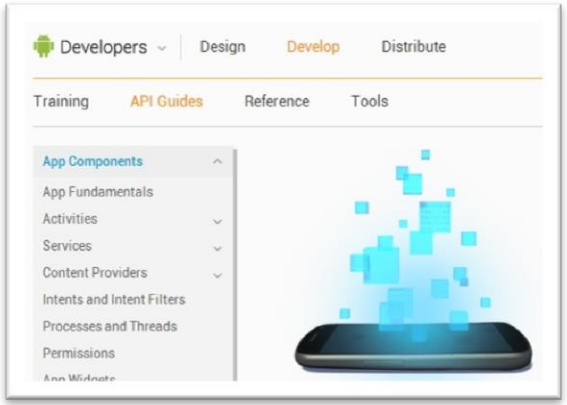
Course Overview

Android Developers Website



http://developer.android.com/guide/components/index.html

Android Development System

Android Website
Design Guide
Eclipse IDE
Android SDK
Android NDK
Emulator
Java
Sample Apps
Testing
Debugging
Publishing Apps
Google Play Developer Console
.
.
.

APP Components and APIs

Activities
Services
Content Providers
Intents
Processes
Messaging
Widgets
Manifest
.
.
.

App Resources

XML
Localization
Colors
Drawables
Layouts
Strings
Dimensions
Menu
Styles
.
.
.

Media and Camera

Playback
Formats
Audio
Camera
.
.
.

Web Apps

WebView
Viewport
Layout
Resizing
.
.
.

Remote Access

Server Access
JSON Data
Google Cloud
.
.
.

User Interface

Layouts
Buttons
Text
Spinners
Pickers
Input Events
Menus
Tabs
Action Bar
Dialogs
Notifications
Progress Bars
Seek Bars
Search
Lists
Styles and Themes
Phone Interface
Email Interface
.
.
.

Connectivity

Bluetooth
NFC
Wi-Fi Direct
USB
SIP
.
.
.

Location and Sensors

Maps
Geocoding
Map Overlays
Motion Sensors
Position Sensors
Environment Sensors
.
.
.

Data Storage

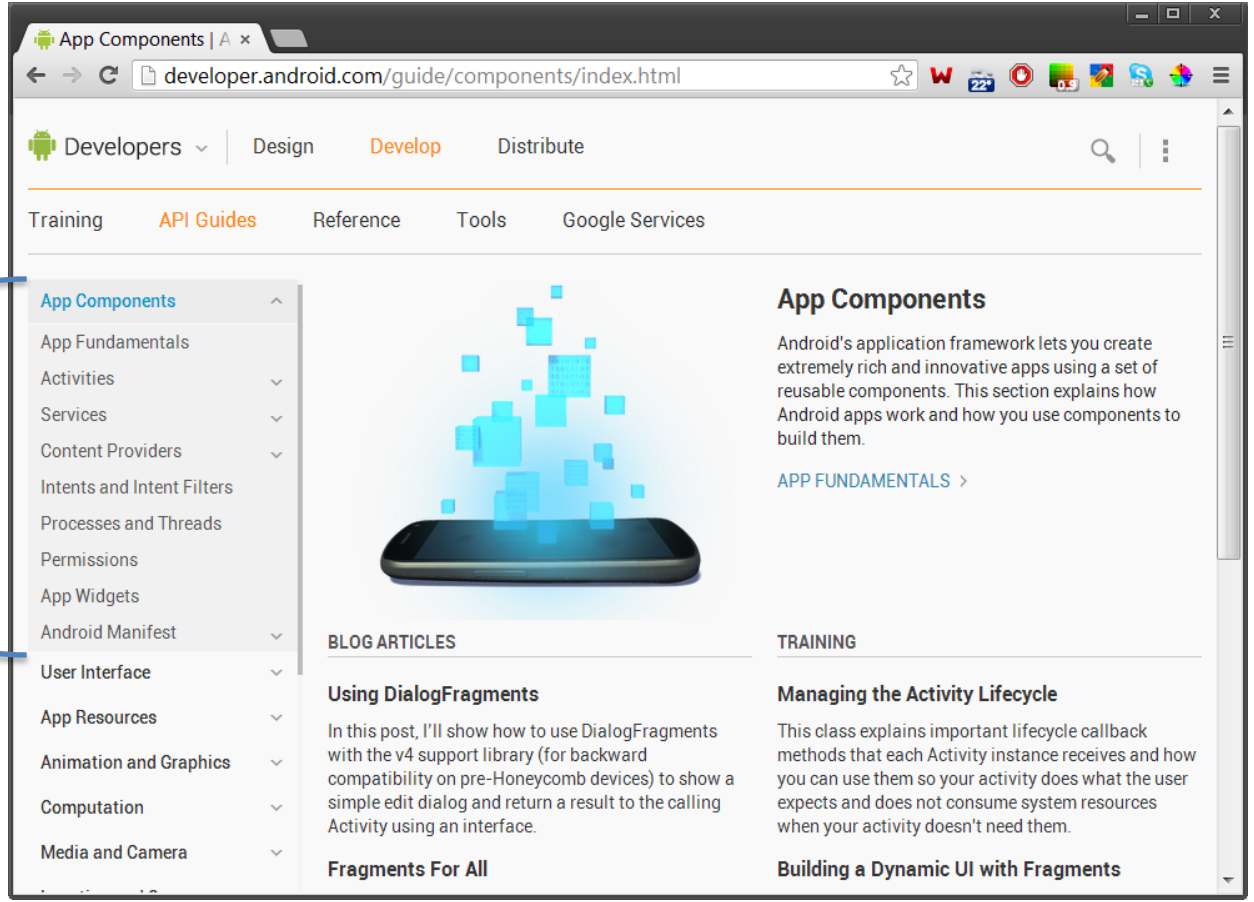
Shared Preferences
SQLite Database
Internal Data Storage
External Data Storage
.
.
.

Animation and Graphics

Property Animation
View Animation
Canvas and Drawables
OpenGL
Hardware Acceleration
.
.
.

App Components and APIs

The basic building blocks of app actions.



http://developer.android.com/guide/components/index.html

App Components and APIs in Use

Android Application

Components
- Activities
- Content Providers
- Services
- Broadcast Receivers

Resources
- XML
- Images

APIs
- Classes
- Methods

- Fragments
- Widgets
- Processes & Threads
- Messaging
- Memory Management
- Permissions
- …

App Development

APK

Android Package

APK Creation

APK Upload

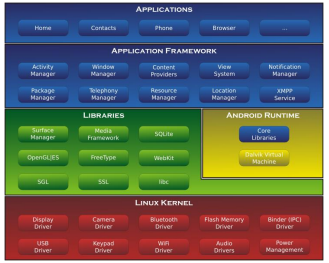
APK Download

Google Play

Device Supplier Load

Android OS

Multi-user Linux System

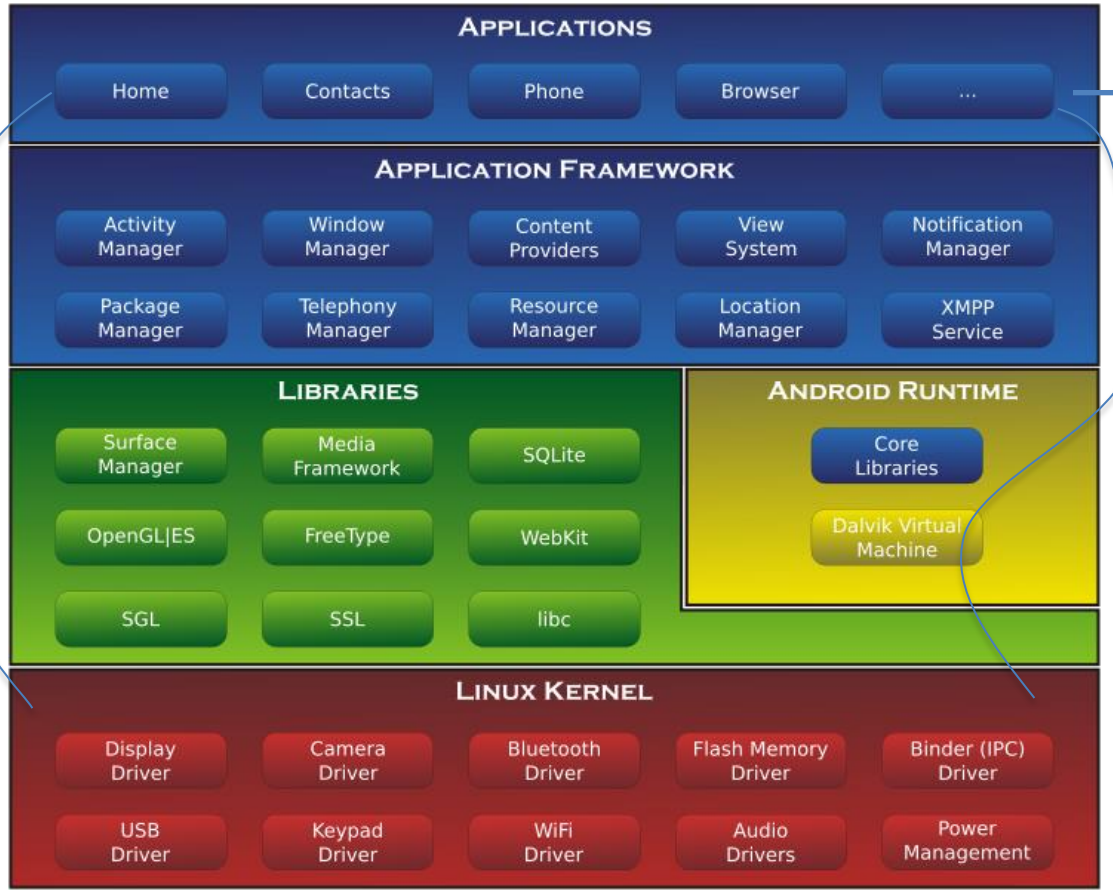


Android Architecture

Multi-user Linux System

Permissions are set for all files in an app so that only the user ID assigned to that app can access them.

Each app is a separate Linux user with its own Linux user ID (unknown to the app).

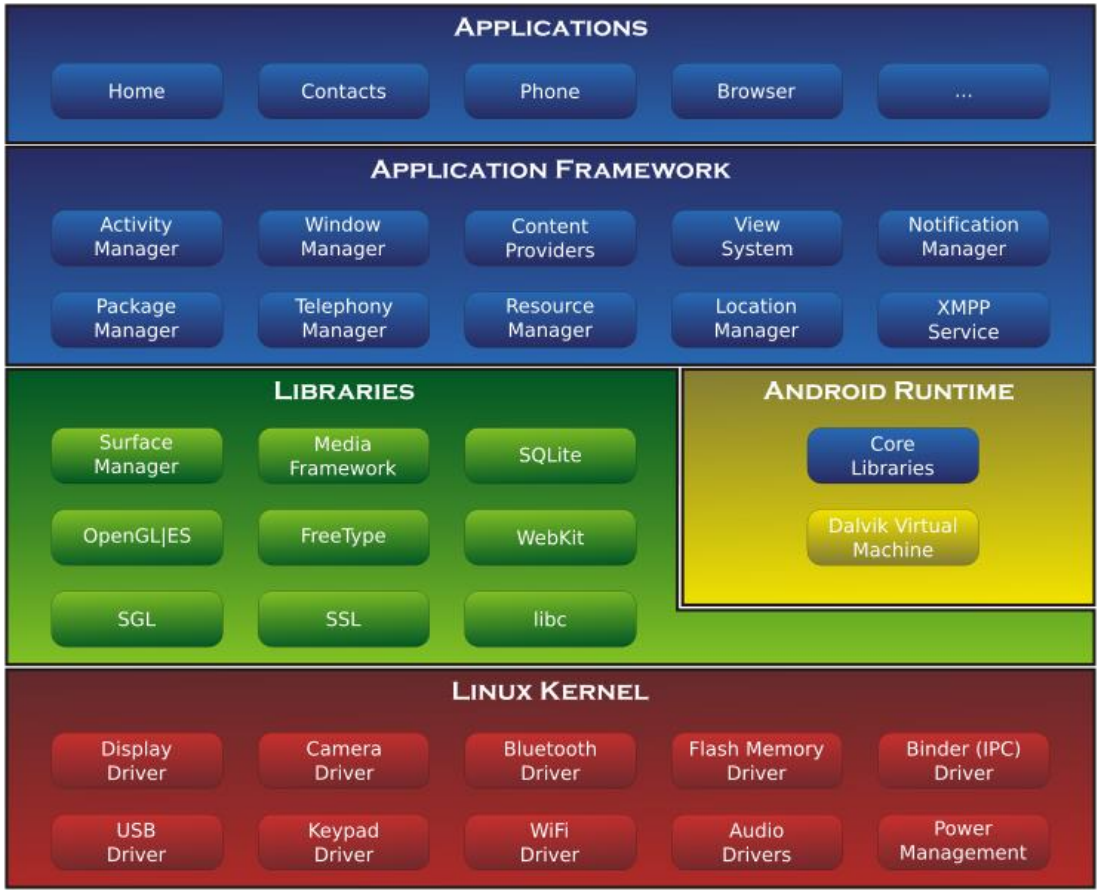


Each app process has its own Dalvik Virtual Machine, which runs in its own Linux process.

http://developer.android.com/guide/components/fundamentals.html

http://en.wikipedia.org/wiki/Android_(operating_system)#Development

Android Linux



Linux is open source and was initially developed in the 1990's:
http://en.wikipedia.org/wiki/Linux

Linux is a branch of the Unix operating system:
http://en.wikipedia.org/wiki/File:Unix_history.svg

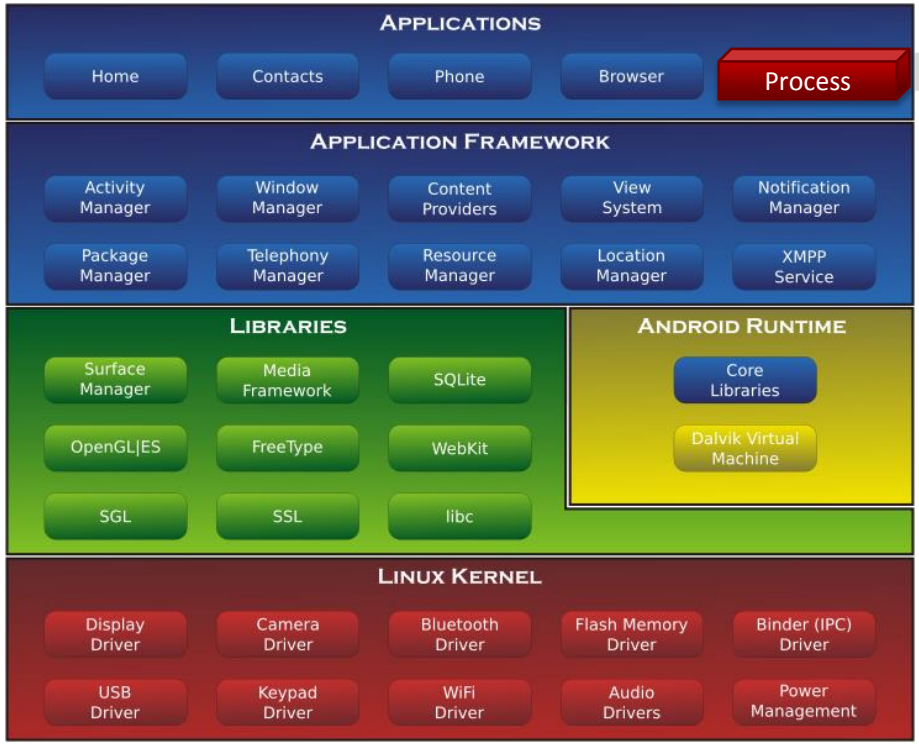
Some Android changes to the Linux Kernel:
http://elinux.org/Android_Kernel_Features

Process

- Scheduling
- Time Sharing
- Prioritization
- Code Execution

http://oreilly.com/catalog/linuxkernel/chapter/ch10.html

App Lifecycles



Lifecycle Phases

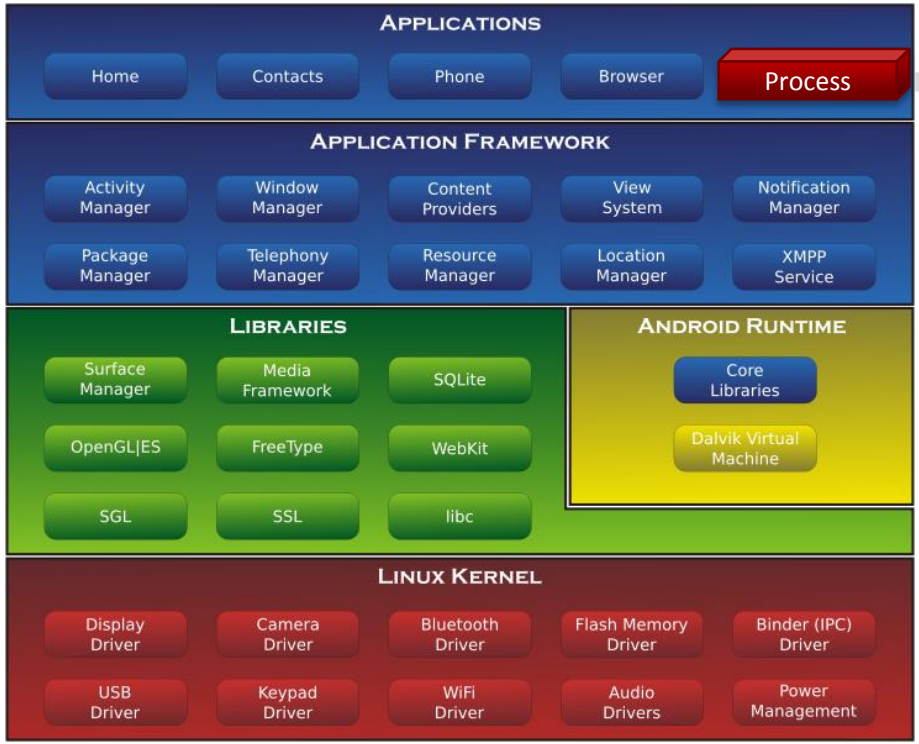
Lifecycle Patterns

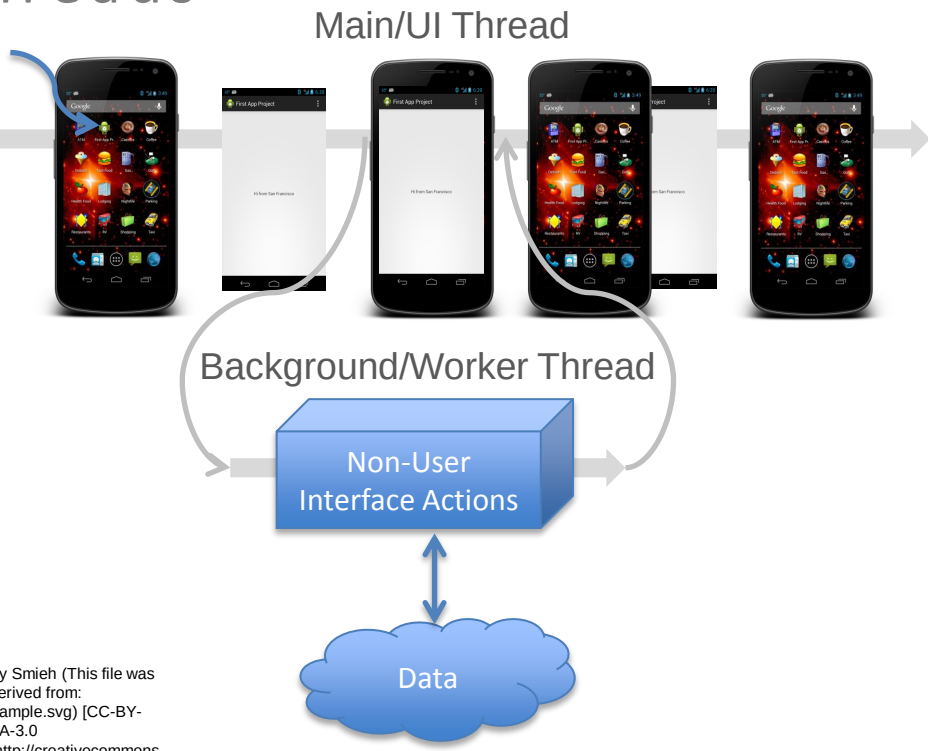
- Activities
- Services
- Content Providers
- Broadcast Receivers

Lifecycle Methods

onPhase1() { . . . }
onPhase2() { . . . }
onPhase3() { . . . }
.
.

App Threads



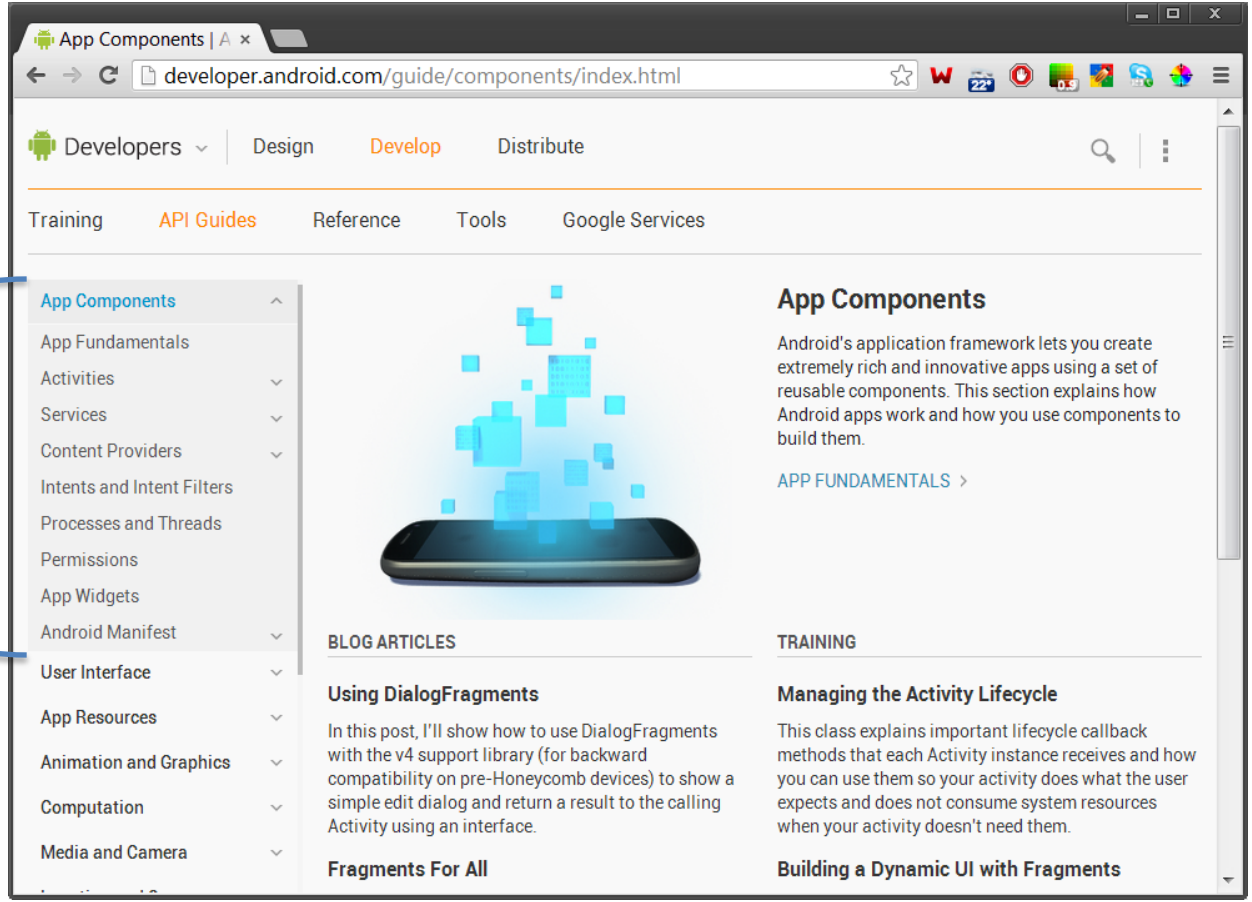


By Smieh (This file was derived from: Sample.svg) [CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

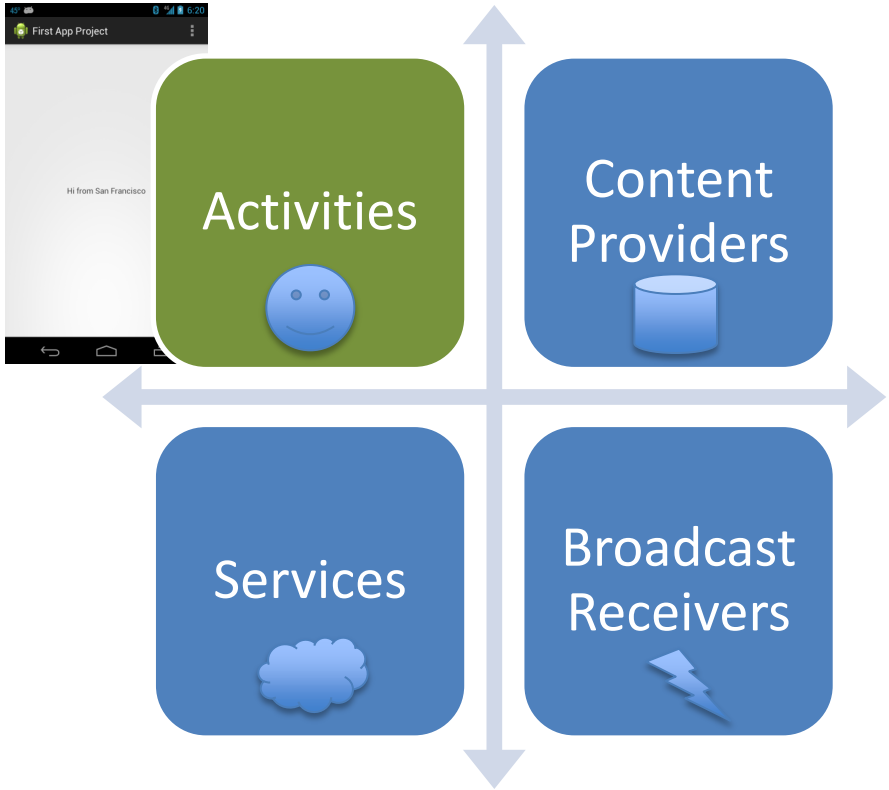
App Components and APIs

The basic building blocks of app actions.

http://developer.android.com/guide/components/index.html



Activities


First App Project
Hi from San Francisco
Activities
Content Providers
Services
Broadcast Receivers

Activities API

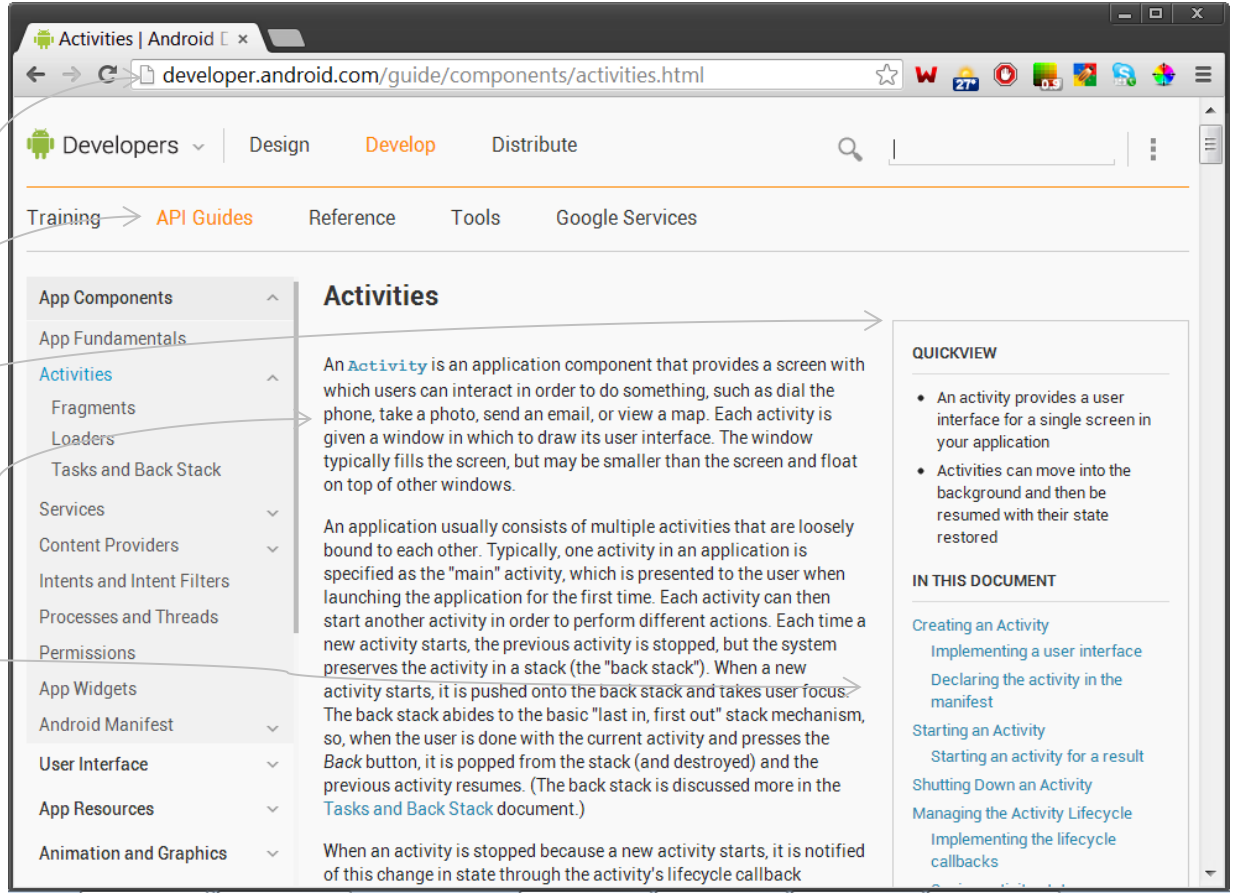
http://developer.android.com/guide/components/activities.html

