Introduction to Mobile Computing with Android

Prof. Manish Kumar Joshi, Assistant Professor
IT & Computer Science

Parul® University

CHAPTER-5

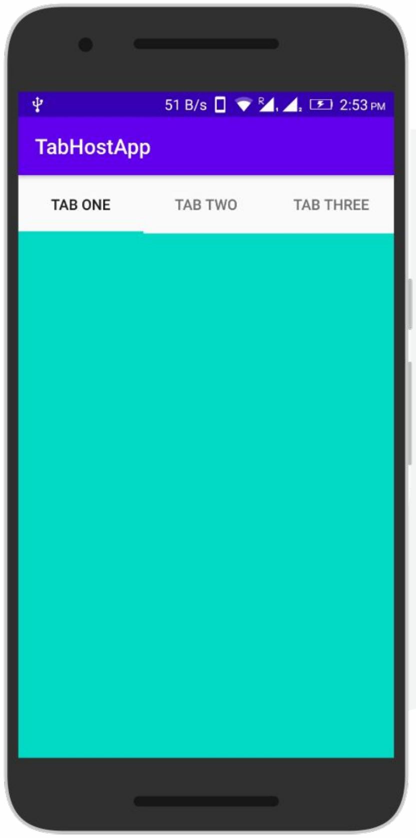
Master Advanced Android Programming

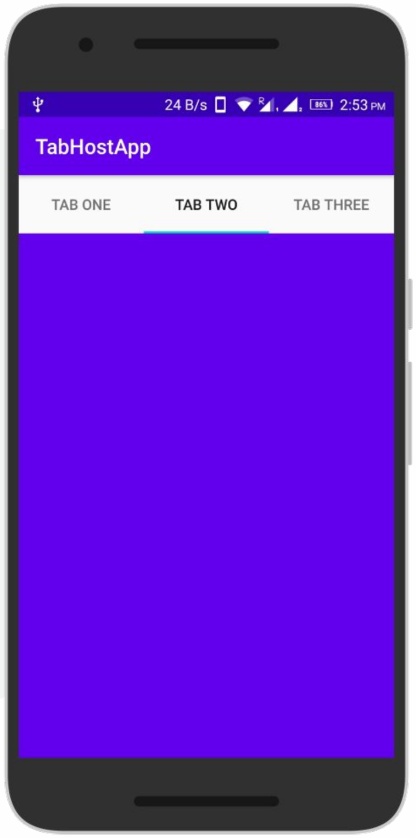
Parul® University

DIGITAL LEARNING CONTENT

TabHost

In Android, TabHost is a Container for tabbed window view. This object holds two children one is set of tab labels that the user clicks to select a specific tab and other is a FrameLayout object that displays the content of that page.





TabHost

Whenever we need to enter or display a lot of information in one activity. A simple and effective method is to use tabs in your interface form which is done using TabHost in Android.

A TabHost holds two children's from which one is use to set the labels that the users clicks to select tab other is a FrameLayout that is used to display the content of that page. It means when you select any label (or you can say change the tab) the FrameLayout is used to display the content for that particular tab.

TabHost.TabSpec

In Android, A tab has a tab indicator, content, and a tag that is used to keep track of it. This builder helps us to choose among these options. For the tab indicator our choices are set a label or set a label and an icon both together. For the tab content our choices are the id of View, a TabHost.TabContentFactory that creates the view content and an Intent that launches an Activity.

Important Methods of TabSpec

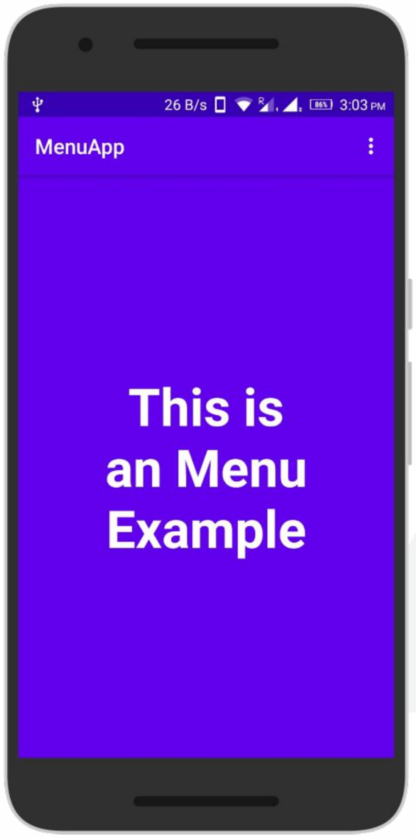
1. setIndicator(CharSequence label)
2. setIndicator(CharSequence label,Drawable icon)
3. setContent(Intent intent)

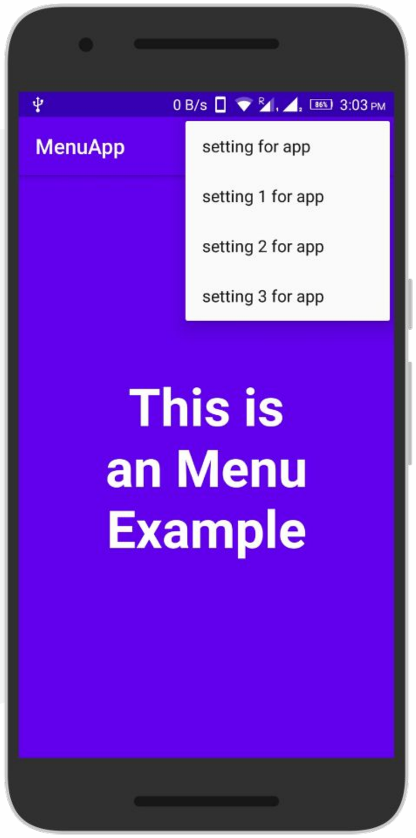
Important Methods of TabHost

1. addTab(TabSpec tabSpec)
2. clearAllTabs()
3. setCurrentTab(int index)
4. setOnTabChangedListener(OnTabChangeListenerl)

Menu

In android, Menu is an important part of UI component which is used to provide some common functionality around the application. With the help of menu user can experience smooth and consistent experience throughout the application.





