

Mobile Application Development

Produced
by

David Drohan (ddrohan@wit.ie)

Department of Computing & Mathematics
Waterford Institute of Technology

<http://www.wit.ie>



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE





Android Persistence using Shared Preferences





Agenda & Goals

- ❑ Be aware of the different approaches to data persistence in Android Development
- ❑ Be able to work with the **SQLiteOpenHelper** & **SQLiteDatabase** classes to implement an SQLite database on an Android device (to manage our Donations)
- ❑ Be able to work with **Realm** to implement a noSQL database on an Android device (again, to manage our Donations)
- ❑ Be able to work with **SharedPreferences** to manage our Login & Register screens



Main Idea – why do we need Persistence?

- ❑ Android can shut down and restart your app
 - When you rotate the screen
 - When you change languages
 - When your app is in background and Android is short on memory
 - When you hit the Back button
- ❑ Problem
 - You risk losing user changes and data
- ❑ Solutions ??



Solutions

- ❑ Android provides several options for you to save persistent application data.
- ❑ The solution you choose depends on your specific needs, such as whether the data should be private to your application or accessible to other applications (and the user) and how much space your data requires.
- ❑ Android provides a way for you to expose your private data to other applications — with a **Content Provider**.
 - A content provider is an optional component that exposes read/write access to your application data, subject to whatever restrictions you want to impose.



Data Storage Solutions *

❑ Shared Preferences

- Store private primitive data in key-value pairs.

❑ Internal Storage

- Store private data on the device memory.

❑ External Storage

- Store public data on the shared external storage.

❑ SQLite Databases

- Store structured data in a private database.

❑ Network Connection

- Store data on the web with your own network server.



Data Storage Solutions *

❑ Bundle Class

- A mapping from String values to various **Parcelable** types and functionally equivalent to a standard **Map**.
- Does not handle Back button scenario. App restarts from scratch with no saved data in that case.

❑ File

- Use **java.io.*** to read/write data on the device's internal storage.

❑ Realm Databases

- Store non-structured data in a private database.

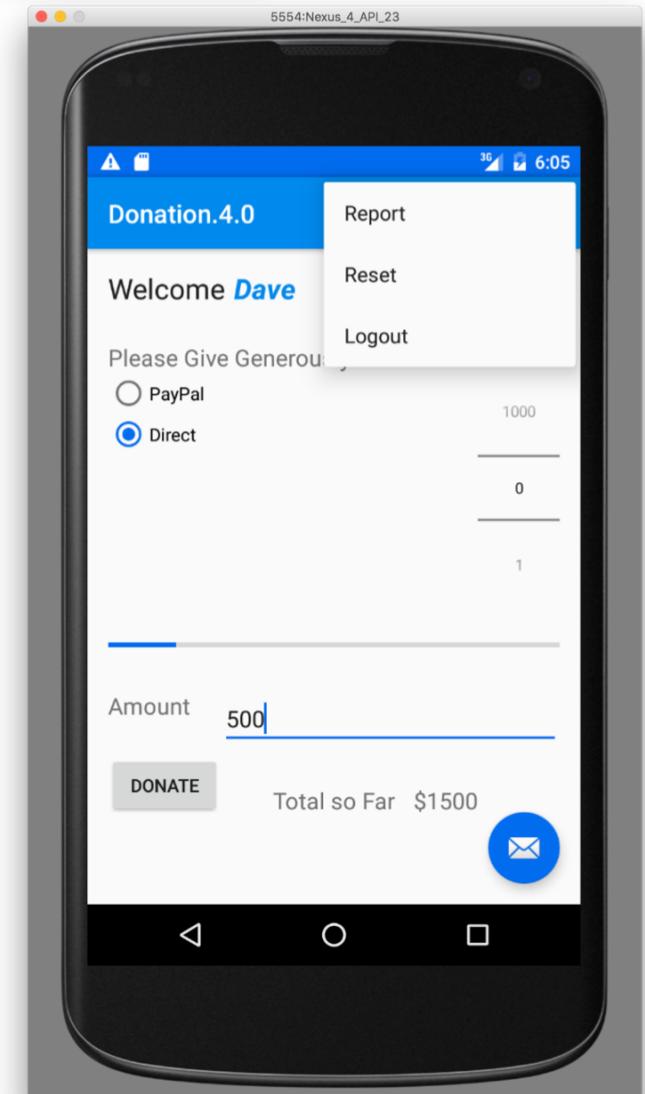


Case Study

□ **Donation** – an Android App to keep track of donations made to ‘*Homers Presidential Campaign*’.