Development API Guide

Overview

Basic Description

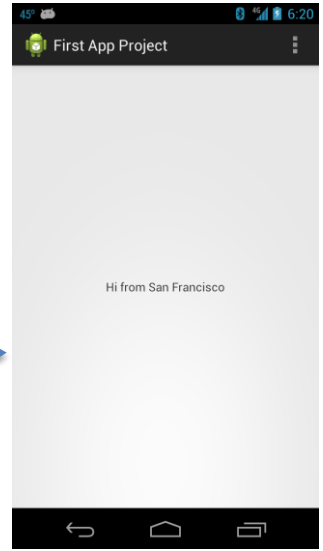
Detailed Description



First App Activity

Extends Activity Class

```
package com.donkcowan.firstappproject;

import android.os.Bundle;
import android.app.Activity;
import android.util.Log;
import android.view.Menu;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        int variableTest = 1;
        int variableUse = variableTest;
        Log.i("AppTest","My message");
        Log.i("AppTest",Integer.toString(variableUse));
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }

}
```



Lifecycle Phases

Activity Methods

Activity Class

http://developer.android.com/reference/android/app/Activity.html

Development Reference

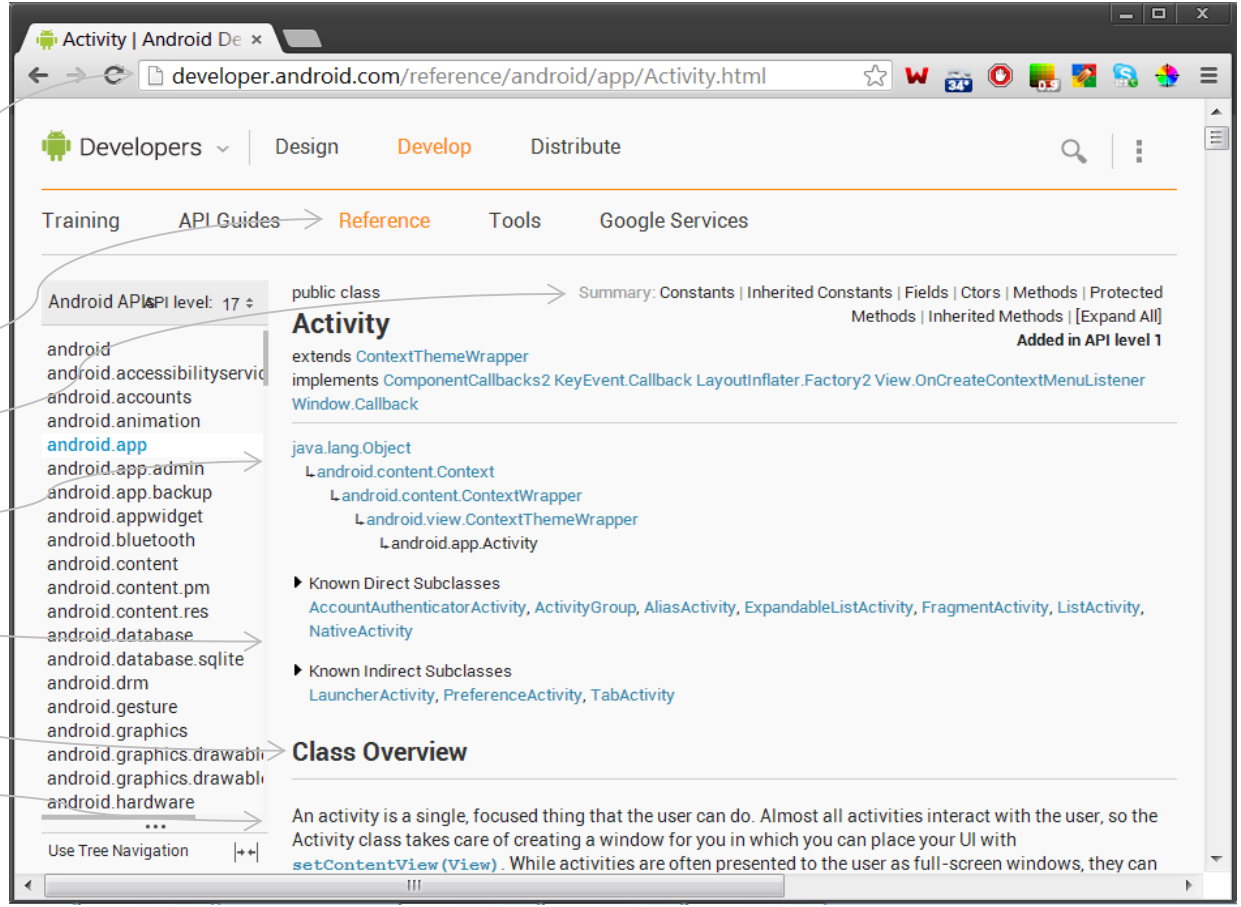
Summary Links

Hierarchy of Class Extensions

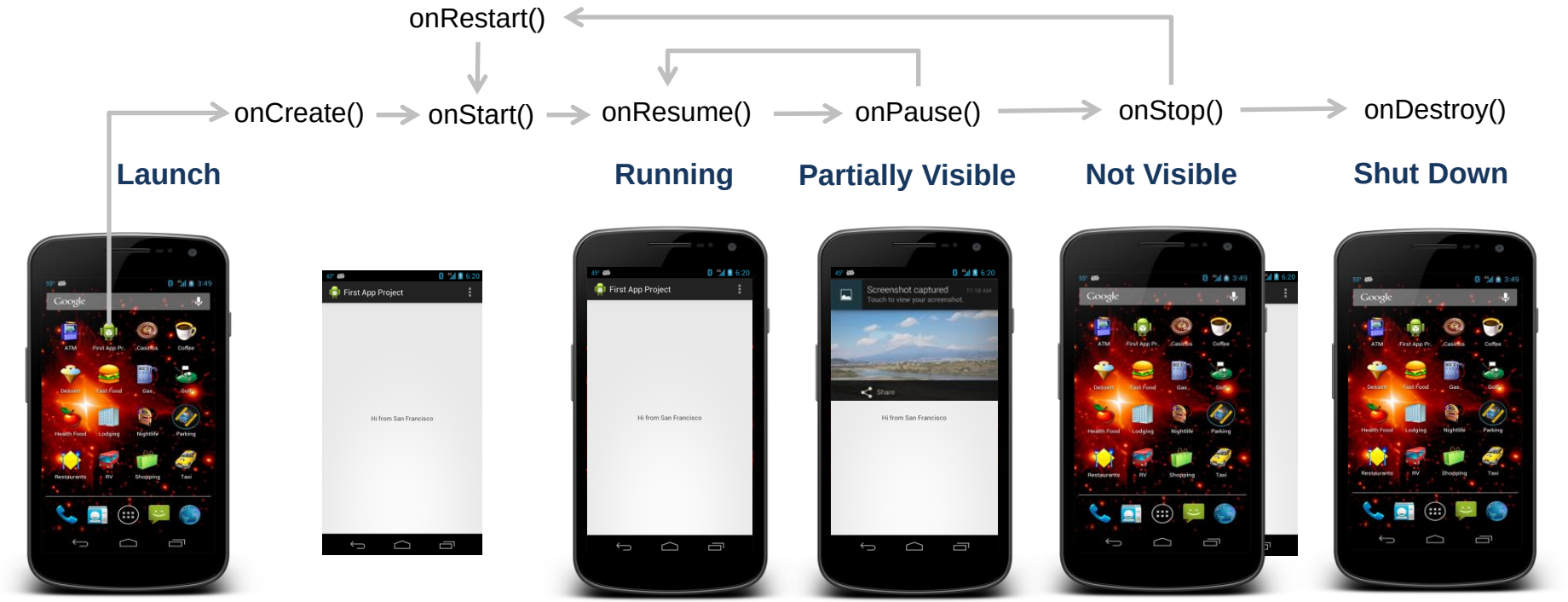
Subclasses

Class Overview

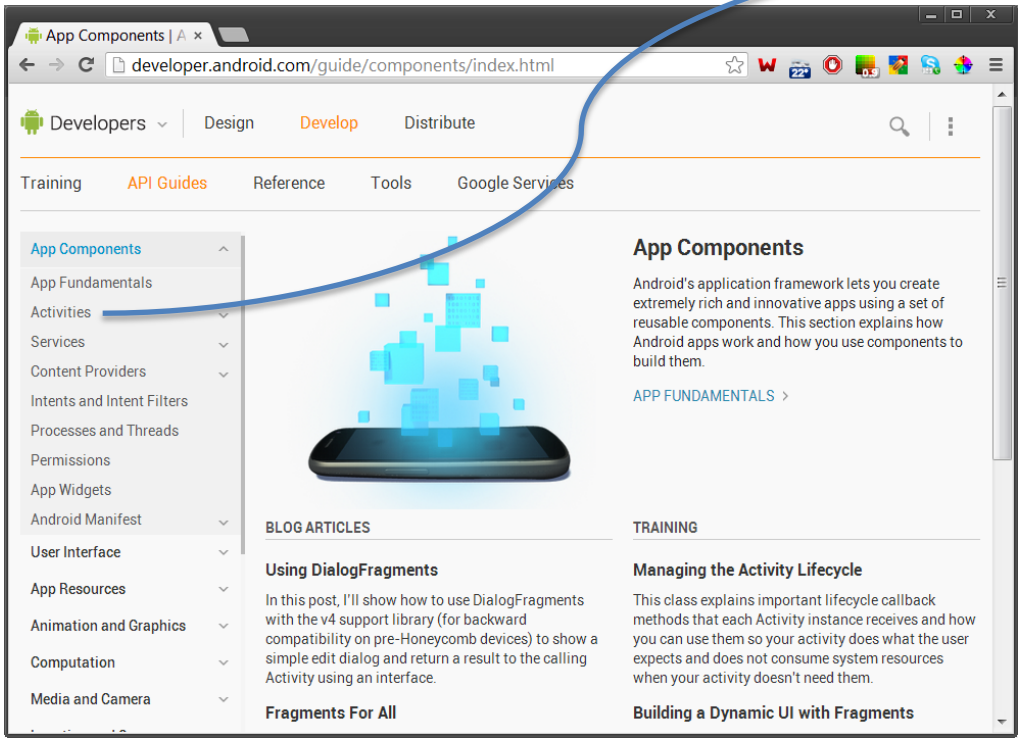
"An activity is a single, focused thing that the user can do."

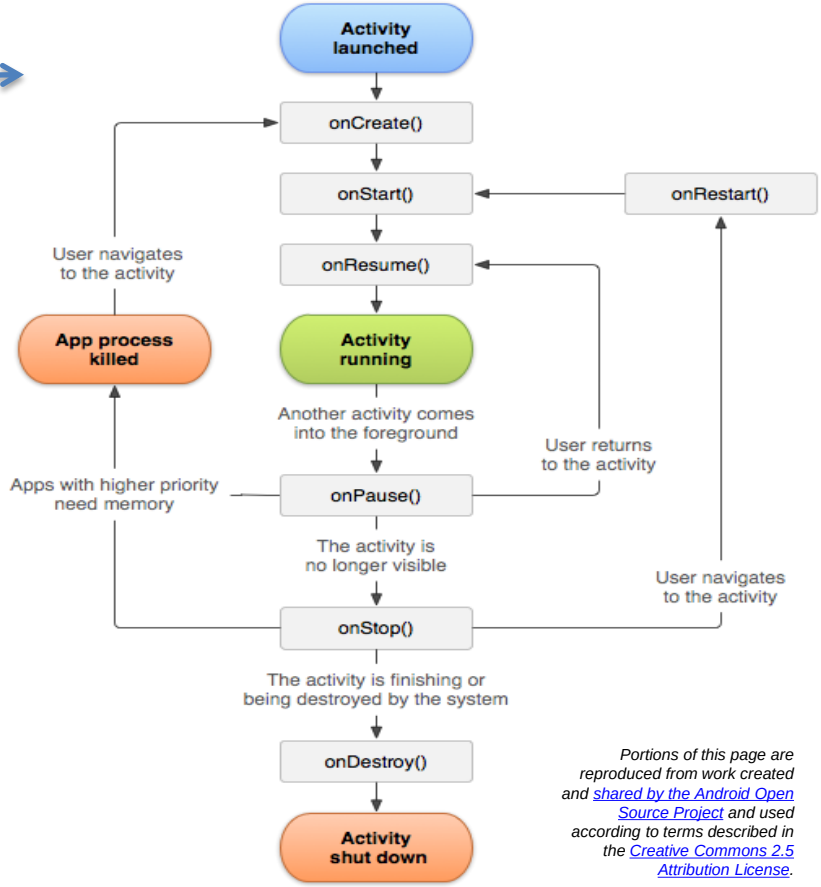


Activity Lifecycle



Activity Lifecycle





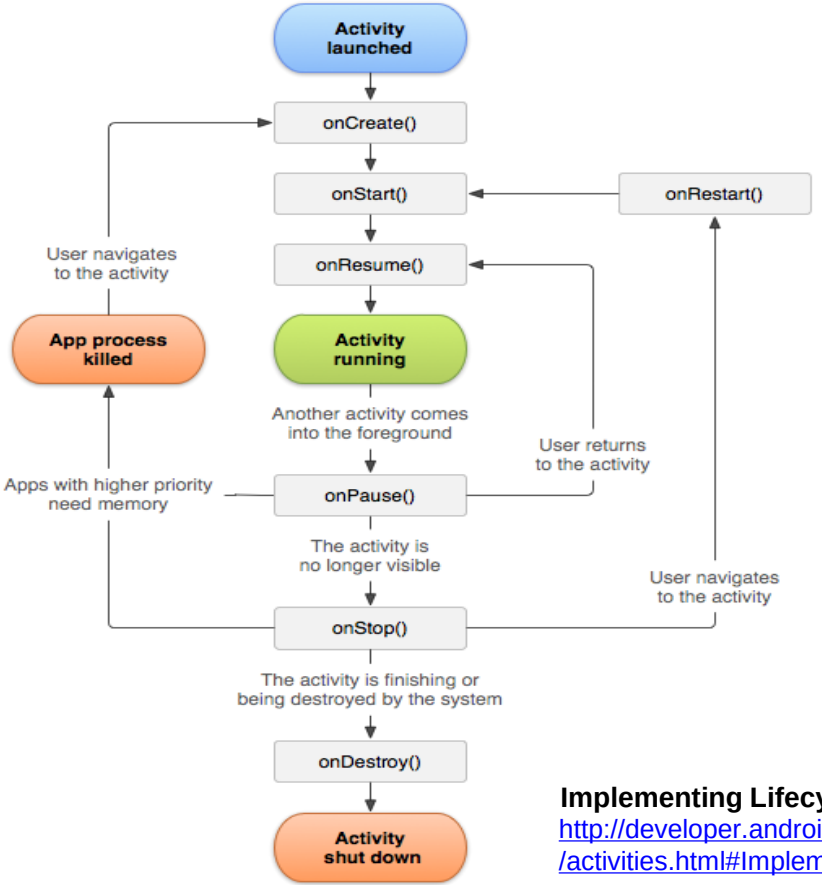
Activity Lifecycle Methods and Typical Uses

onCreate()
- *Must* be implemented
- Call setContentView()
- Initialize display layout
- Initialize variables
- Start background thread

onResume()
- Restore variables
- Restart animations
- Lightweight code

onPause()
- Commit changes
- Save variables
- Save preferences
- Stop animations
- Lightweight code

onDestroy()
- Close databases
- Release remaining resources
- Stop background thread



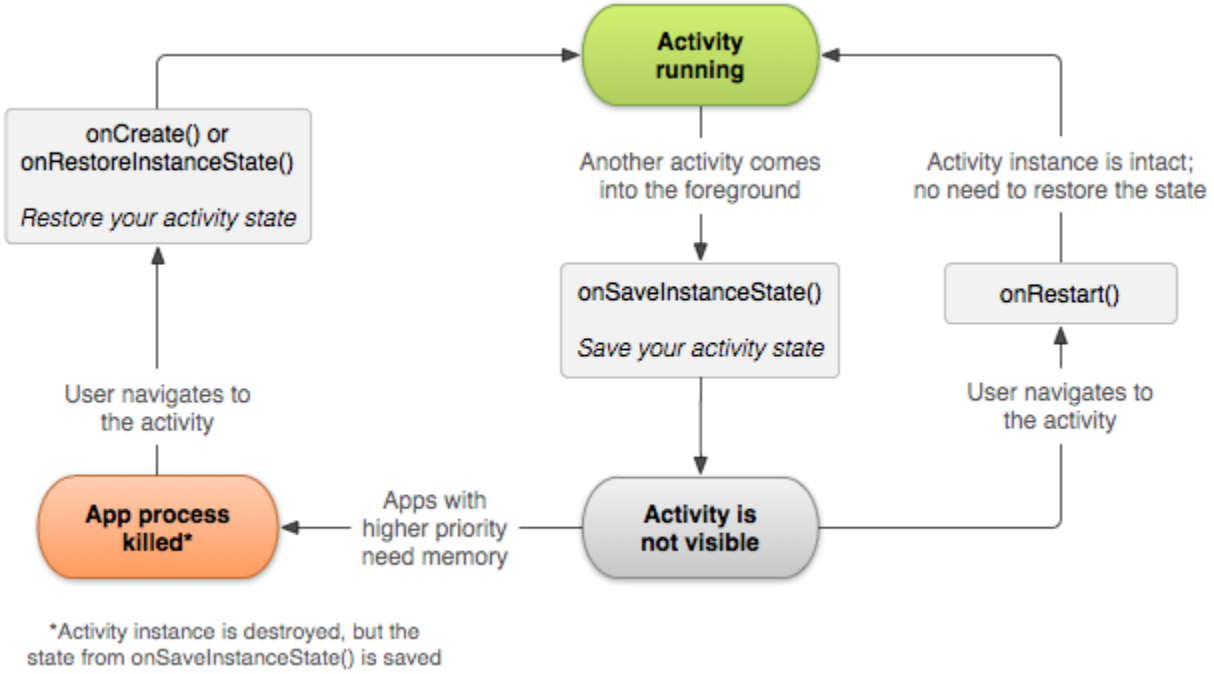
onStart()
- Register Broadcast Receiver
- Reinitialize states

onRestart()
- Open server connections
- Allocate resources

onStop()
- Close server connections
- Free resources
- Unregister Broadcast Receiver

Implementing Lifecycle Callbacks
http://developer.android.com/guide/components/activities.html#ImplementingLifecycleCallbacks

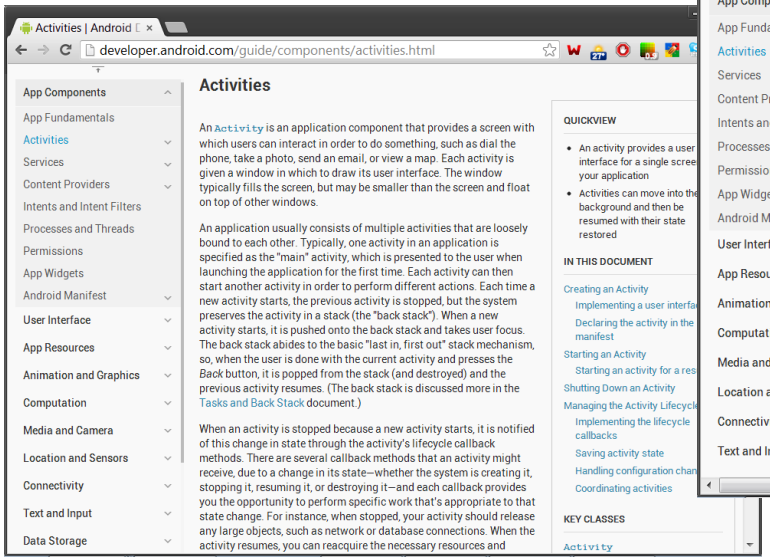
Saving Activity State



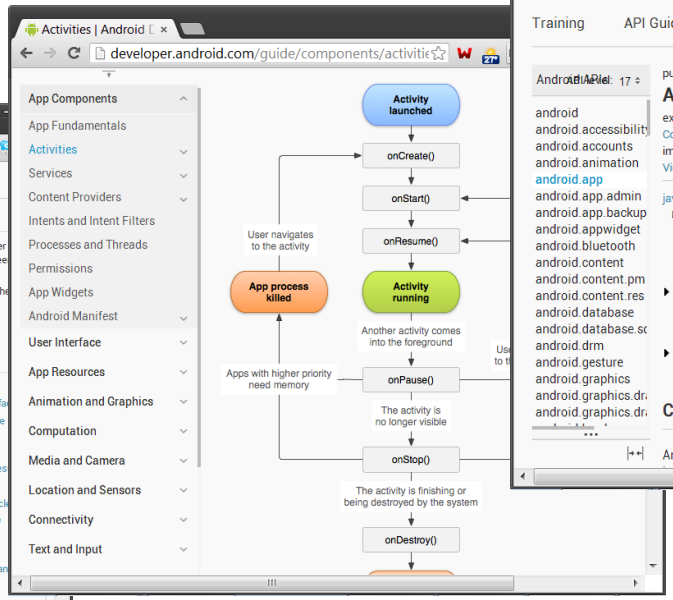
http://developer.android.com/guide/components/activities.html#SavingActivityState

Understanding an API

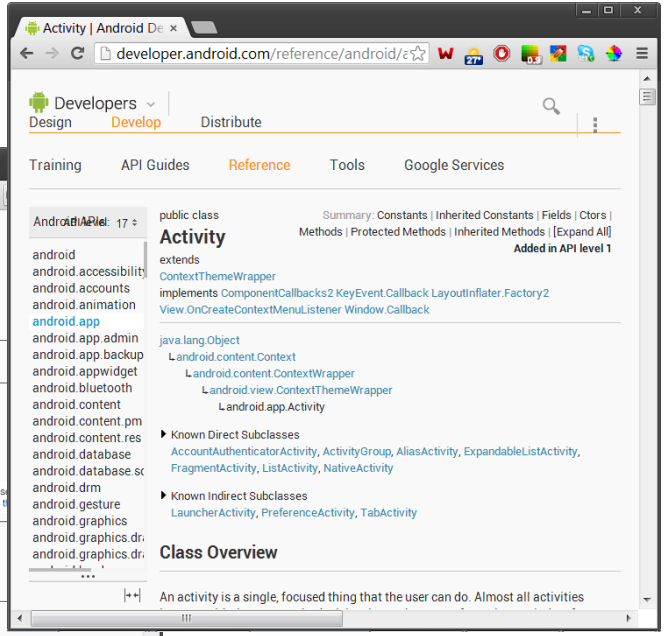
Familiarity with major aspects



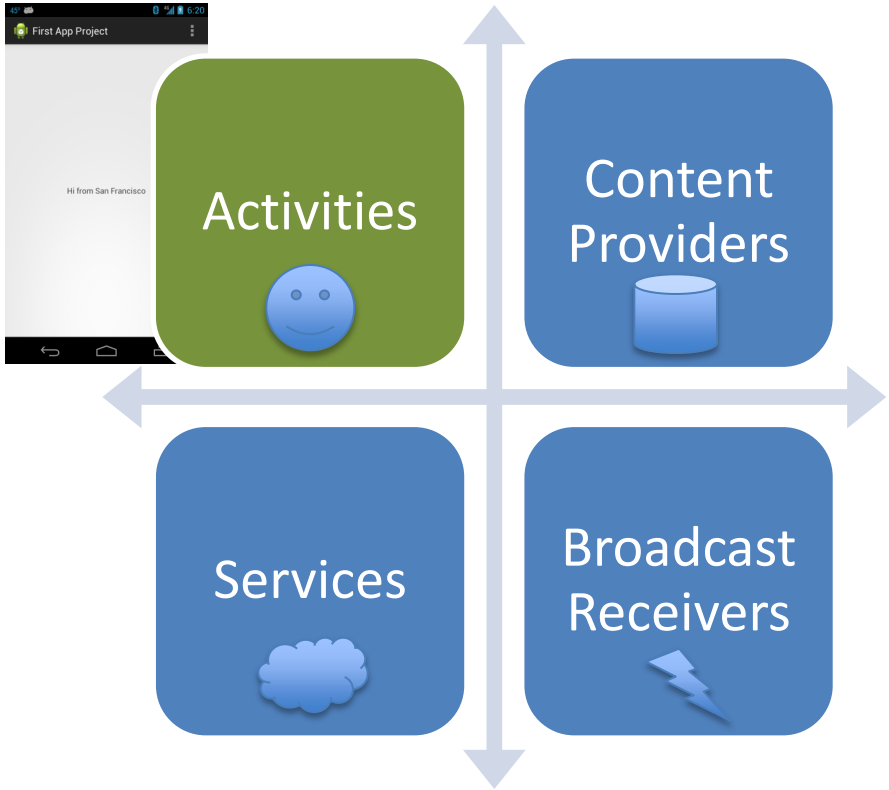
Understand key and required capabilities



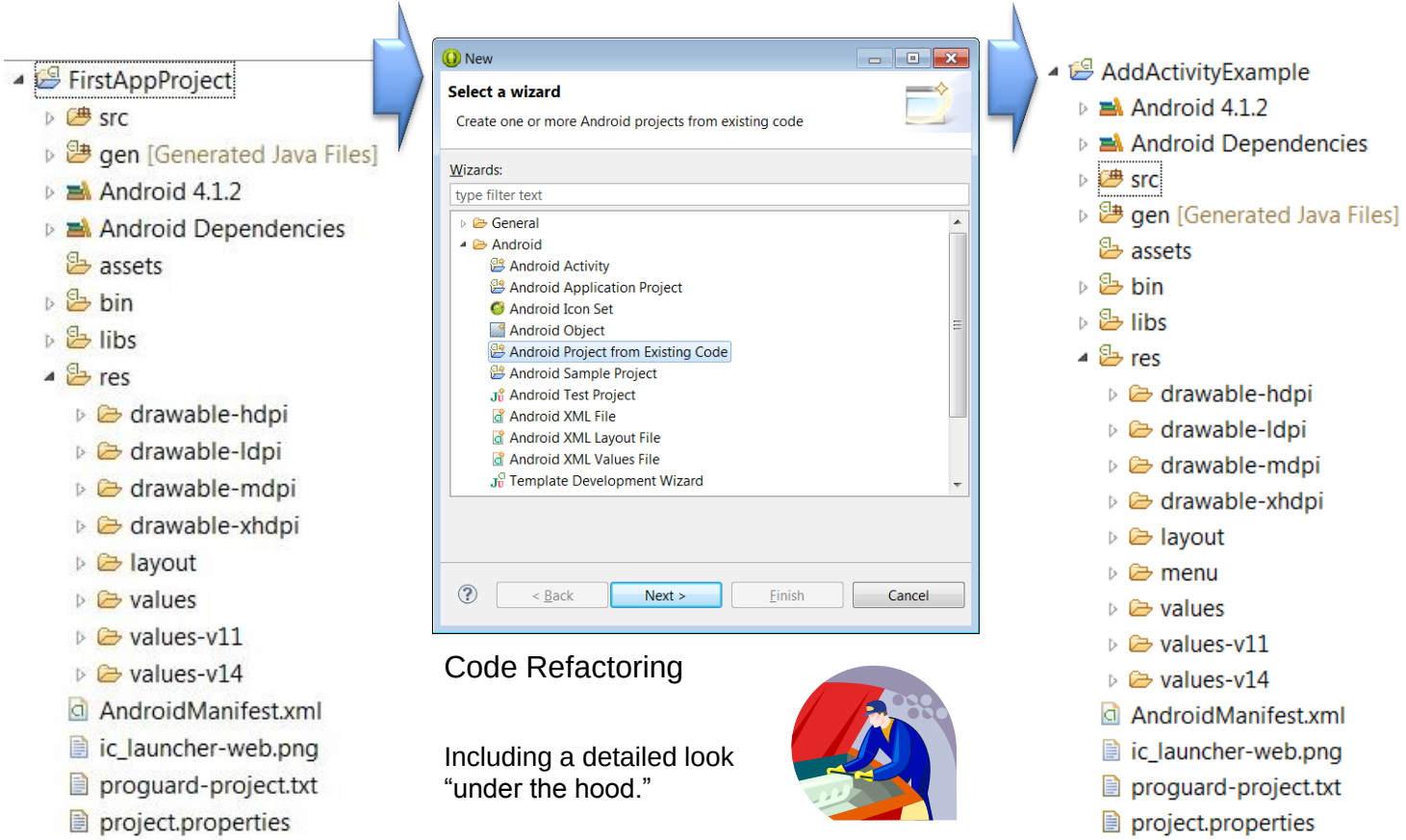
Research as needed



Activities


First App Project
Hi from San Francisco
Activities
Content Providers
Services
Broadcast Receivers

