

ANDROID

CH - 4

**Location Based Services (LBS) ,
Common Android API,
Notification, Services &
Deployment of App.**

Prepared By :

Ms. Kakadiya Jainam A.

POINTS OF LEARN :

- Introduction to LBS**
- Using Global Positioning Services (GPS)**
- Location and Maps**




5.1 BASIC INTRODUCTION

- A GPS receiver calculates its position by precisely timing the signals sent by GPS satellites high above the Earth.
- Each satellite continually transmits messages that include :
 - The time the message was transmitted and,
 - Satellite position at time of message transmission.



GLOBAL POSITIONING SYSTEM (GPS)

- The GPS project was developed in 1973 to overcome the limitations of previous navigation systems, integrating ideas from several predecessors, including a number of classified engineering design studies from the 1960s.
 - The **Global Positioning System (GPS)** is a space-based satellite navigation system that provides location and time information in all weather conditions, anywhere on or near the Earth.
- 

How GPS WORKS ?

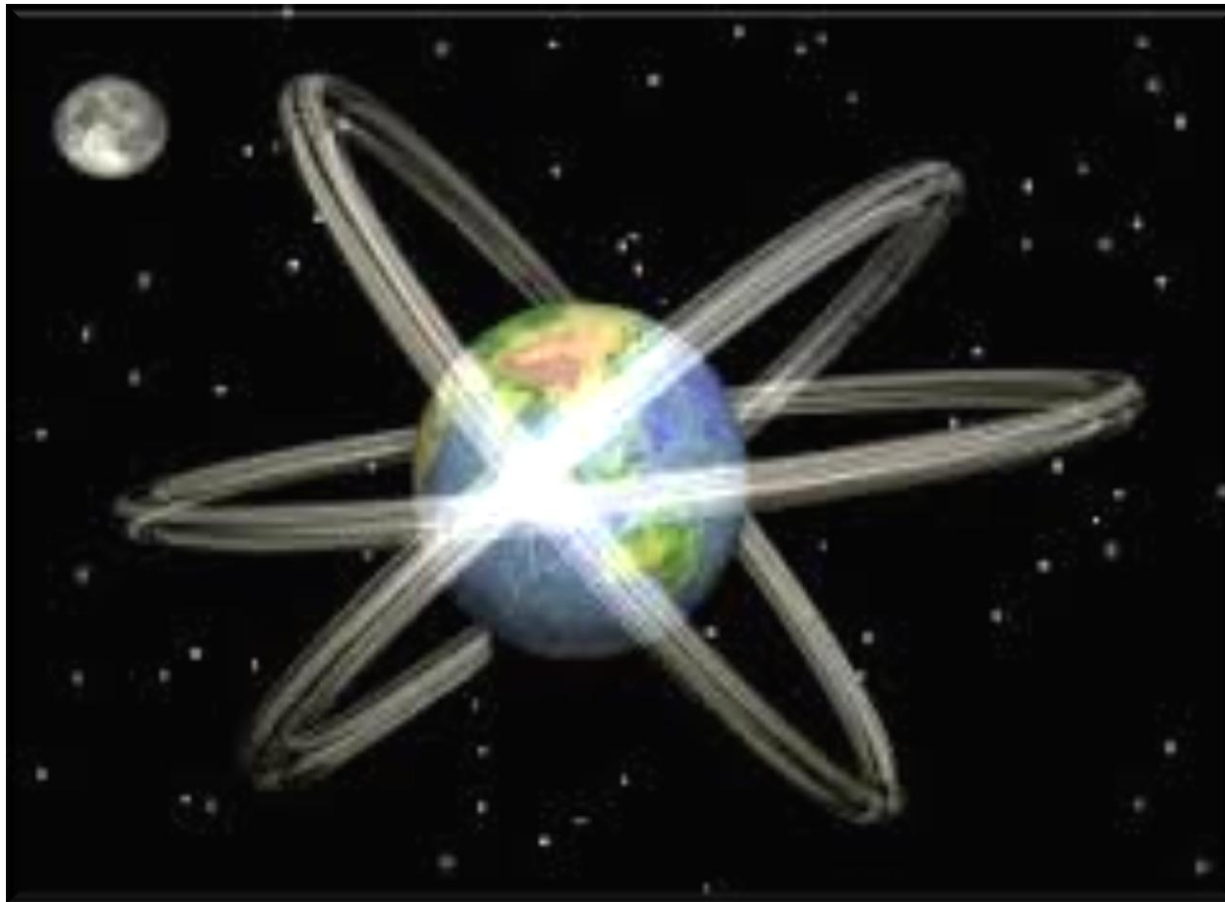
- The receiver uses the messages it receives to determine the transit time of each message and computes the distance to each satellite using the speed of light.
- Each of these distances and satellites locations defines a area.
- The receiver is on the surface of each of these areas when the distances and the satellite's locations are correct.