□ App Features

- Accept donation via number picker or typed amount
- Keep a running total of donations
- Display report on donation amounts and types
- Display running total on progress bar





Ultimate Case Study – Donation 4.0





Using A Splash Screen & Login Screen / Register Screen



What do we want exactly?



- Display Splash Screen for a few seconds
- Display Login Screen
- Allow User to Register
- Only show Home Screen once valid details entered
- Logout Menu Option
- AND Manage our DB via Application Object



Donation 4.0 - Splash

```
public class Splash extends Activity {  
    // used to know if the back button was pressed in the splash screen activity  
    // and avoid opening the next activity  
    private boolean          mIsBackPressed;  
    private static final int  SPLASH_DURATION = 2000; // 2 seconds  
  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_splash);  
        Handler handler = new Handler();  
        // run a thread after 2 seconds to start the home screen  
        handler.postDelayed(new Runnable() {  
            @Override  
            public void run() {  
                // make sure we close the splash screen so the user  
                // won't come back when it presses back key  
                finish();  
  
                if (!mIsBackPressed) {  
                    // start the home screen if the back button wasn't pressed already  
                    Splash.this.startActivity(new Intent(Splash.this, Login.class));  
                }  
            }  
        }, SPLASH_DURATION); // time in milliseconds to delay call to run()  
    }  
  
    @Override  
    public void onBackPressed() {  
        // set the flag to true so the next activity won't start up  
        mIsBackPressed = true;  
    }  
}
```

Handler object
associated with single
thread

Start Login Screen via
Intent



Update Manifest File

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ie.app" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Donation.4.0"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        android:name="ie.app.main.DonationApp">

        <activity
            android:name=".activities.Splash"
            android:configChanges="orientation|keyboardHidden"
            android:screenOrientation="portrait"
            android:theme="@android:style/Theme.NoTitleBar" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity...>
        <activity android:name=".activities.Report"
            android:label="Donation.4.0">
        </activity>
        <activity android:name=".activities.Login"
            android:label="Donation.4.0">
        </activity>
    </application>
</manifest>
```

Activity to Launch



Using SharedPreferences



Overview

- ❑ Shared Preferences allows activities and applications to keep preferences, in the form of **key-value pairs** similar to a Map that will persist even when the user closes the application.
- ❑ Android stores Shared Preferences settings as an XML file in **shared_prefs** folder under *DATA/data/{application package}* directory.
- ❑ It's application specific, i.e. the data is lost on uninstalling the application or clearing the application data (through Settings)
- ❑ Primary purpose is to store user-specified configuration details, such as user settings, keeping the user logged in etc.



Key Points to Note

- ❑ Interface for accessing and modifying preference data returned by [getSharedPreferences\(String, int\)](#).
- ❑ For any particular set of preferences, there is a single instance of this class that all clients share.
- ❑ Modifications to the preferences must go through an [SharedPreferences.Editor](#) object to ensure the preference values remain in a consistent state and control when they are committed to storage.
- ❑ Objects that are returned from the various get methods must be treated as immutable by the application.



SharedPreferences

- Three forms:
 - `getPreferences()` : used from within your Activity, to access activity-specific preferences
 - `getSharedPreferences()` : used from within your Activity (or other application Context), to access application-level preferences
 - `getDefaultSharedPreferences()` : used on the *PreferenceManager*, to get the shared preferences that work in concert with Android's overall preference framework



Initialization

- We need an editor to edit and save the changes in shared preferences. The following code can be used to get the shared preferences.

```
SharedPreferences pref = getApplicationContext()
                      .getSharedPreferences("MyPref", 0);
// 0 - for private mode
Editor editor = pref.edit();
```



Storing Data

- ☐ `editor.commit()` is used in order to save changes to shared preferences.

```
editor.putBoolean("key_name", true); // Storing boolean - true/false  
editor.putString("key_name", "string value"); // Storing string  
editor.putInt("key_name", "int value"); // Storing integer  
editor.putFloat("key_name", "float value"); // Storing float  
editor.putLong("key_name", "long value"); // Storing long  
editor.commit(); // commit changes
```



Retrieving Data

- Data can be retrieved from saved preferences by calling `getString()` (or whatever) as follows:

```
pref.getString("key_name", null); // getting String  
pref.getInt("key_name", null); // getting Integer  
pref.getFloat("key_name", null); // getting Float  
pref.getLong("key_name", null); // getting Long  
pref.getBoolean("key_name", null); // getting boolean
```



Clearing or Deleting Data

- ❑ `remove("key_name")` is used to delete that particular value.
- ❑ `clear()` is used to remove all data

