big
        myEditor.putString("textStyle", "bold"); // fancy bold
        myEditor.putInt("layoutColor", Color.GREEN); // fancy green
    }
    myEditor.commit();
    applySavedPreferences();
}

```

# Preferences

## Example1: Saving/Retrieving a SharedPreferences Object

```
@Override
protected void onPause() {
    // warning: activity is on its last state of visibility!.
    // It's on the edge of being killed! Better save all current
    // state data into Preference object (be quick!)
    myEditor.putString("DateLastExecution", new Date().toLocaleString());
    myEditor.commit();
    super.onPause();
}
```

19

# Preferences

## Example1: Saving/Retrieving a SharedPreferences Object

```
public void applySavedPreferences() {
    // extract the <key/value> pairs, use default param for missing data
    int backColor = mySharedPreferences.getInt("backColor", Color.BLACK);
    int textSize = mySharedPreferences.getInt("textSize", 12);
    String textStyle = mySharedPreferences.getString("textStyle", "normal");
    int layoutColor = mySharedPreferences.getInt("layoutColor", Color.DKGRAY);
    String msg = "color " + backColor + "\n"
        + "size " + textSize + "\n"
        + "style " + textStyle;
    Toast.makeText(getApplicationContext(), msg, 1).show();

    txtCaption1.setBackgroundColor(backColor);
    txtCaption1.setTextSize(textSize);
    if (textStyle.compareTo("normal")==0){
        txtCaption1.setTypeface(Typeface.SERIF, Typeface.NORMAL);
    }
    else {
        txtCaption1.setTypeface(Typeface.SERIF, Typeface.BOLD);
    }
    myLayout1Vertical.setBackgroundColor(layoutColor);
} // applySavedPreferences

} // class
```

20

## Preferences

### Example 2

1. In this example a persistent *SharedPreferences* object is created at the end of an activity lifecycle. It contains data (name, phone, credit, etc. of a fictional customer)
2. The process is interrupted using the “*Back Button*” and re-executed later.
3. Just before been killed, the state of the running application is saved in the designated *Preference* object.
4. When re-executed, it finds the saved *Preference* and uses its persistent data.

#### Warning

Make sure you test from a ‘fresh’ configuration. If necessary use DDMS and *delete* existing Preferences held in the application’s name-space.

21

## Preferences

### Example 2: Saving/Retrieving a SharedPreferences Object containing ‘business’ data.

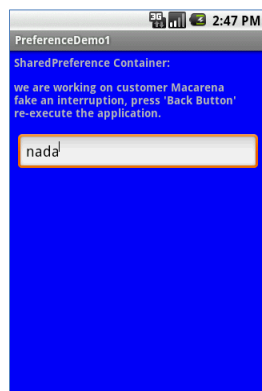


Image of the data held in the SharedPreferences object displayed the first time the Activity **Preferences1** is executed.

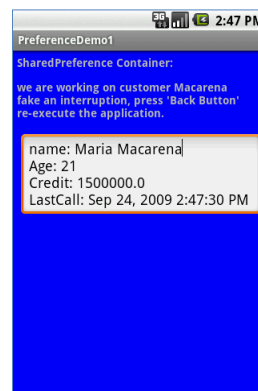


Image of the saved Preference data displayed the second time the Activity **Preferences1** is executed.

22

# Preferences

Example2: Saving/Retrieving a SharedPreferences Object

The screenshot shows the Eclipse IDE interface. The 'DDMS' window is open, and the 'File Explorer' tab is active. It displays a file tree for the application. The file 'MySharedPreferences001.xml' is highlighted. A blue arrow points from the top of the slide to the File Explorer window. Another blue arrow points from the right side of the slide to the 'MySharedPreferences001.xml' file. The console at the bottom shows log output from the emulator.

Use DDMS to see persistent data set

# Preferences

Example2: Saving/Retrieving a SharedPreferences Object

The screenshot shows a Windows Internet Explorer browser window. The address bar shows the file path 'C:\Downloads\MySharedPreferences001.xml'. The main content area displays the XML content of the file. A blue arrow points from the right side of the slide to the address bar. A text box on the right explains that the data is saved in the phone's memory as an XML file and was pulled from the device using DDMS.

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
- <map>
  <string name="custDateLastCall">Sep 24, 2009 2:47:30 PM</string>
  <int name="custAge" value="21" />
  <float name="custCredit" value="1500000.0" />
  <long name="custNumber" value="9876543210" />
  <string name="custName">Maria Macarena</string>
</map>
```

Persistent data is saved in the phone's memory as an XML file. This image was pulled from the device using DDMS.