- These distances and satellites location are used to compute the location of the receiver using the navigation equations. This location is then displayed, perhaps with a moving map display or **Latitude** (અક્ષાંશ) and **Longitude** (રેખાંશ) .
- Basic GPS measurements surrender only a position, and neither speed nor direction. However, most GPS units can automatically derive speed and direction of movement from two or more position measurements.



- A visual example of a 24 satellites GPS constellation in motion with the Earth rotating. Each circle contain 8 satellites for continues monitoring...



5.2 USING GLOBAL POSITIONING SERVICES (GPS)

- The android SDK provides resources for accessing location via a built-in GPS hardware, if available in hardware. Generally speaking, just about every Android phone has some LBS (Location Based Services) capabilities.
- For example, in the United States, mobile phone location information is used by emergency services. That said, not all android devices are phones, nor do all phones enable consumer-usage of LBS services. ●

- If GPS features are disabled, or an Android device does not have LBS hardware, the Android SDK provides additional APIs for determining alternate location providers.



5.2.1 WORKING WITH GPS


- LBS services and hardware such as a built-in accuracy GPS are optional features for Android devices. In addition to requiring the appropriate permissions, you can specify which optional features your application requires within the Android manifest file.
- If your application will only function well on devices with some sort of method for determining the current location, you could use the following `<uses-feature>` tag in your application's manifest file :
- `<uses-feature android:name "android.hardware.location" />`

5.2.2 GEOCODING LOCATIONS

- GeoCoding is a processes of assigning locations to addresses to that they can be placed as points on a map, similar to putting pins on a paper map, and analyzed with other spatial data. The process assigns geographic coordinates to the original data, hence the name geocoding.
- The Android platform API provides a feature that returns an estimated street addresses for latitude and longitude values.



5.2.3 DEFINE THE ADDRESS LOOKUP TASK

- To get an address for a given latitude and longitude, call `Geocoder.getFromLocation()`, which returns a list of addresses.
- The method is synchronous, and may take a long time to do its work, so you should call the method from the `doInBackground()` method of an `AsyncTask`.
- Let's consider the example to get location name based on longitude and latitude. 

- **Following the steps :**

- 1) Add following line of code to you manifest file.

- `<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />`

- `<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />`

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

- 2) Add xml file.

- 3) Add Java file.



5.3 LOCATION AND MAPS

- Location and map-based apps offer a compelling experience on mobile devices.
- You can build these capabilities into your app using the classes of the `android.location` package and the Google Maps Android API.
- The sections below provides an introduction to how you can add the features.



5.3.1 LOCATION SERVICES


- Android gives your applications access to the location services supported by the device through classes in the `android.location` package and the Google Maps Android API. The sections below provides an introduction to how you can add the features.
- The central component of the location framework is the `LocationManager` system service, which provides APIs to determine location and behaviour of the basic device.



- As with other system services, you do not instantiate a `LocationManager` directly. Rather, you request an instance from the system by calling `getSystemService(Context.LOCATION_SERVICE)`. The method returns a handle to a new `LocationManager` instance.



5.3.2 GOOGLE MAPS ANDROID API

- With the Google Maps Android API, you can add maps to your app that are based on Google Maps data. The API automatically handles access to Google Maps servers, data downloading, map display, and touch gestures on the map.
- The key class in the Google Maps Android API is `MapView`. A `MapView` displays a map with data obtained from the Google Maps services. 

POINTS OF LEARN :

- **Connection to the Network**
- **Managing network usage**
- **Android Web API**
- **Building Web Apps in WebView**




INTRODUCTION TO ANDROID NETWORK API

- API class includes basic tasks involves in connecting to the network, monitoring the network connection(including connection changes), and giving control over an app's network usage.
- API are describes how to parse and consume XML data.