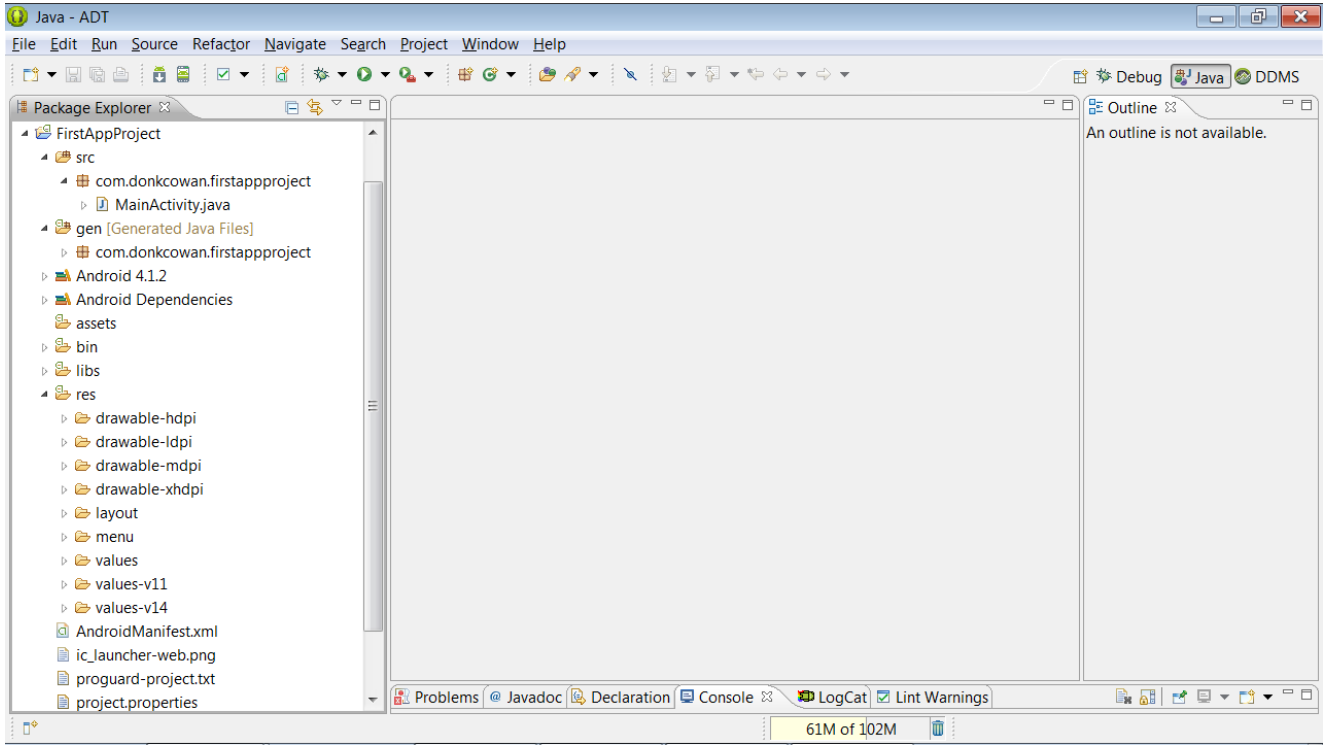
Creating a Project from Existing Code



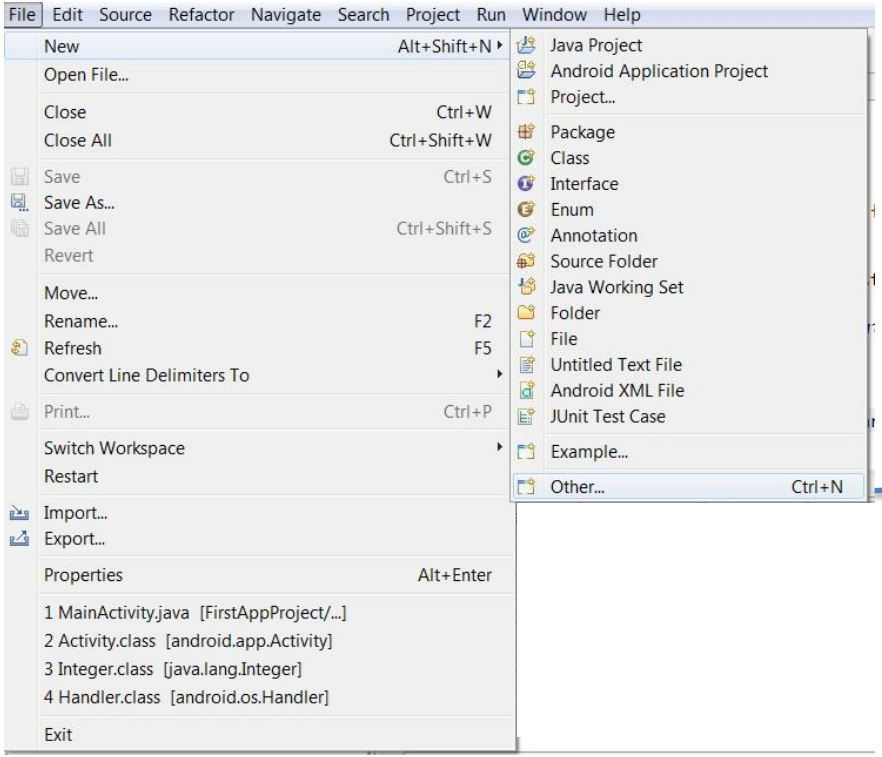
Code Refactoring

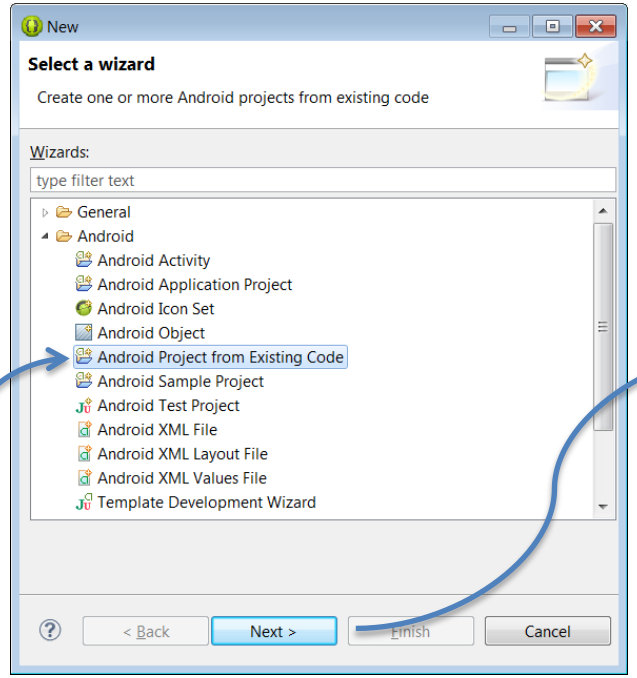
Including a detailed look "under the hood."

Opening Your Workspace

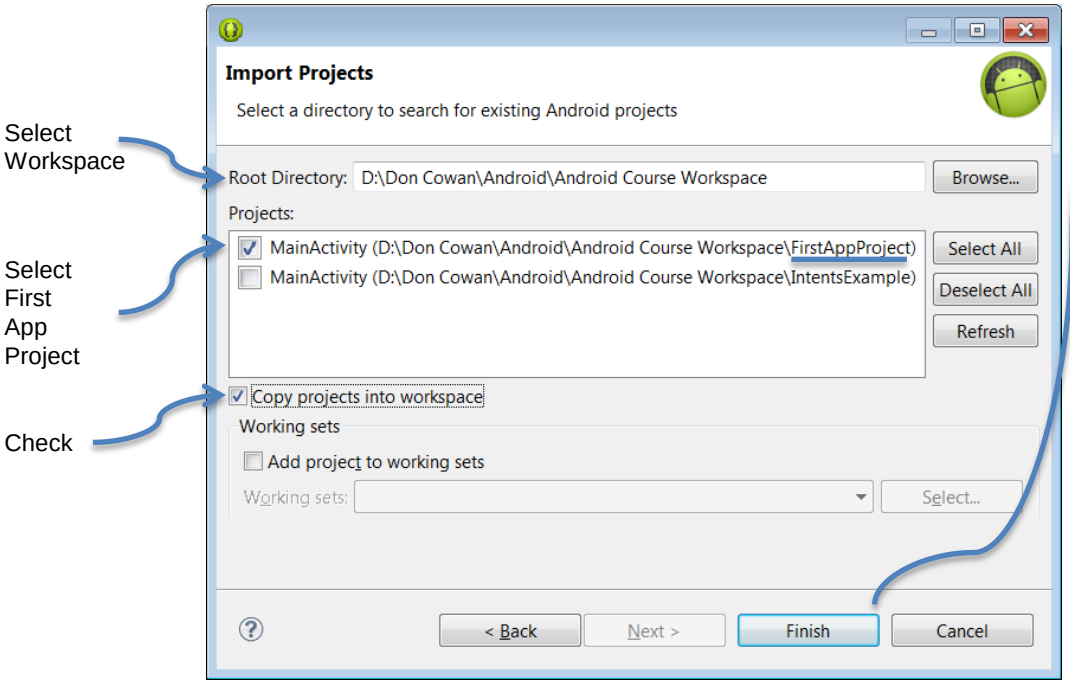


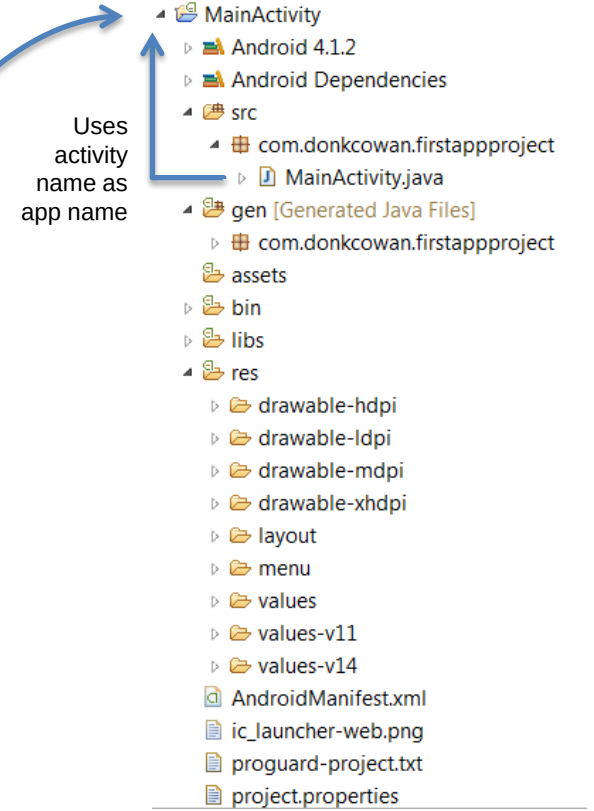
Selecting the Project from Existing Code Wizard



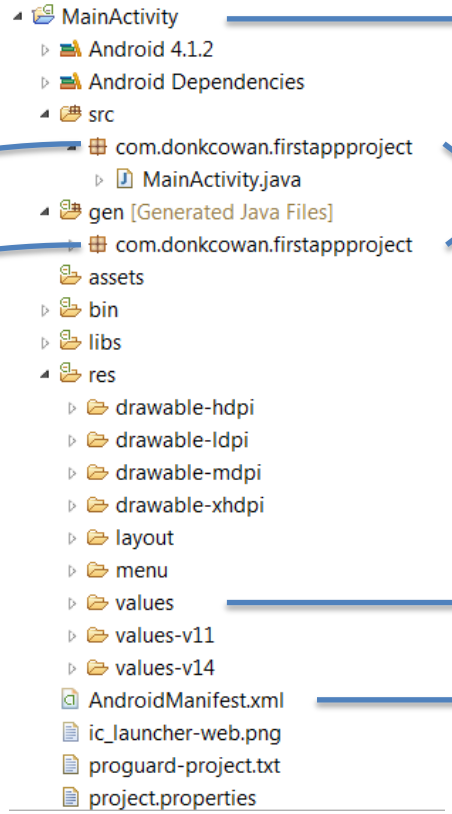


Selecting the Project to Import





Renaming App Project and Package



Rename Project to AddActivityExample

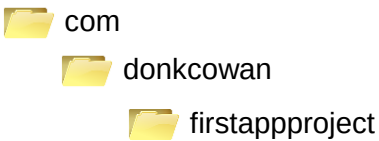
Rename Package to com.donkcowan.addactivityexample

You can use your own name or any other name you choose for the middle qualifier – or leave it as donkcowan for now. When you publish apps to Google Play, you'll need a unique qualifier, usually a domain name which assures uniqueness.

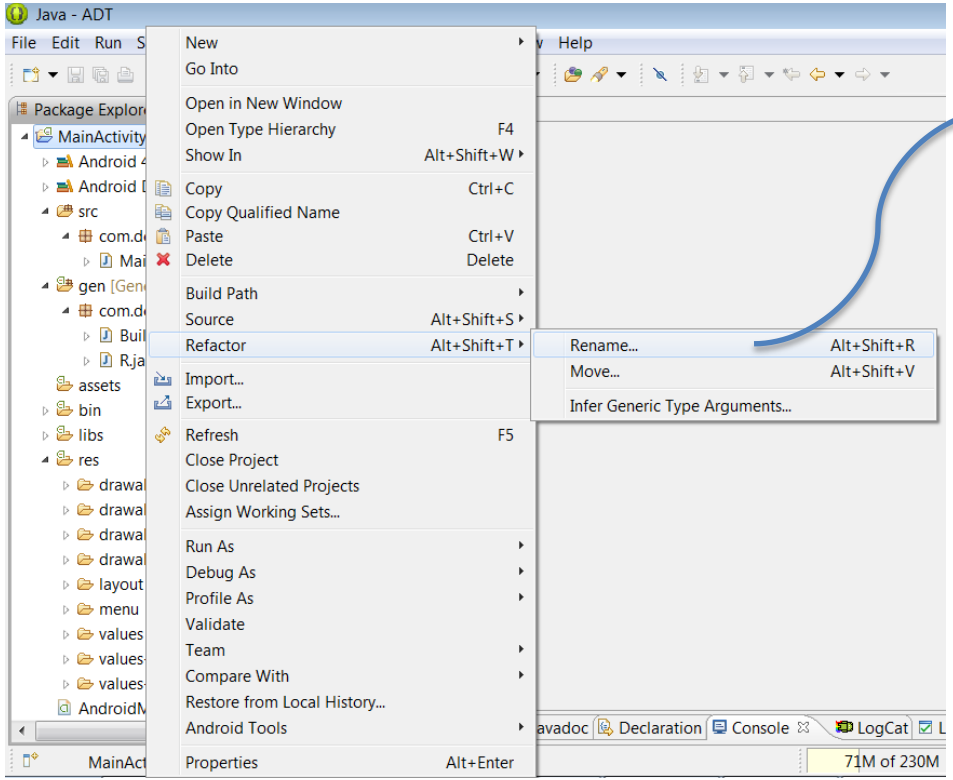
Change app_name in string values

Verify changes in Manifest

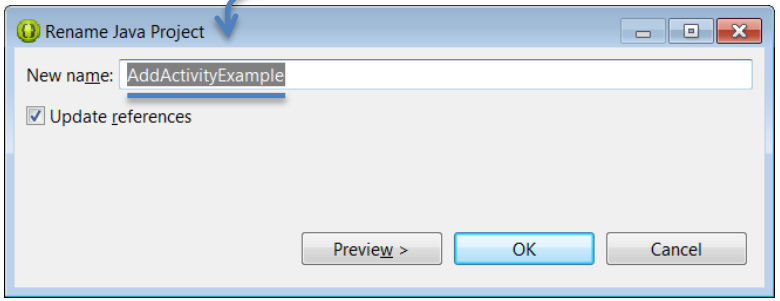
Actual folder structure under src and gen:



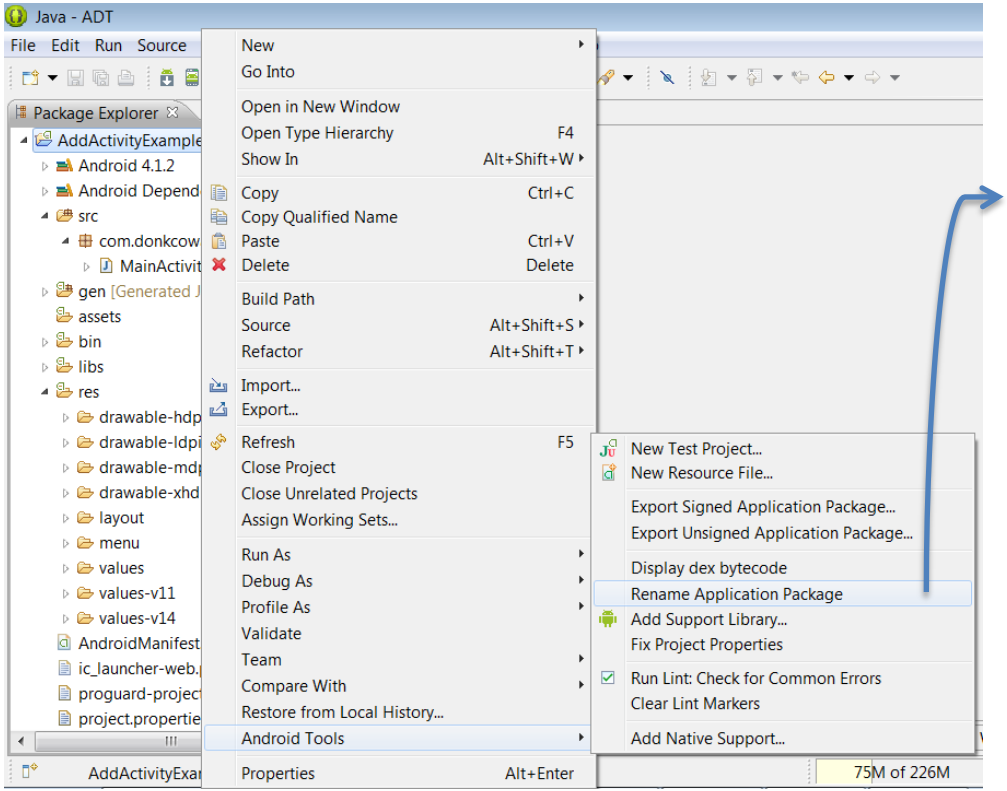
Renaming the Project



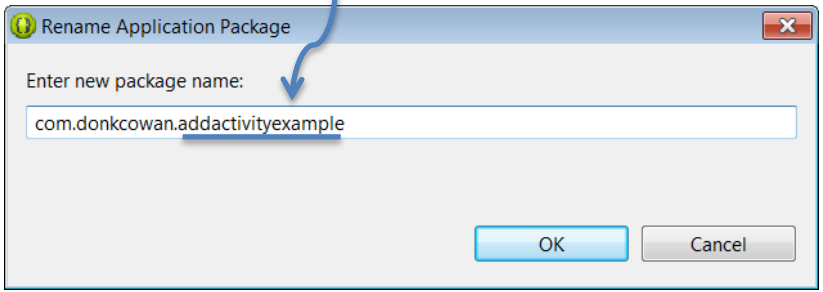
Change to AddActivityExample



Renaming the Application Package

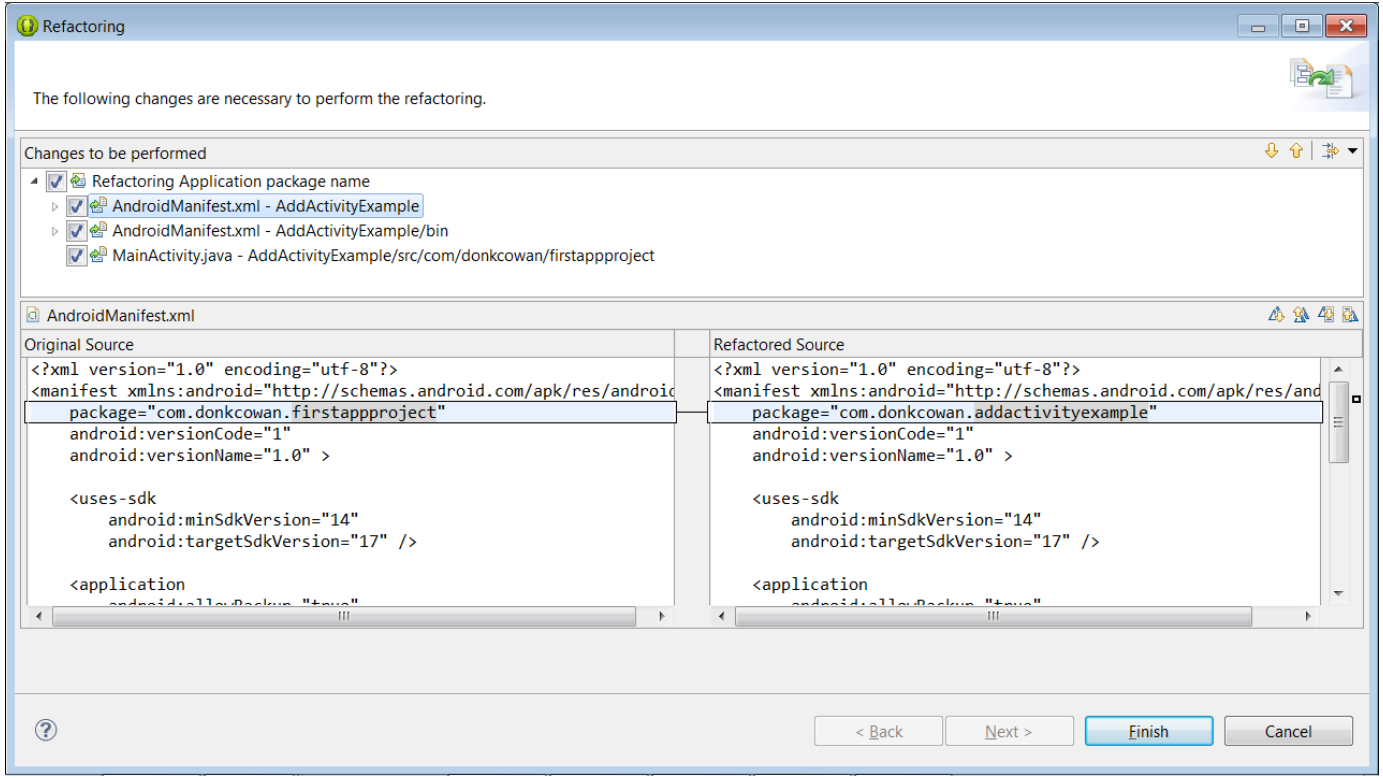


Change to addactivityexample

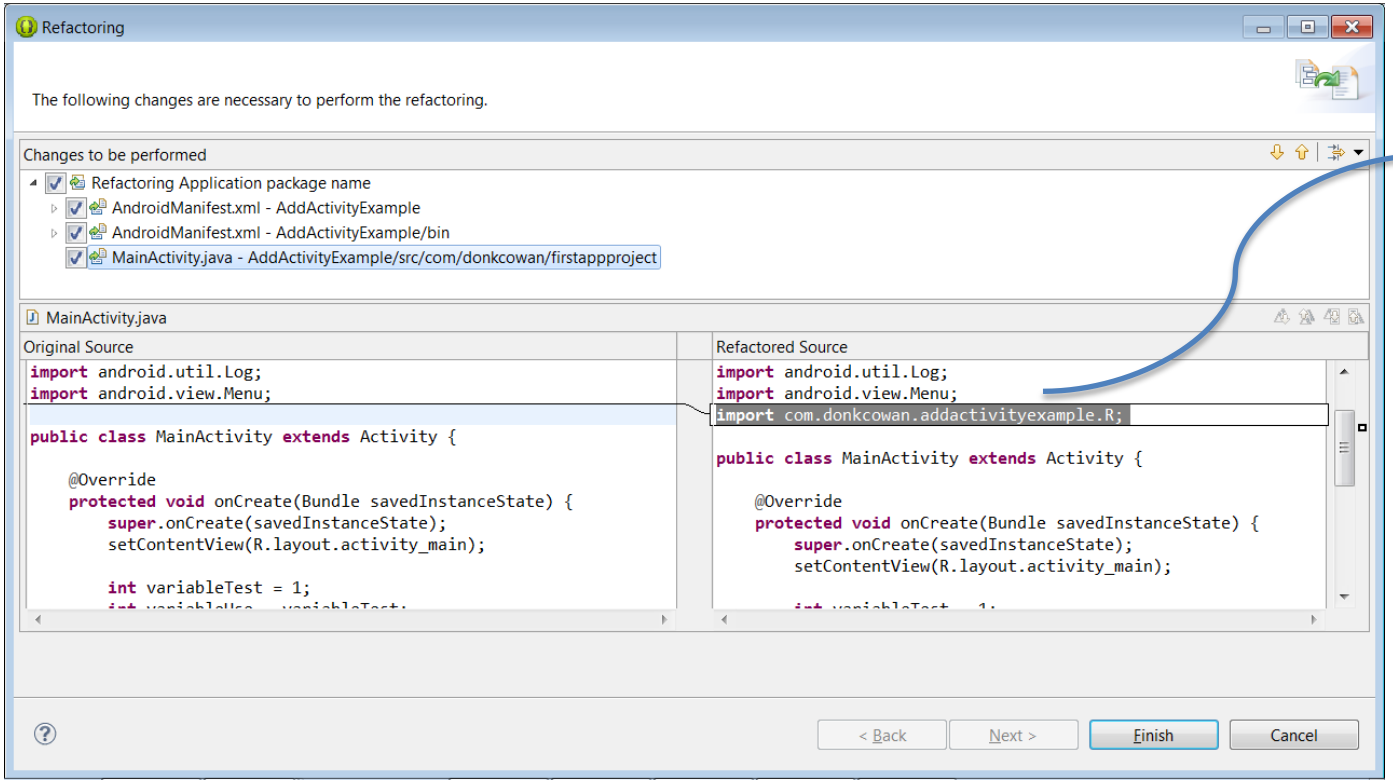


Change Verification

Gives you an opportunity to preview the changes that will be made to your application.

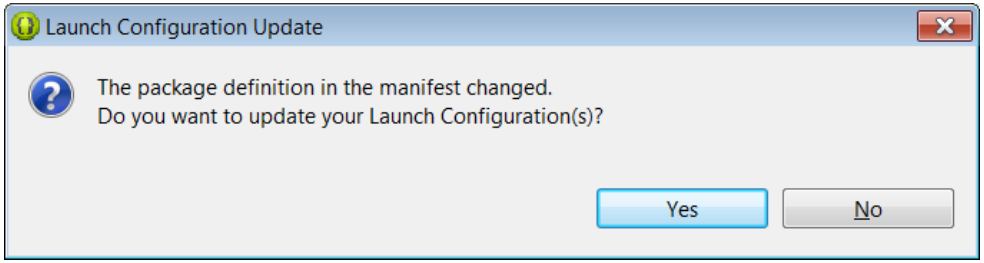


Added Import Statement



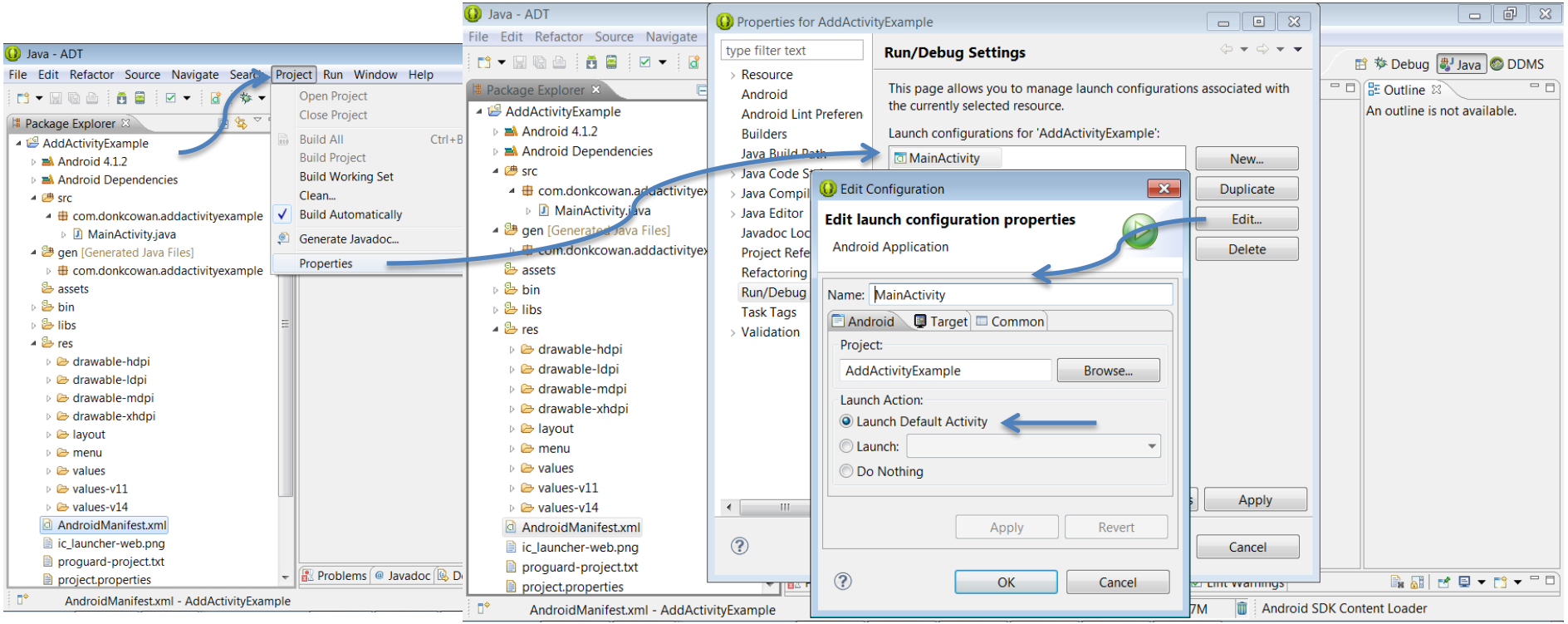
An import statement may be added to your Java source for the .R file. Once you've done all your renaming you won't need this, but it does no harm to include it.

