Android Fundamentals

by Kiran Kumar Bali

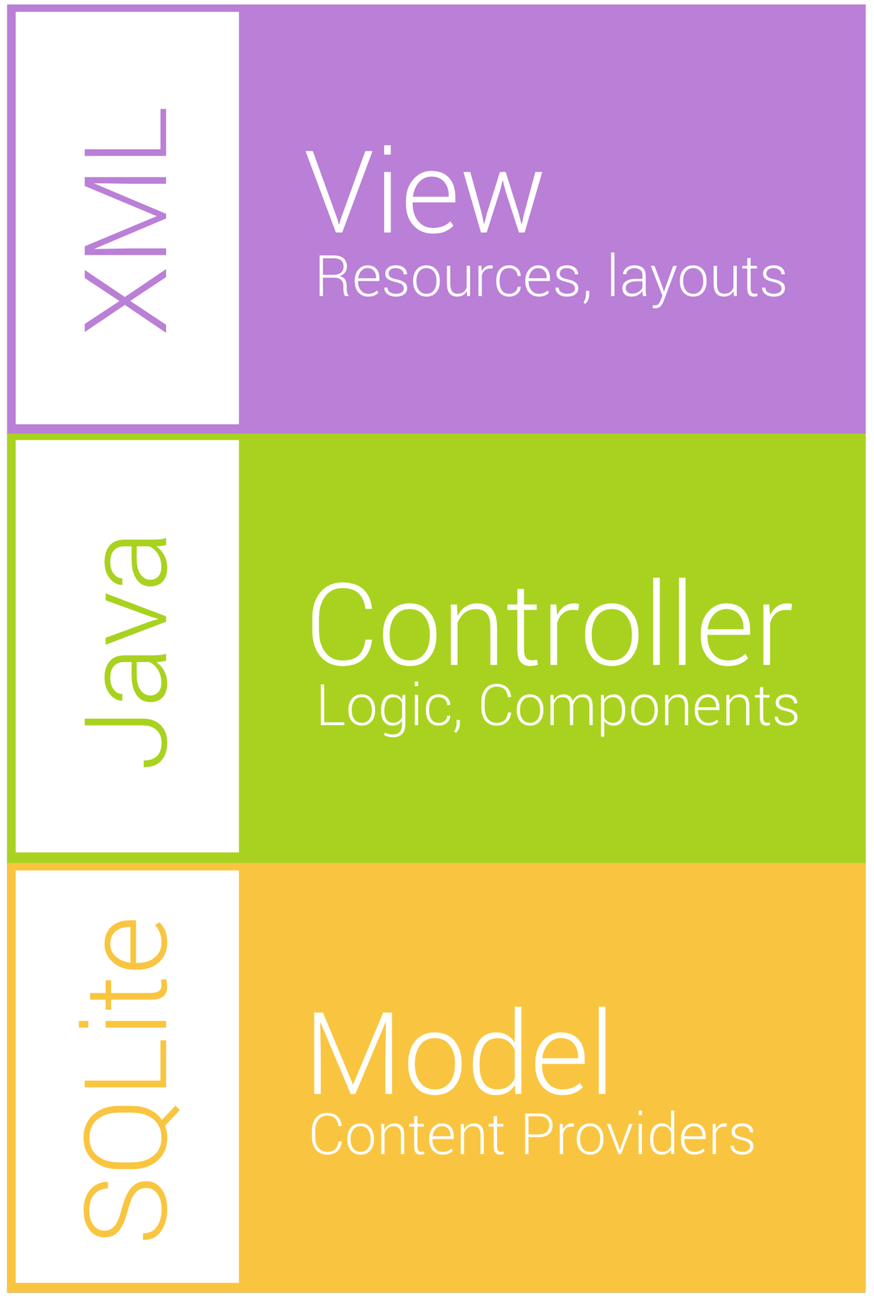
Introduction to Android

- Open software platform for mobile application development
- Powered by Linux Operating System.
- Application development in Java :Android SDK
- Native Libraries in C++ :Android NDK

