Mobile Application Development

Produced by

David Drohan (ddrohan@wit.ie)

Department of Computing & Mathematics
Waterford Institute of Technology
http://www.wit.ie

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 Coffees)
- ❑ Be able to work with **`Realm`** to implement a noSQL database on an Android device (again, to manage our Coffees)
- ❑ Be able to work with **`SharedPreferences`** to manage, for example, basic 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.

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
```

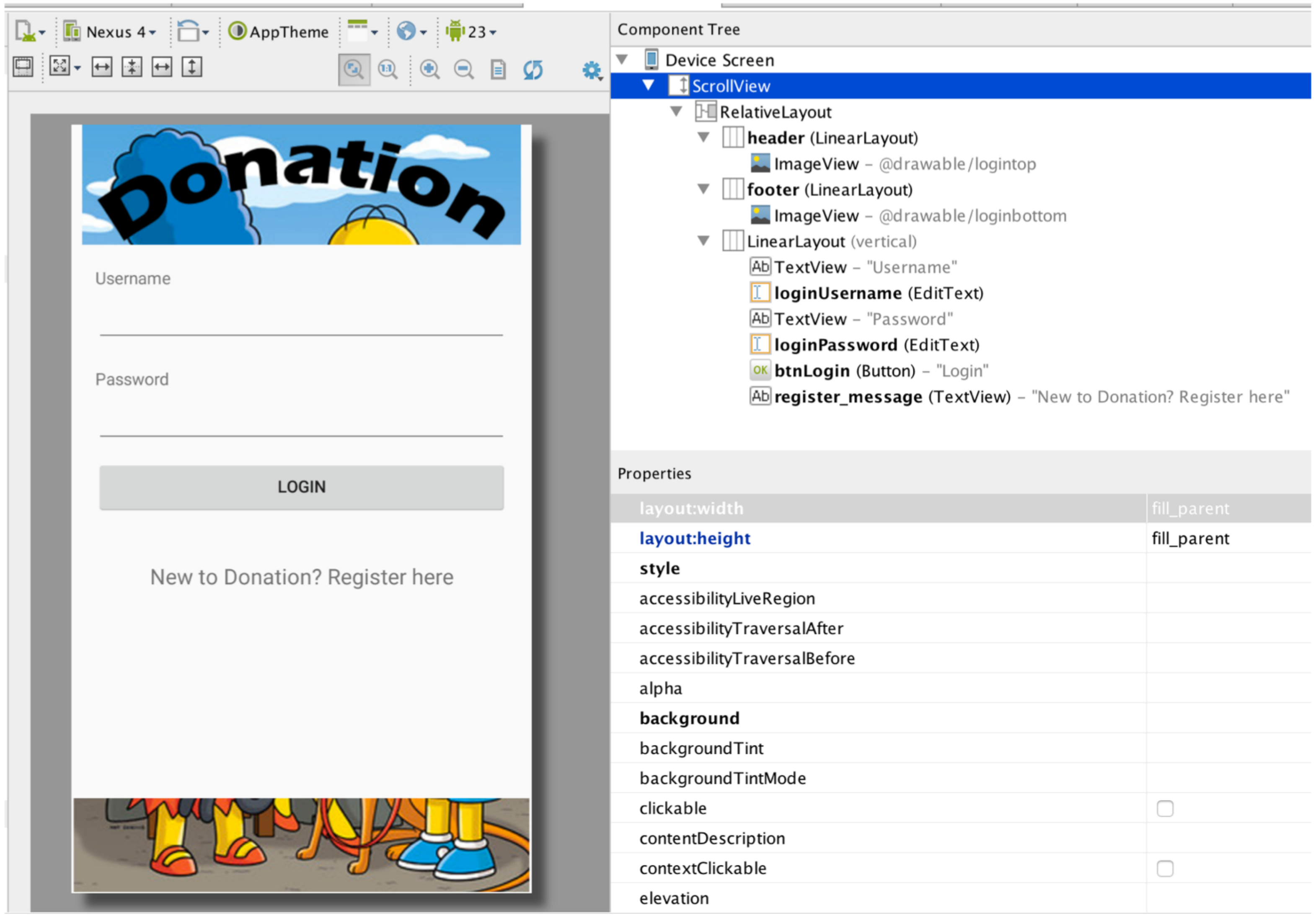
Clearing or Deleting Data

- ❑ 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 with a Login & Register Screen

activity_login.xml



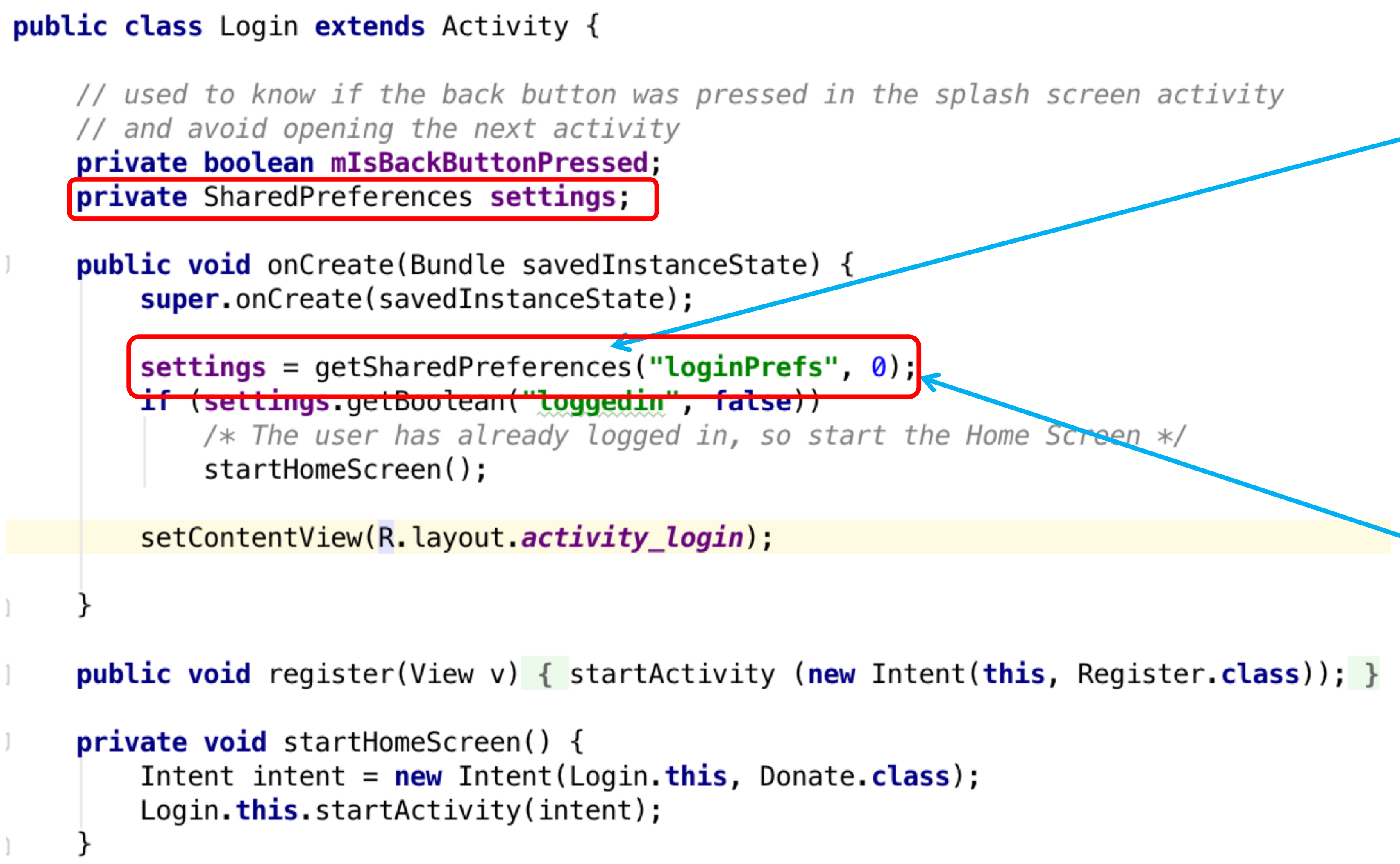
Component Tree