Updating the Launch Configuration

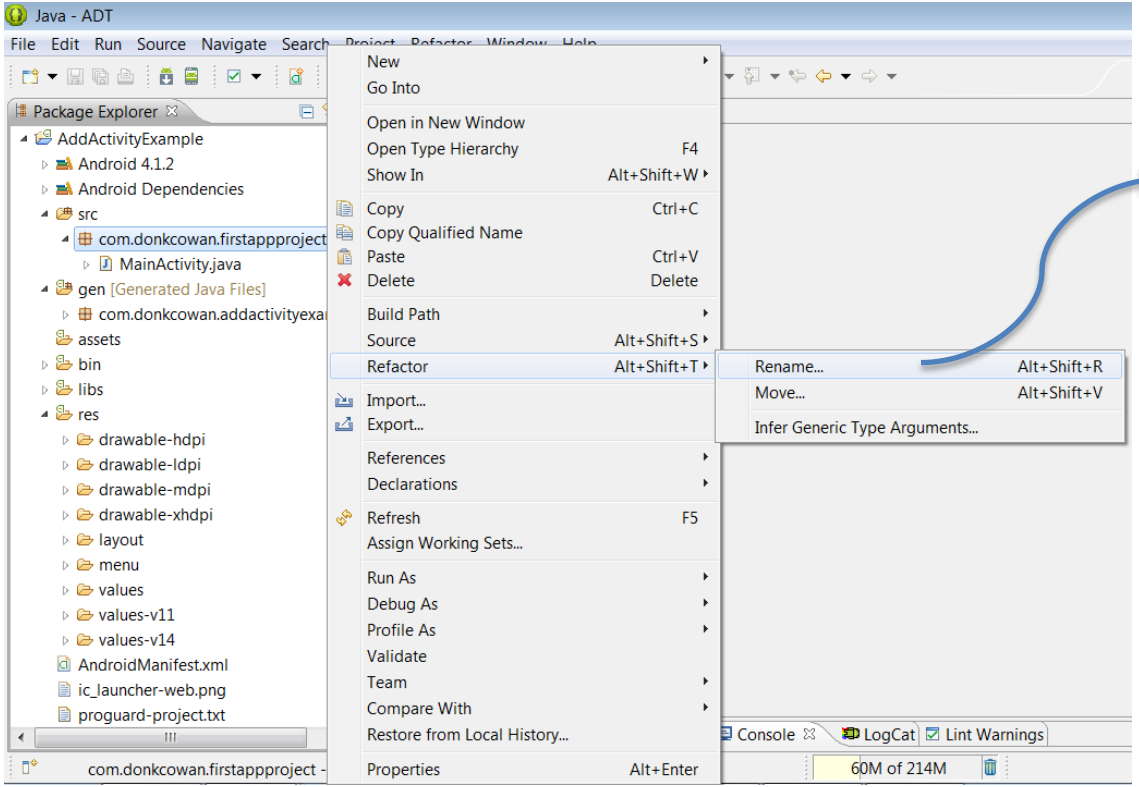


Infinite Skills

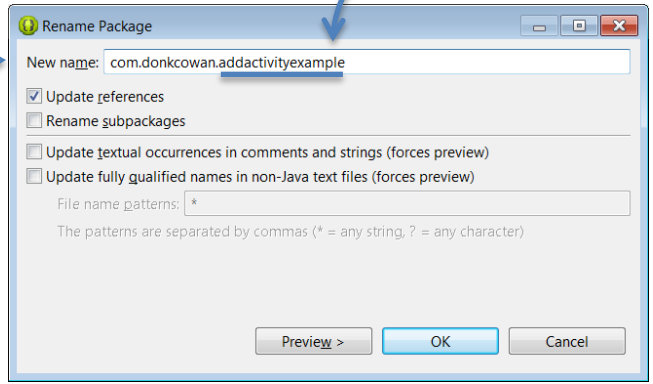
Launch Configuration Management



Renaming the Src Package Name

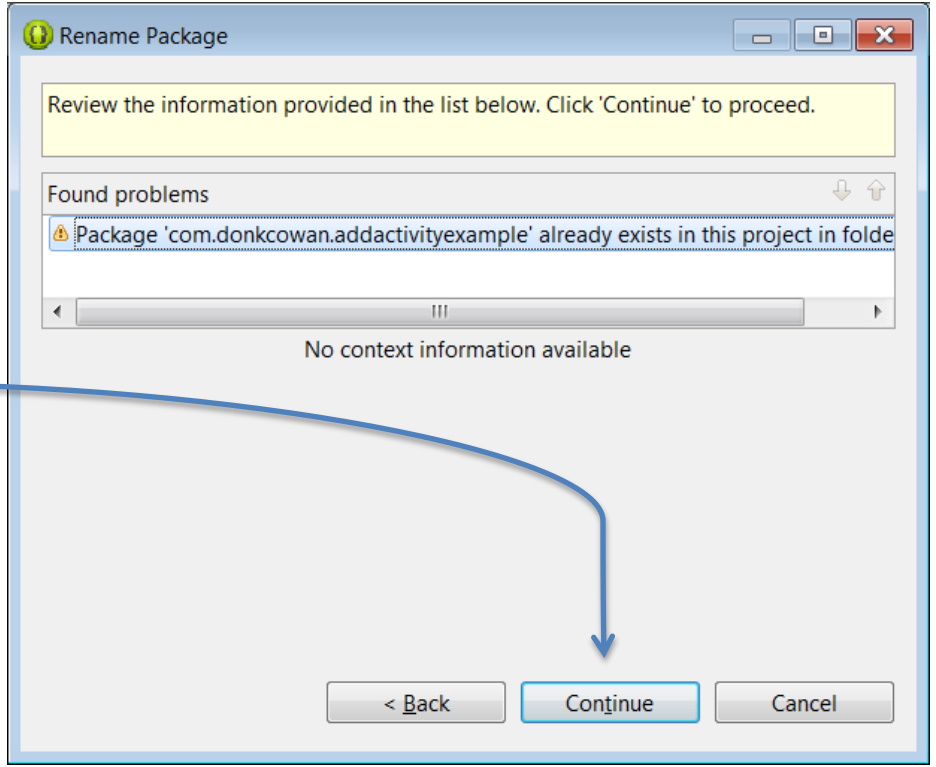


Change to addactivityexample



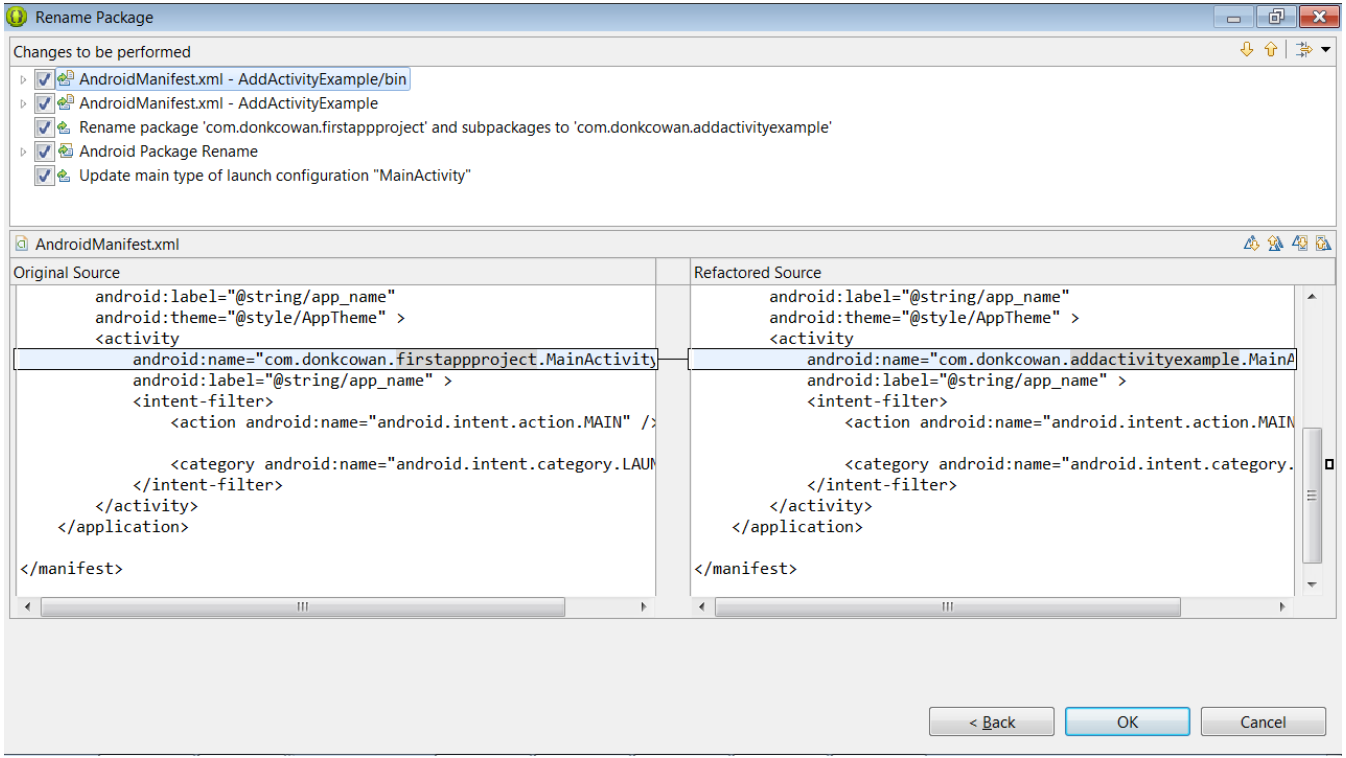
Package Already Exists

Because you've already made changes using the addactivityexample package name, this warning appears. Just ignore it and click continue.

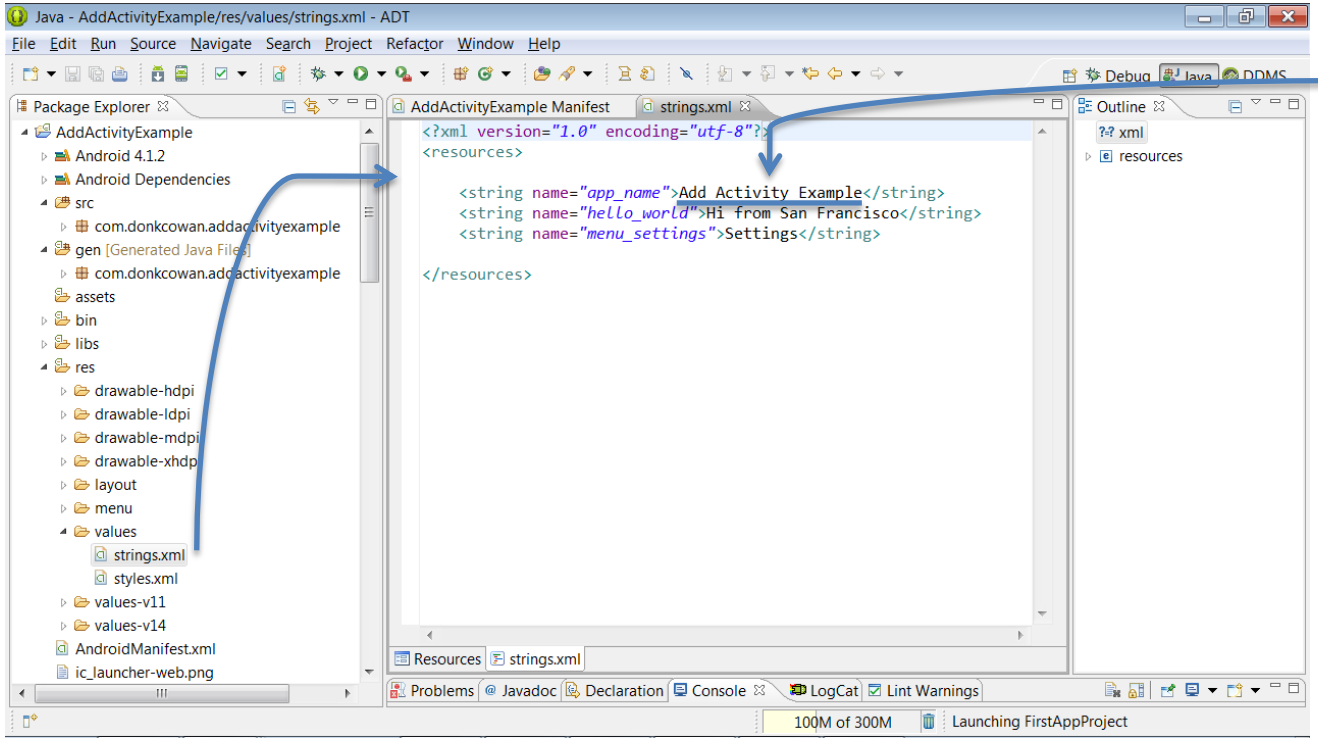


Preview of Changes

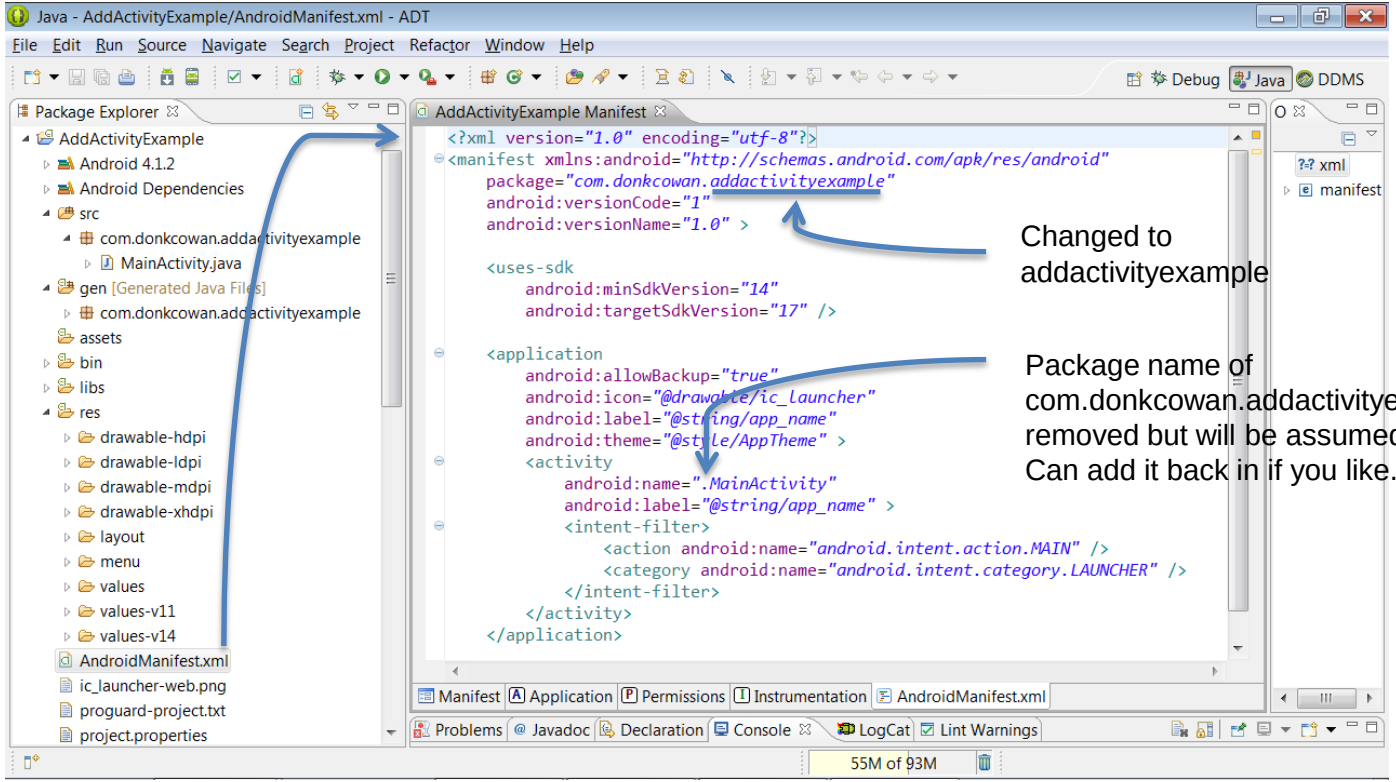
Gives you an opportunity to preview the changes that will be made to your application.



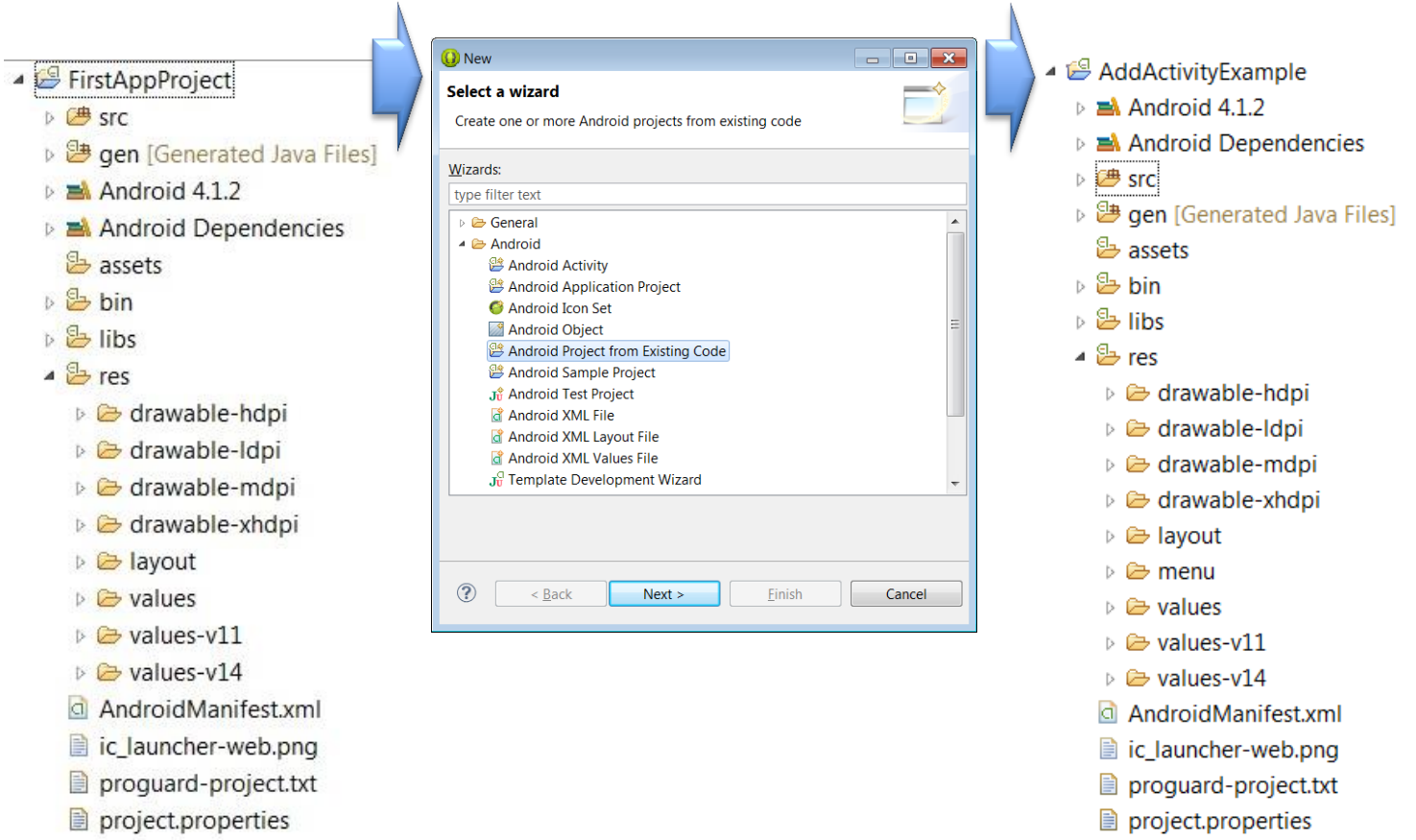
Changing the App Name String



Changes in the Manifest

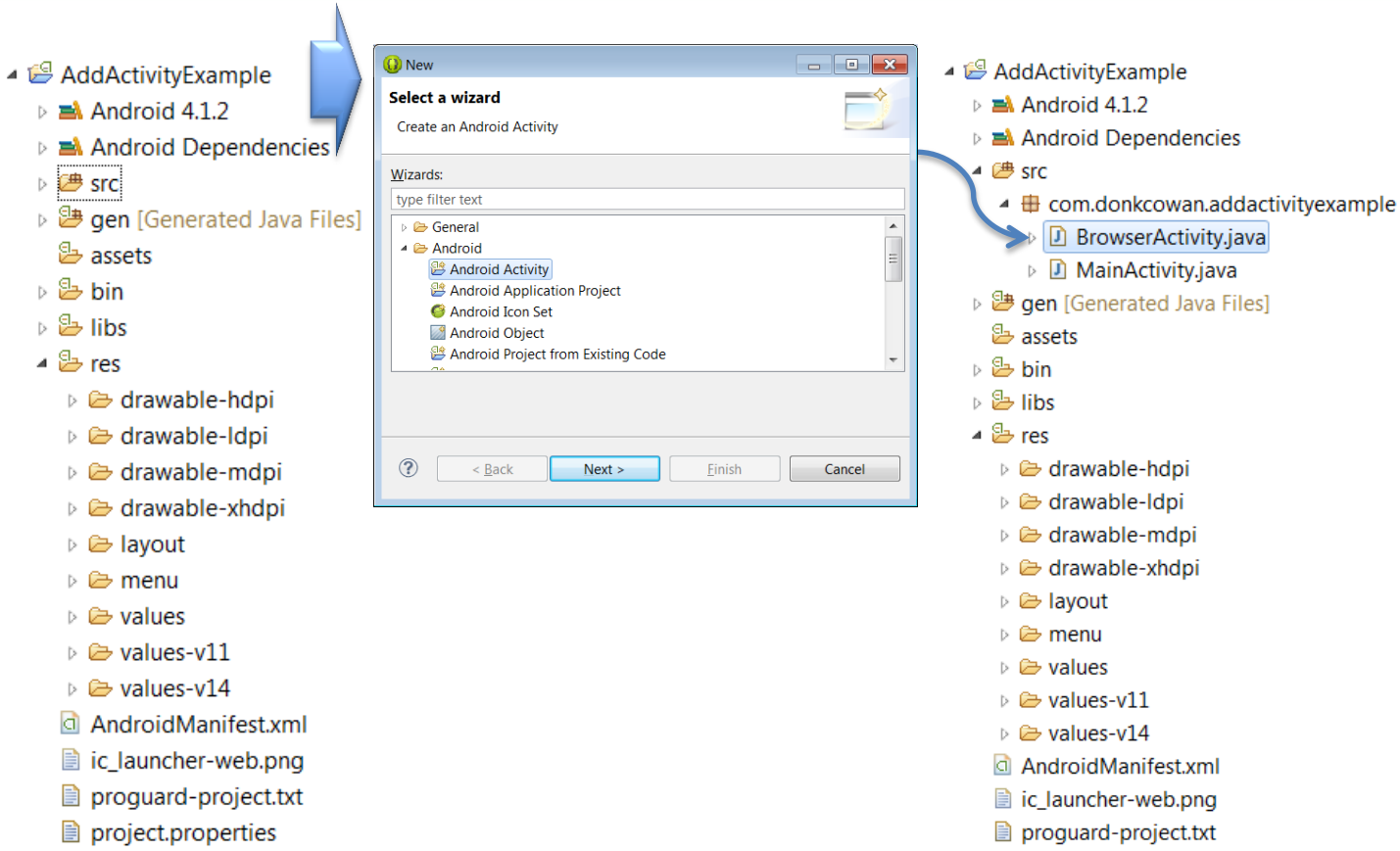


Creating a Project from Existing Code

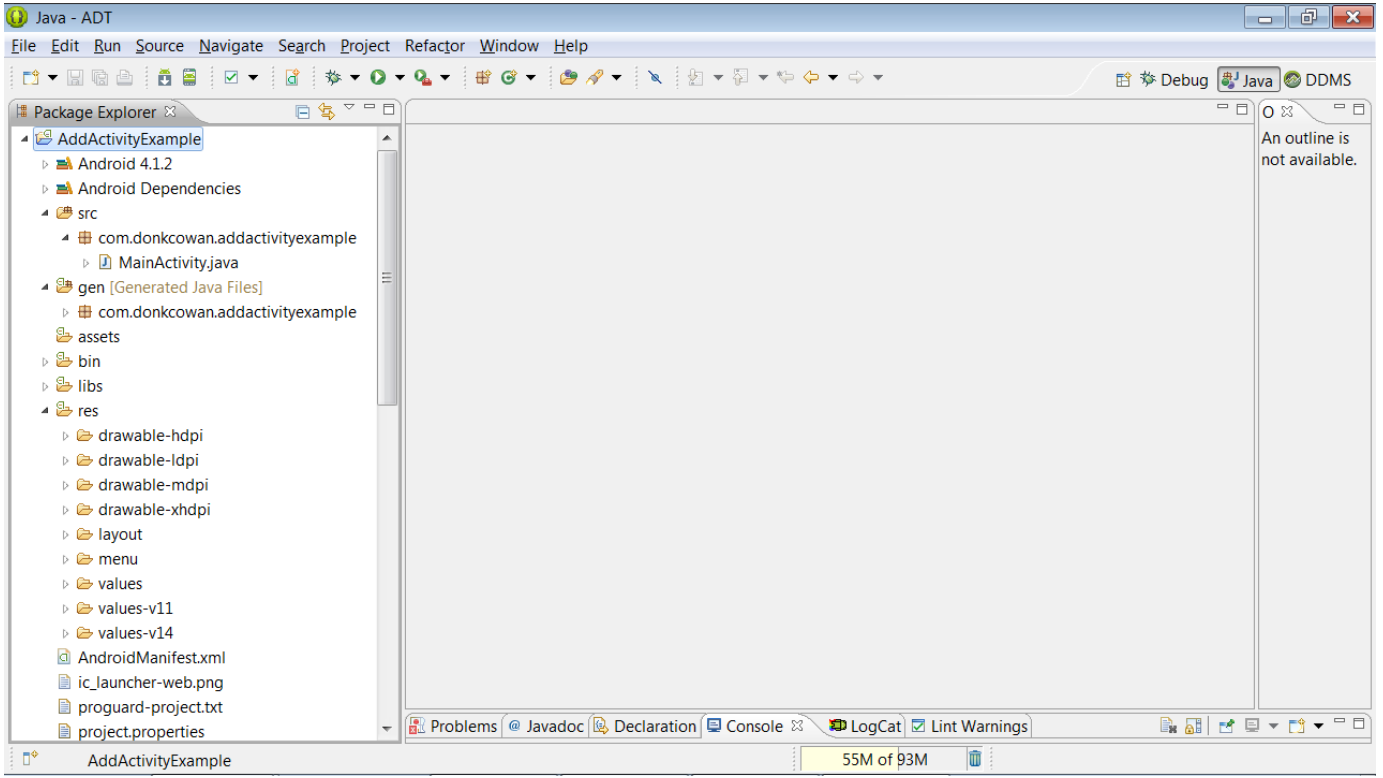


Infinite Skills

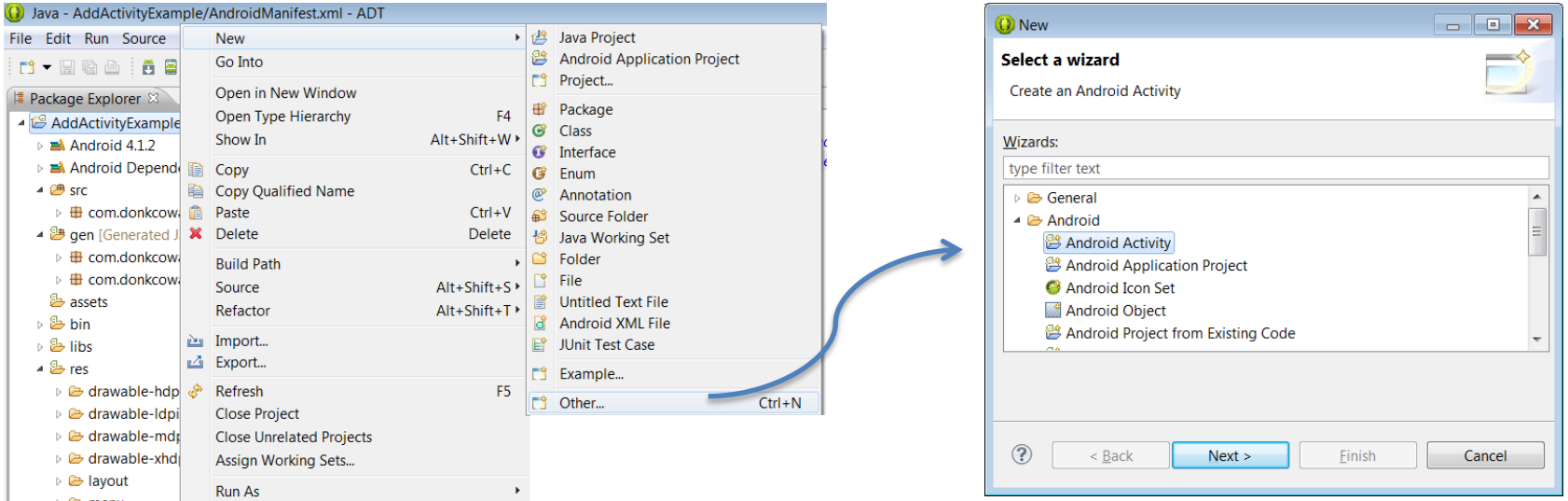
Adding an Activity to an Existing Project