History

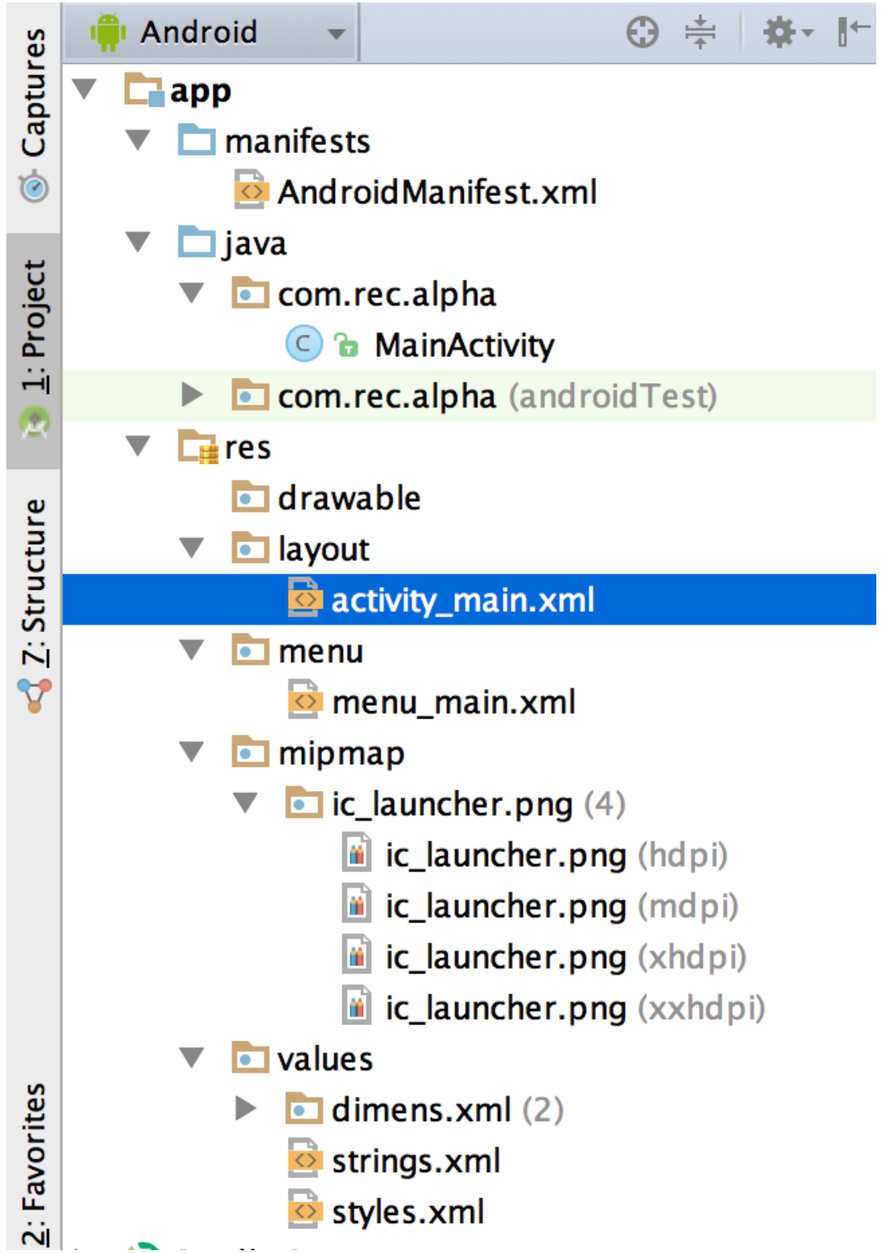
Android 5.0, Lollipop

MVC


XML
View
Resources, layouts
Java
Controller
Logic, Components
SQLite
Model
Content Providers

Project Structure

- AndroidManifest
- Activities/Fragments/POJO
- Resources(res)
  - Layouts/Menus
  - Drawable/mipmap
  - values



App Manifest

- allows us to define the packages, API, libraries needed for the application.
  - Basic building blocks of application like activities, services and etc.
  - Details about permissions.
  - Set of classes needed before launch.
- http://code.tutsplus.com/tutorials/android-sdk-project-manifest--mobile-20606

App Resources

- Different types of resources belong in different subdirectories of res/
- Alternative resources provide configuration-specific resource files
- Always include default resources so your app does not depend on specific device configurations

```
MyProject/
    src/
        MyActivity.java
    res/
        drawable/
            graphic.png
        layout/
            main.xml
            info.xml
        mipmap/
            icon.png
        values/
            strings.xml
```

```
res/
    drawable/
        icon.png
        background.png
    drawable-hdpi/
        icon.png
        background.png
```

Accessing Resources

- Resources can be referenced from code using integers from R.java, such as R.drawable.myimage
- Resources can be referenced from resources using a special XML syntax, such as @drawable/myimage
- You can also access your app resources with methods in Resources