Menu

In order to use menu, we should define it in separate XML file and use that file in our application based on our requirements. Also, we can use menu APIs to represent user actions and other options in our android application activities.

Types of Menu

- Android Options Menu
- Android Context Menu
- Android Popup Menu

Types of Menu

Android Options Menu – Android Options Menu is a primary collection of menu items in an android application and useful for actions that have a global impact on the searching application.

Android Context Menu – Android Context Menu is a floating menu only appears when user click for a long time on an element and useful for elements that effect the selected content or context frame.

Android Popup Menu – Android Popup Menu displays a list of items in a vertical list which presents to the view that invoked the menu and useful to provide an overflow of actions that related to specific content.

How to Define Menu in XML File?

Android Studio provides a standard XML format for type of menus to define menu items. We can simply define the menu and all its items in XML menu resource instead of building the menu in the code and also load menu resource as menu object in the activity or fragment used in our android application.

Elements in Menu

- <menu> It is the root element which helps in defining Menu in XML file and it also holds multiple elements.
- <item> It is used to create a single item in menu. It also contains nested <menu> element in order to create a submenu.
- <group> It is an optional and invisible for <item> elements to categorize the menu items so they can share properties like active state, visibility.

Elements in Menu

- <menu> It is the root element which helps in defining Menu in XML file and it also holds multiple elements.
- <item> It is used to create a single item in menu. It also contains nested <menu> element in order to create a submenu.
- <group> It is an optional and invisible for <item> elements to categorize the menu items so they can share properties like active state, visibility.

SharedPreferences

One of the most Interesting Data Storage option Android provides its users is Shared Preferences. Shared Preferences is the way in which one can store and retrieve small amounts of primitive data as key/value pairs to a file on the device storage such as String, int, float, Boolean that make up your preferences in an XML file inside the app on the device storage.

SharedPreferences

Shared Preferences can be thought of as a dictionary or a key/value pair. For example, you might have a key being “username” and for the value, you might store the user’s username. And then you could retrieve that by its key (here username). You can have simple shared preferences API that you can use to store preferences and pull them back as and when needed. Shared Preferences class provides APIs for reading, writing and managing this data.

SharedPreferences

Shared Preferences is suitable in different situations. For example, when the user's settings need to be saved or to store data that can be used in different activities within the app. As you know, onPause() will always be called before your activity is placed in the background or destroyed, So for the data to be saved persistently we prefer saving it in onPause(), which could be restored in onCreate() of the activity. The data stored using shared preferences are kept private within the scope of the application. However, shared preferences are different from that activity's instance state.

SharedPreferences

Shared Preferences is suitable in different situations. For example, when the user’s settings need to be saved or to store data that can be used in different activities within the app. As you know, onPause() will always be called before your activity is placed in the background or destroyed, So for the data to be saved persistently we prefer saving it in onPause(), which could be restored in onCreate() of the activity. The data stored using shared preferences are kept private within the scope of the application. However, shared preferences are different from that activity’s instance state.

How to Create SharedPreferences

The first thing we need to do is to create one shared preferences file per app. So name it with the package name of your app- unique and easy to associate the app. When you want to get the values, call getSharedPreferences() method. Shared Preferences provide modes of storing the data (private mode and public mode). It is for the backward compatibility- use only MODE_PRIVATE to be secure.