6.1 CONNECTING TO THE NETWORK

- Here we will see how to implement a simple application that connects to the network.
 - To perform the network operations described here, your application manifest must include the following permissions :
 - `<uses-permission android:name="android.permission.INTERNET">`
 - `<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE">`
- 

6.1.1 CHOOSE AN HTTP CLIENT

- Most network-connected Android apps are HTTP to send and receive data. Android includes two HTTP clients : HttpURLConnection and Apache HttpClient.
- Support HTTPS, streaming uploads and downloads, configurable timeout, IPv6, and connection pooling. We recommend using HttpURLConnection for applications targeted at Gingerbread and higher.




6.1.2 CHECK THE NETWORK CONNECTION

- Before your app attempts to connect to the network, it is good practice to check whether a network connection is available using `getActiveNetworkInfo()` and `isConnected()`.
- Remember, the device may be out of range of network, or the user may have disabled both Wi-Fi and mobile data access.



6.1.2 CHECK THE NETWORK CONNECTION

- Let's consider a basic example to check if network is available or not.
 - 1) Define permission in Manifest.xml :
 - `<uses-permission android:name="android.permission.INTERNET">`
 - `<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE">`
 - 2) Create Activity_main.xml
 - 3) Create MainActivity.java
- 

6.1.3 PERFORM NETWORK OPERATIONS A SEPARATE THREAD

- Network operations can involve unpredictable delays. To prevent this from causing a poor user experience, always perform network operations on a separate thread from the UI.
- AsyncTask class provides one of the simplest ways to fire off a new task from the UI thread.
- In this section, the myClickHandler() method invokes new DownloadWebpageTask() execute. The DownloadWebpageTask class is a subclass of AsyncTask.

- **DownloadWebpageTask** implements the following AsyncTask methods:
 - **1) doInBackground()** executes the method `downloadUrl()`. It passes the web page URL as a parameter.
 - **2) onPostExecute()** takes the returned string and displays it in the UI.



○ Now lets consider an example that will display data from your webpage.

○ **1) provide following uses permission to your application :**

- `<uses-permission android:name="android.permission.INTERNET">`

- `<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE">`

○ **2) Make the Activity_main.xml file**

○ **3) Add code to MainActivity.java**



6.1.4 CONNECT AND DOWNLOAD DATA

- In your thread that performs your network transactions, you can use `HttpURLConnection` to perform a GET and download your data. After you call `connect()`, you can get an `InputStream` of the data by calling `getInputStream()`.
- The `doInBackground()` method calls the method `downloadUrl()`. The `downloadUrl()` method takes the given URL and uses it to connect to the network via `HttpURLConnection`. Once a connection has been established, the app uses the method `getInputStream()` to retrieve the data as an `InputStream`.

6.1.5 CONVERT THE INPUT STREAM TO A STRING

- An InputStream is a readable source of bytes. Once you get an inputStream, it's common to code or convert it into a target data type.
- For example, if you were downloading image data, you might decode and display it like this:
- This example converts the InputStream to a string so that the activity can display it in the UI :



- o //read an InputStream and converts it to a String.

```
public String readIt(InputStream stream, int len)
throws IOException,
UnsupportedEncodingException
{
    Reader reader=null;
    reader=new InputStreamReader(stream, "UTF-8");
    char[] buffer=new char[len];
    reader.read(buffer);
    return new string(buffer); }
```



6.2 MANAGING NETWORK USAGE

- If your application performs a lot of network operations, you should provide user settings that allow users to control your app's data habits, such as how often your app syncs data, whether to perform uploads/downloads only when on Wi-Fi, and so on.
- With these controls available to them, users are much less likely to disable your app's access to background data when they approach their limits, because they can instead precisely control how much data your app uses.

6.2.1 CHECK A DEVICE'S NETWORK CONNECTION

- A device can have various type of network connections. At present we focus on Wi-Fi and Data connection. Wi-Fi is typically faster. Also, mobile data is faster if you are using 3g or in near future 4g.
- To check the network connection, you typically use the following classes :
 - **ConnectivityManager** : Answer queries about the state of network connectivity. It also notifies applications when network connectivity changes.