```
Device Screen
  ScrollView
    RelativeLayout
      header (LinearLayout)
        ImageView – @drawable/logintop
      footer (LinearLayout)
        ImageView – @drawable/loginbottom
      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	☐
contentDescription	
contextClickable	☐
elevation	

Login *

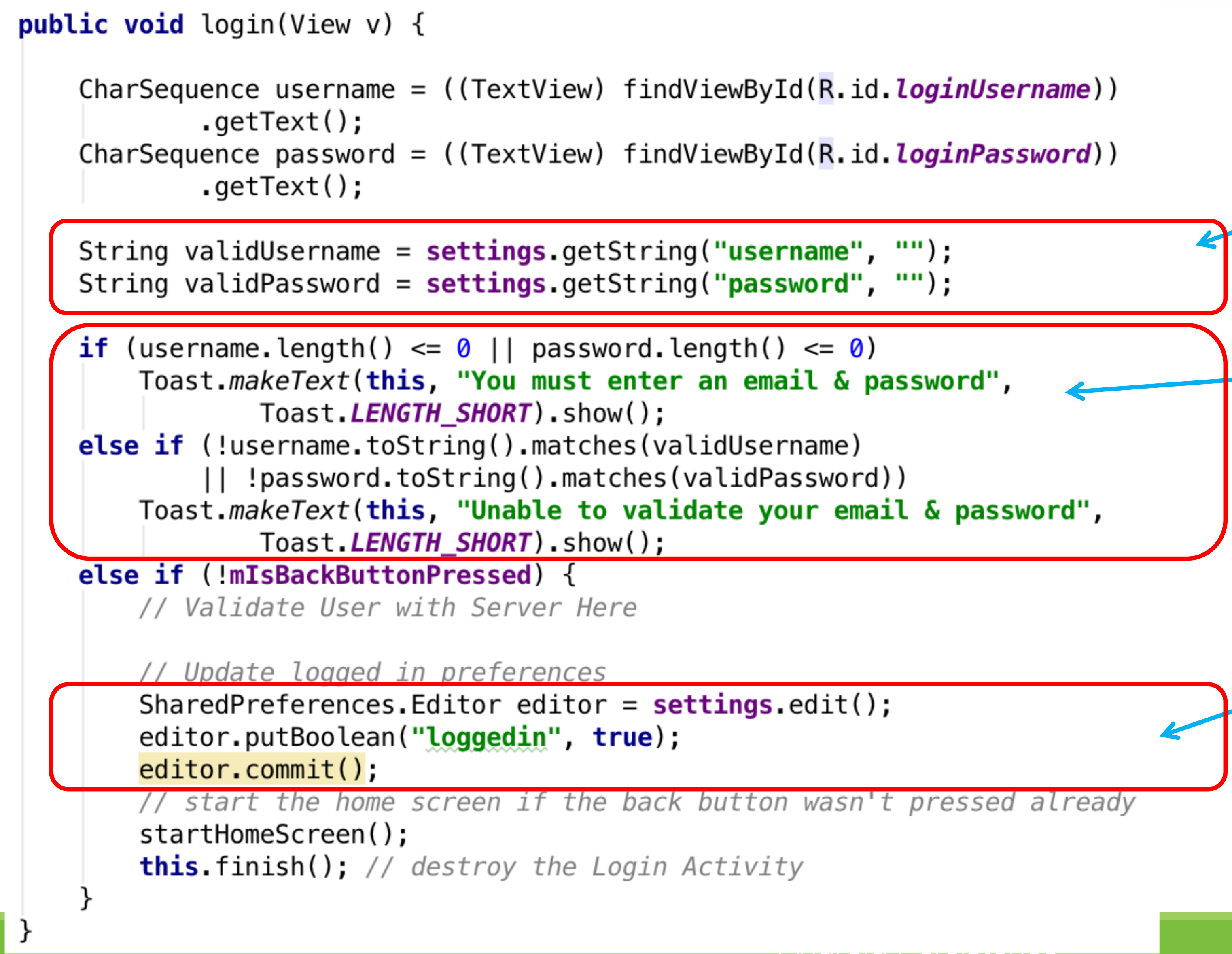


```
public class Login extends Activity {

    // used to know if the back button was pressed in the splash screen activity
    // and avoid opening the next activity
    private boolean mIsBackButtonPressed;
    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

Login *



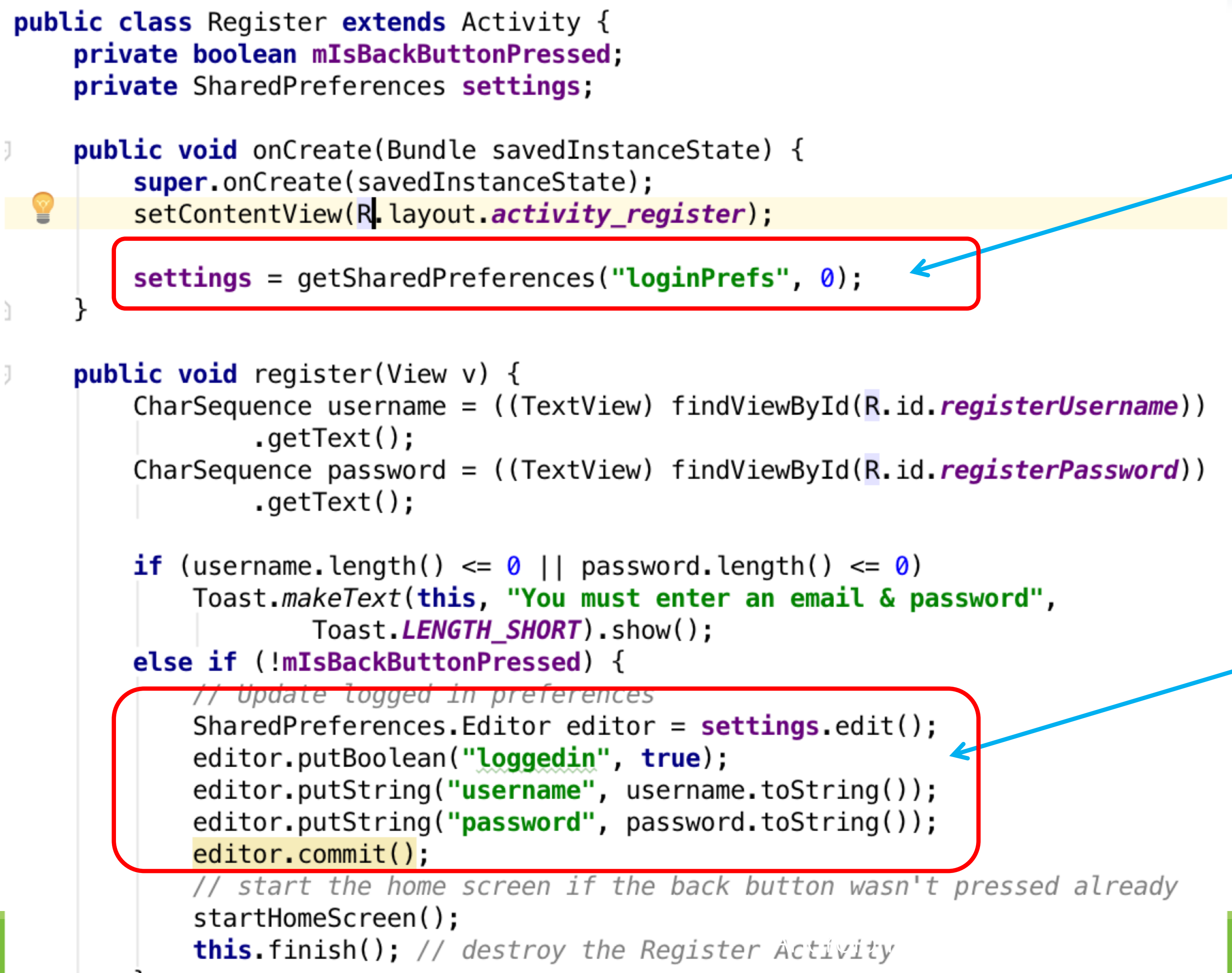
```
public void login(View v) {