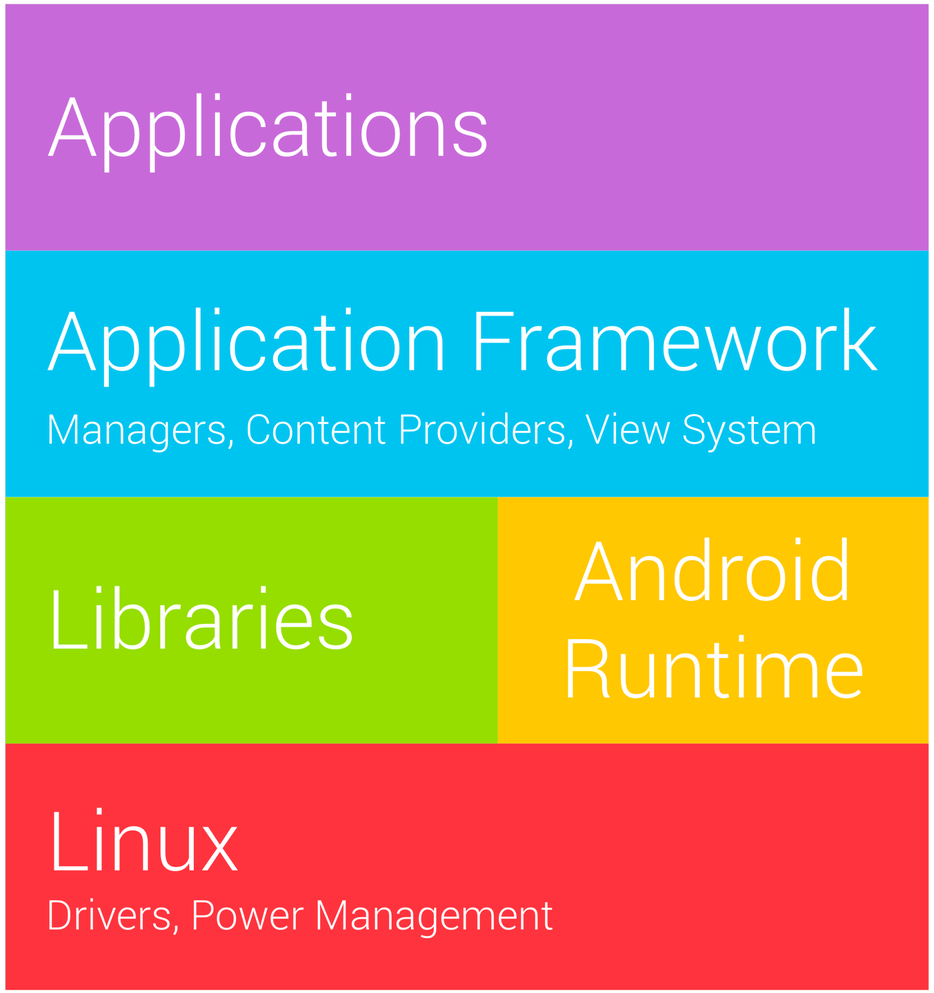
Project Bolt

DEMO

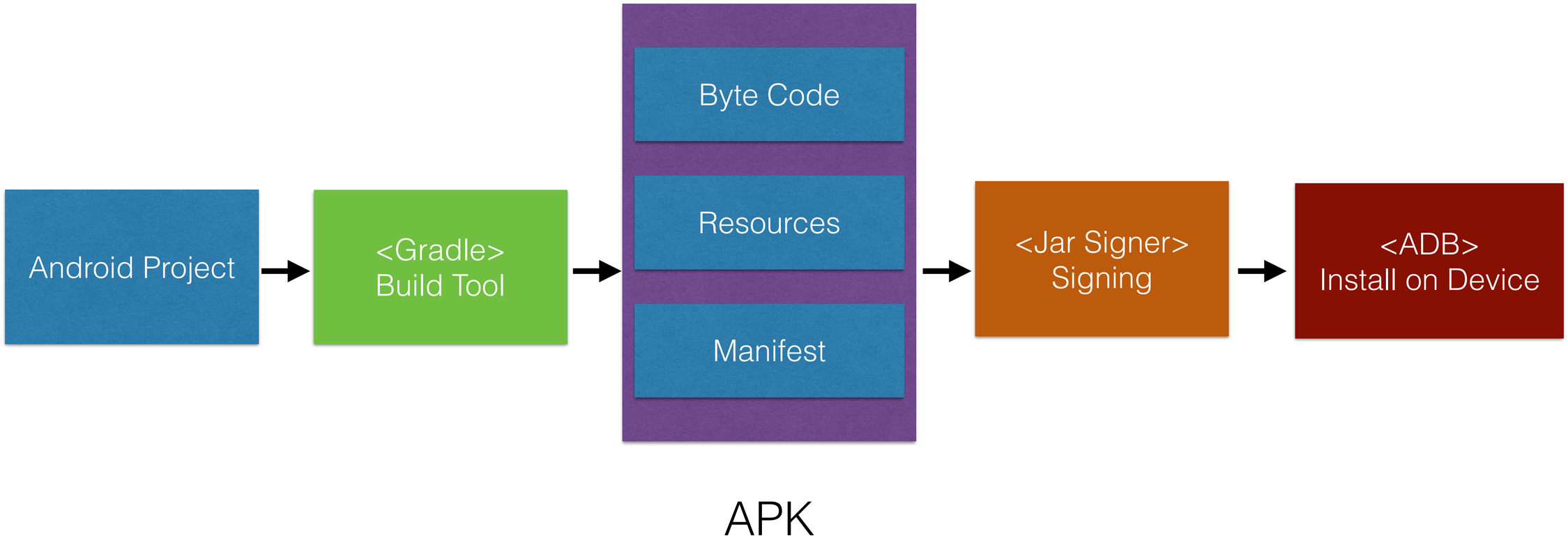
Create a New Android Studio Project

- Project Name, Company, Location
- Select Minimum and Target SDK. ##
- Select Template : (Activity + Fragment)
- Provide Template Creation Info and Click Finish
- Connect your device to the machine and Launch the app.##