6.2.1 CHECK A DEVICE'S NETWORK CONNECTION

- **NetworkInfo** : Describe the status of the network interface of given type (currently either mobile or Wi-Fi).
- This code section tests network connectivity for Wi-Fi and mobile. It determines whether these network interface are available or not.





```
Package com.bagdais.chknet;  
Import com.bagdais.chknet.r.id;  
Import android.net.ConnectivityManager;  
Import android.net.NetworkInfo;  
Import android.net.Bundle;  
Import android.net.app.Activity;  
Import android.net.content.Context;  
Import android.view.Menu;  
Import android.widget.TextView;
```

```
Public class MainActivity extends Activity  
{  
    TextView tv;
```




```
Protected void onCreate(Bundle savedInstanceState)
{ super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  tv=(TextView)findViewById(id.textview1);
  ConnectivityManager
  cm=(ConnectivityManager)getSystemService(conte
  xt.CONNECTIVITY_SERVICE);
  NetworkInfo
  ni=cm.getNetworkInfo(ConnectivityManager.TYPE_
  MOBILE);
  boolean isMobileConn=ni.isConnected();
  string finalout="wifi connected"+isWifiConn+"\n
  mobile connected"+isMobileConn;
  tv.setText(finalout); } }
```



6.2.2 MANAGE NETWORK USAGE

- You can implement a preferences activity that gives users explicit control over your app using network resources. For example :
 - You might allow users to upload videos only when the device is connected to a Wi-Fi network.
 - You might sync(or not)depending on specific criteria such as network availability, time interval, and so on.



- To write an app that supports network access and managing network usage, your manifest must have right permissions and intent filters.
 - The manifest excerpted below includes the following permissions :
 - **Android.permission.INTERNET** : Allows applications to open network socket.
 - **Android.permission.ACCESS_NETWORK_STATE** : Allows applications to access information about networks.
- 

6.2.3 IMPLEMENT A PREFERENCES ACTIVITY

- The sample app's activity `UserSettingActivity` is a subclass of `PreferenceActivity`. It displays a preferences screen that lets users specify the following :
 - Whether to display summaries for each XML feed entry, or just a link for each entry.
 - Whether to download the XML feed if any network connection is available, or only if Wi-Fi is available.

STEPS TO CREATE APPLICATION

- [1] Define permission in Android manifests file
- [2] Create **activity_main.xml** file
- [3] Create **menu.xml** to display settings options to user
- [4] Create on **xml** folder under **res**. And add settings.xml file
- [5] Create **Arrays.xml** file under **res/values**
- [6] Modify **string.xml** file in **res/values**
- [7] Create file named **UserSettingActivity.java** under **src** folder.
- [8] Modify **MainActivity.java** file



6.3 ANDROID WEB API

- **Web Apps** : There are essentially two ways to deliver an application on Android as a client-side application (developed using the Android SDK and installed on user devices in an APK) or as a web application.
- If you chose to provide a web-based app for Android-powered devices, you can rest assured that major web browsers for Android allow you to specify viewport and style properties that make your web pages appear at the proper size and scale on all screen configuration.

6.4 BUILDING WEB APPS IN WEBVIEW

- If you want to deliver a web application as a part of client application, you can do it using WebView. The WebView class is an extension of Android's View class that allows you to display web page as part of your activity layout.
- It does not include any features of a fully developed web browser, such as navigation controls or an address bar. All that WebView does, by default, is show web page.



- A common scenario in which using WebView is useful is when you want to provide information in your application that you might need to update, such as an end-user agreement or a user guide.
- Within your Android application, you can create an Activity that contains a WebView, and that to display your document that's hosted online.
- Another scenario in which WebView can help is if your application provides data to the user that always requires an Internet connection to retrieve data, such as email.

6.4.1 ADDING A WEBVIEW TO YOUR APPLICATION

- To add a WebView to your Application, simply include the <WebView> element in your activity layout. For example, here's a layout file in which the WebView fills the screen :
- <WebView
xmlns:android=http://schemas.android.com/apk/res/android
android:id="@+id/webview"
android:layout_width="fill_parent"
android:layout_height="fill_parent"

/>



- To load a web page in the WebView use `loadUrl()`. For example :
 - `Webview mywebview = (WebView) findViewById(R.id.webview);`
- Before this will work, however, your application must have access to the Internet. To get Internet access, request the `INTERNET` permission in your manifest file. For example :
 - `<manifest....>`
 - `<uses-permission android:name="android.permission.INTERNET"> </manifest>`

6.4.2 USING JAVASCRIPT IN WEBVIEW

- If the web page you plan to load in your WebView use JavaScript, you must enable JavaScript for your WebView.
- Once JavaScript is enable, you can also create interfaces between your application code and your JavaScript code.
 - **Enabling JavaScript** : JavaScript is disable in a WebView by default. You can enable it through the WebSettings attached to your WebView.

- You can retrieve WebSettings with `getSettings()`, then enable JavaScript with `setJavaScriptEnabled()`.