The Context Modes

- MODE_PUBLIC
- MODE_PRIVATE
- MODE_APPEND

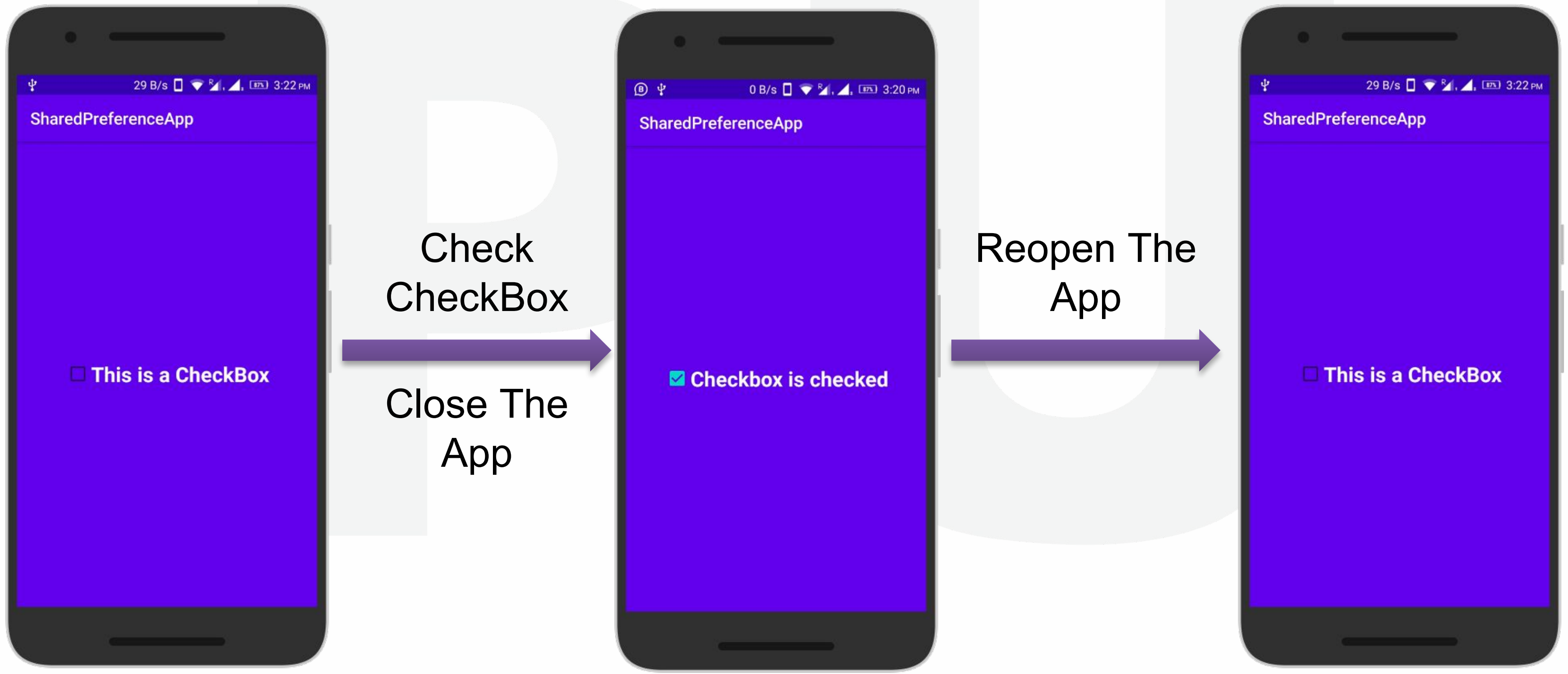
Methods of SharedPreferences

1. contains(String key)
2. edit()
3. getAll()
4. getBoolean(String key, boolean defValue)
5. getFloat(String key, float defValue)
6. getInt(String key, int defValue)
7. getLong(String key, long defValue)
8. getString(String key, String defValue)
9. getStringSet(String key, Set defValues)
10. registerOnSharedPreferencechangeListener(SharedPreferences.OnSharedPreferencechangeListener listener)
11. unregisterOnSharedPreferencechangeListener(SharedPreferences.OnSharedPreferencechangeListener listener)

Parul® University

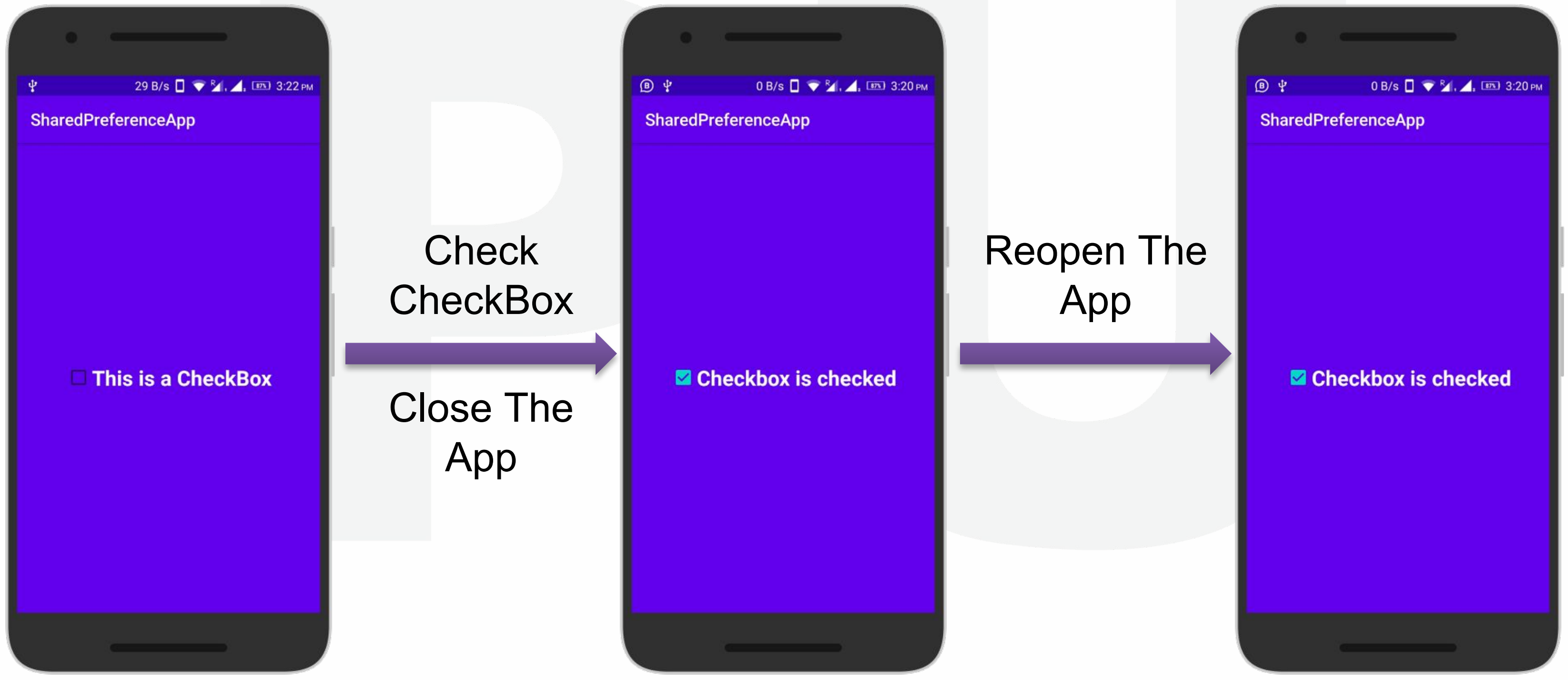
Examples of SharedPreferences

Before using SharedPreferences.



