ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

# Mobile Programing

## Shared Preference

# Note

- This slide is based on Google Android code labs slides
- Original slides: https://drive.google.com/drive/folders/1eu-LXxiHocSktGYpG04PfE9Xmr_pBY5P

# 9.1 Shared Preferences

# Contents

- Shared Preferences
- Listening to changes

# What is Shared Preferences?

- Read and write small amounts of primitive data as key/value pairs to a file on the device storage
- SharedPreference class provides APIs for reading, writing, and managing this data
- Save data in onPause()
  restore in onCreate()

# Shared Preferences and Saved Instance State

- Small number of key/value pairs
- Data is private to the application

# Shared Preferences vs. Saved Instance State

- Persist data across user sessions, even if app is killed and restarted, or device is rebooted
- Data that should be remembered across sessions, such as a user's preferred settings or their game score
- Common use is to store user preferences

- Preserves state data across activity instances in same user session
- Data that should not be remembered across sessions, such as the currently selected tab or current state of activity.
- Common use is to recreate state after the device has been rotated

# Creating Shared Preferences

- Need only one Shared Preferences file per app
- Name it with package name of your app—unique and easy to associate with app
- MODE argument for getSharedPreferences() is for backwards compatibility—use only MODE_PRIVATE to be secure

# getSharedPreferences()

```
private String sharedPrefFile =
    "com.example.android.hellosharedprefs";
mPreferences =
    getSharedPreferences(sharedPrefFile,
                         MODE_PRIVATE);
```

# Saving Shared Preferences

- SharedPreferences.Editor interface
- Takes care of all file operations
- put methods overwrite if key exists
- apply() saves asynchronously and safely

# SharedPreferences.Editor

```
@Override
protected void onPause() {
    super.onPause();
    SharedPreferences.Editor preferencesEditor =
        mPreferences.edit();
    preferencesEditor.putInt("count", mCount);
    preferencesEditor.putInt("color", mCurrentColor);
    preferencesEditor.apply();
}
```

# Restoring Shared Preferences

- Restore in onCreate() in Activity
- Get methods take two arguments—the key, and the default value if the key cannot be found
- Use default argument so you do not have to test whether the preference exists in the file

# Getting data in onCreate()

```
mPreferences = getSharedPreferences(sharedPrefFile, MODE_PRIVATE);
if (savedInstanceState != null) {
    mCount = mPreferences.getInt("count", 1);
    mShowCount.setText(String.format("%s", mCount));

    mCurrentColor = mPreferences.getInt("color", mCurrentColor);
    mShowCount.setBackgroundColor(mCurrentColor);

    mNewText = mPreferences.getString("text", "");
} else { … }
```

# Clearing

- Call clear() on the SharedPreferences.Editor and apply changes

- You can combine calls to put and clear. However, when you apply(),  clear() is always done first, regardless of order!

# clear()

```
SharedPreferences.Editor preferencesEditor
=
                    mPreferences.edit();
preferencesEditor.clear();
preferencesEditor.apply();
```

# Listening to Changes

# Listening to changes

- Implement interface SharedPreference.OnSharedPreferenceChangeListener
- Register listener with registerOnSharedPreferenceChangeListener()
- Register and unregister listener in onResume() and onPause()
- Implement on onSharedPreferenceChanged() callback

# Interface and callback

```
public class SettingsActivity extends AppCompatActivity
    implements OnSharedPreferenceChangeListener { ...

    public void onSharedPreferenceChanged(
        SharedPreferences sharedPreferences, String key) {
        if (key.equals(MY_KEY)) {
            // Do something
        }
    }
}
```

# Creating and registering listener

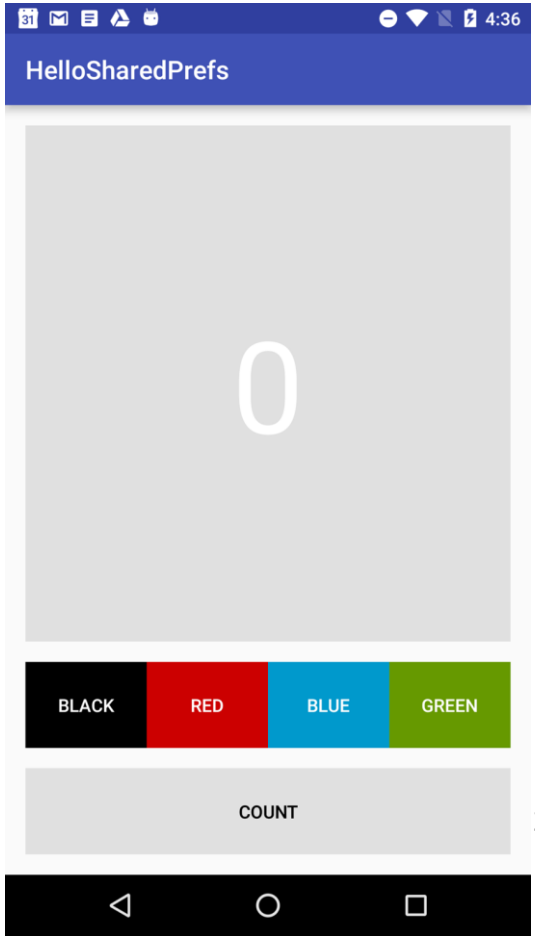
```
SharedPreferences.OnSharedPreferenceChangeListener listener =
    new SharedPreferences.OnSharedPreferenceChangeListener() {
  public void onSharedPreferenceChanged(
     SharedPreferences prefs, String key) {
        // Implement listener here
  }
};
prefs.registerOnSharedPreferenceChangeListener(listener);
```

# You need a STRONG reference to the listener

- When registering the listener the preference manager does not store a strong reference to the listener
- You must store a strong reference to the listener, or it will be susceptible to garbage collection
- Keep a reference to the listener in the instance data of an object that will exist as long as you need the listener

# Practical: HelloSharedPrefs

- Add Shared Preferences to a starter app
- Add a "Reset" button that clears both the app state and the preferences for the app



# Learn more

- Saving Data
- Storage Options
- Saving Key-Value Sets
- SharedPreferences
- SharedPreferences.Editor

Stackoverflow

- How to use SharedPreferences in Android to store, fetch and edit values
- onSavedInstanceState vs. SharedPreferences

# What's Next?

- Concept Chapter: 9.1 Shared Preferences
- Practical: 9.1 Shared Preferences

END