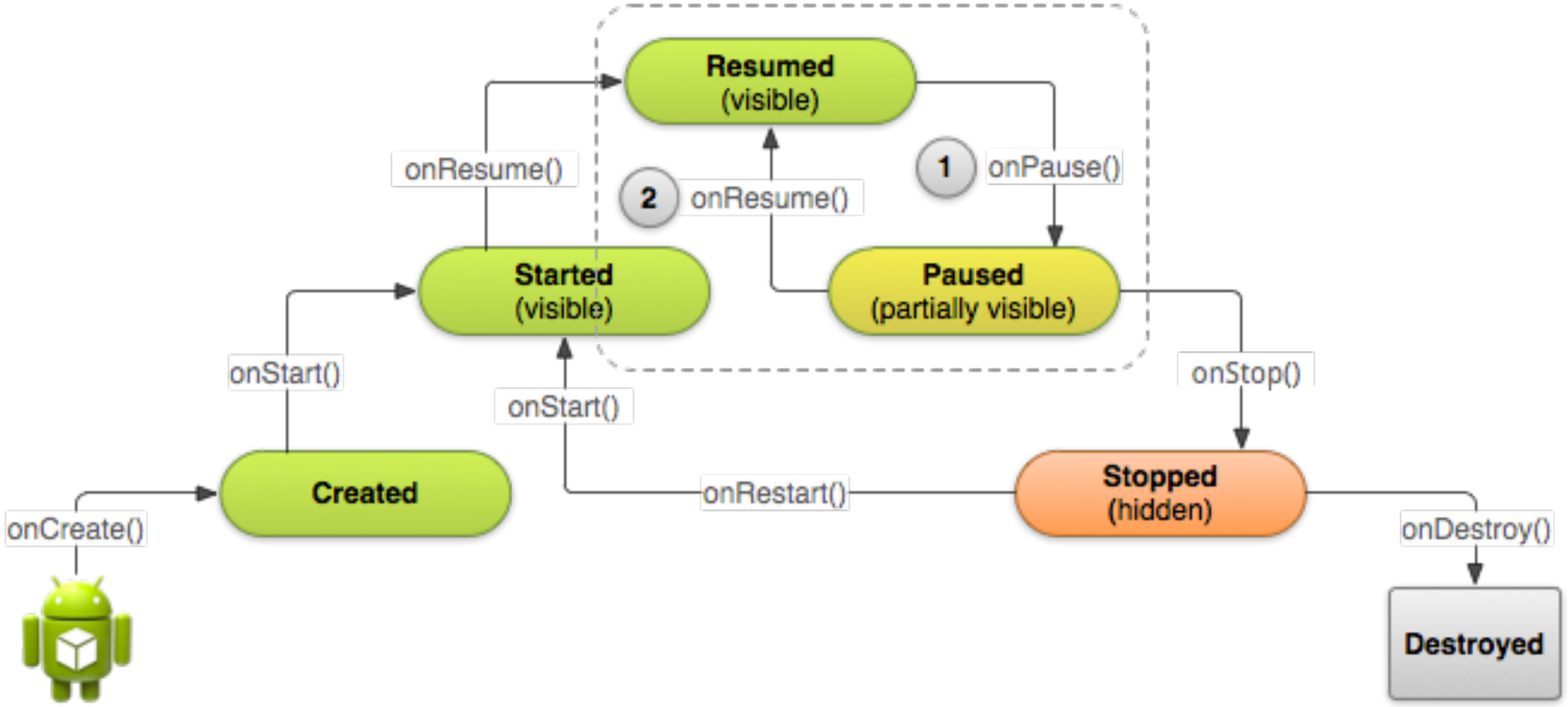
Android Software Stack



Gradle


Android Project
<Gradle>
Build Tool
Byte Code
Resources
Manifest
APK
<Jar Signer>
Signing
<ADB>
Install on Device