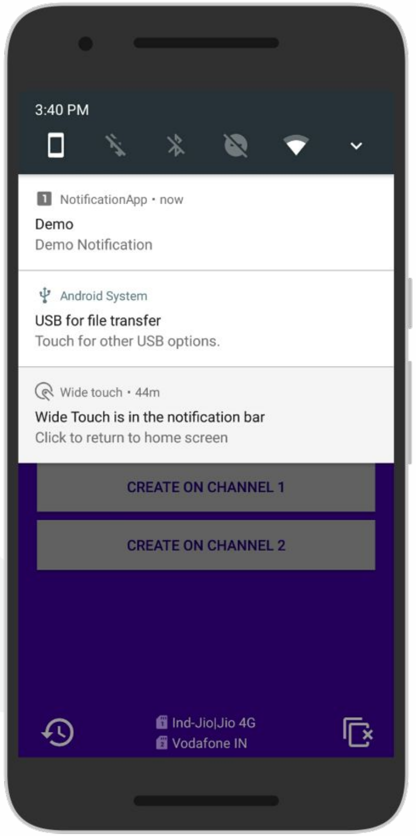
Examples of SharedPreferences

After using SharedPreferences.



Notifications in Android

A notification is a message you can display to the user outside of your application's normal UI. When you tell the system to issue a notification, it first appears as an icon in the notification area. To see the details of the notification, the user opens the notification drawer. Both the notification area and the notification drawer are system-controlled areas that the user can view at any time.



3:40 PM

NotificationApp • now
Demo
Demo Notification

Android System
USB for file transfer
Touch for other USB options.

Wide touch • 44m
Wide Touch is in the notification bar
Click to return to home screen

Notifications in Android

Android Toast class provides a handy way to show users alerts but problem is that these alerts are not persistent which means alert flashes on the screen for a few seconds and then disappears.

NotificationCompat.Builder Class

The NotificationCompat.Builder class allows easier control over all the flags, as well as help constructing the typical notification layouts. Following are few important and most frequently used methods available as a part of NotificationCompat.Builder class.

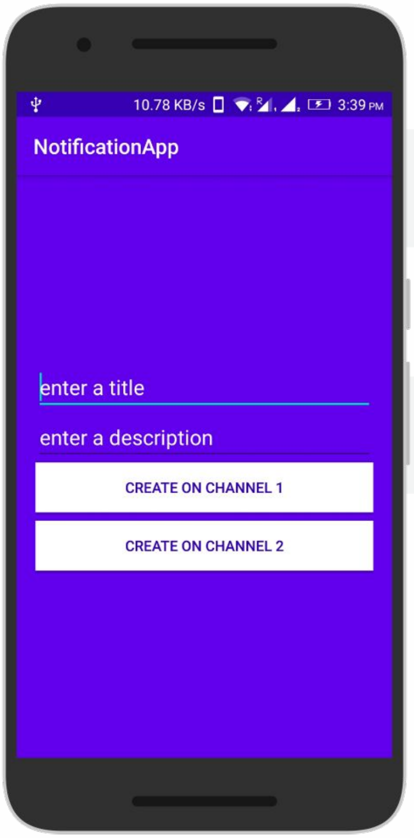
Methods

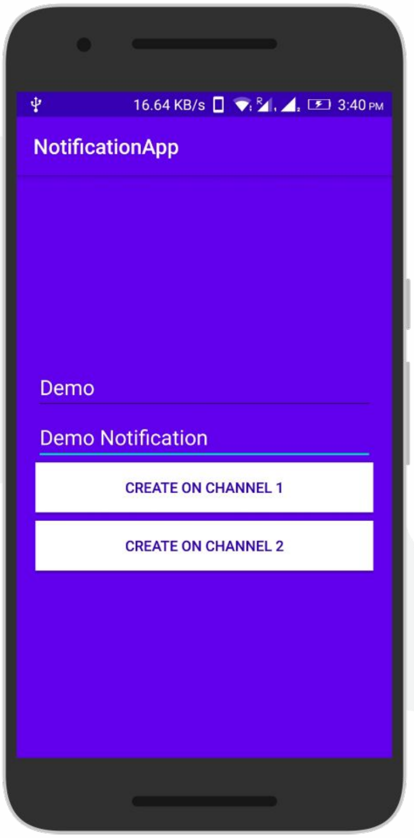
1. Notification build()
2. NotificationCompat.Builder setAutoCancel (boolean autoCancel)
3. NotificationCompat.Builder setContent (RemoteViews views)
4. NotificationCompat.Builder setContentInfo (CharSequence info)
5. NotificationCompat.Builder setContentIntent (PendingIntent intent)
6. NotificationCompat.Builder setContentText (CharSequence text)
7. NotificationCompat.Builder setContentTitle (CharSequence title)