```
editor.remove("name"); // will delete key name  
editor.remove("email"); // will delete key email  
  
editor.commit(); // commit changes
```



Using SharedPreferences in Donation.4.0



Donation 4.0 – activity_login.xml

The screenshot shows the Android Studio XML Editor with the following details:

- Device Screen:** Nexus 4, AppTheme.
- Component Tree:**
 - Device Screen
 - ScrollView
 - RelativeLayout
 - header (LinearLayout)
 - ImageView – @drawable/login_top
 - footer (LinearLayout)
 - ImageView – @drawable/login_bottom
 - LinearLayout (vertical)
 - TextView – "Username"
 - loginUsername (EditText)
 - TextView – "Password"
 - loginPassword (EditText)
 - btnLogin (Button) – "Login"
 - register_message (TextView) – "New to Donation? Register here"
- Properties:**

layout:width	fill_parent
layout:height	fill_parent
style	
accessibilityLiveRegion	
accessibilityTraversalAfter	
accessibilityTraversalBefore	
alpha	
background	
backgroundTint	
backgroundTintMode	
clickable	<input type="checkbox"/>
contentDescription	
contextClickable	<input type="checkbox"/>
elevation	



Donation – Login *

```
public class Login extends Activity {  
  
    // used to know if the back button was pressed in the splash screen activity  
    // and avoid opening the next activity  
    private boolean mIsBackPressed;  
    private SharedPreferences settings;  
  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        settings = getSharedPreferences("loginPrefs", 0);  
        if (settings.getBoolean("loggedin", false))  
            /* The user has already logged in, so start the Home Screen */  
            startHomeScreen();  
  
        setContentView(R.layout.activity_login);  
    }  
  
    public void register(View v) { startActivity (new Intent(this, Register.class)); }  
  
    private void startHomeScreen() {  
        Intent intent = new Intent(Login.this, Donate.class);  
        Login.this.startActivity(intent);  
    }  
}
```

Load your Preferences

Default value



Donation – Login *

```
public void login(View v) {  
  
    CharSequence username = ((TextView) findViewById(R.id.loginUsername))  
        .getText();  
    CharSequence password = ((TextView) findViewById(R.id.loginPassword))  
        .getText();  
  
    String validUsername = settings.getString("username", "");  
    String validPassword = settings.getString("password", "");  
  
    if (username.length() <= 0 || password.length() <= 0)  
        Toast.makeText(this, "You must enter an email & password",  
            Toast.LENGTH_SHORT).show();  
    else if (!username.toString().matches(validUsername)  
        || !password.toString().matches(validPassword))  
        Toast.makeText(this, "Unable to validate your email & password",  
            Toast.LENGTH_SHORT).show();  
    else if (!mIsBackPressed) {  
        // Validate User with Server Here  
  
        // Update logged in preferences  
        SharedPreferences.Editor editor = settings.edit();  
        editor.putBoolean("loggedin", true);  
        editor.commit();  
        // start the home screen if the back button wasn't pressed already  
        startHomeScreen();  
        this.finish(); // destroy the Login Activity  
    }  
}
```

Retrieving existing details

Verifying entered details

Update Preferences with data



Donation – Register *

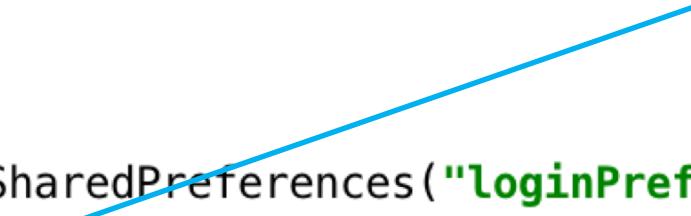
```
public class Register extends Activity {  
    private boolean mIsBackPressed;  
    private SharedPreferences settings;  
  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_register);  
  
        settings = getSharedPreferences("loginPrefs", 0);  
    }  
  
    public void register(View v) {  
        CharSequence username = ((TextView) findViewById(R.id.registerUsername))  
            .getText();  
        CharSequence password = ((TextView) findViewById(R.id.registerPassword))  
            .getText();  
  
        if (username.length() <= 0 || password.length() <= 0)  
            Toast.makeText(this, "You must enter an email & password",  
                Toast.LENGTH_SHORT).show();  
        else if (!mIsBackPressed) {  
            // Update logged in preferences  
            SharedPreferences.Editor editor = settings.edit();  
            editor.putBoolean("loggedin", true);  
            editor.putString("username", username.toString());  
            editor.putString("password", password.toString());  
            editor.commit();  
  
            // start the home screen if the back button wasn't pressed already  
            startHomeScreen();  
            this.finish(); // destroy the Register Activity  
        }  
    }  
}
```

Retrieving existing
prefs file

Update Preferences
with new data



Donation – Logout method (in Base)