    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 (!mIsBackButtonPressed) {
        // 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

Register *



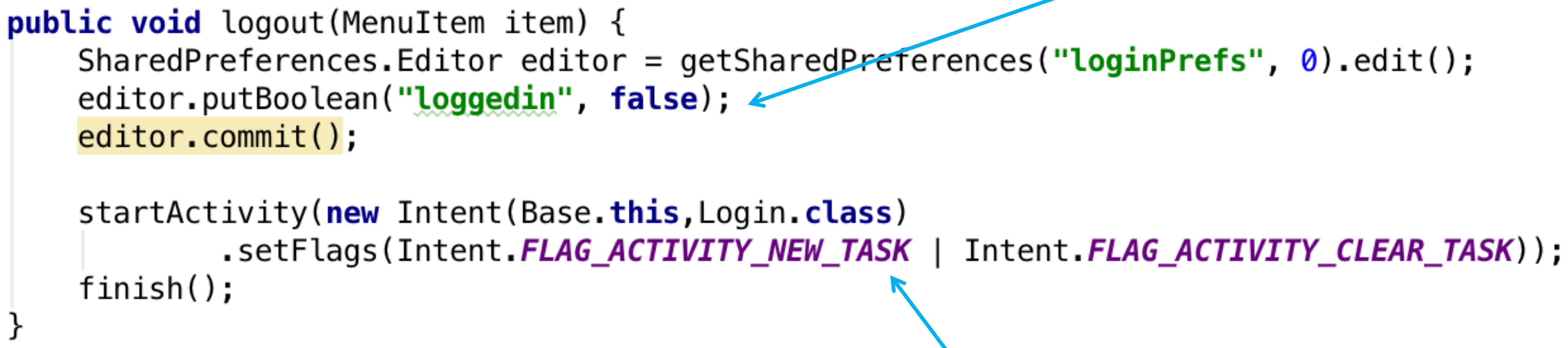
```
public class Register extends Activity {
    private boolean mIsBackButtonPressed;
    private SharedPreferences settings;

    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 (!mIsBackButtonPressed) {
            // 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

Logout method

Resetting 'loggedin' to false



```
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();
}
```

Returning to the 'Login' screen

More Reading

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

Questions?