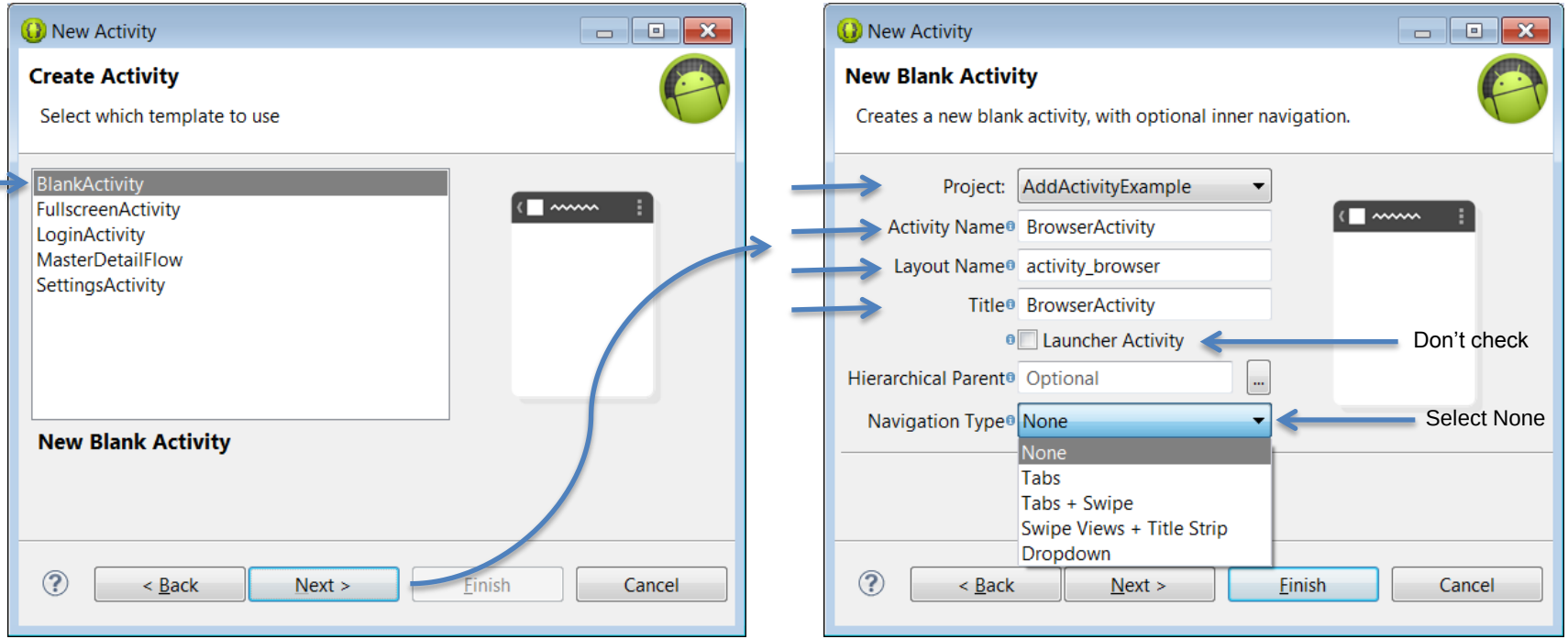
Opening Your Workspace



Selecting the Activity Wizard



Naming a New Activity



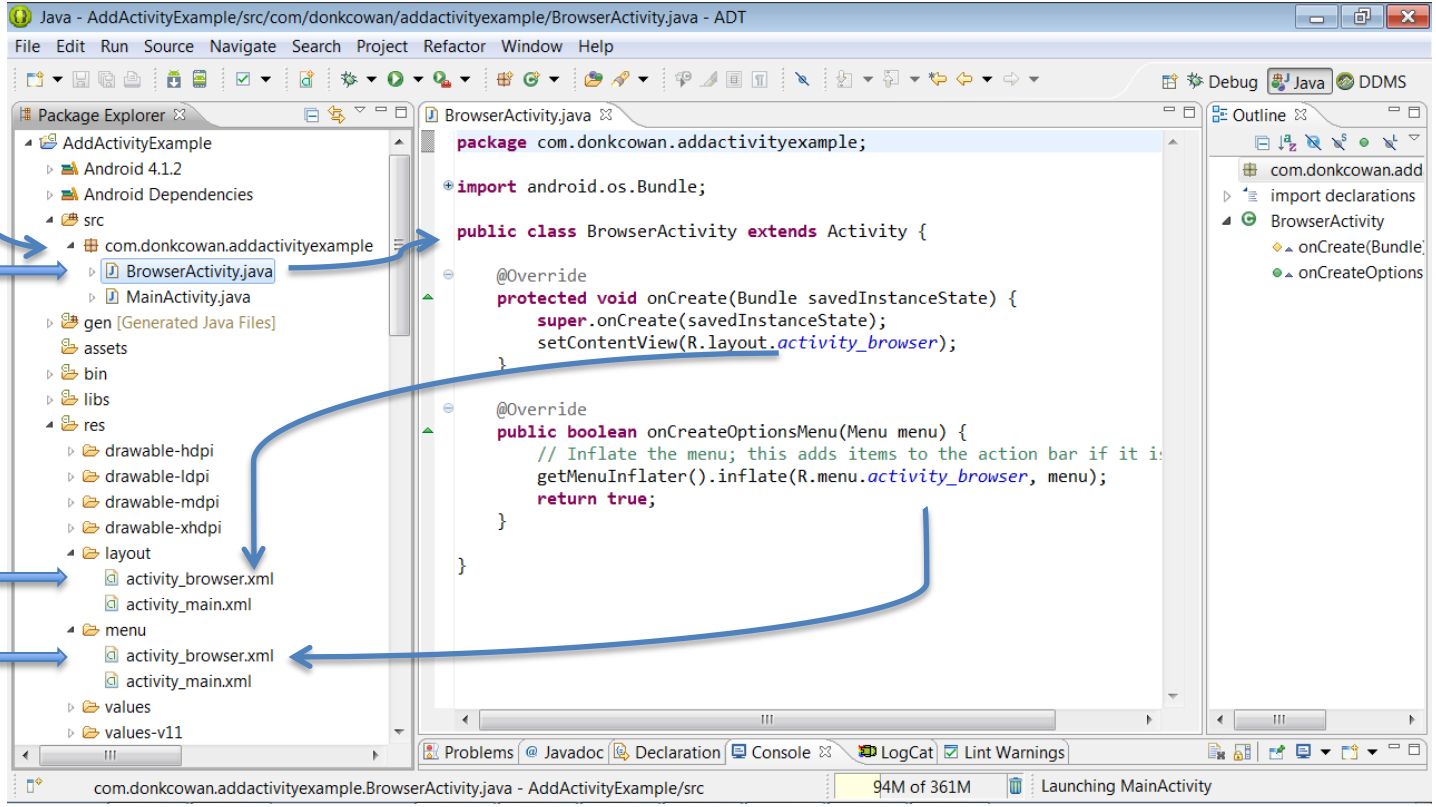
New Activity File Additions

If source is placed in a different project folder, move it to the addactivityexample folder.

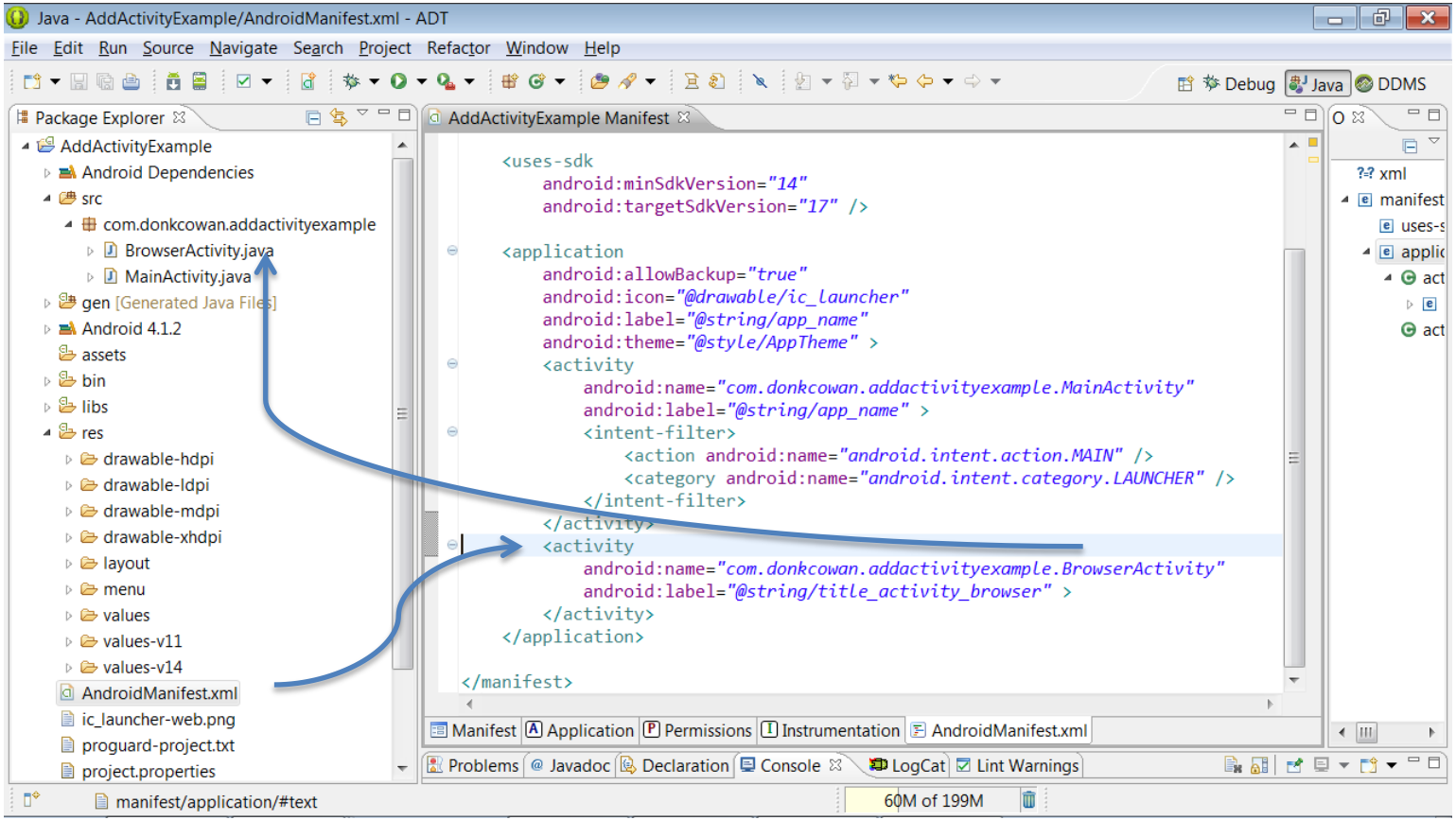
Source

Layout

Menu

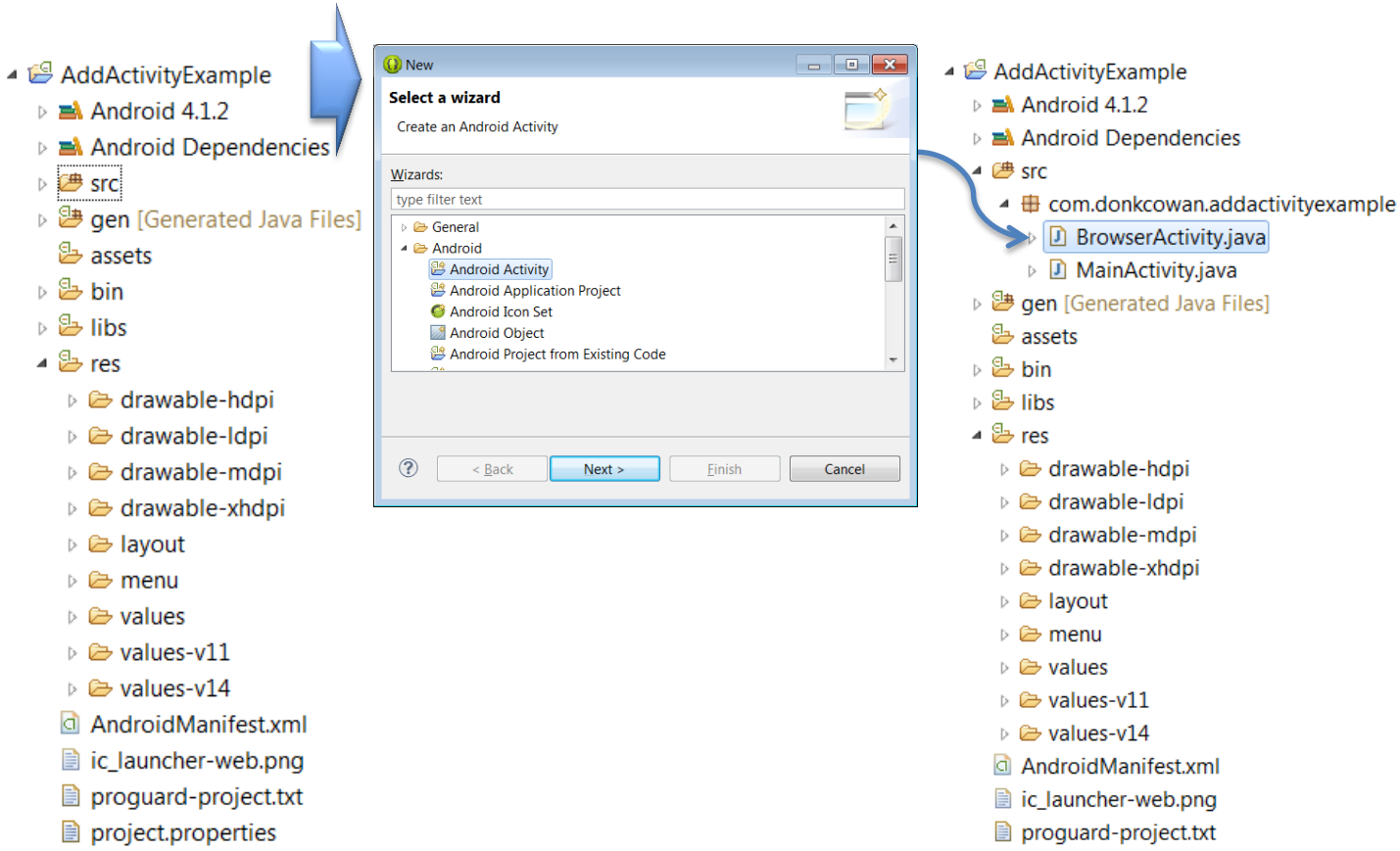


New Activity Manifest Code

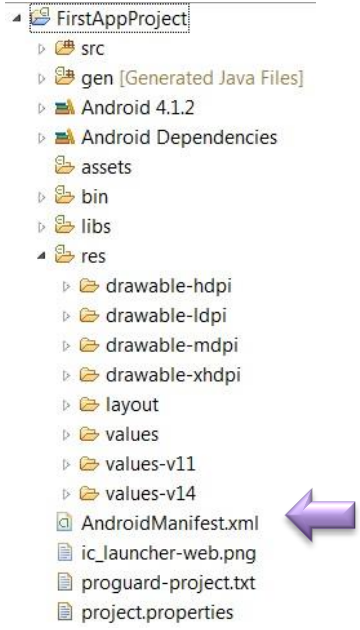


Infinite Skills

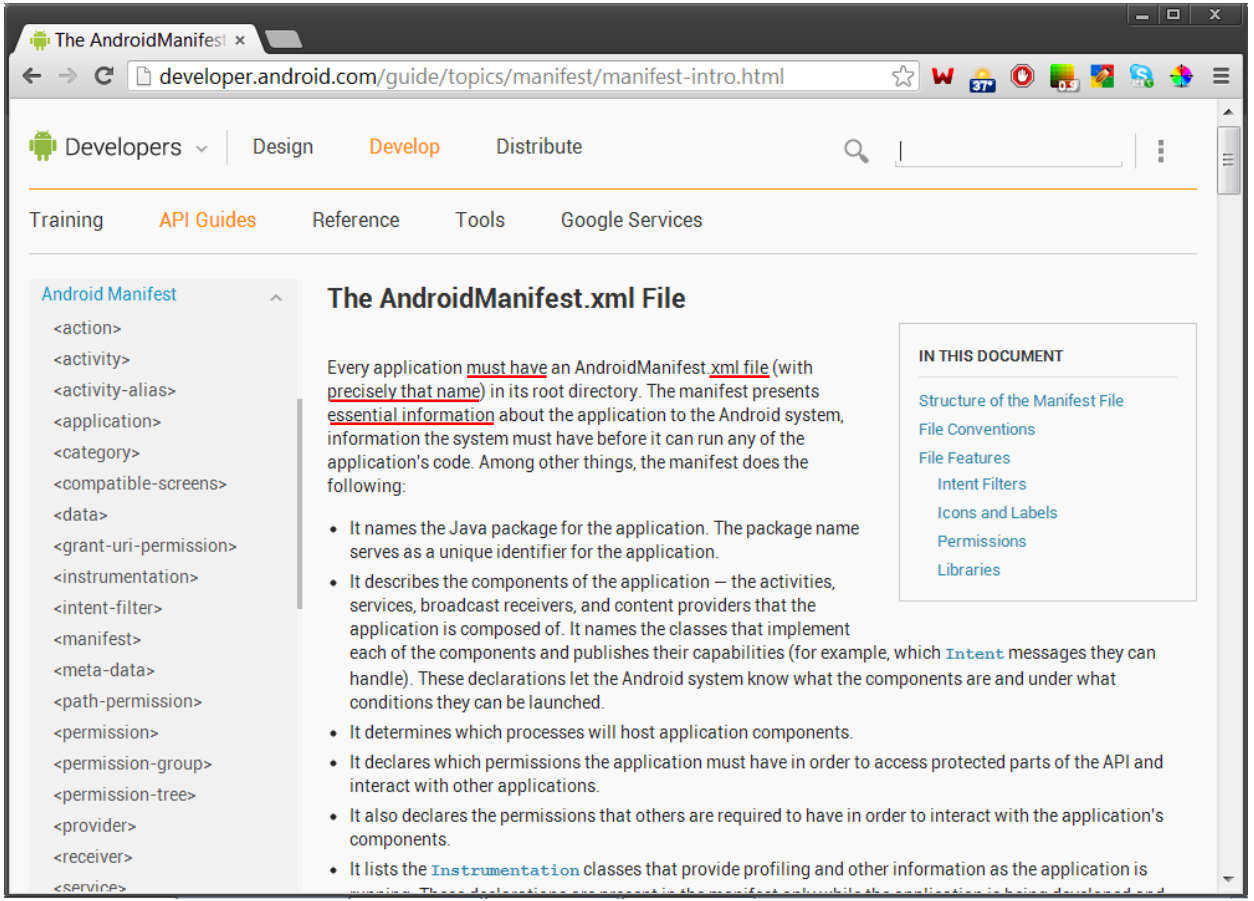
Adding an Activity to an Existing Project



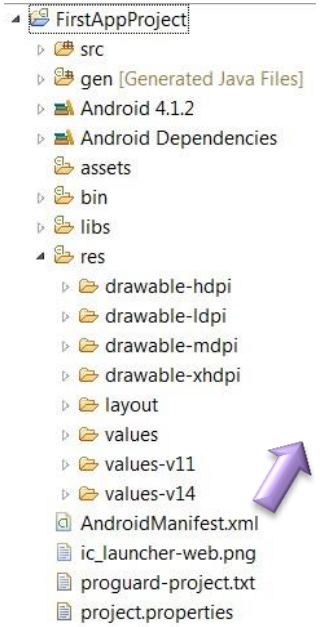
The Android Manifest



http://developer.android.com/guide/topics/manifest/manifest-intro.html



First App Manifest



Package Name

Version IDs

SDK Levels

App Attributes

Components

Intent Filters

-

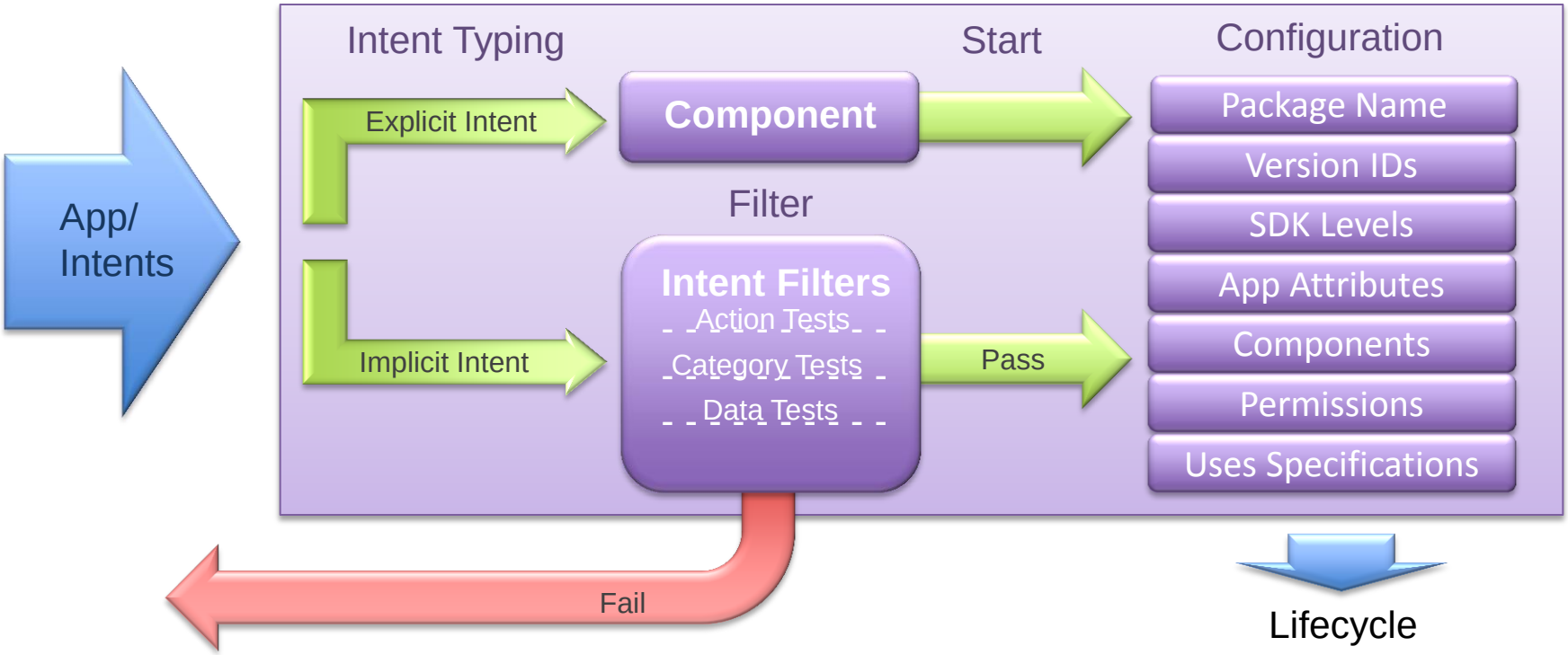
-

-

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.donkcowan.firstappproject"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="17" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.donkcowan.firstappproject.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Names defined in resources

Manifest Uses



Intent Filter Standard Actions

Most important and frequently used

Start up component as the initial activity of a task.

Display data for the user to edit.

Standard Activity Actions
(to be performed)

ACTION_MAIN
ACTION_VIEW
ACTION_ATTACH_DATA
ACTION_EDIT
ACTION_PICK
ACTION_CHOOSER
ACTION_GET_CONTENT
ACTION_DIAL
ACTION_CALL
ACTION_SEND
ACTION_SENDTO
ACTION_ANSWER
ACTION_INSERT
ACTION_DELETE
ACTION_RUN
ACTION_SYNC
ACTION_PICK_ACTIVITY
ACTION_SEARCH
ACTION_WEB_SEARCH
ACTION_FACTORY_TEST

Standard Broadcast Actions
(were performed)

ACTION_TIME_TICK
ACTION_TIME_CHANGED
ACTION_TIMEZONE_CHANGED
ACTION_BOOT_COMPLETED
ACTION_PACKAGE_ADDED
ACTION_PACKAGE_CHANGED
ACTION_PACKAGE_REMOVED
ACTION_PACKAGE_RESTARTED
ACTION_PACKAGE_DATA_CLEARED
ACTION_UID_REMOVED
ACTION_BATTERY_CHANGED
ACTION_POWER_CONNECTED
ACTION_POWER_DISCONNECTED
ACTION_SHUTDOWN

First App:

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Intent Filter Standard Categories

The activity can be the initial activity of a task and is listed in the top-level application launcher.

CATEGORY_DEFAULT
CATEGORY_BROWSABLE
CATEGORY_TAB
CATEGORY_ALTERNATIVE
CATEGORY_SELECTED_ALTERNATIVE
CATEGORY_LAUNCHER
CATEGORY_INFO
CATEGORY_HOME
CATEGORY_PREFERENCE
CATEGORY_TEST
CATEGORY_CAR_DOCK
CATEGORY_DESK_DOCK
CATEGORY_LE_DESK_DOCK
CATEGORY_HE_DESK_DOCK
CATEGORY_CAR_MODE
CATEGORY_APP_MARKET

First App:

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

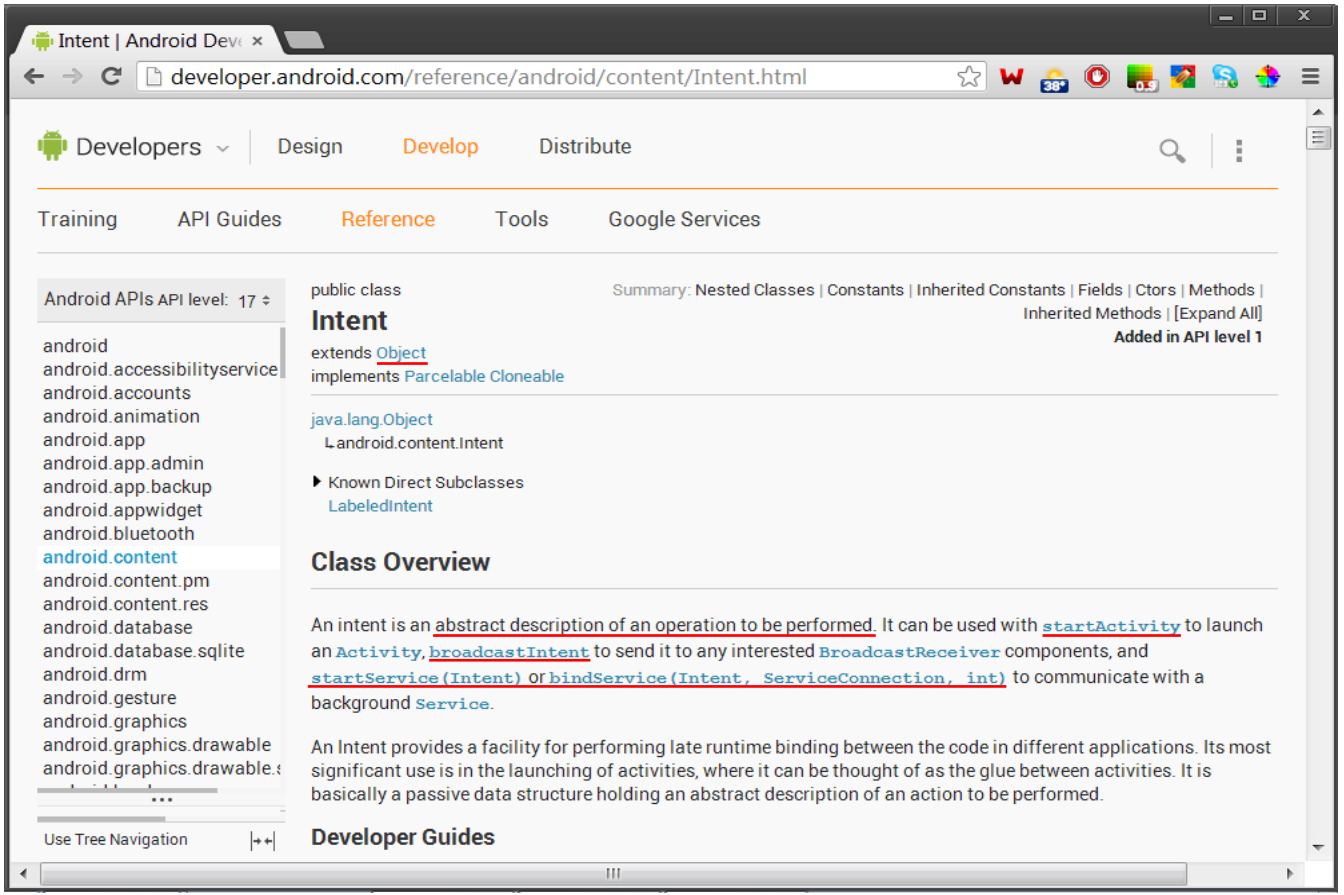
Manifest XML Tags

<action>
<activity>
<activity-alias>
<application>
<category>
<data>
<grant-uri-permission>
<instrumentation>
<intent-filter>
<manifest>
<meta-data>
<permission>
<permission-group>
<permission-tree>
<provider>
<receiver>
<service>
<supports-screens>
<uses-configuration>
<uses-feature>
<uses-library>
<uses-permission>
<uses-sdk>

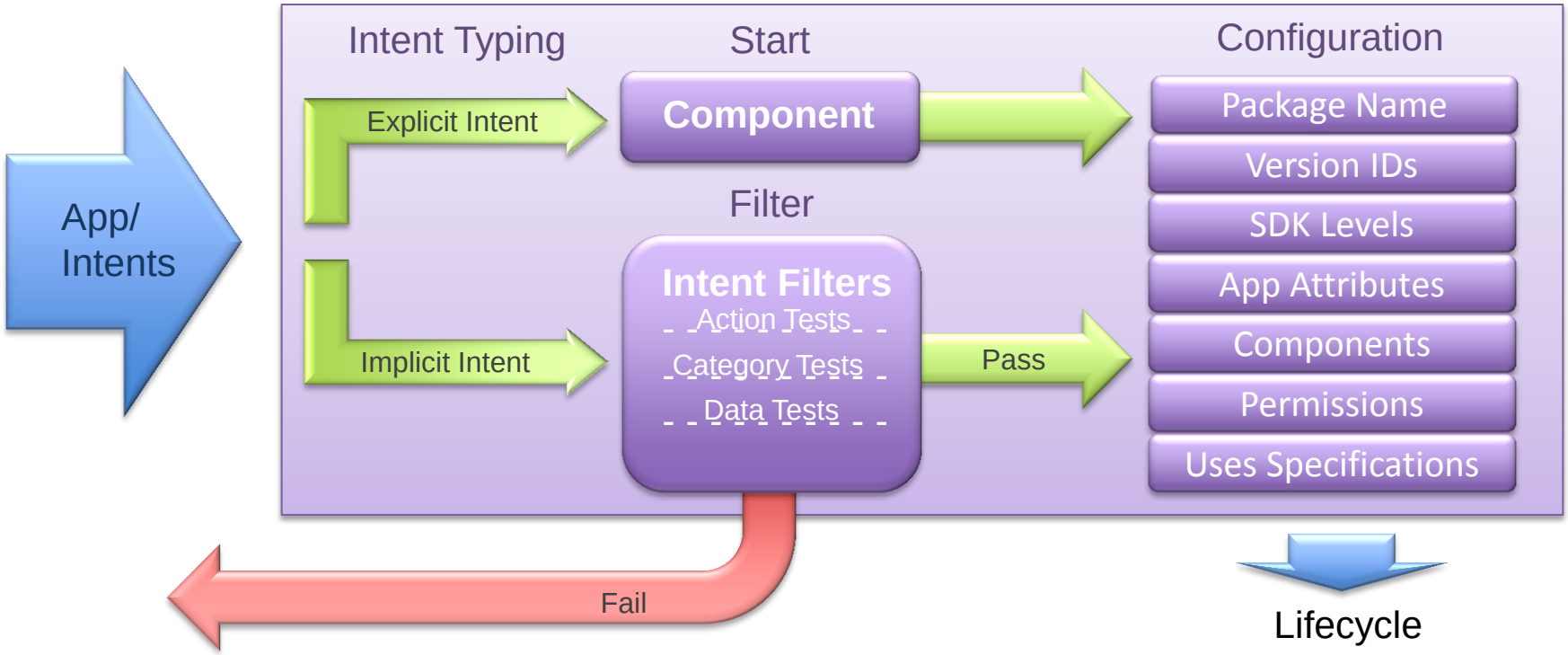
http://developer.android.com/guide/topics/manifest/manifest-intro.html#filestruct

Intents

http://developer.android.com
/reference/android/content/Intent.html



Intent Typing and Filtering



Primary Intent Information

action -- The general action to be performed, such as `ACTION_VIEW`, `ACTION_EDIT`, `ACTION_MAIN`, etc.
data -- The data to operate on, such as a person record in the contacts database, expressed as a `Uri`, such as http://www.google.com

Examples

action	data	
`ACTION_VIEW`	***content://contacts/people/1***	-- Display information about the person whose identifier is "1".
`ACTION_DIAL`	***content://contacts/people/1***	-- Display the phone dialer with the person filled in.
`ACTION_DIAL`	***tel:123***	-- Display the phone dialer with the given number filled in.
`ACTION_EDIT`	***content://contacts/people/1***	-- Edit information about the person whose identifier is "1".
`ACTION_VIEW`	***content://contacts/people/***	-- Display a list of people, which the user can browse through.