- **For example :**

- `WebView mywebview=(WebView) findViewById(R.id.webview);`

- `WebSettings websettings= mywebview.getSettings();`

- `WebSettings.setJavaScriptEnable(true);`



6.4.3 HANDLING PAGE NAVIGATION

- When the user clicks a link from a web page in your `WebView`, the default behaviour is for Android to launch an application that handles URLs. You can allow the user to navigate backward and forward through their web page history that's maintained by your `WebView`.
- To open links clicked by the user, simply provide a `WebViewClient` for your `WebView`, using `setWebViewClient()`.



○ For example :

- `WebView mywebview=(WebView) findViewById(R.id.webview);`

```
mywebview.setWebViewClient(new  
WebViewClient());
```

- **Navigation web page history :**

- When your `WebView` overrides URL loading, it automatically collect a history of visited web page. You can navigate backward and forward through the history with `goBack()` and `goForward()`.

6.5 ANDROID TELEPHONY API

- As a software developer for mobile platform, you may be interested in incorporating telephony features in your app.
- Telephony manager provides access of information about the telephony services on the device.
- You do not instantiate this class directly instead, you retrieve a reference to an instance through `Context.getSystemService(Context.TELEPHONY_SERVICE)`

- **Permission Required** : To work with Telephony Manager and to read the phone details we need to add this permission statement in AndroidManifest file.

- `<uses-permission android:name="android.permission.READ_PHONE_STATE">`



6.5.1 GET PHONE STATUS


- Using Telephony Manager we can retrieve the current status of mobile phone and also get some basic information about device.
- Following statement will give you current telephony object using that you can interact.
 - `TelephonyManager tm=(TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);`



6.5.2 GET PHONE TYPE CDMA/GSM/NONE

- Get the type of network you are connected with

```
Int phoneType=tm.getPhoneType();
switch(phoneType)
{ case (TelephonyManager.PHONE_TYPE_ CDMA)
// YOUR CODE ; break;
case (TelephonyManager.PHONE_TYPE_ GSM)
// YOUR CODE ; break;
case (TelephonyManager.PHONE_TYPE_ CDMA)
// YOUR CODE
break; }
```



- Find whether the Phone is in Roming, return true if in roming :

```
if(isRoming)
```

```
    phoneDetails+="Is in Roming"+"YES";
```

```
else
```

```
    phoneDetails+="Is in Roming"+"No";
```



GET THE SIM STATE/DETAILS

- Using the Object of Telephony manager class we can get the details like SIM serial number, Country code, Network Provider code and other Details.

```
int SIMstate=tm.getSimState();
switch(SIMstate)
{
case TelephonyAMnager.SIM_STATE_ABSENT
// YOUR CODE
Break;
```



```
case TelephonyAMnager.SIM_STATE_NETWORK_
LOCKED
```

```
    // your code
```

```
    break;
```

```
Case TelephonyAMnager.SIM_STATE_PIN_REQUEST
```

```
    // your code
```

```
    break;
```

```
case TelephonyAMnager.SIM_STATE_PUK_REQUEST
```

```
    // your code
```

```
    break;
```

```
case
```

```
    TelephonyAMnager.SIM_STATE_STATE_UNKNOWN
```

```
    // your code
```

```
    break;
```

POINTS OF LEARN :

- **Introduction**
- **Building a Notification**
- **Updating Notification**



7.1 INTRODUCTION TO NOTIFICATION

- User needs to be updating even application is not Active at present so that user don't miss the important news, mails, or messages.
- We can notify user using different techniques like Vibrating the phone, Ringing, Blinking etc.



7.1.1 NOTIFYING THE USER

- A notification is a user interface element that you display outside your app's normal UI to indicate that an event has occurred.
- Users can choose to view the notification while using other apps.



7.1.2 NOTIFYING WITH THE STATUS BAR

- The standard location for displaying notifications and indicators on an Android device is the status bar that runs along the top of the screen.
- Typically, the status bar shows information such as the current date and time.
- It also displays notifications like incoming SMS message.



7.2 BUILDING A NOTIFICATION

○ 1) Creating a Notification Builder :

- When creating a notification specify UI content and actions with a `NotificationCompat.Builder` object.