Application Lifecycle



Add ListItem Layout xml

- Create a new layout resource file with name list_item_forecast_textview

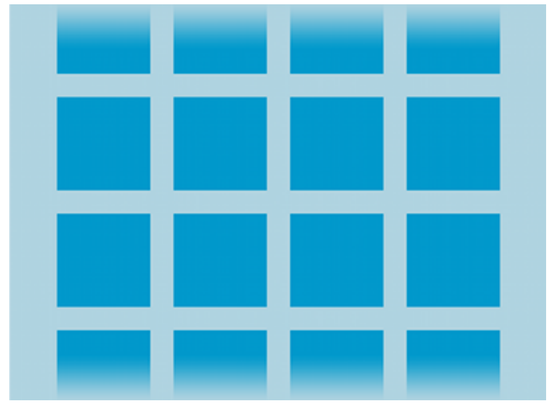
- Root element to be TextView for now,

Layout Manager

LinearLayout

RelativeLayout

ListView

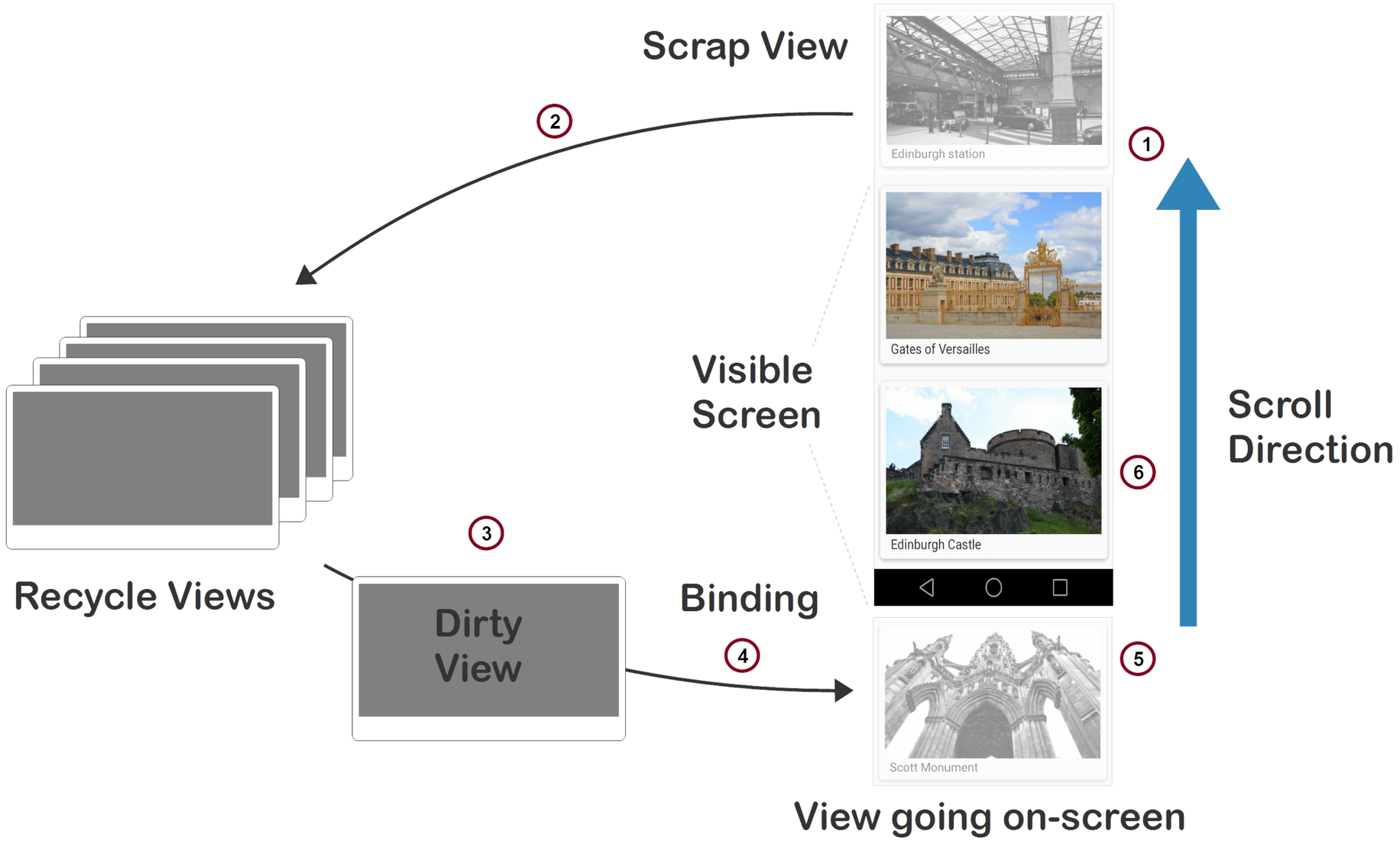


GridView

ScrollView vs ListView

- Limitations
- Limited resources

ListView and Recycling



Adapters

- ArrayAdapter
  - Create a class that extends ArrayAdapter
  - override getView(int position, View convertView, ViewGroup parent) method.