```
public void logout(MenuItem item) {  
    SharedPreferences.Editor editor = getSharedPreferences("loginPrefs", 0).edit();  
    editor.putBoolean("loggedin", false);   
    editor.commit();  
  
    startActivity(new Intent(Base.this, Login.class)  
        .setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK));  
    finish();  
}
```

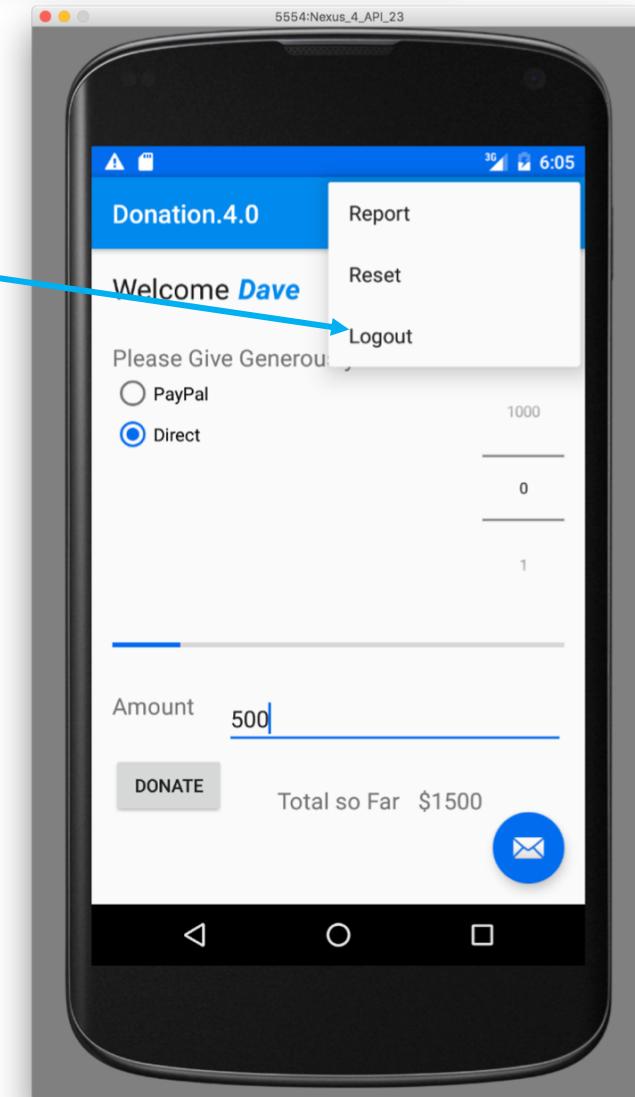
Resetting 'loggedin' to false

Returning to the 'Login' screen



Donation – ‘Logout’ Menu Item

```
<item  
    android:id="@+id/menuLogout"  
    android:title="@string/menuLogout"  
    android:orderInCategory="100"  
    android:showAsAction="never"  
    android:onClick="logout"/>
```





Donation – activity_donate.xml

Nexus 4 | AppTheme | Donate | 23 | Component Tree | Properties

Component Tree

- Shown in @layout/activity_donate
 - RelativeLayout
 - donateTitle (TextView) – @string/donateTitle
 - donateUsername (TextView) – @string/donateUsername **selected**
 - donateSubtitle (TextView) – @string/donateSubtitle
 - donateButton (Button) – @string/donateButton
 - paymentMethod (RadioGroup)
 - amountPicker (NumberPicker)
 - progressBar
 - amount (TextView) – @string/amount
 - paymentAmount (EditText)
 - textView – @string/total
 - totalSoFar (TextView) – @string/totalSoFar

Properties

layout:width	wrap_content
layout:height	wrap_content
layout:margin	[?, 10dp, ?, ?]
layout:alignEnd	
layout:verticalParentEnd	



Donation 4.0 – Donate ‘onCreate()’ extract *

Retrieving existing
Username from Prefs file

```
settings = getSharedPreferences("loginPrefs", 0);  
String username = settings.getString("username", "");
```

```
TextView donateUsername = (TextView) findViewById(R.id.donateUsername);
```

```
SpannableString spanString = new SpannableString(username);  
//spanString.setSpan(new UnderlineSpan(), 0, spanString.length(), 0);  
spanString.setSpan(new StyleSpan(Typeface.BOLD), 0, spanString.length(), 0);  
spanString.setSpan(new StyleSpan(Typeface.ITALIC), 0, spanString.length(), 0);  
donateUsername.setText(spanString);
```

Applying some ‘styling’ and
updating our TextView



End Result – Donation 4.0





More Reading

- ❑ <https://www.journaldev.com/9412/android-shared-preferences-example-tutorial>



Questions?