Udacity / Google Developing Android Apps Cheat Sheet

by ScottHOC via cheatography.com/45425/cs/14625/

Shared Preferences

Reading From Shared Preferences:

`getDefaultSharedPreferences(context)`

Gets a SharedPreferences instance that points to the**default** file that is used by the preference framework in the given context.

getSharedPreferences(context)

Gets a specific SharedPreferences instance **by name** in case you have more than one preference in the same context.

Note that there is no type checking, so if you ask shared preferences for a type that is different from what is actually stored in that key, the app will crash.

Implementing OnSharedPreferenceChangeListener

```
public class MyClass extends ExtendedClass implements
  SharedPreferences.OnSharedPreferenceChangeListener {
...
```

Error Levels

Error (Remains in release versions)

Warning (Remains in release versions)

Info (Remains in release versions)

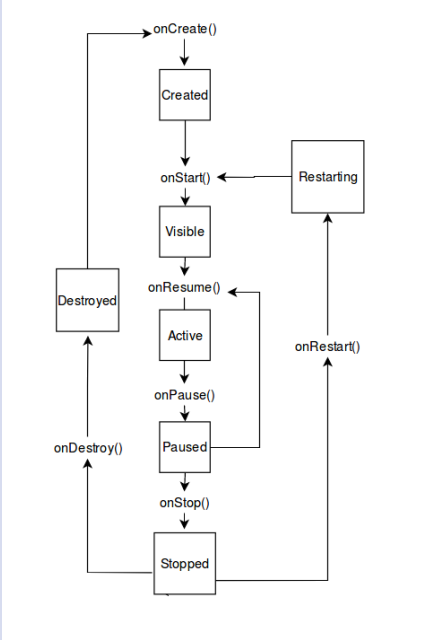
Debug (Not in release versions)

Verbose (Not in release versions)

Permissions

```
Add Permissions to AndroidManifest.xml
<uses-permission
android:name="android.permission.XXXXXXXXX">
```

Lifecycle



A Simple Way to Launch URL in Browser

```
// In mainifest.xml ----------------------------
<uses-permission
android:name="android.permission.INTERNET"/>
```

```
// Imports -------------------------------------
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.Window;
```


By **ScottHOC**

cheatography.com/scotthoc/

Published 14th February, 2018.

Last updated 20th February, 2018.

A Simple Way to Launch URL in Browser (cont)

** The block below will not allow more than 500 characters and the formatting tags caused it to go over. If you cut and paste this code, use auto-format (Alt-L in intelliJ or AS) to format it properly. If you use another IDE, consult your setup or help file to find the auto-format hotkey.

```
// The code -------------------------------------

public class MainActivity extends Activity {

@Override
protected void onCreate(Bundle savedInstanceState) {
requestWindowFeature(Window.FEATURE_NO_TITLE);
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
Intent browserIntent = new Intent(Intent.ACTION_VIEW,
Uri.parse("http://www.google.com"));
startActivity(browserIntent);
}

}
```

Loaders

To create a loader:

1) Auto-import the Loader class as you type, Then, have the IDE fill in the Loader Callbacks for you:

```
LoaderManager.LoaderCallbacks<String>{
    onCreateLoader()
    onLoadFinished()
    onLoaderReset()
}
```

2) Create an integer constant for a loader ID:

```
private static final int myLoaderName = 20;
```

3) Initialize the loader with the Loader Manager

Preferences

The SharedPreference class saves key-value pairs of preferences.

The PreferenceFragment subclass was specifically designed to display preferences and has replaced the now depricated Preference Activity.

Preference Fragements populate themselves with data that's defined in an XML document. The XML is used to generate UI widgets in the fragment.

When the user updates preferences in the widget, these changes are automatically updated in the SharedPreferences file.

Recycler View

Layout Manager > **RECYCLER VIEW** < View Holder < Adapter < Data

"Whereas the View Holder determines how an individual entry is displayed, the Layout Manager determines how the entire collection of entries is displayed."

"Layout Manager is a key part of the way recycling works in Recycler View since it determines when to recycle views that are no longer visible to the user."

RecyclerView Adapter:

An adaper is called by Recycler View to:

Create new items in the form of ViewHolders

Populate (or bind) items with data

Return information abou the data (IE # of items)

The Adapter requires 3 functions to be overridden:

OnCreateViewHolder() : Called when RecyclerView instantiates a new ViewHolder instance (inflates from XML or generates it in code)

RecyclerView Adapter: (cont)

OnBindViewHolder() : Called when RecyclerView wants to populate ViewHolder with data

GetItemCount() : Returns the number of items in teh data source. (This might be called a number of times during the layout process so it needs to be a fast, efficient process.)

Recycler View Layout Manager

"Whereas the View Holder determines how an individual entry is displayed, the Layout Manager determines how the entire collection of entries is displayed."

"Layout Manager is a key part of the way recycling works in Recycler View since it determines when to recycle views that are no longer visible to the user."

There are 3 implementations of Layout Manager:
LinearLayoutManager
GridLayoutManager
StaggeredGridLayoutManger

LinearLayoutManager allows for vertical or horizontal scrolling. Vertical is the default.

GridLayoutManager is a subclass of LinearLayoutManager that is laid out in a grid and can scroll vertically or horizontally.

StaggeredGridLayoutManager displays a offset grid of items. Commonly used for situations where the content is of varying dimentions.

It is also possible to directly extend from Layout Manager and create your own.

Data Persistence

There are 5 different ways to persist data:

1) "Saved Instance State" uses key-value pairs (a map) to save the state of one of your views. It's usually used to save state when the view must be quickly destroyed and restarted such as when the phone is rotated or if the activity has to be destroyed because of memory constraints but will need to be recreated at some point, when it returns to the foreground. If the user quits the app or restarts the phone, this data is lost.

If you need to have data persist beyond app closures and phone restarts, it needs to be saved to a file system.

2) The SharedPreferences class saves simple key-value pairs (a map) to a file. Keys are always strings. values are primitives (as opposed to objects). This can be used for things like saving a String for the user name between sessions or the URL (as a String) of the last web page someone was on and returning to it when restarting the app.

3) For more complex data (IE: an object) we use a database, and Android uses SQL lite. Android also has various framework components (such as Content Providers) that allow you to manage and share data in databases.

4) For larger items (IE: audio, video, image or e-book files) you can save to internal or external storage.

5) We can also save larger files to a Cloud or Firebase, rather than taking up space in the device's storage.