Other Intent Information

In addition to these primary attributes, there are a number of secondary attributes that you can also include with an intent:

category -- Gives additional information about the action to execute. For example, CATEGORY_LAUNCHER means it should appear in the Launcher as a top-level application, while CATEGORY_ALTERNATIVE means it should be included in a list of alternative actions the user can perform on a piece of data.

type -- Specifies an explicit type (a MIME type) of the intent data. Normally the type is inferred from the data itself. By setting this attribute, you disable that evaluation and force an explicit type.

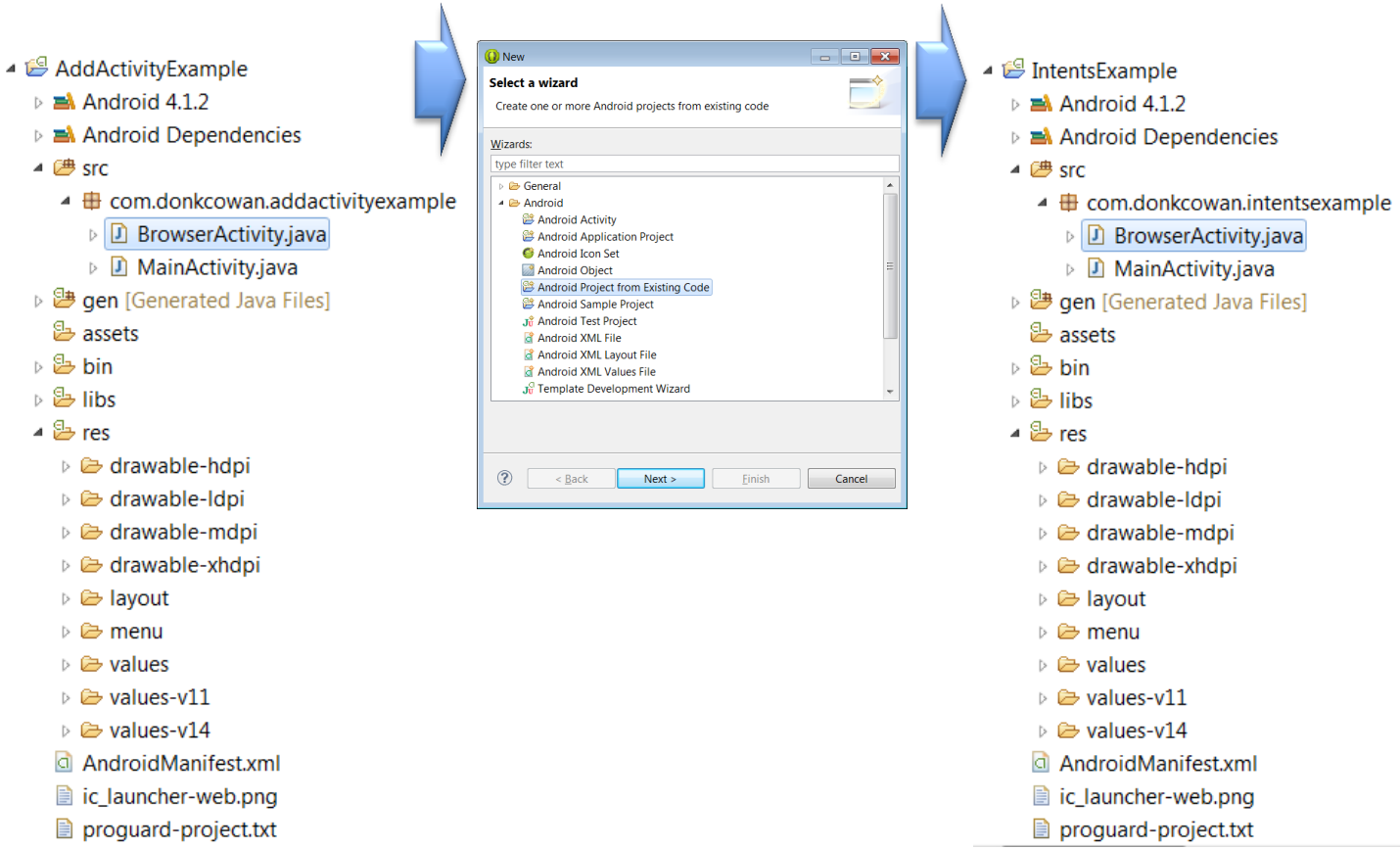
component -- Specifies an explicit name of a component class to use for the intent. Normally this is determined by looking at the other information in the intent (the action, data/type, and categories) and matching that with a component that can handle it. If this attribute is set then none of the evaluation is performed, and this component is used exactly as is. By specifying this attribute, all of the other Intent attributes become optional.

extras -- This is a Bundle of any additional information. This can be used to provide extended information to the component. For example, if we have a action to send an e-mail message, we could also include extra pieces of data here to supply a subject, body, etc.

For more information see: http://developer.android.com/reference/android/content/Intent.html

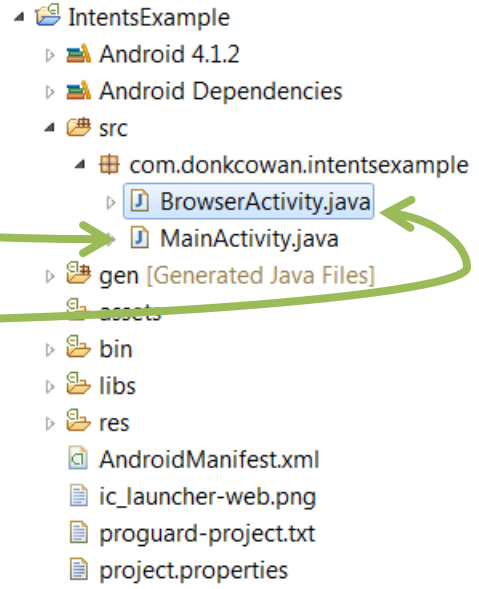
Infinite Skills

Creating a New Project to Demonstrate Intents



Explicit Intent Example

```
package com.donkcowan.intentsexample;

import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Intent browserIntent = new Intent(this, BrowserActivity.class);
        startActivity(browserIntent);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
}
```



Explicit Intent for Activity Start

Declares object as an intent

name of intent object

new instance of object

Intent class

context of **this** activity

Name of component class

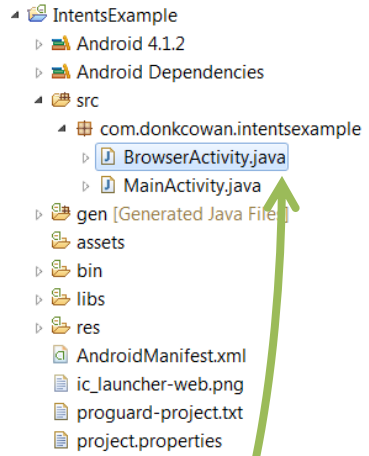
class indication

```
Intent browserIntent = new Intent(this, BrowserActivity.class);
startActivity(browserIntent);
```

Intent for activity to be started

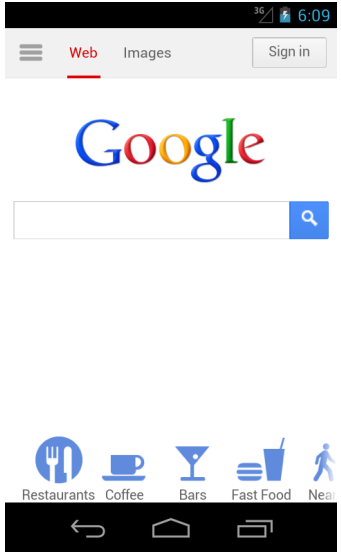
startActivity method

method signature to create an intent for a specific component

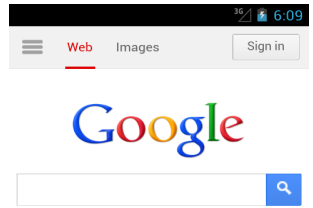


Implicit Intent Example

```
package com.donkcowan.intentsexample;

import android.net.Uri;

public class BrowserActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_browser);

        Intent browserIntent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.com"));
        startActivity(browserIntent);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.activity_browser, menu);
        return true;
    }

}
```



Implicit Intent for Browser Display



Declares object as an intent

name of intent object

new instance of object

Intent object

Intent class

Action View string

Uri class

parse method

Scheme name

Scheme part

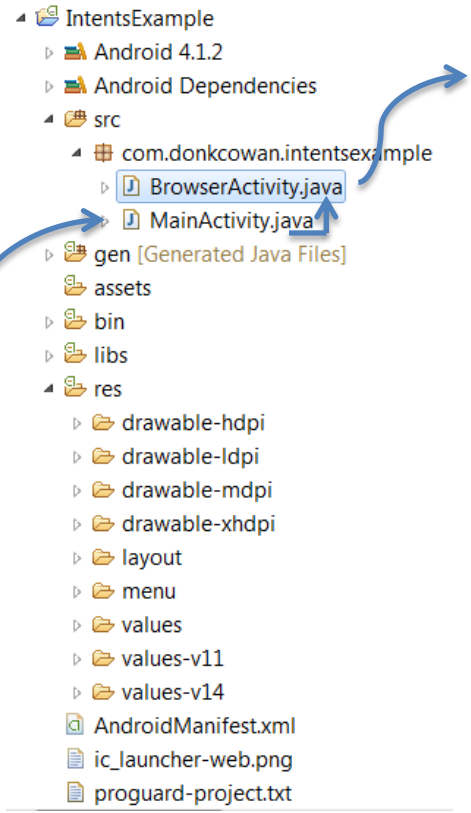
```
Intent browserIntent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.com"));
startActivity(browserIntent);
```

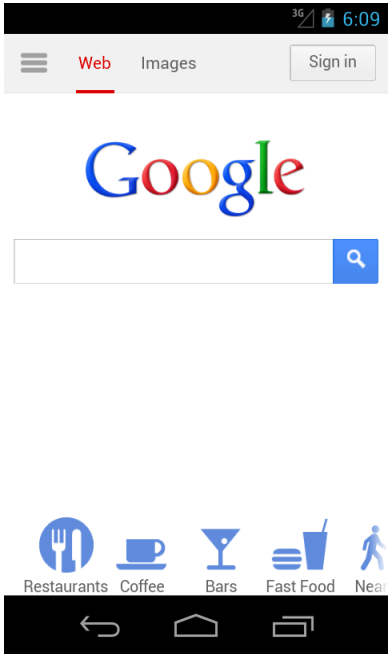
Intent for activity to be started

startActivity method

method signature to create an intent with a given action with a specific data url

How Intents were used in Our Example





Listener for Button Click

declares object as a button

name of button object

casting as a button

find view of button in layout

id of button

```
    // Declare button for Get Answer.
    Button getAnswer = (Button)findViewById(R.id.getAnswerButton);

    // Set listener for button.
    getAnswer.setOnClickListener(getAnswerListener);
}
// Define OnClickListener for getAnswer button.
private OnClickListener getAnswerListener = new OnClickListener() {
    public void onClick(View v){ ...
```

Set listener on button object

Listener object

onClick method

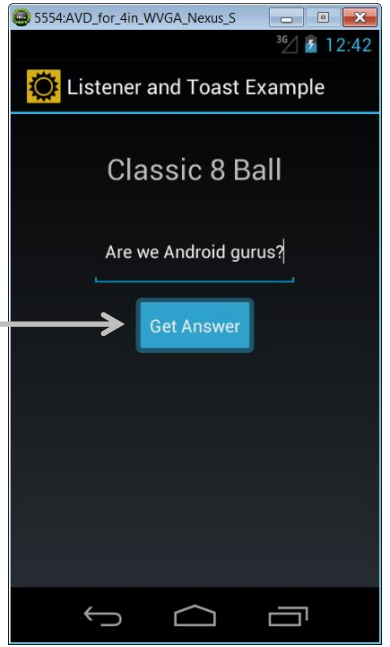
call back code

OnClickListener object

declares OnClickListener object

View object parameter

new instance of object



Standard sequence of statements used for setting up listeners for user interaction with screen objects.

Display Toast

```
Context context = getApplicationContext();
int duration = Toast.LENGTH_LONG;
Toast toast = Toast.makeText(context, text, duration);
toast.show();
```

declares object as a Context

name of context object

duration variable

getApplicationContext method

Toast class constant

Declaration of Toast object

Toast object name

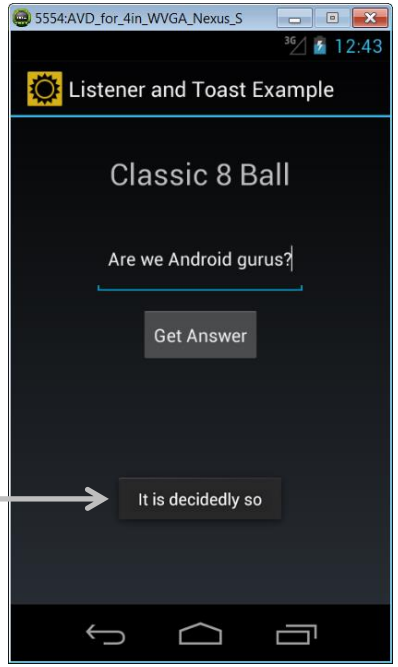
makeText method of Toast class

Text variable (defined elsewhere)

duration variable

name of context object

show method to display toast



Listeners and Toasts

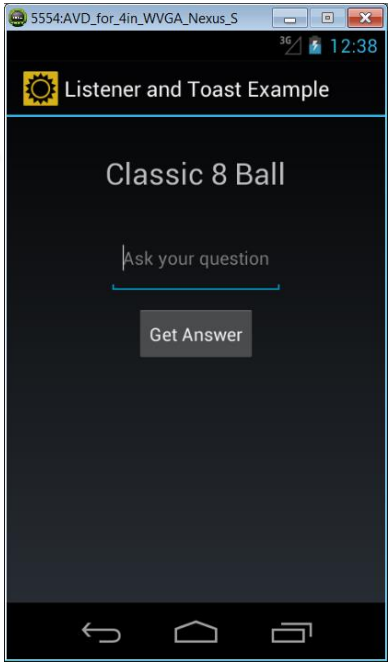
Set up display

Accept input

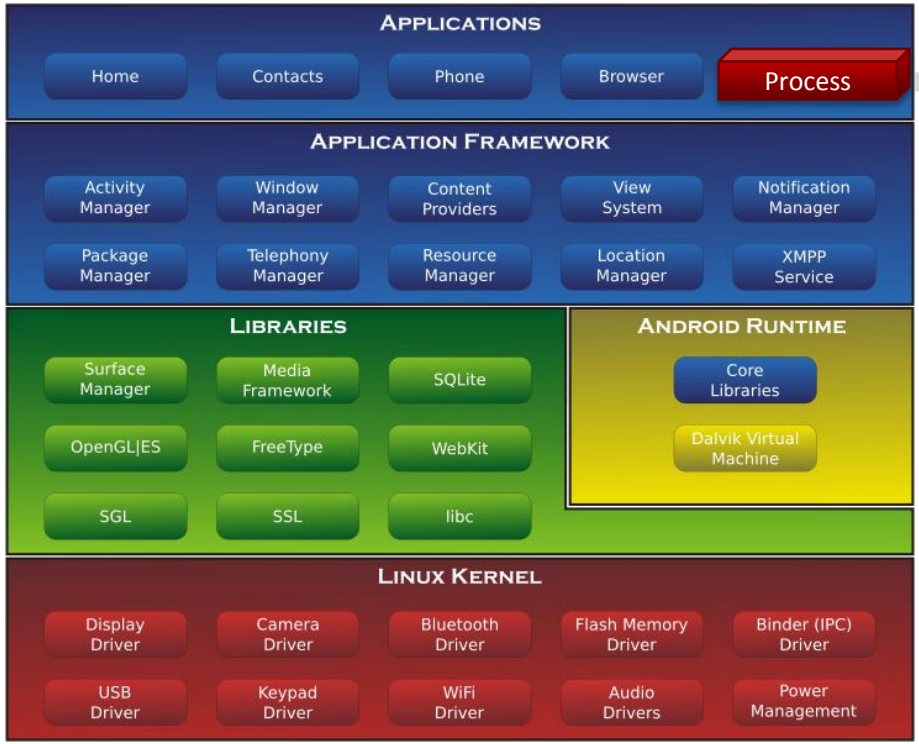
Set up listener for button click

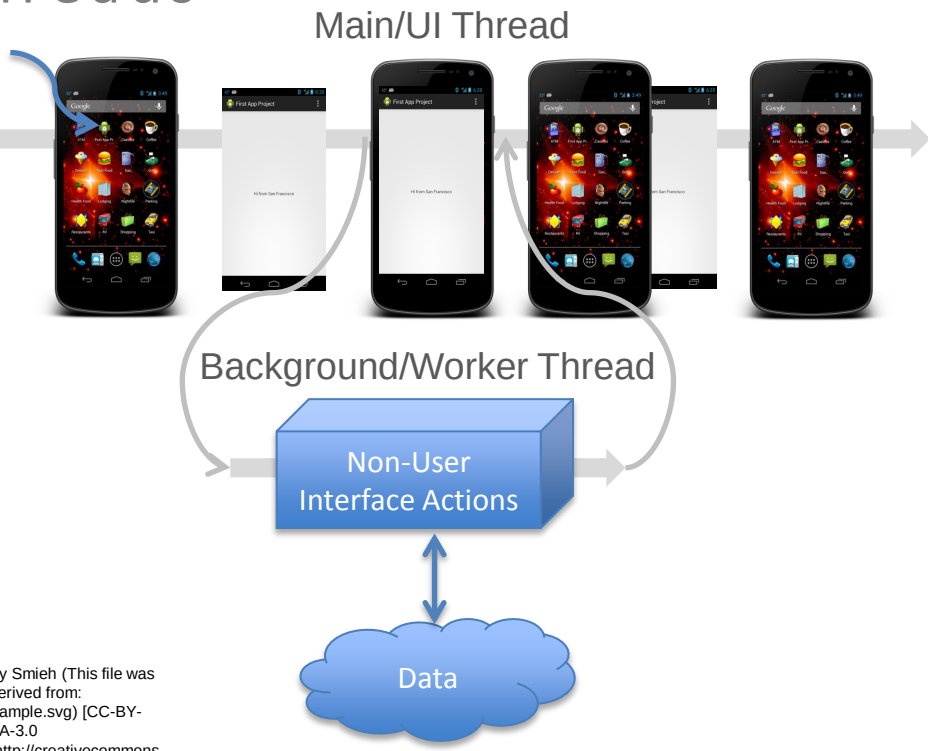
Respond to click listener

Toast answer



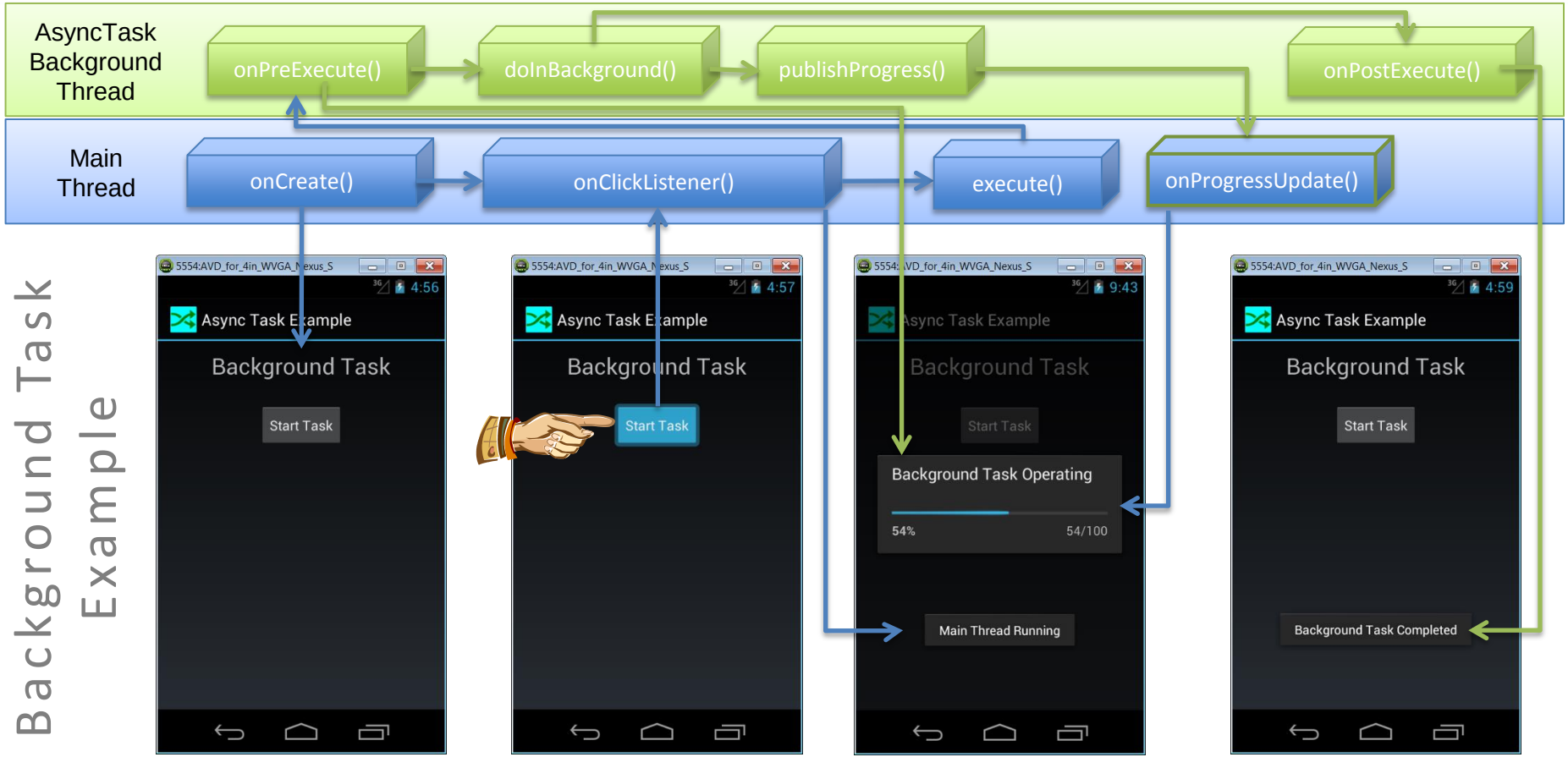
App Threads





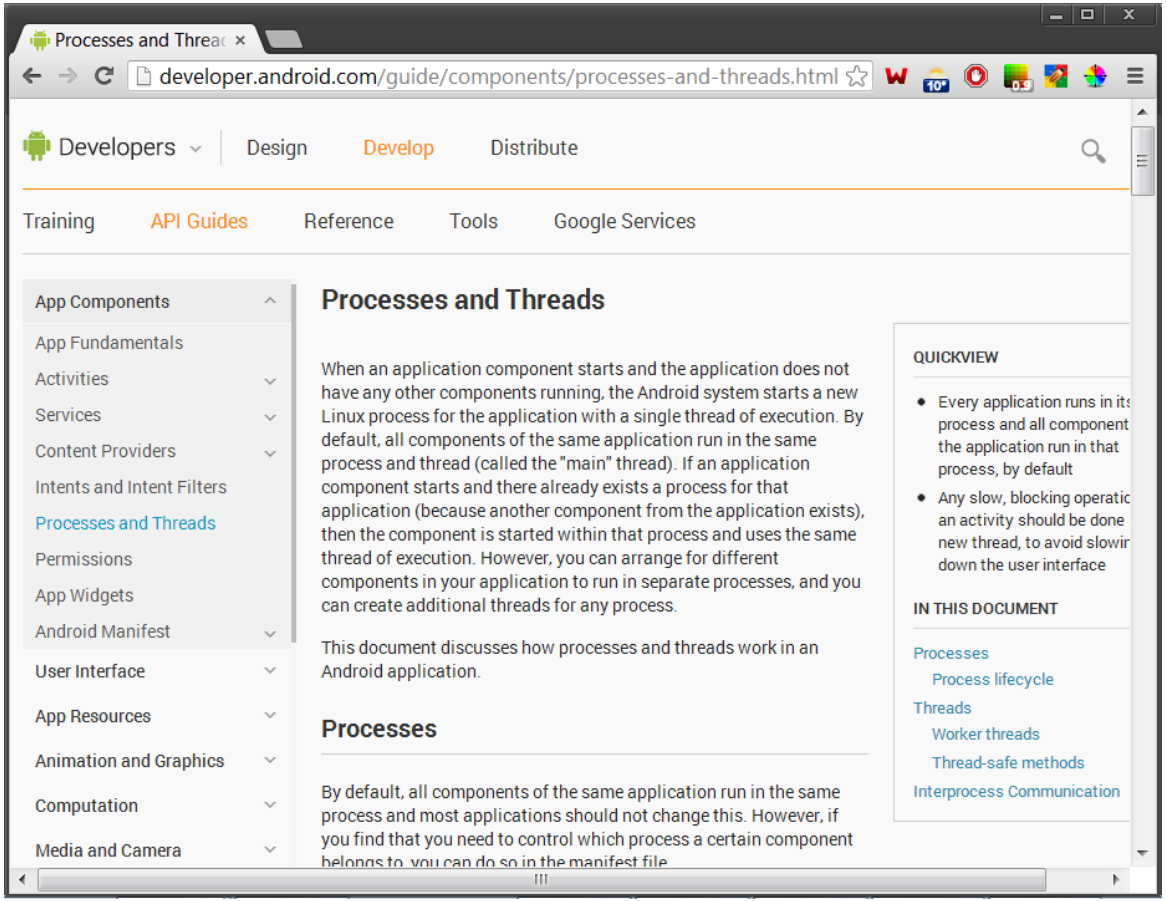
By Smieh (This file was derived from: Sample.svg) [CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

Background Task Example



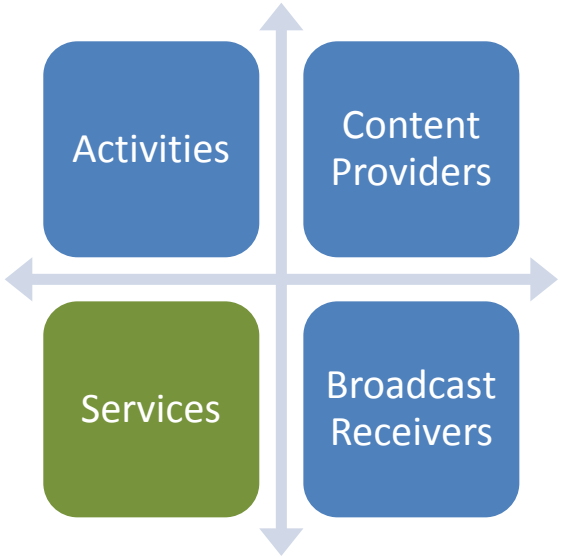
Processes and Threads

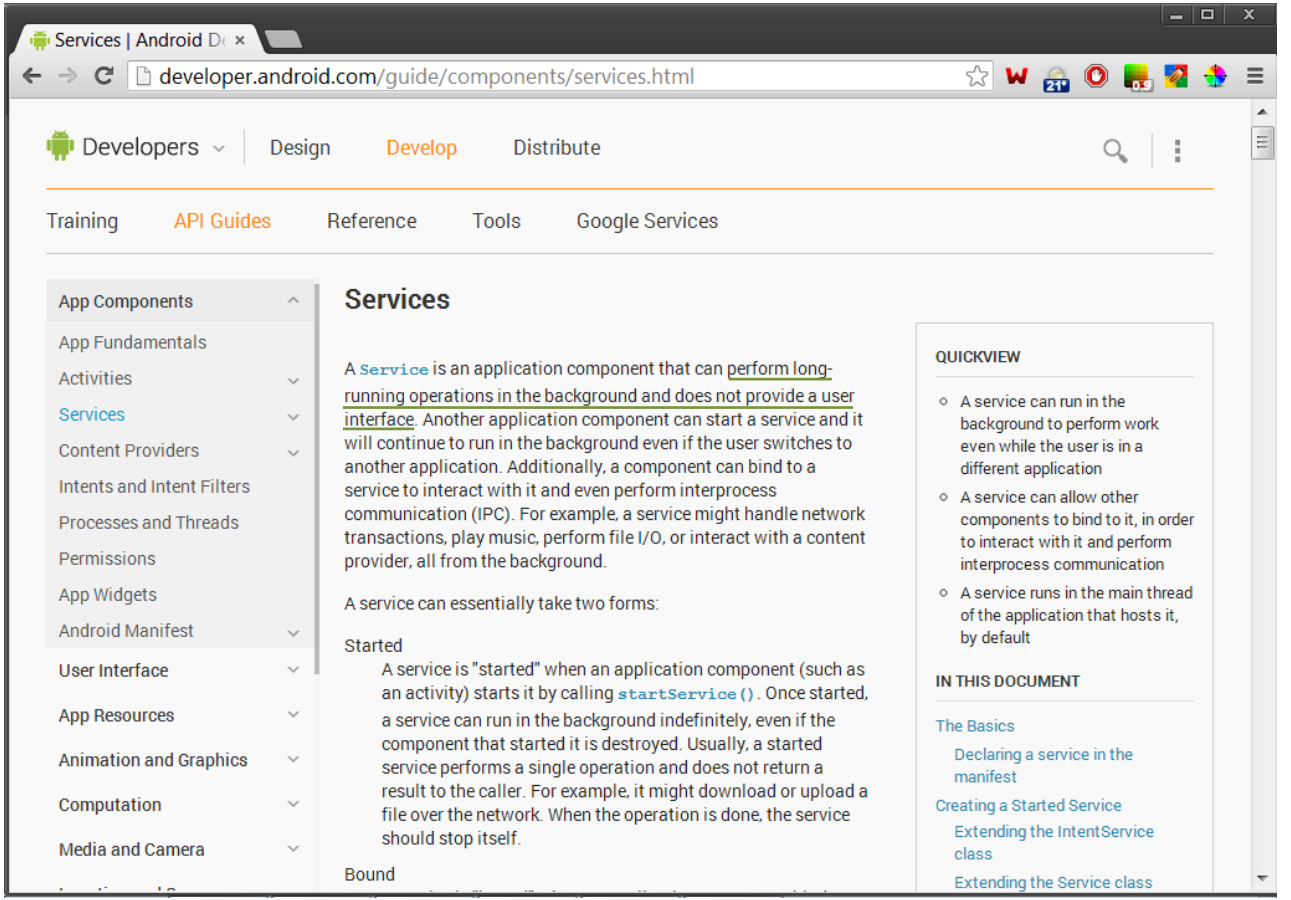
http://developer.android.com/guide/components/processes-and-threads.html