# Preferences

## Example2: Saving/Retrieving a SharedPreferences Object

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:id="@+id/linLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ff0000ff"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <TextView
        android:id="@+id/captionBox"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="SharedPreferences Container: Customer Data"
        android:layout_margin="5px" android:textStyle="bold">
    </TextView>
    <EditText
        android:id="@+id/txtPref"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10px"
        >
    </EditText>
</LinearLayout>
```

25

# Preferences

## Example2: Saving/Retrieving a SharedPreferences Object

```
package cis493.preferences;

import java.util.Date;

import android.app.Activity;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.widget.*;

public class Preference1 extends Activity {
    public static final String MYPREFS = "MySharedPreferences001";
    //this data values describe a typical customer record
    String custName = "n.a.";
    int custAge = 0;
    float custCredit = 0;
    long custNumber = 0;
    String custDateLastCall;

    TextView captionBox;
    EditText txtPref;
    final int mode = Activity.MODE_PRIVATE;
```

26

# Preferences

## Example2: Saving/Retrieving a SharedPreferences Object

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    txtPref = (EditText) findViewById(R.id.txtPref);
    captionBox = (TextView) findViewById(R.id.captionBox);
    captionBox.setText("SharedPreferences Container: \n\n"+
        "we are working on customer Macarena \n" +
        "fake an interruption, press 'Back Button' \n" +
        "re-execute the application.");

    //create a reference to the shared preferences object
    int mode = Activity.MODE_PRIVATE;
    SharedPreferences mySharedPreferences = getSharedPreferences(MYPREFS, mode);
    //is there an existing Preferences from previous executions of this app?
    if (mySharedPreferences != null &&
        mySharedPreferences.contains("custName")) {
        //object and key found, show all saved values
        showSavedPreferences();
    }
    else
    {
        txtPref.setText("nada");
    }
}
} //onCreate

```

# Preferences

## Example2: Saving/Retrieving a SharedPreferences Object

```

@Override
protected void onPause() {
    //warning: activity is on last state of visibility! We are on the
    //edge of been killed! Better save current state in Preference object
    savePreferences();
    super.onPause();
}

protected void savePreferences() {
    //create the shared preferences object
    SharedPreferences mySharedPreferences =
        getSharedPreferences(MYPREFS, mode);

    //obtain an editor to add data to (my)SharedPreferences object
    SharedPreferences.Editor myEditor = mySharedPreferences.edit();

    //put some <key/value> data in the preferences object
    myEditor.putString("custName", "Maria Macarena");
    myEditor.putInt("custAge", 21);
    myEditor.putFloat("custCredit", 1500000.00F);
    myEditor.putLong("custNumber", 9876543210L);
    myEditor.putString("custDateLastCall", new Date().toLocaleString());
    myEditor.commit();
} //savePreferences

```

## Preferences

### Example2: Saving/Retrieving a SharedPreferences Object

```
public void showSavedPreferences() {
    //retrieve the SharedPreferences object

    SharedPreferences mySharedPreferences =
        getSharedPreferences(MYPREFS, mode);

    //extract the <key/value> pairs, use default param for missing data
    custName = mySharedPreferences.getString("custName", "defNameValue");
    custAge = mySharedPreferences.getInt("custAge", 18);
    custCredit = mySharedPreferences.getFloat("custCredit", 1000.00F);
    custNumber = mySharedPreferences.getLong("custNumber", 1L);
    custDateLastCall = mySharedPreferences.getString("custDateLastCall",
        new Date().toLocaleDateString());

    //show saved data on screen
    String msg = "name: " + custName + "\nAge: " + custAge +
        "\nCredit: " + custCredit +
        "\nLastCall: " + custDateLastCall;
    txtPref.setText(msg);
} //loadPreferences

} //Preferences1
```

## Preferences

## Questions ?