- CursorAdapter

OnItemClickListener

- <Your class> implements AdapterView.OnItemClickListener
- override the method:
  onItemClick(AdapterView<?> adapterView, View view, int position, long l);
- make sure you have activity/context scope.

Toast Messages

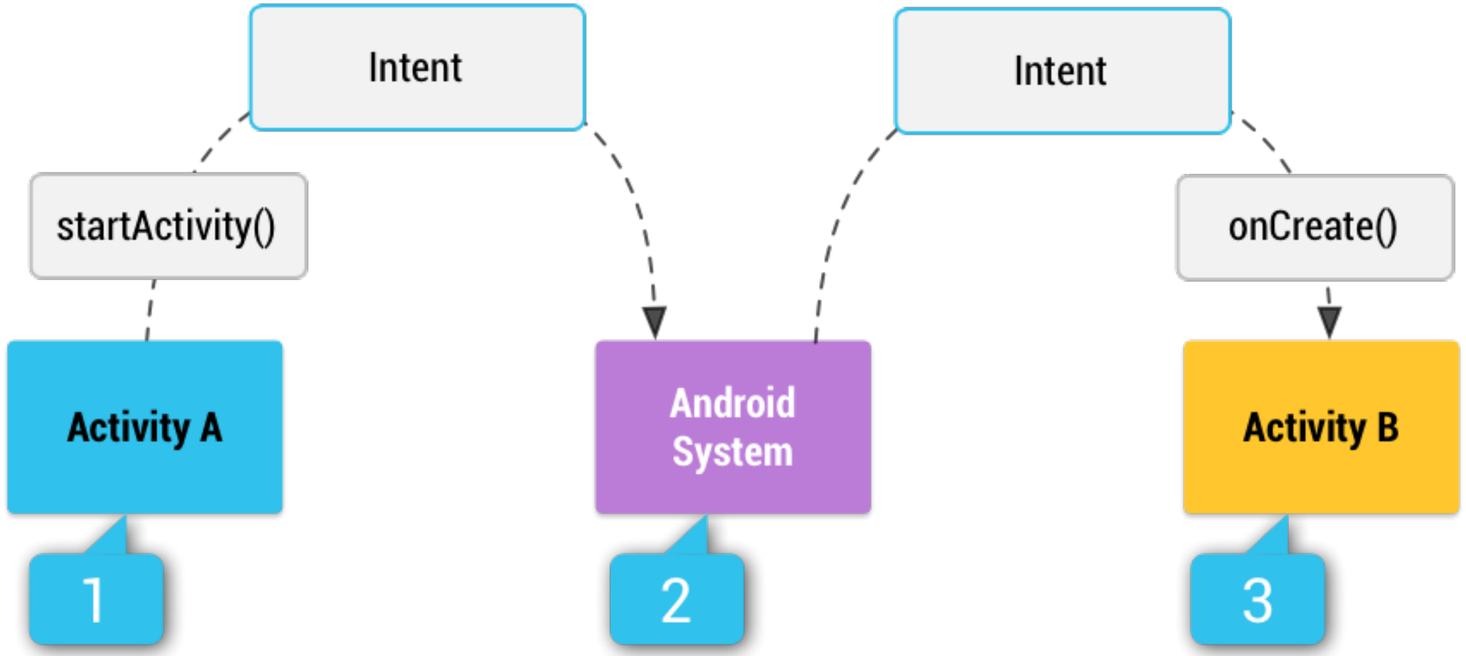
```
Toast myToast = new Toast(mActivity);

myToast.setView(myView);

myToast.setDuration(Toast.LENGTH_SHORT);

myToast.setGravity(Gravity.BOTTOM | Gravity.CENTER, 0, 0);

myToast.show();
```

