Android Developer Fundamentals V2

Data Storage

Shared Preferences

Shared Preferences

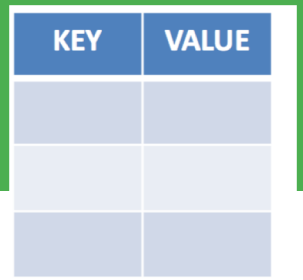
Contents

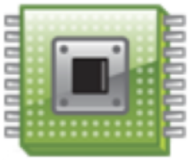
- Shared Preferences
- Listening to changes

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





1. Data in this type of container is saved as **<Key, Value>** pairs where
   a. The key is a string and
   b. its associated value must be a primitive data type.
2. Data is stored in the device's internal main memory.
3. PREFERENCES are typically used to keep state information and shared data among several activities of an application.

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()

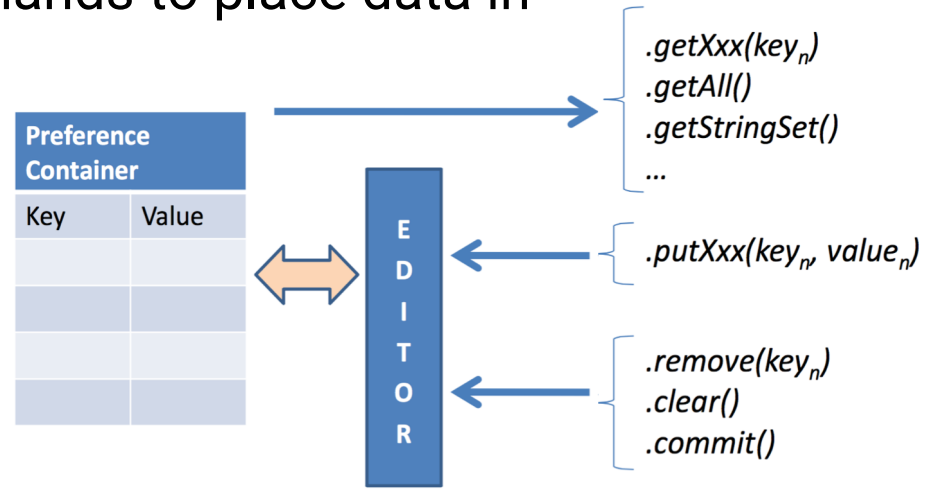
```
private String sharedPrefFile =
    "com.example.android.hellosharedprefs";

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

Shared Preferences

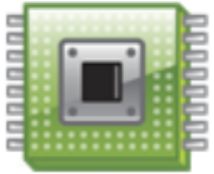
Each of the Preference mutator methods carries a typed-value content that can be manipulated by an **editor** that allows **putXxx**... and **getXxx**... commands to place data in and out of the Preference container.

Xxx = { **Long, Int, Double, Boolean, String** }



Shared Preferences

Key	Value
chosenColor	RED
chosenNumber	7



```
private void usingPreferences(){
  // Save data in a SharedPreferences container
  // We need an Editor object to make preference changes.

1 SharedPreferences myPrefs = getSharedPreferences("my_preferred_choices",
                                                   Activity.MODE_PRIVATE);
  SharedPreferences.Editor editor = myPrefs.edit();
2         editor.putString("chosenColor", "RED");
          editor.putInt("chosenNumber", 7 );
  editor.commit();

  // retrieving data from SharedPreferences container (apply default if needed)
3 String favoriteColor = myPrefs.getString("chosenColor", "BLACK");
  int favoriteNumber = myPrefs.getInt("chosenNumber", 11 );

}
```

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

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