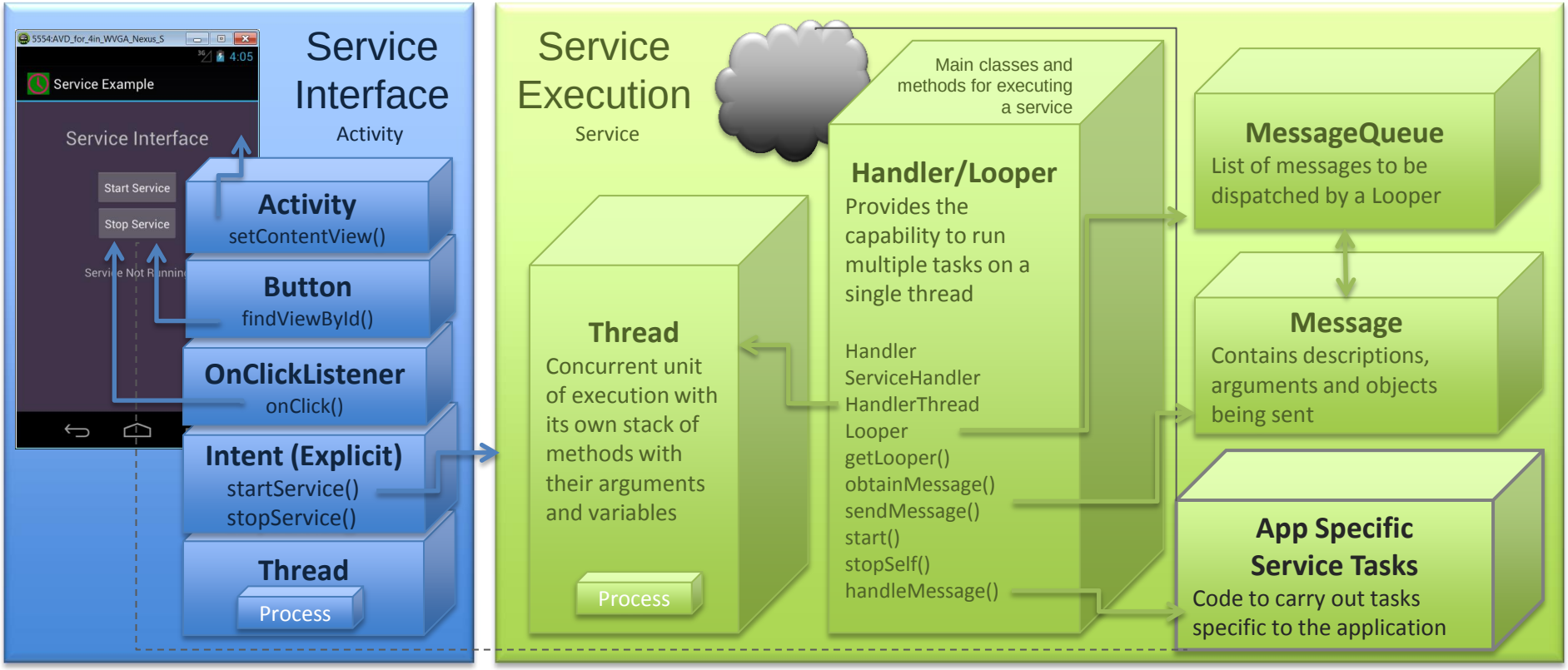
Services

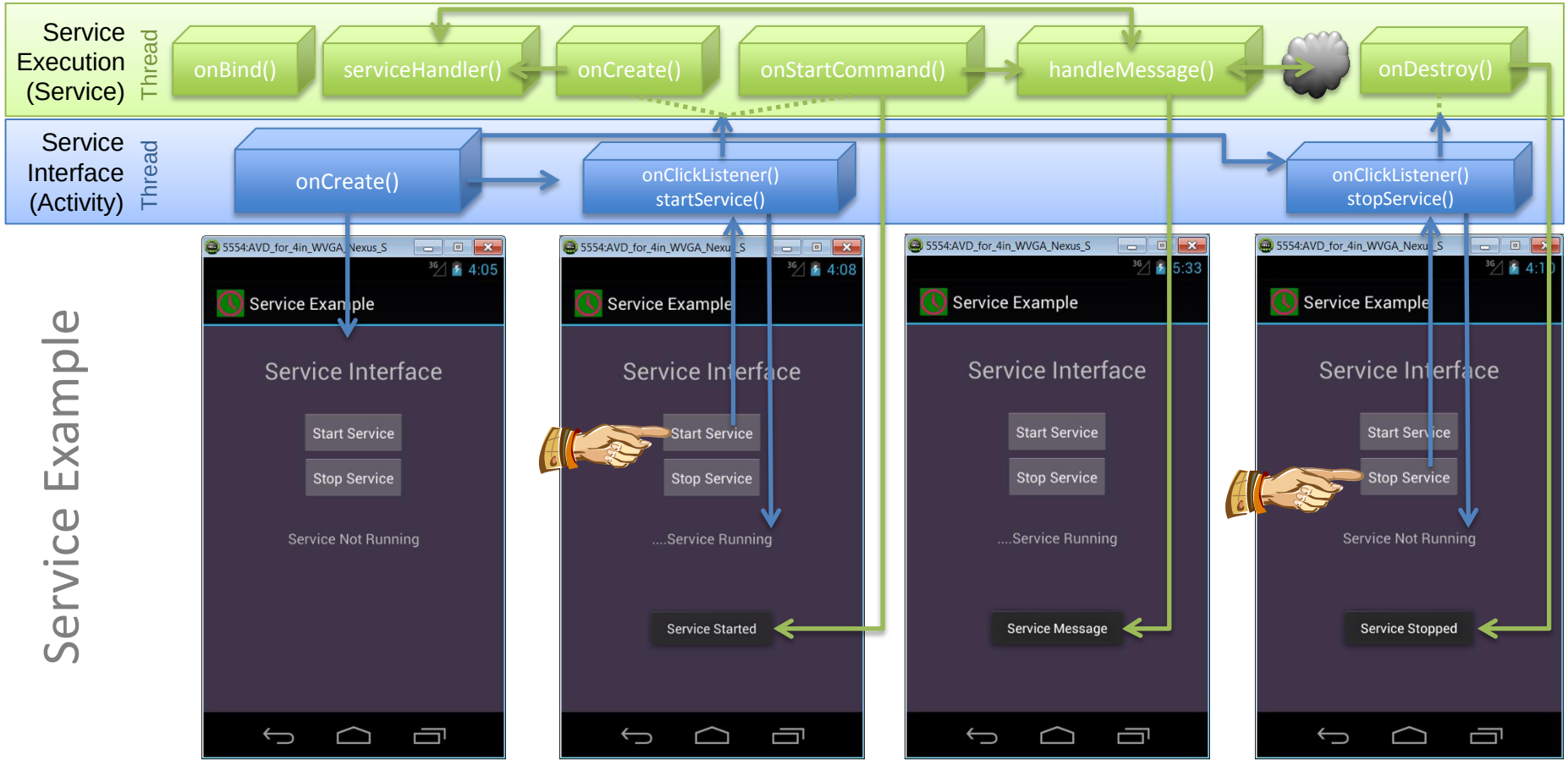
http://developer.android.com/guide/components/services.html

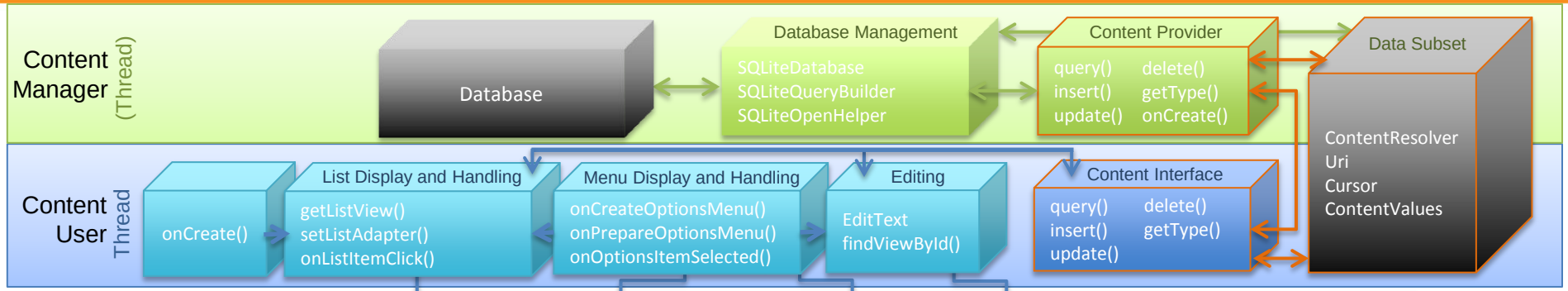




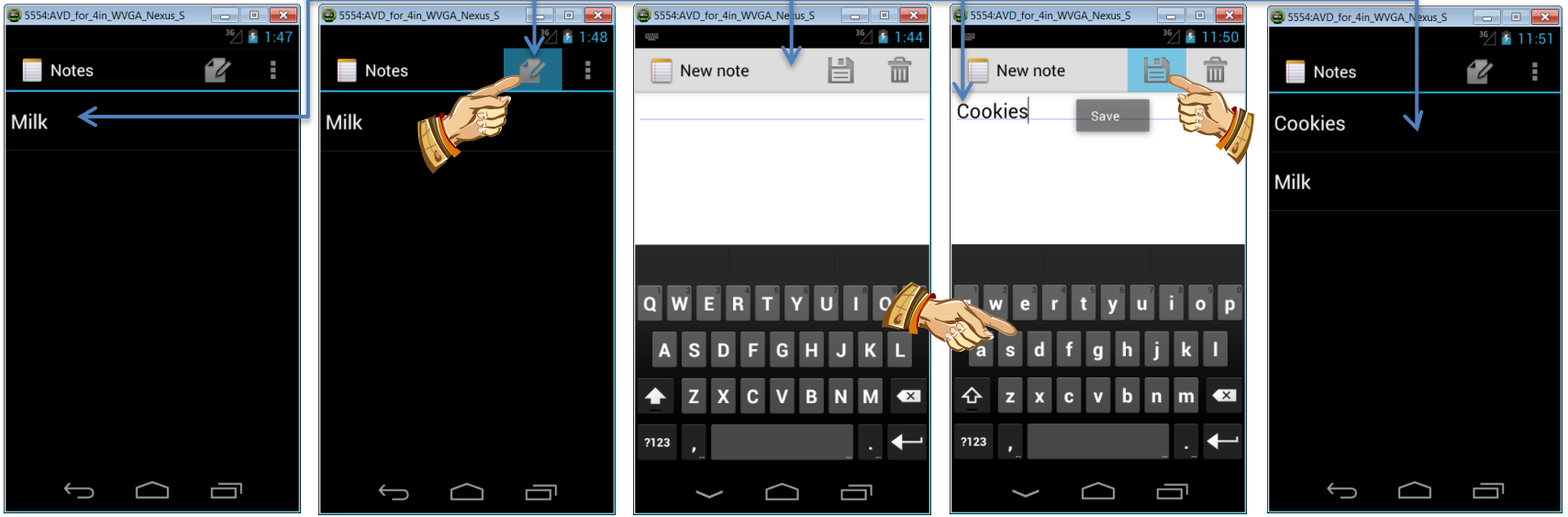
Objects Used to Implement a Service




Service Execution (Service) Thread
onBind()
serviceHandler()
onCreate()
onStartCommand()
handleMessage()
onDestroy()
Service Interface (Activity) Thread
onCreate()
onClickListener()
startService()
onClickListener()
stopService()
Service Example
Service Example
Service Interface
Start Service
Stop Service
Service Not Running
Service Example
Service Interface
Start Service
Stop Service
....Service Running
Service Started
Service Example
Service Interface
Start Service
Stop Service
....Service Running
Service Message
Service Example
Service Interface
Start Service
Stop Service
Service Not Running
Service Stopped

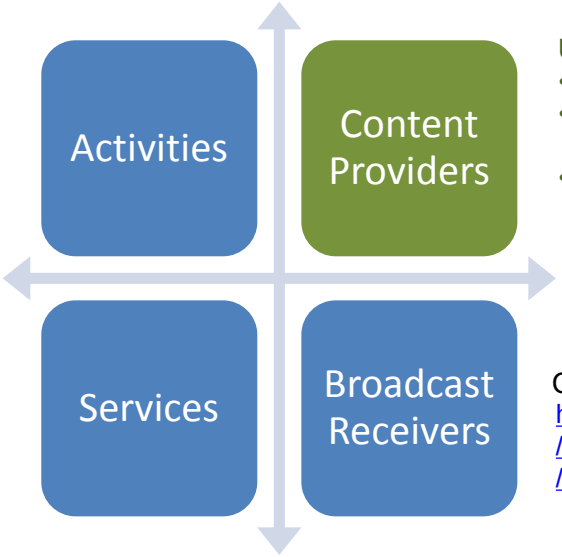


NotePad App Content Provider



Content Providers

http://developer.android.com/guide/topics/providers/content-providers.html

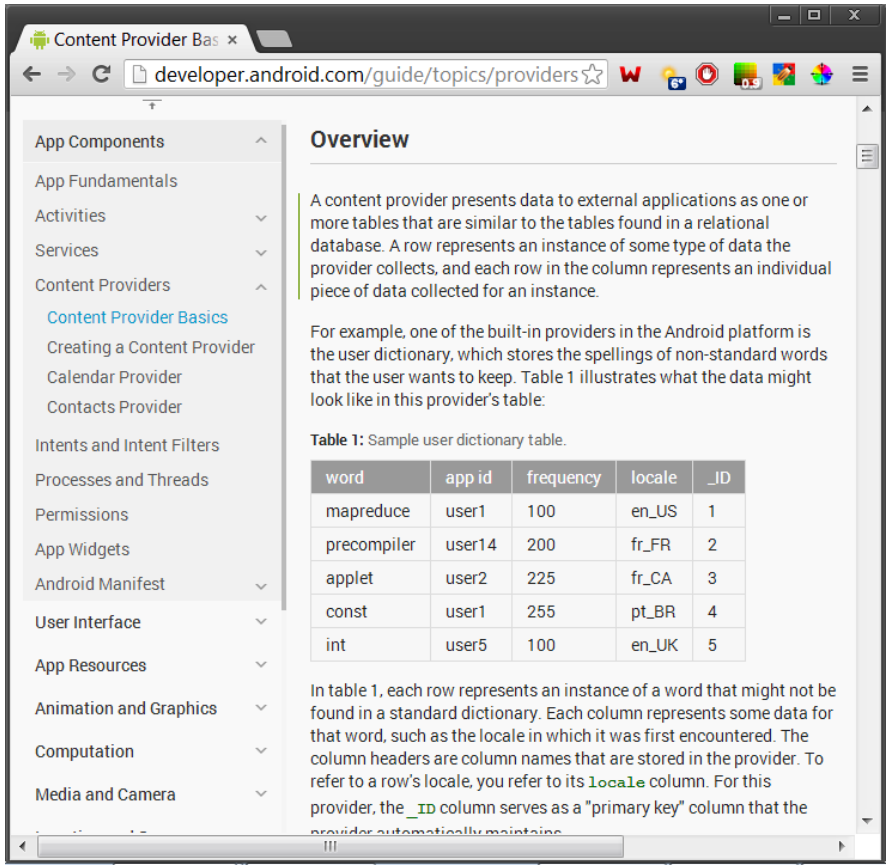


Use when your app needs to:

- Share data with other apps
- Provide custom search suggestions
- Copy and paste complex data or files from your app to other apps

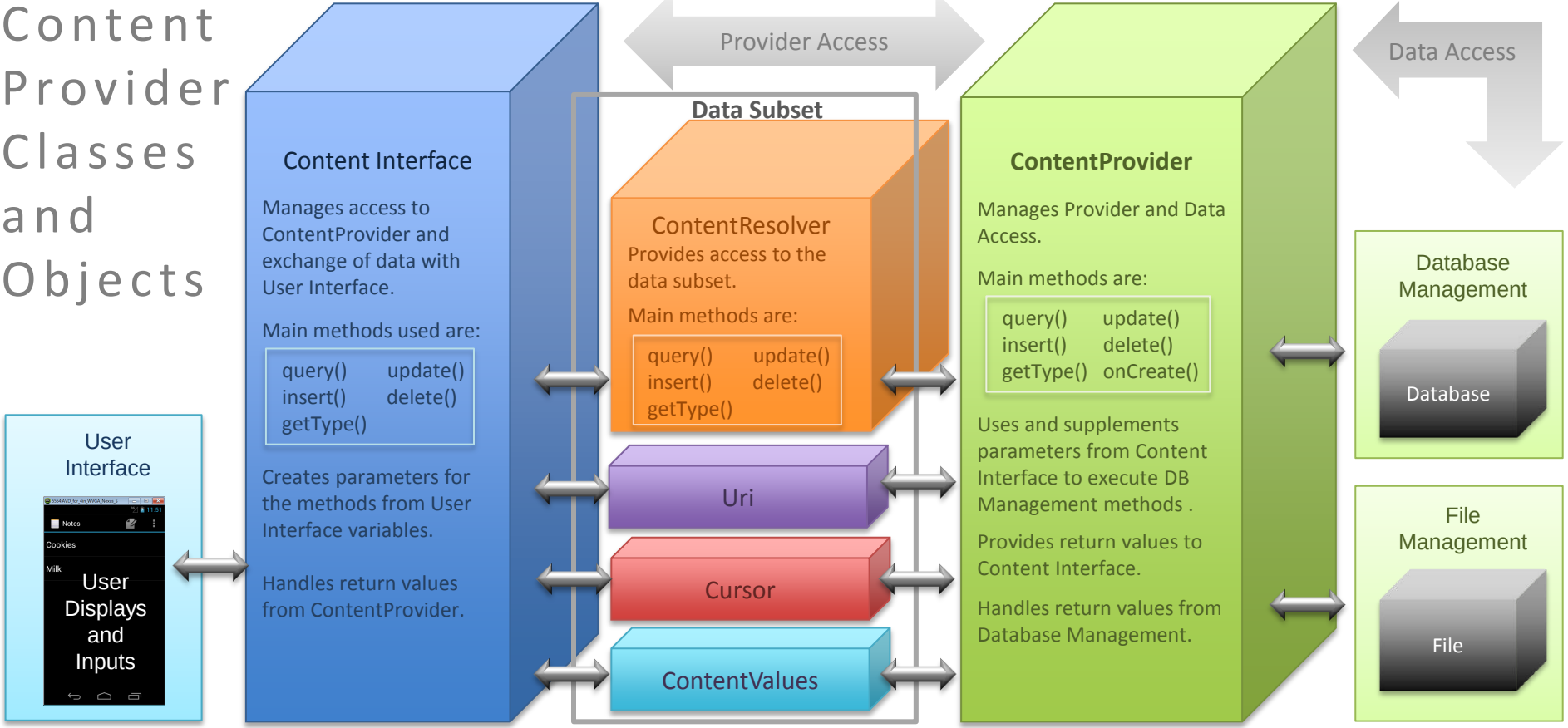
Custom Search
http://developer.android.com/guide/topics/search/adding-custom-suggestions.html

Copy and Paste
http://developer.android.com/guide/topics/text/copy-paste.html

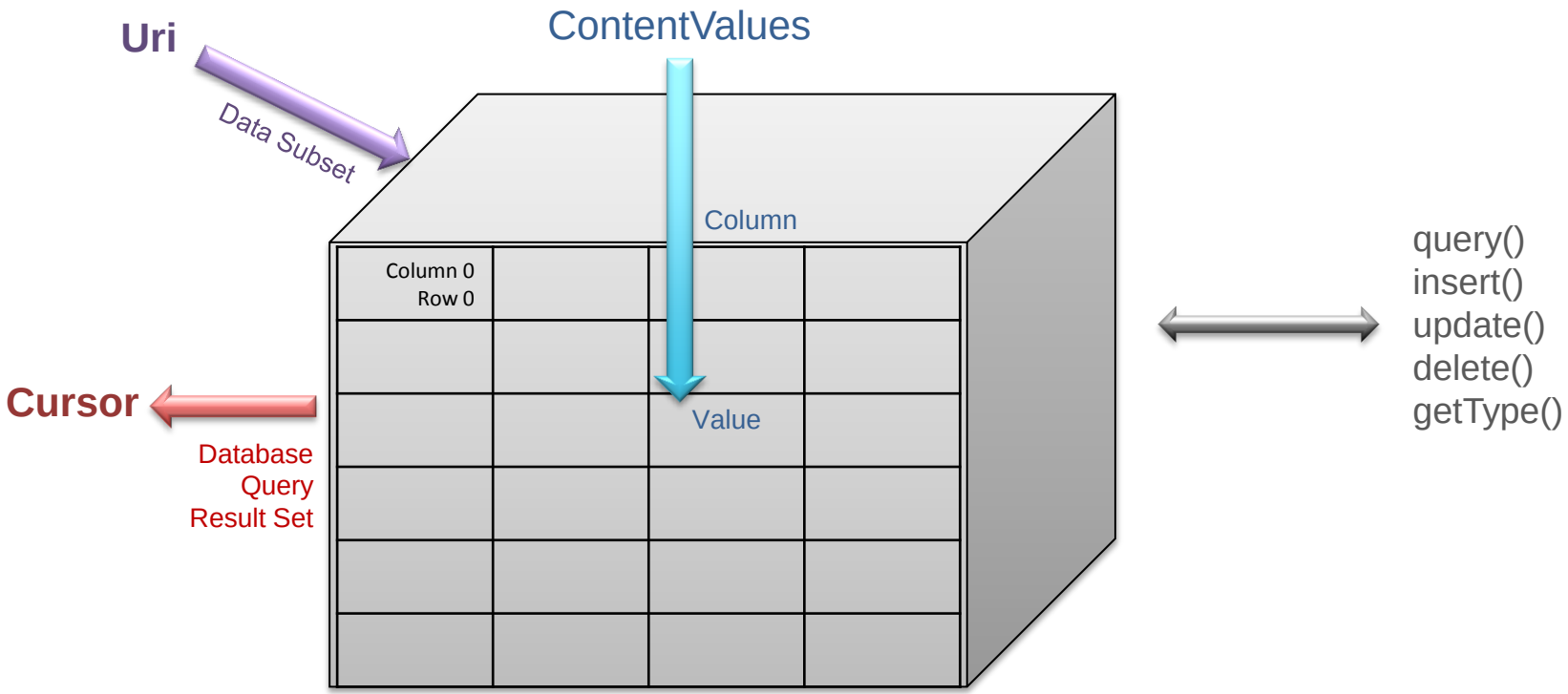


Infinite Skills

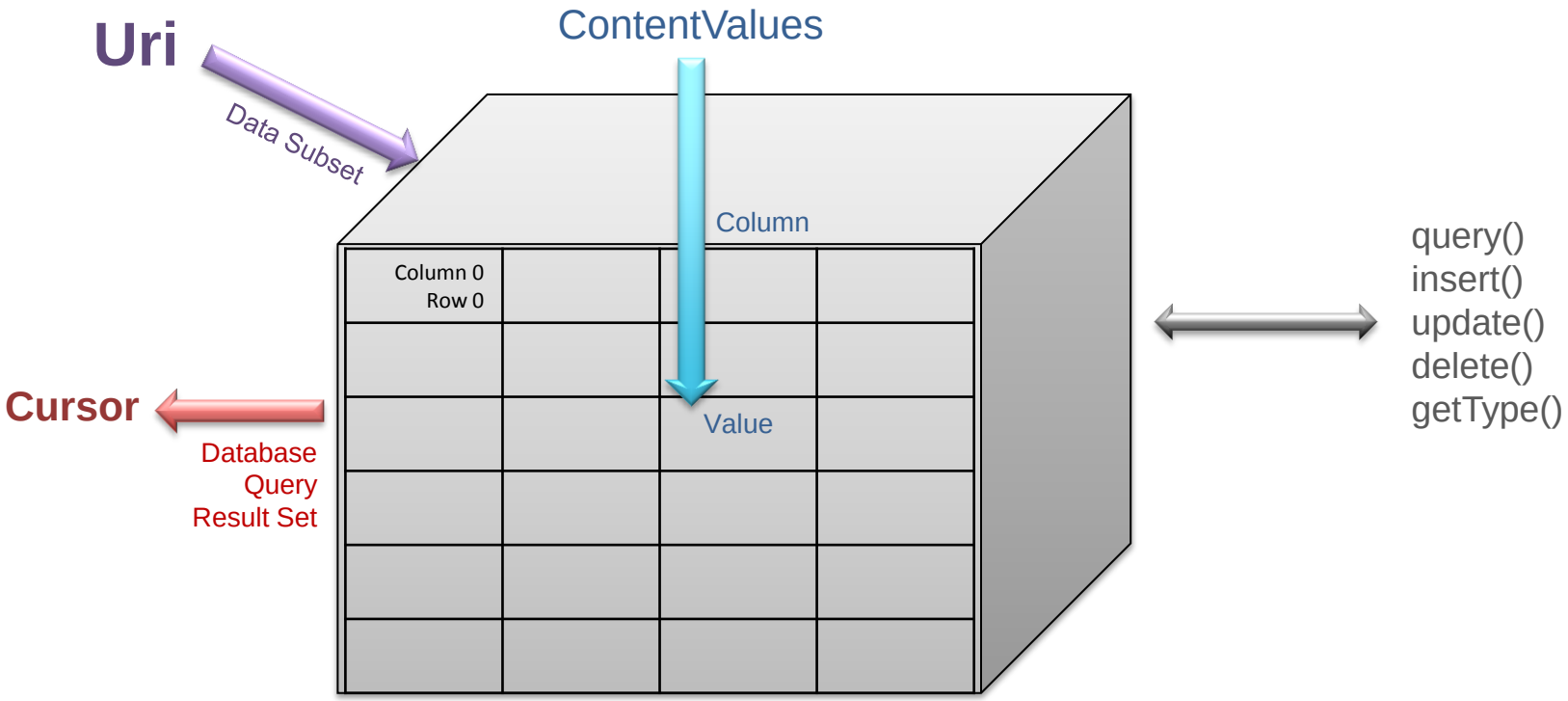
Content Provider Classes and Objects



Data Specification



URI (Uniform Resource Identifier)



URI Format

```
scheme:scheme-specific-part
```

Examples:

Web address	`http://example.org/absolute/URI/with/absolute/path/to/resource.txt`
File transfer	`ftp://example.org/resource.txt`
Resource name	`urn:isbn:0-486-27557-4`
Android content provider	`content://com.google.provider.NotePad/notes/4`

http://en.wikipedia.org/wiki/Uniform_resource_identifier

http://en.wikipedia.org/wiki/URI_scheme

Content Provider URI

content://authority/path/id

Identifies a data subset

Used for methods:
query()
insert()
update()
delete()
getType()

```
// Define URI component strings.
String SCHEME        = ContentResolver.SCHEME.CONTENT;  // Scheme (content://)
String AUTHORITY     = "com.google.provider.NotePad";   // Authority for controlling access
String PATH_NOTE_ID  = “/notes";                        // Path to identify an individual group
String SAMPLE_Id     = “/4";                            // ID to specify a single row number

// Defines a URI in the required format.
// In this example the resulting string is “content://com.google.provider.NotePad/notes/4”
Uri SAMPLE_URI = Uri.parse(                             // Use parse() method to format a Uri
                     SCHEME +                           // Scheme
                     AUTHORITY +                        // Authority
                     PATH_NOTE_ID +                     // Path – none, one or multiple paths
                     SAMPLE_ID);                        // ID - optional
```

Description

```
UriMatcher(int code)
- code to match for the root URI
```

```
addURI(String authority, String path, int code)
- authority controls access to data
- path identifies individual elements
        * used to match any character
        # used to match number
- code is the number returned on a match
```

```
match(Uri uri)      returns int code
- uri controls access to data
- code is the number returned on a match
```