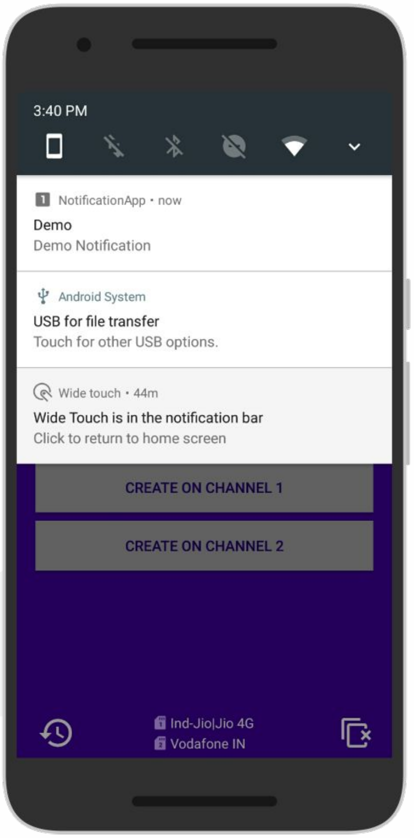
Methods

8. NotificationCompat.Builder setDefaults (int defaults)
9. NotificationCompat.Builder setLargeIcon (Bitmap icon)
10. NotificationCompat.Builder setNumber (int number)
11. NotificationCompat.Builder setOngoing (boolean ongoing)
12. NotificationCompat.Builder setSmallIcon (int icon)
13. NotificationCompat.Builder setStyle (NotificationCompat.Style style)
14. NotificationCompat.Builder setTicker (CharSequence tickerText)
15. NotificationCompat.Builder setVibrate (long[] pattern)
16. NotificationCompat.Builder setWhen (long when)

Example






3:40 PM

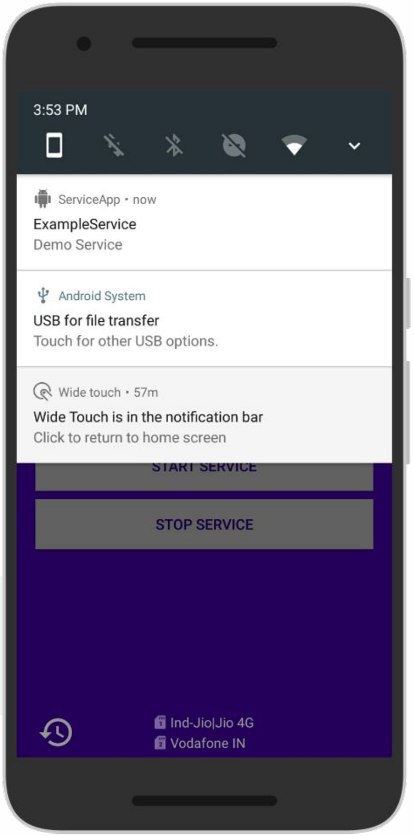
NotificationApp • now
Demo
Demo Notification

Android System
USB for file transfer
Touch for other USB options.

Wide touch • 44m
Wide Touch is in the notification bar
Click to return to home screen

Services

A service is a component that runs in the background to perform long-running operations without needing to interact with the user and it works even if application is destroyed. A service can essentially take two states −



3:53 PM

ServiceApp • now
ExampleService
Demo Service

Android System
USB for file transfer
Touch for other USB options.

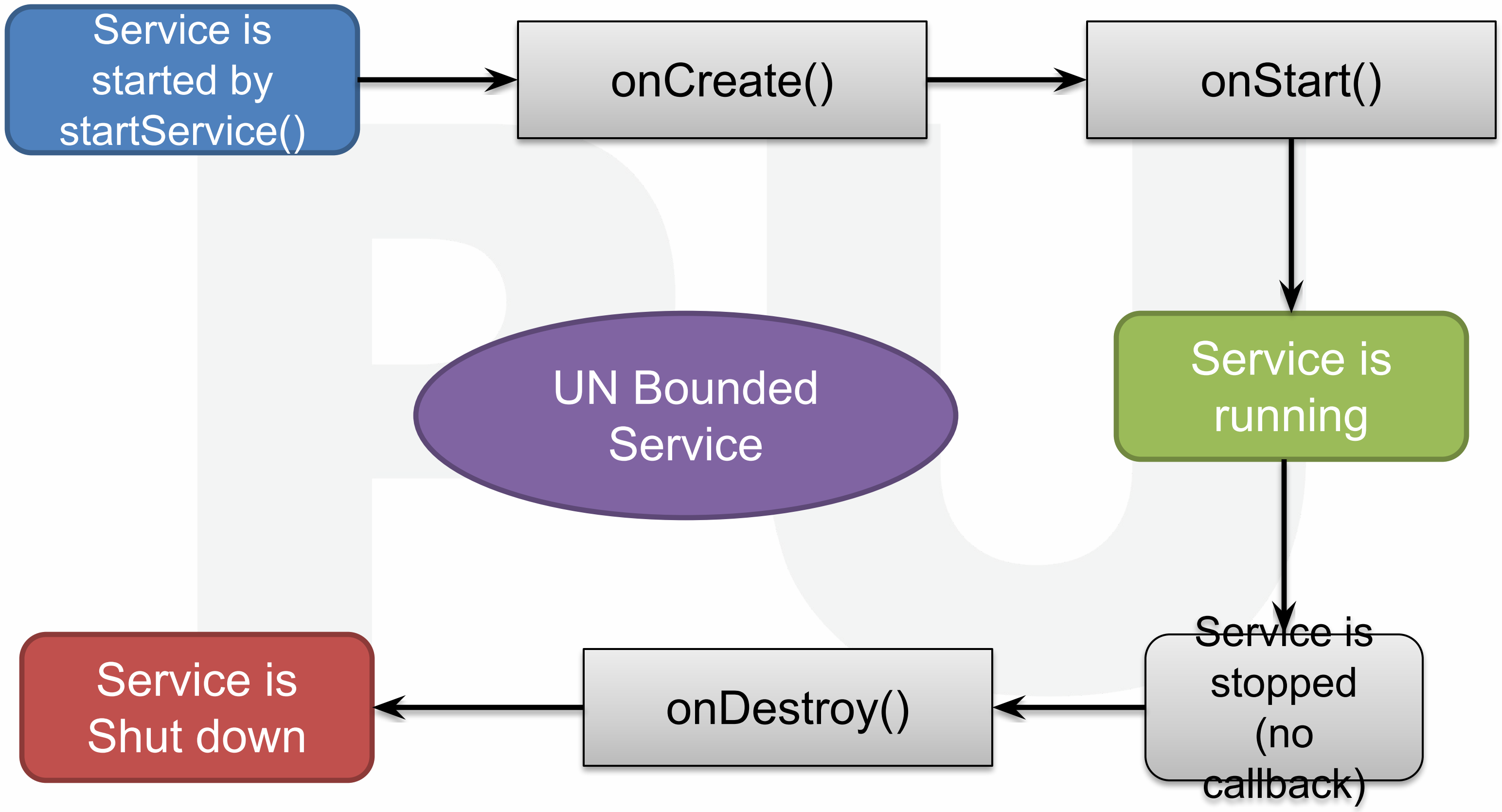
Wide touch • 57m
Wide Touch is in the notification bar
Click to return to home screen