Logging

- **Log** is a logging class that you can utilize in your code to print out messages to the **LogCat**. Common logging methods include:
  - v(String, String) (verbose)
  - d(String, String) (debug)
  - i(String, String) (information)
  - w(String, String) (warning)
  - e(String, String) (error)

Intents

- Types: Explicit & Implicit
- Used to start a Activity or Service and to deliver a broadcast.



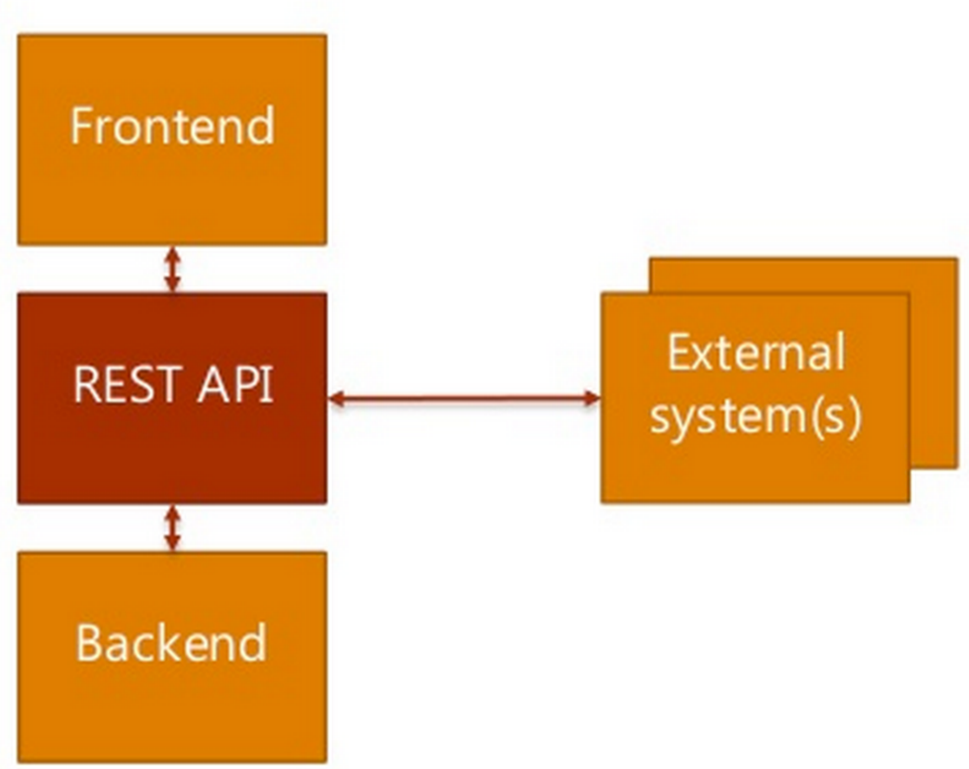
IntentFilters

- To receive an implicit Intent.

```
<activity android:name="ShareActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
</activity>
```