- **For example :**

- `NotificationCompat.Builder ncb=new NotificationCompat.Builder(this).setSmallIcon(R.drawable.ic_luncher).setContentTitle("From Notif").setContentText("Welcome");`

- 1) A small icon, set by **`setSmallIcon()`**

- 2) A title, set by **`setContentTitle()`**

- 3) Detail text, set by **`setContentText();`**



7.2 BUILDING A NOTIFICATION

○ 2) Define the Notification's Action :

- Although actions are optional, you should add at least one action to your notification.
- An action takes users directly from the notification to an Activity in your application, where they can look at the event that caused the notification or do further work.



7.2 BUILDING A NOTIFICATION

- **3) Preserving Navigation when Starting an Activity :**
 - When you start an Activity from a notification, you must preserve the user's expected navigation experience. Clicking back should take the user back through the application's normal work flow to home screen.
 - To preserve the navigation experience, there are two general situation : **1) Regular Activity**
 - **2) Special Activity**

7.3 UPDATING NOTIFICATIONS

- **1) Modify a Notification** : To set up a notification so it can be updated, issue it with a notification ID by calling **NotificationManager.notify(ID,notification).**
- **Example of updating notification**
- **Intent inte= new Intent(this, setActivity.class)
PendingIntent
pi=PendingIntent.getActivity(this ,0,inte,
PendingIntent. FLAG_UPDATE_CURRENT);
ncb.setContentIntent(pi);**

7.3 UPDATING NOTIFICATIONS

- **2) Vibrating the Phone** : Making Android phones vibrate is a good way to provide feedback to users or to interact with users even when phone volume is low.
- Android phone to vibrate in one of the following Vibrating methods:
 - **i) Grant vibrating permissions** : Before you start adding the code necessary to cause your application to vibrate, you must first notify Android that your application expects to have permission to use the vibrator.

- Add the uses-permission line to your Manifest.xml.
- `<uses-permission android:name="android.permission.VIBRATE"/>`
- **ii) Vibrating in a given pattern** : This method of vibration is useful when you need to user with a one-time notification. Such as receiving a text message.
- **iii) Vibrating Repeatedly Until Cancelled** : This method of vibrating is useful when you need to notify the user of something that requires more immediate action, such as an incoming phone call.

- **3) Blinking the lights:** Blinking lights are a great way to pass information silently to the user when other forms of alert are not appropriate.
- The SDK provides reasonable control over a multicolored indicator light, when such a light is available on the device.
- **For example :**
notificationCompat.Builder ncb=new
NotificationCompat.Builder(this).setLights(Color.
GREEN,1,1);



- **4) Customizing the Notification** : Although the default behavior in the expanded status bar tray is sufficient for most purpose, developers can customize how notifications are displayed if they so choose.
- **To do so, developers can use the RemoteView object to customize the notification :**
- `RemoteViews remote=new RemoteViews(getPackageName(),R.layout.remote);
remote.setTextviewText(R.id.text1, “Text here”);`

POINTS OF LEARN :

- **Services**
- **Creating a Started Service**
- **Bound Services**
- **How to connect Android with PHP, MYSQL**



8.1 INTRODUCTION TO SERVICES

- A service is an application component that can perform long-running operations in the background and does not provide a user interface.
- For example, a service might handle network transactions, play music, perform file I/O etc.
- **A service can essentially two forms :**
 - 1) Started
 - 2) Bound



- **1) Started :**

- A service is “started” when an application component (Such as an activity) starts it by calling `startService()`.

- **2) Bound :**

- A service “bound” when an application component binds to it by calling `bindService()`.



8.2 CALLBACK METHODS

- To create a service, you must create a subclass of Service. In your implementation, you need to override some callback methods that handle key aspects of the service lifecycle and provide a mechanism for components to bind to the service. The most important callback methods are :
 - **1) onStartCommand()** : The system calls this method when another component, such as an activity, requests that the service be started, by calling startService().


- **2) onBind()** : The system calls this method when another component wants to bind with the service (such as to perform RPC), by calling `bindService()`.
- **3) onCreate()** : The system calls this method when service is first created, is first created.
- **4) onDestroy()** : The system calls this method when the service is no longer used and is being destroyed.




8.2.1 DECLARING A SERVICE IN THE MANIFEST

- To declare your service, add a `<service>` element as a child of the `<application>` element.