Service States

- **Started -** A service is **started** when an application component, such as an activity, starts it by calling *startService()*. Once started, a service can run in the background indefinitely, even if the component that started it is destroyed.

- **Bound -** A service is **bound** when an application component binds to it by calling *bindService()*. A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with interprocess communication (IPC).

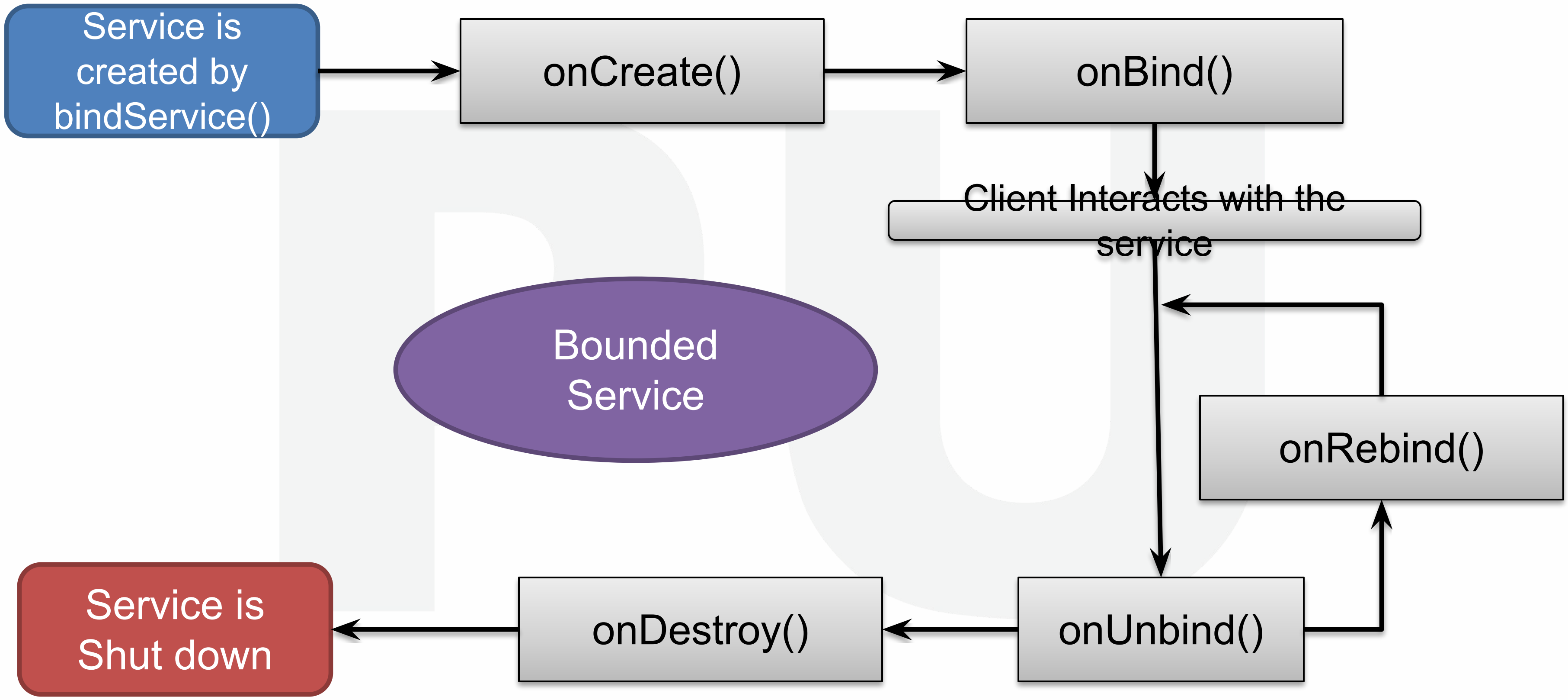
UNBounded Service Lifecycle



Parul® University

DIGITAL LEARNING CONTENT

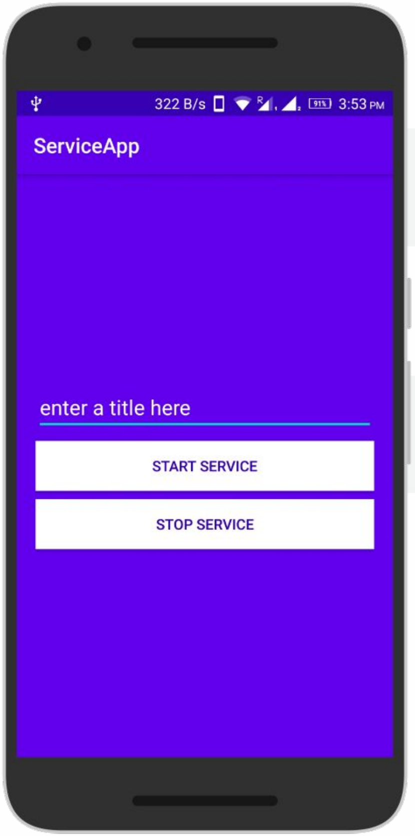
Bounded Service Lifecycle

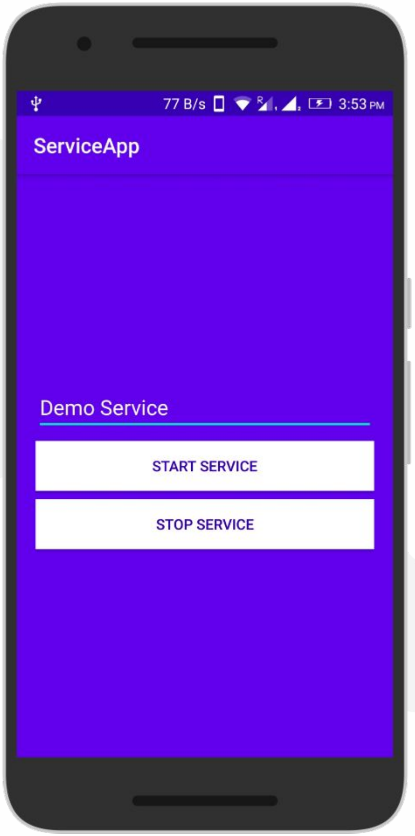


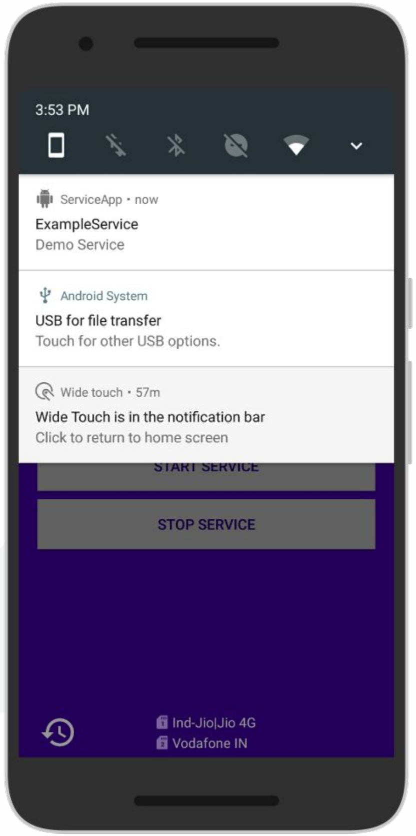
Service Callbacks

- onStartCommand()
- onBind()
- onUnbind()
- onRebind()
- onCreate()
- onDestroy()

Example

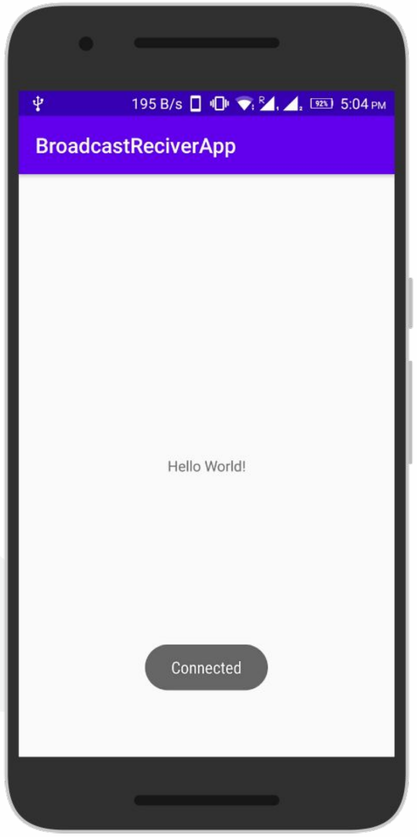

322 B/s 3:53 PM
ServiceApp
enter a title here
START SERVICE
STOP SERVICE


77 B/s 3:53 PM
ServiceApp
Demo Service
START SERVICE
STOP SERVICE


3:53 PM
ServiceApp • now
ExampleService
Demo Service
Android System
USB for file transfer
Touch for other USB options.
Wide touch • 57m
Wide Touch is in the notification bar
Click to return to home screen
START SERVICE
STOP SERVICE
Ind-Jio|Jio 4G
Vodafone IN

Broadcast Receivers

Broadcast Receivers simply respond to broadcast messages from other applications or from the system itself. These messages are sometime called events or intents. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.



Broadcast Receivers

There are following two important steps to make BroadcastReceiver works for the system broadcasted intents:

1. Creating the Broadcast Receiver.
2. Registering Broadcast Receiver

Creating Broadcast Receivers

A broadcast receiver is implemented as a subclass of **BroadcastReceiver** class and overriding the onReceive() method where each message is received as a **Intent** object parameter.