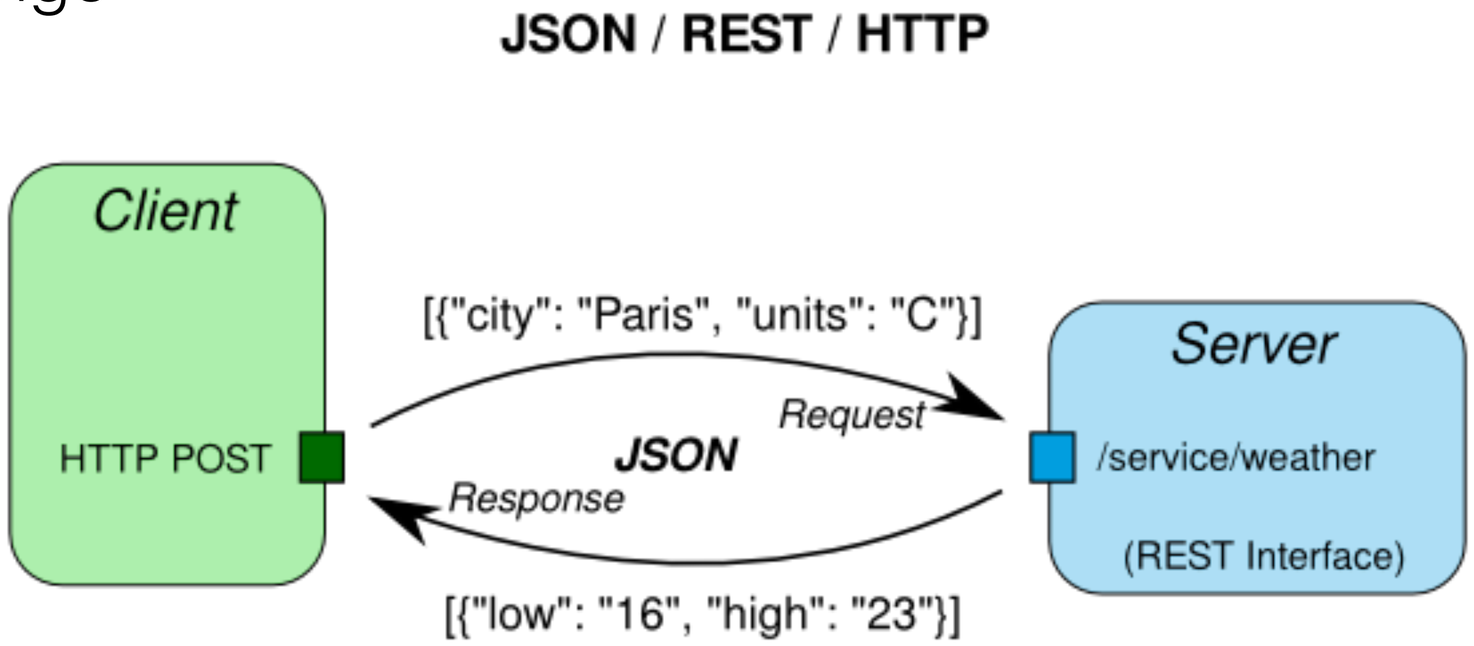
Calling REST API

- Uses HTTP
- CRUD Operations
  - read : GET
    write : POST
    update : PUT
    delete : DELETE



JSON

- Javascript Object Notation
- lightweight data-interchange format.



AsyncTask

- allows us to perform long/background operations and show its result on the UI thread without having to manipulate threads.

- Important methods to override: AsyncTask <TypeOfVarArgParams , ProgressValue , ResultValue>

  - **doInBackground**: Code performing long running operation goes in this method. When onClick method is executed on click of button, it calls execute method which accepts parameters and automatically calls doInBackground method with the parameters passed.

  - **onPostExecute**: This method is called after doInBackground method completes processing. Result from doInBackground is passed to this method.

  - **onPreExecute**: This method is called before doInBackground method is called.

  - **onProgressUpdate**: This method is invoked by calling publishProgress anytime from doInBackground call this method.

AsyncTask Example

```
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
    protected Long doInBackground(URL... urls) {
        int count = urls.length;
        long totalSize = 0;
        for (int i = 0; i < count; i++) {
            totalSize += Downloader.downloadFile(urls[i]);
            publishProgress((int) ((i / (float) count) * 100));
            // Escape early if cancel() is called
            if (isCancelled()) break;
        }
        return totalSize;
    }

    protected void onProgressUpdate(Integer... progress) {
        setProgressPercent(progress[0]);
    }

    protected void onPostExecute(Long result) {
        showDialog("Downloaded " + result + " bytes");
    }
}
```

Menu

Settings

Q&A

Time to take revenge on me!

Thank You!

www.kirankumarbali.in