- **For example :**

- `<manifest>`
 - `<appliccation>`
 - `<service`
`android:name=".ExampleService"/>`
 - `</appliccation>`
 - `</manifest>`
- 

8.3 CREATING A STANDARD SERVICE

- A standard service is one another component starts by calling `startService()`, resulting in a call to the service's `onStartCommand()` method.
 - Traditionally, there are two classes you can extend to create a standard service.
 - **1) Service** : This is the base class for all service, when you extends this class, it's important that you create a new thread in which to do all the service's work.
- 

- **2) IntentService:** This is a subclass of Service that uses a worker thread to handle all start requests, one at a time. This is the best option if you don't required that your service handle multiple request simultaneously.
- The onStartCommand() method must return an integer.
- The return value from onStartCommand() must be one of the following constants :



- **i) START_NOT_STICKY** : If the system kills the service after onStartCommand() returns, do not recreate the service.
- **ii) START_STICKY** : If the system kills the service after onStartCommand() returns recreate the service and call onStartCommand().
- **ii) START_REDELIVER_INTENT** : If the system kills the service after onStartCommand() returns recreate the service and call onStartCommand() with the last was delivered to the service.

○ 3) Starting a Service :

- You can start service to activity or another application component by passing an Intent to `startService()`.

○ For example :

```
Intent int=new Intent(this, HelloService.class);  
startService(int);
```



○ 4) Creating a Bound Service :


- A bound service is one that allows application components to bind to it by calling `bindService()` in order to create a long-standing connection.
- You should create a bound service when you want to interact with the service from activities and other components in your application, through interposes communication(**IPC**).
- To create a bound service, you must implement the **onBind()** to return `Ibinder` that defines the interface for communication with the service.

8.4 BOUND SERVICE

- A bound service is the server in a client-side interface. A bound service allows components to bind to the service.
- A bound service is an implementation of the Service class that allows other applications to bind to it and interact with it.
- To provide binding for a service, you must implement the onBind() method.



8.4.1 CREATING A BOUND SERVICE

- When creating a service that provides binding, you must provide an IBinder that provides the programming interface that clients can use to interact with the service.
 - There are three ways you can define a interface :
 - 1) Extending the Binder class
 - 2) Using a Messenger
 - 3) Using AIDL (Android Interface Definition Language).
- 

8.4.2 MANAGING THE LIFECYCLE OF A BOUND SERVICE

- When a service is unbound from all clients, the Android system destroys it. As such, you don't have to manage the lifecycle of your service if it's purely a bound service.
- The Android system manages it for you based on whether it is bound to any client.



8.5 HOW TO CONNECT ANDROID WITH PHP, MYSQL

- We are going see how to make a very simple Android app that will call a PHP script to basic CRUD (Create, Read, Update, Delete) operation.
- The PHP script then connects to your MySQL database to perform the operation.
- So the data flow from your Android app to PHP script then finally is stored in your MySQL db. ●

○ 1) WAMP Server :

- WAMP software is one click installer which creates an environment for developing PHP, MySQL web application. By installing this software you will be installing Apache, MySQL and PHP. Alternatively can use XAMP server also.


○ 2) Installing and Running WAMP Server :

- Download & Install WAMP server from www.wampserver.com/en.



- Once you have installed wamp server, launch the program deom start -> All Programs -> WampServer -> StartWampServer.

○ 3) **Creating and Running PHP Project :**

- Now you have the environment ready to develop a PHP & MySQL project.
- Go to the location where you installed WAMP server and go WWW folder and create a new for your project. You have to place all your project files inside this folder. 

- After placing following code try to open <http://localhost/Bagdaisexa/test.php> and you should see a message called “Welcome, Example from Bagdais to connect with PHP, MySQL”.

- **Test.php**

```
<?php
```

```
    Echo “Example from Bagdais to connect with  
    PHP, MySQL”
```

```
?>
```



○ 4) Creating MySQL Database and Tables :

- Create a simple db with one table. Now open phpmyadmin by opening the address <http://localhost/phpmyadmin/> in your browser.
- You can use the PhpMyadmin tool to create a database and a table.
- ```
CREATE DATABASE bagdaisdb;
CREATE TABLE product(pid int(10), name
varchar(20),price decimal(10,2));
```



- **5) Connecting to MySQL database using PHP**
- **6) Basic MySQL CRUD Operation using PHP**
- **7) Creating Android Application**



## 9 DEPLOYMENT OF APPLICATIONS

- Android application publishing is a process that makes your Android applications available to users.
- Infact, publishing is the last phase of the Android application development process.

Design

Coding

Testing

Publish



- Here is a simplified check list which will help you in launching your Android application:
- **1) Is Application tested ?**
  - If the answer to this is “no” then you need to test it again and do some precise testing. In addition to basic tests on the emulator, you should also test your application on real, physical device.






