Shared Preferences

Jigisha Patel

Shared Preferences

- You should use the SharedPreferences APIs, if you have a relatively small collection of **key-value** pairs that you'd like to save.
- A SharedPreferences object points to a file containing key-value pairs and provides simple methods to read and write them.
- Each SharedPreferences file is
  - managed by the framework and can be private or shared,
  - can be accessed throughout the app,
  - can store only primitive types and String sets and
  - not accessible to other apps.

Get a handle to shared preferences

- You can create a new shared preference file or access an existing one by calling one of these methods:

1. getSharedPreferences()

Use this if you need multiple shared preference files identified by name, which you specify with the first parameter. You can call this from any Context in your app.

When naming your shared preference files, you should use a name that's uniquely identifiable to your app. An easy way to do this is prefix the file name with your application ID. For example: "com.example.myapp.PREFERENCE_FILE_KEY".

```
Context context = getActivity();
SharedPreferences sharedPref = context.getSharedPreferences(
        getString(R.string.preference_file_key), Context.MODE_PRIVATE);
```

Get a handle to shared preferences

2. getPreferences()

Use this from an Activity if you need to use only one shared preference file for the activity. Because this retrieves a default shared preference file that belongs to the activity, you don't need to supply a name.

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
```

3. getDefaultSharedPreferences()

If you're using the SharedPreferences API to save app settings, you should instead use getDefaultSharedPreferences() to get the default shared preference file for your entire app.

Write to shared preferences

- To write to a shared preferences file, create a SharedPreferences.Editor by calling edit() on your SharedPreferences.
- Pass the keys and values you want to write with methods such as putInt() and putString().
- Then call apply() or commit() to save the changes.
- apply() writes the updates to disk asynchronously.
- commit() writes the data to disk synchronously. You should avoid calling it from your main thread because it could pause your UI rendering.

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();
editor.putInt(getString(R.string.saved_high_score_key), newHighScore);
editor.apply();
```

Read from shared preferences

- To retrieve values from a shared preferences file, call methods such as getInt() and getString(), providing the key for the value you want, and optionally a default value to return if the key isn't present.

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
int defaultValue = getResources().getInteger(R.integer.saved_high_score_default_key);
int highScore = sharedPref.getInt(getString(R.string.saved_high_score_key), defaultValue);
```

Shared Preferences Methods

- You can use following methods and listener for additional operations with SharedPreferences :
- contains(String key) - Checks whether the preferences contains a preference
- getAll() - Retrieve all values from the preferences.
- remove(String key) – marks the sharedpreferences editor that a preference should be removed which will be removed when commit() is called.

- registerOnSharedPreferenceChangeListener(SharedPreferences.OnSharedPreferenceChangeListener listener) - Registers a callback to be invoked when a change happens to a preference.

AndroidX Preference Library

AndroidX Preference Library

- The recommended way to integrate user configurable settings into your application is to use the AndroidX Preference Library.
- This library manages the user interface and interacts with storage so that you define only the individual settings that the user can configure.
- The library comes with a Material theme that provides a consistent user experience across devices and OS versions.

Cont…

- A Preference is the basic building block of the Preference Library.
- A settings screen contains a Preference hierarchy.
- You can define this hierarchy as an XML resource, or you can build a hierarchy in code.

- If you have several related Preferences on a single screen, you can group them with a PreferenceCategory.
- A PreferenceCategory displays a category title and visually separates the category.

References

- https://developer.android.com/training/data-storage/shared-preferences
- https://developer.android.com/reference/android/content/SharedPreferences
- https://developer.android.com/guide/topics/ui/settings