ITW202: Mobile Application

Unit IV: Developing for Android

Ms. Sonam Wangmo

Gyalpozhing College of Information Technology
Royal University of Bhutan

May 27, 2021

Persistent Data in Android

Three approaches to data persistence

1. SharedPreferences
2. Standard flat file input/output
3. SQLite database system

What is Shared Preferences?

- Shared preferences allow you to store small amounts of primitive data as key/value pairs in a file on the device.
- SharedPreference class provides APIs for reading, writing, and managing this data.
- Save data in onPause(),restore in onCreate()

Shared preferences vs. saved instance state

Shared preferences	Saved instance state
Persists across user sessions, even if your app is stopped and restarted, or if the device is rebooted.	Preserves state data across activity instances in the same user session.
Used for data that should be remembered across user sessions, such as a user's preferred settings or their game score.	Used for data that should not be remembered across sessions, such as the currently selected tab, or any current state of an activity.
Represented by a small number of key/value pairs.	Represented by a small number of key/value pairs.
Data is private to the app.	Data is private to the app.
Common use is to store user preferences.	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

Creating Shared Preferences

```
private String sharedPrefFile =
    "com.example.android.hellosharedprefs";

mPreferences =
    getSharedPreferences(sharedPrefFile,
                         MODE_PRIVATE);
```

Saving shared preferences

You save preferences in the onPause() state of the activity lifecycle using the SharedPreferences.Editor interface.

Saving shared preferences

Steps to save the shared preferences are:

- Get a SharedPreferences.Editor.
- Add key/value pairs to the editor using the "put" method appropriate for the data type, for example, putInt() or putString().
- Call apply() to write out your changes.

Saving shared preferences

```
@Override
protected void onPause() {
    super.onPause();
    SharedPreferences.Editor preferencesEditor = mPreferences.edit();
    preferencesEditor.putInt(COUNT_KEY, mCount);
    preferencesEditor.putInt(COLOR_KEY, mColor);
    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

Restoring shared preferences

```
mPreferences = getSharedPreferences(sharedPrefFile,MODE_PRIVATE);
mCount = mPreferences.getInt(COUNT_KEY, defValue: 0);
mShowCountTextView.setText(String.format("%s",mCount));
mColor = mPreferences.getInt(COLOR_KEY,mColor);
mShowCountTextView.setBackgroundColor(mColor);
```

Clearing shared preferences

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

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


Mobile Application
Development
THANK YOU