## ○ 2) Application Performs well

- Performance is really important, especially if you're programming a game. If your application is not responsive enough in certain cases, try to see if you can optimize those parts.

## ○ 3) Checked SDK compatibility

- According to data collected around August 12, 2014, Android 2.2 is still active on 0.7% devices that have accessed the Market.
- 

| <b>Version</b>       | <b>Code name</b>          | <b>Release date</b>  | <b>API level</b> | <b>Distribution</b> |
|----------------------|---------------------------|----------------------|------------------|---------------------|
| <b>4.4</b>           | <b>Kitkat</b>             | <b>Oct. 31,2013</b>  | <b>19</b>        | <b>20.9 %</b>       |
| <b>4.3 – 4.2</b>     | <b>Jelly Bean</b>         | <b>July 24, 2013</b> | <b>18 , 17</b>   | <b>7.9 %</b>        |
| <b>4.1</b>           | <b>Jelly Bean</b>         | <b>July 9,2012</b>   | <b>16</b>        | <b>26.5 %</b>       |
| <b>4.0.3 – 4.0.4</b> | <b>Ice Cream Sandwich</b> | <b>Dec. 16, 2011</b> | <b>15</b>        | <b>10.6 %</b>       |
| <b>2.3.3-2.3.7</b>   | <b>Gingerbread</b>        | <b>Feb. 9,2011</b>   | <b>10</b>        | <b>13.6 %</b>       |
| <b>2.2</b>           | <b>Froyo</b>              | <b>May 20, 2010</b>  | <b>8</b>         | <b>0.7 %</b>        |

## ○ Now check configuration for final launch

### ○ 1) Request necessary Android permissions

- `<uses-permission android:name="android.permission.VIBRATE">`
- `<uses-permission android:name="android.permission.INTERNET">`

### ○ 2) Specify a name and icon

- `<application android:label="@string/app_name" android:icon="@drawable/myIcon">`

### ○ 3) Configure version manifest data

- `<manifest xmlns:android=http://schemas.android.com/apk/res/android package="com.example" android:versionCode="1" android:versionName="1.0.0">`

### ○ 4) Set compatibility options

- **android:minSdkVersion** The minimum Android platform API level on which your application will be able to run.



- **android:targetSdkVersion** The API level that your application was designed to run on.
- **android:maxSdkVersion** An upper limit for compatibility.

## ○ 5) Cleanup files and remove logging

- Go through your project and remove any logging calls, old files, private data and resource files.



## ○ 6) Export Android Application

- To export an application, just open that application project in Eclipse and select **File->Export from your Eclipse and follow the simple steps to export your application:**



# Export

Select



Select an export destination:

type filter text

- ▶ General
- ▼ Android
  - Export Android Application**
  - Generate Gradle build files
- ▶ C/C++
- ▶ Install
- ▶ Java
- ▶ Run/Debug
- ▶ Team
- ▶ XML
- ▶ Other



< Back

Next >

Cancel

Finish

- 7) Next select, **Export Android Application** option as shown in the above screen shot and then click **Next** and again **Next** so that you get following screen where you will choose **Create new keystore** to store your application.





## Select Keystore Name

Save As:

Where:

Cancel

Save

### Keystore selection

 Enter path to keystore.

Use existing keystore

Create new keystore

Location:

Browse...

Password:

Confirm:



< Back

Next >

Cancel

Finish


- 8) Enter your password to protect your application and click on **Next button once again. It will display following screen to let you create a key for your application:**



## Export Android Application

### Key Creation



 A 25 year certificate validity is recommended.

|                      |                                                 |
|----------------------|-------------------------------------------------|
| Alias:               | <input type="text" value="PhoneDemo"/>          |
| Password:            | <input type="password" value="••••••••"/>       |
| Confirm:             | <input type="password" value="••••••••"/>       |
| Validity (years):    | <input type="text" value="2"/>                  |
| First and Last Name: | <input type="text" value="John Imrahim"/>       |
| Organizational Unit: | <input type="text" value="Mobile Development"/> |
| Organization:        | <input type="text" value="tutorialspoint"/>     |
| City or Locality:    | <input type="text" value="Hyderabad"/>          |
| State or Province:   | <input type="text" value="AP"/>                 |
| Country Code (XX):   | <input type="text" value="INDIA"/>              |



< Back

Next >

Cancel

Finish

Destination and key/certificate checks



Destination APK file:

Certificate expires in 25 years.