Registering Broadcast Receivers

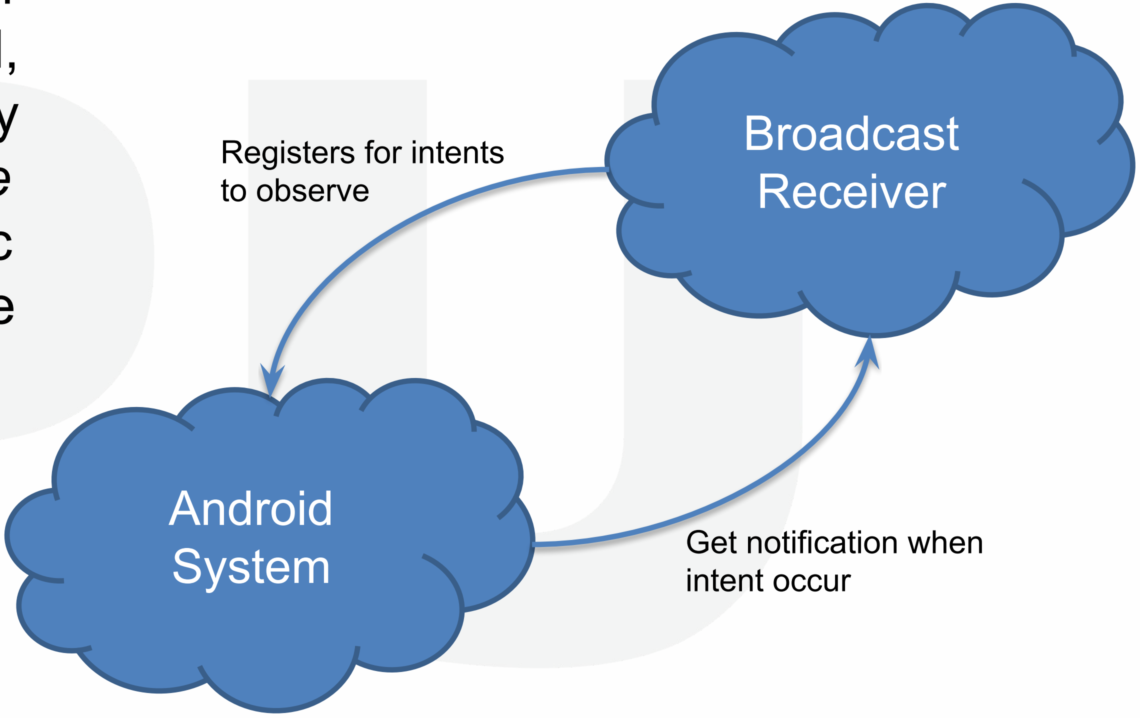
An application listens for specific broadcast intents by registering a broadcast receiver in *AndroidManifest.xml* file. Consider we are going to register *MyReceiver* for system generated event ACTION_BOOT_COMPLETED which is fired by the system once the Android system has completed the boot process.

Parul® University

DIGITAL LEARNING CONTENT

Registering Broadcast Receivers

Now whenever your Android device gets booted, it will be intercepted by BroadcastReceiver *MyReceiver* and implemented logic inside *onReceive()* will be executed.

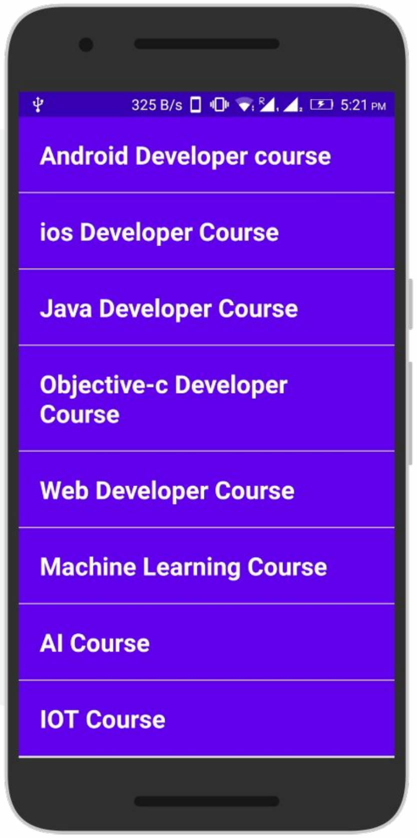

Registers for intents to observe
Broadcast Receiver
Android System
Get notification when intent occur

Events Broadcast Receivers

- android.intent.action.BATTERY_CHANGED
- android.intent.action.BATTERY_LOW
- android.intent.action.BATTERY_OKAY
- android.intent.action.BOOT_COMPLETED
- android.intent.action.BUG_REPORT
- android.intent.action.CALL
- android.intent.action.CALL_BUTTON
- android.intent.action.DATE_CHANGED
- android.intent.action.REBOOT

ListView

Android ListView is a view which groups several items and display them in vertical scrollable list. The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database.



Android Developer course

ios Developer Course

Java Developer Course

Objective-c Developer Course

Web Developer Course

Machine Learning Course

AI Course

IOT Course

ListView

An adapter actually bridges between UI components and the data source that fill data into UI Component. Adapter holds the data and send the data to adapter view, the view can takes the data from adapter view and shows the data on different views like as spinner, list view, grid view etc.

ListView

The **ListView** and **GridView** are subclasses of **AdapterView** and they can be populated by binding them to an **Adapter**, which retrieves data from an external source and creates a View that represents each data entry.

ListView

Android provides several subclasses of Adapter that are useful for retrieving different kinds of data and building views for an AdapterView (i.e. ListView or GridView). The common adapters are

- ArrayAdapter
- BaseAdapter
- CursorAdapter
- SimpleCursorAdapter
- SpinnerAdapter
- WrapperListAdapter

ListView Attributes

1. android:id
2. android:divider
3. android:dividerHeight
4. android:entries
5. android:footerDividersEnabled
6. android:headerDividersEnabled

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in