Steps

Create

Setup

Switch

Example

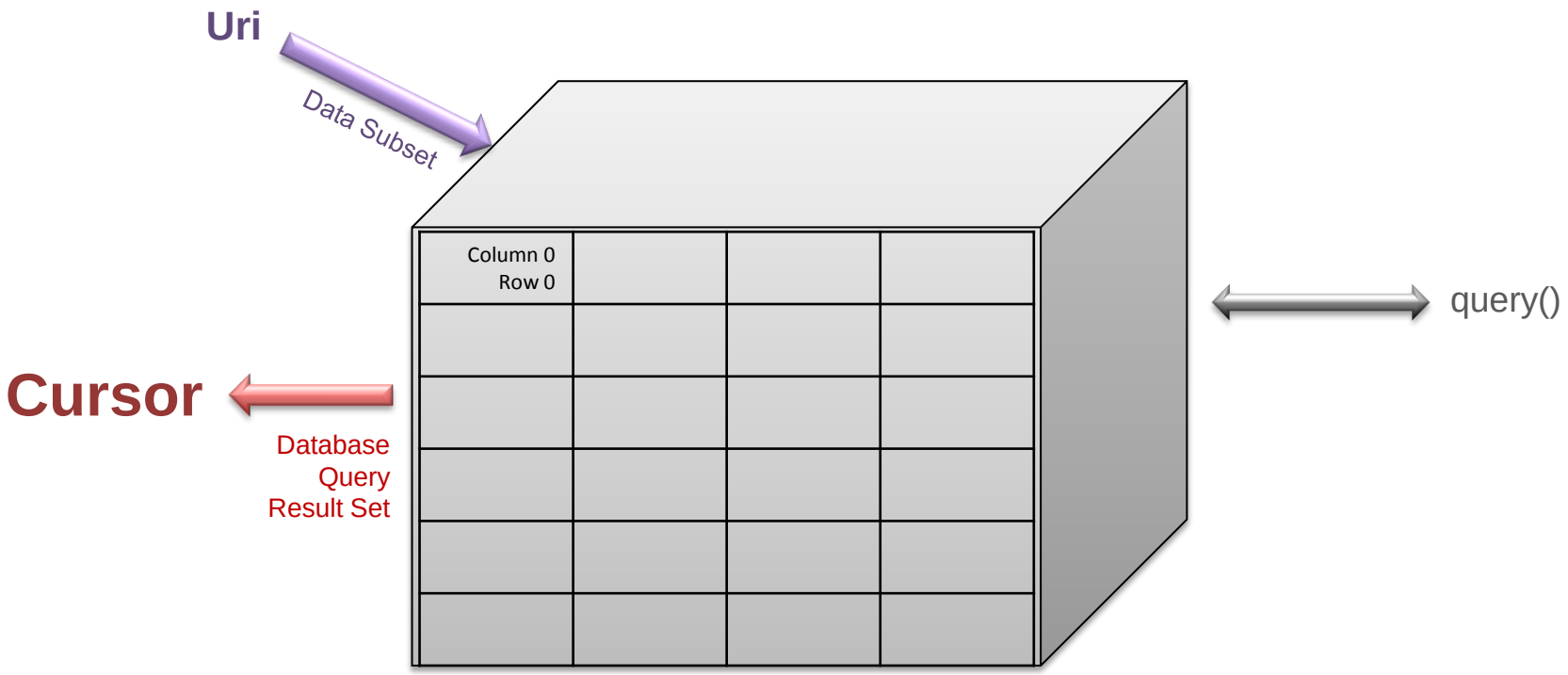
```
// Create a new instance of UriMatcher object
// NO_MATCH = -1 for no matched nodes
private static final UriMatcher sUriMatcher;
sUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
```

```
// Add entries to UriMatcher:
//                      authority        path      code
//                  -----------------  --------   -------
sUriMatcher.addURI(NotePad.AUTHORITY, "notes",   NOTES  );
sUriMatcher.addURI(NotePad.AUTHORITY, "notes/#", NOTE_ID);
```

```
// Use match() method to retrieve code for the uri
switch (sUriMatcher.match(uri)) {

    // Take action based on code
    case NOTES:
         qb.setProjectionMap(sNotesProjectionMap);
         break;
    case NOTE_ID:
         qb.setProjectionMap(sNotesProjectionMap);
         .
         .
         .
```

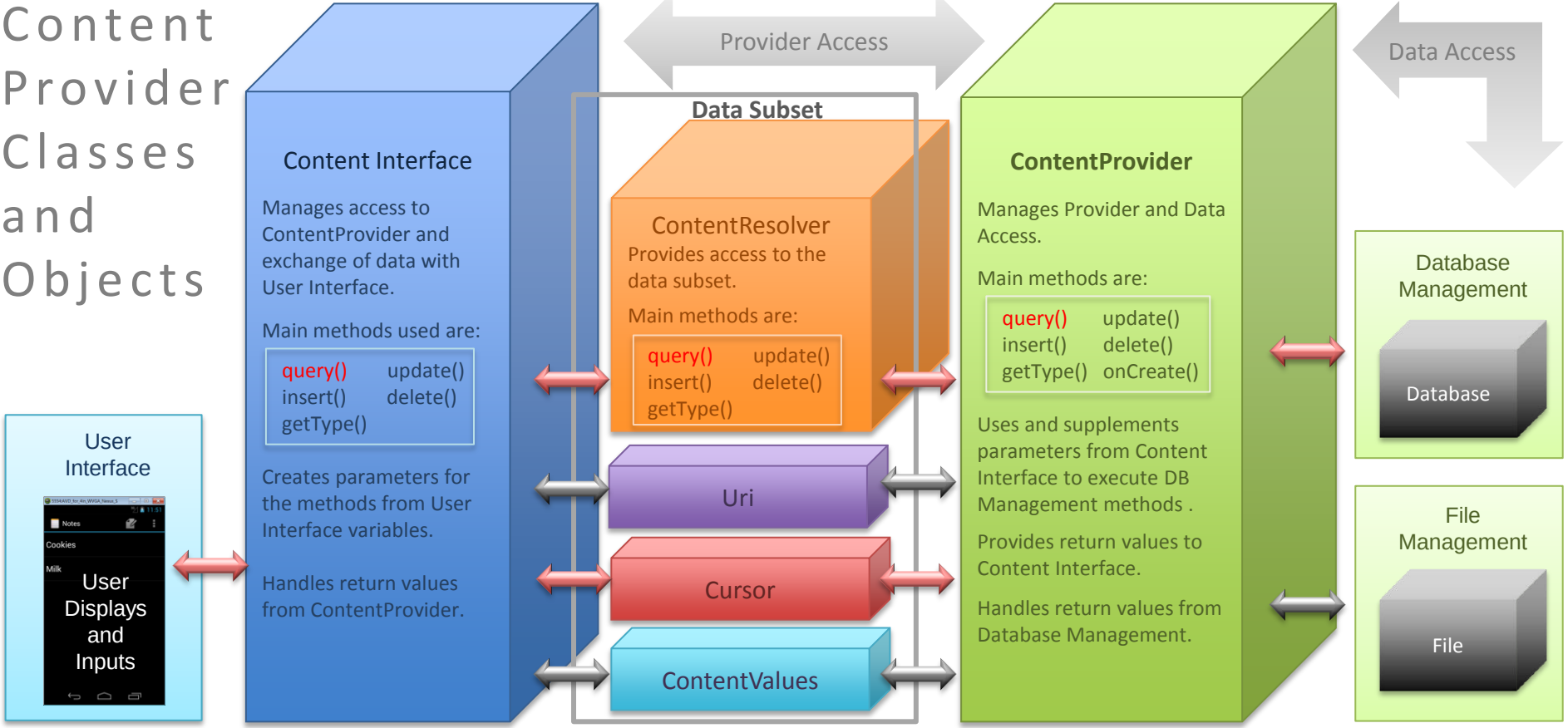
URI Matcher

Cursor Use in Content Providers



Infinite Skills

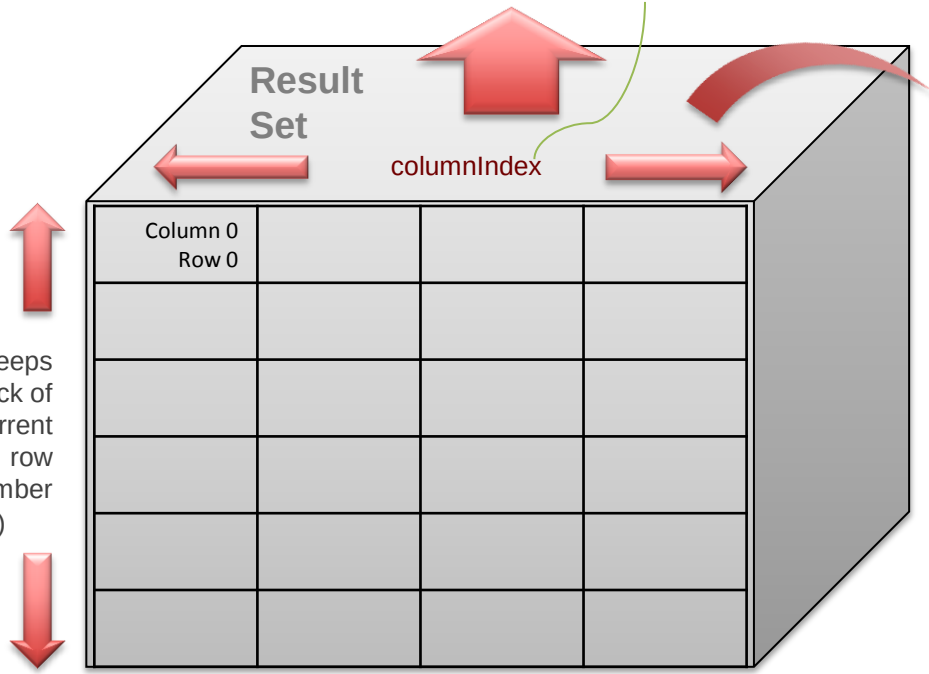
Content Provider Classes and Objects



Cursor Class and Objects

Provide read/write access to a result set returned from a database query.

getDouble(int columnIndex)
getFloat(int columnIndex)
getLong(int columnIndex)
getShort(int columnIndex)
getString(int columnIndex)
getDouble(int columnIndex)



move(int offset)
moveToFirst()
moveToLast()
moveToNext()
moveToPrevious()
moveToPosition(int position)

Keeps track of current row number

Used for methods:
query()

Cursor c *(return from a query)*

Cursor is used by other classes and methods such as cursor adapters for list displays.

getColumnIntdex(String columnName)
getColumnCount()
getColumnNames()
getCount() *(returns total number of rows)*
setNotificationUri(ContentResolver cr, Uri uri)

Infinite Skills

Content Provider Methods

User Interface

Content Interface

Manages access to ContentProvider and exchange of data with User Interface.

Main methods used are:

query() update()
insert() delete()
getType()

Creates parameters for the methods from User Interface variables.

Handles return values from ContentProvider.

Provider Access

Data Subset

ContentResolver

Provides access to the data subset.

Main methods are:

query() update()
insert() delete()
getType()

Uri

Cursor

ContentValues

ContentProvider

Manages Provider and Data Access.

Main methods are:

query() update()
insert() delete()
getType() onCreate()

Uses and supplements parameters from Content Interface to execute DB Management methods .

Provides return values to Content Interface.

Handles return values from Database Management.

Data Access

Database Management

Database

File Management

File

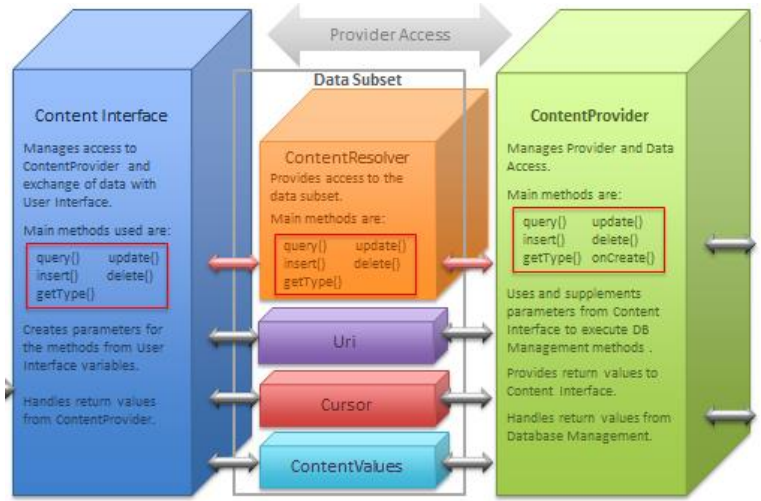
onCreate() Method

Content Interface uses a ContentResolver object to access some data for the first time

ContentResolver tries to access ContentProvider for first time

ContentProvider object created

ContentProvider onCreate method invoked



ContentProvider onCreate() method

- Initialize variables
- Open files/database
- Keep minimal

Key Methods

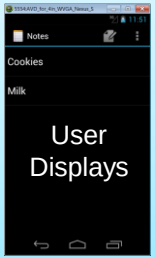
Parameter Use

Applicable

Can be null

Parameters	query()	insert()	update()	delete()	getType()
Uri	✓	✓	✓	✓	✓
Projection	✓ ○				
Selection Clause	✓ ○		✓ ○	✓ ○	
Selection Arguments	✓ ○		✓ ○	✓ ○	
Sort Order	✓ ○				
ContentValues		✓ ○	✓ ○		
Cancellation Signal (Introduced in API 16)	✓ ○				
	Cursor ○	Uri ○	int number of rows affected	int number of rows affected	String of MIME type ○

User Interface

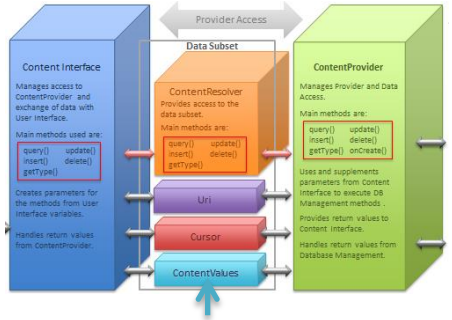


Database Management

Return Use

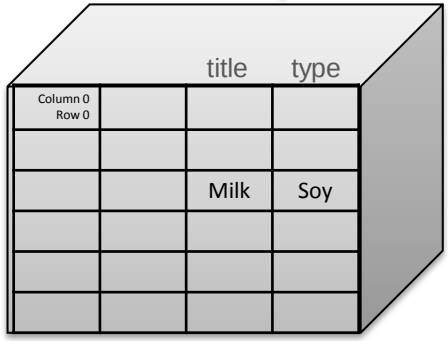
ContentValues

Identifies column/value pairs



Used for methods:
insert()
update()

```
// Defines an object to contain new values to insert.
ContentValues mValues = new ContentValues();


// Sets values in the ContentValues object
// using the put() method of the ContentValues class
// with Column Name and Value string as parameters.
//
//                  Key (Column Name)           Value
//          -------------------------------  -------
mValues.put( NotePad.Notes.COLUMN_NAME_TITLE, "Milk“ );
mValues.put( NotePad.Notes.COLUMN_NAME_TYPE,  "Soy“  );
```



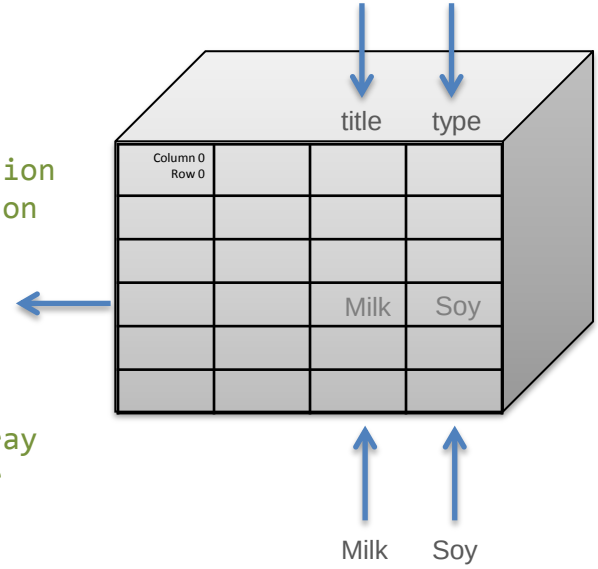

```
// Sets title Milk
// Sets type Soy
```

Selection Clause and Arguments

Identifies criteria for access operations

Used for methods:
query()
update()
delete()

```
// Defines the criteria for which rows to return,
// formatted as an SQL WHERE clause (excluding the WHERE itself).
// Passing null will return all rows for the given table.
// Question marks will be replaced with corresponding entry in the
// selection arguments parameter below.

String mSelectionClause =                          // Define a string
    "NotePad.Notes.COLUMN_NAME_TITLE=?   AND       // Title column selection
     NotePad.Notes.COLUMN_NAME_TYPE=?";            // Type column selection


// Defines values input by user to be paired
// with selection clause fields above.

String[] mSelectionArguments = new String[] {      // Create a string array
    “Milk”,                                        // Title input by user
    “Soy”};                                        // Type input by user
```

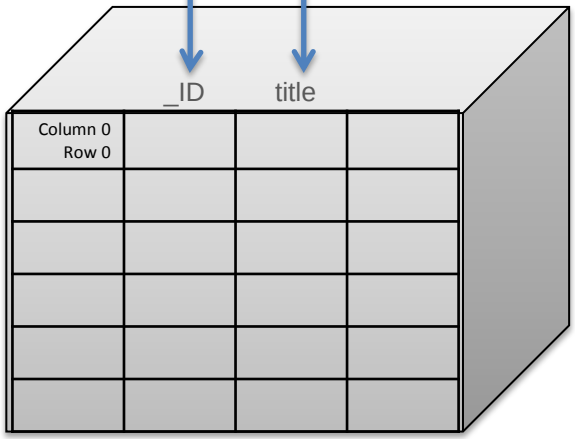


Projection

Identifies columns for data retrieval

Used for methods:
query()

```
// Defines the columns to be returned for each row returned from a query.

String[] mProjection = new String[] {                    // Create a string array
                       NotePad.Notes._ID,                // Column = "_ID"
                       NotePad.Notes.COLUMN_NAME_TITLE   }; // Column = "title"
```

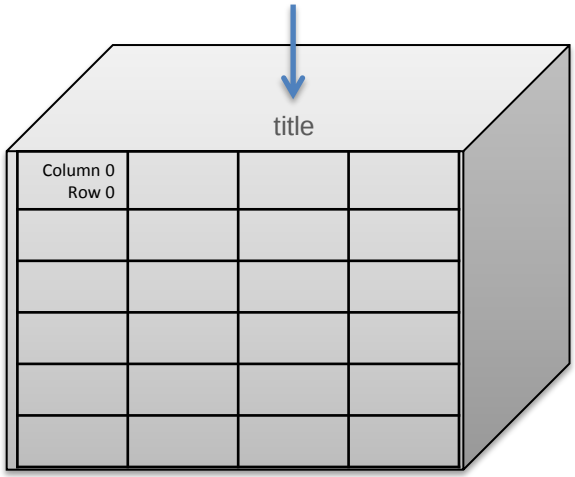


Sort Order

Identifies column for sort

Used for methods:
query()

```
// Specifies the column to be used to sort query results.

String mSortOrder = "NotePad.Notes.COLUMN_NAME_TITLE"        // Title column sort
```



Cancellation Signal

Cancels operation in progress

Used for methods:
query()

```
// Create a CancellationSignal object.
// Introduced in API 16.
// If this parameter is used, you must also implement
// a query() method without it to support previous versions.
CancellationSignal mCancellationSignal = new CancellationSignal()   // Instantiate CancellationSignal

.
.
.

// Cancel the query operation.
mCancellationSignal.cancel();
```

MIME Type

Provides information about the file type

Used for methods: getType()

String in the format: `type/subtype/optional-parameters`

Standard

```
Types:                  application, audio, image, message, text, video, . . .
Text subtype examples:  calendar, plain, html, xml
Example:                "text/html"
```

http://en.wikipedia.org/wiki/Internet_media_type

http://www.iana.org/assignments/media-types

Android Vendor-Specific Custom

```
Type for multiple rows:  vnd.android.cursor.dir
Type for Single row:     vnd.android.cursor.item
Subtype example:         vnd.google.note
Example:                 "vnd.android.cursor.dir/vnd.google.note"
```

http://developer.android.com/guide/topics/providers/content-provider-basics.html#MIMETypeReference

Infinite Skills

Content Provider Methods

User Interface

Content Interface

Manages access to ContentProvider and exchange of data with User Interface.

Main methods used are:

query() update()
insert() delete()
getType()

Creates parameters for the methods from User Interface variables.

Handles return values from ContentProvider.

Provider Access

Data Subset

ContentResolver

Provides access to the data subset.

Main methods are:

query() update()
insert() delete()
getType()

Uri

Cursor

ContentValues

ContentProvider

Manages Provider and Data Access.

Main methods are:

query() update()
insert() delete()
getType() onCreate()

Uses and supplements parameters from Content Interface to execute DB Management methods .

Provides return values to Content Interface.

Handles return values from Database Management.

Data Access

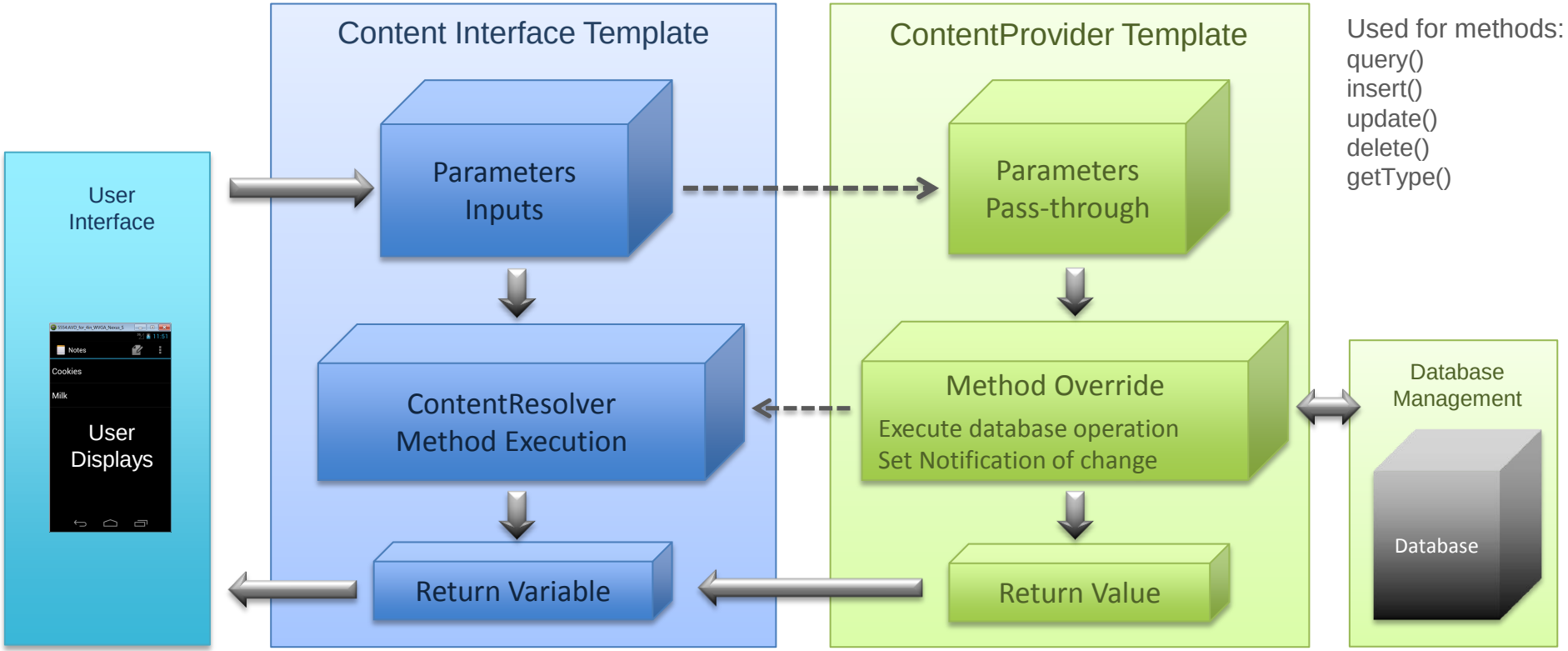
Database Management

Database

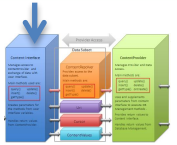
File Management

File

Content Interface and Provider Method Templates



Content Interface Method Template



Use appropriate:

```
// Define parameters needed for access. (see parameter specific graphics)
. . .

// Define a variable to hold return depending on the type of access.  Use one of:
Cursor mReturn = new Cursor();          // Define Cursor for query operation.
Uri    mReturn = new Uri();             // Define Uri for insert operation.
int    mReturn = 0;                     // Define int for update or delete operation.
String mReturn = "";                    // Define MIME type String for getType operation.

// Execute operation and assign return result.
// Replace access with one of query, insert, update, delete, getType method names.
mReturn = getContentResolver().access(  // Get Content Resolver and perform operation.
            mUri,                       // Parameter values - which are set by
            mProjection,                // code executed prior to this statement.
            mSelectionClause,           // Use parameters depending on access type.
            mSelectionArguments,        // Some parameters can be null.
            mSortOrder,
            mValues,
            mCancellationSignal);
```

parameters

return

access

parameters

Key Methods

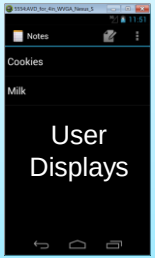
Parameter Use

Applicable

Can be null

Parameters	query()	insert()	update()	delete()	getType()
Uri	✓	✓	✓	✓	✓
Projection	✓ ○				
Selection Clause	✓ ○		✓ ○	✓ ○	
Selection Arguments	✓ ○		✓ ○	✓ ○	
Sort Order	✓ ○				
ContentValues		✓ ○	✓ ○		
Cancellation Signal (Introduced in API 16)	✓ ○				
	Cursor ○	Uri ○	int number of rows affected	int number of rows affected	String of MIME type ○

User Interface



Database Management

Return Use

Key Methods

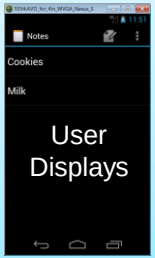
Parameter Use

Applicable

Can be null

Parameters	query()	insert()	update()	delete()	getType()
Uri	✓	✓	✓	✓	✓
Projection	✓ ○				
Selection Clause	✓ ○		✓ ○	✓ ○	
Selection Arguments	✓ ○		✓ ○	✓ ○	
Sort Order	✓ ○				
ContentValues		✓ ○	✓ ○		
Cancellation Signal (Introduced in API 16)	✓ ○				
	Cursor ○	Uri ○	int number of rows affected	int number of rows affected	String of MIME type ○

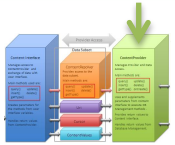
User Interface



Database Management

Return Use

ContentProvider Method Template



Use appropriate:

Uri matching

```
// Use UriMatcher to take actions based on the Uri.  (see Uri Matcher graphic)
. . .

// Execute operation and return result.
// Replace return with one of Cursor, Uri, int, String return types.
// Replace access with one of query, insert, update, delete, getType method names.

@Override                                       // Override to sync with Content Interface
public return access(                           // Declare method with result
          Uri                 uri,              // These are the parameters passed through
          String[]            projection,       // from the Content Interface use
          String              selectionClause,  // of the access method. Use parameters
          String[]            selectionArgs,    // depending on access type.
          String              sortOrder,        // Parameters other than URI can be null.
          ContentValues       values,
          CancellationSignal  cancellationSignal)
          {
               // Body of access method (see next graphic)
          }
```

return & access

parameters

ContentProvider Method Template

```
// Body of access method – code executed when the method is invoked.
{
   // Define a variable to hold result depending on the type of access. Use one of:
   Cursor mReturn;       // Define Cursor for a query.
   Uri    mReturn;       // Define Uri for an insert.
   int    mReturn;       // Define integer for row count for an update or delete.
   String mReturn;       // Define String for MIME type for a getType.

   // Perform database access operation based on the method parameters
   // which were given values by the Content Interface.
   . . .                                          // Prepare for access
   mReturn = database.access(parameters . . . )   // Access SQLite database & get return

   // Perform Notification to set callbacks for access type.
   mReturn.setNotificationUri(getContext().getContentResolver(), uri); // query
   getContext().getContentResolver().notifyChange(uri, null);  // update, delete, insert

   // Return result
   return mReturn        // Return result for access.
}
```

Use appropriate:

return

access

notification

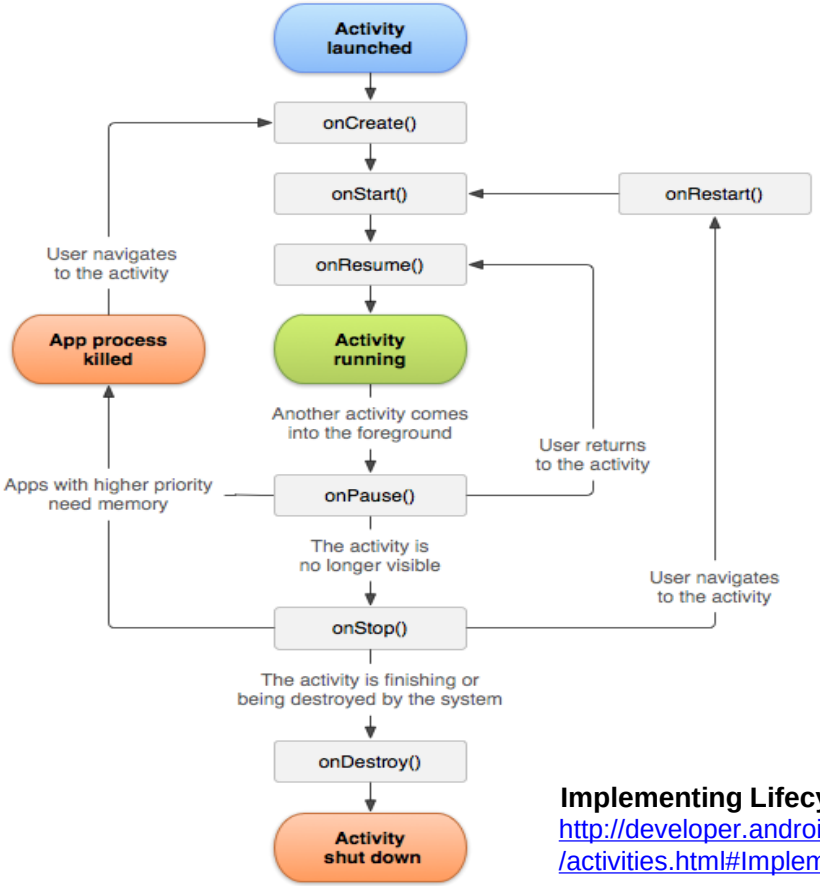
Activity Lifecycle Methods and Typical Uses

onCreate()
- *Must* be implemented
- Call setContentView()
- Initialize display layout
- Initialize variables
- Start background thread

onResume()
- Register Broadcast Receivers
- Restore variables
- Restart animations
- Lightweight code

onPause()
- Commit changes
- Save variables
- Save preferences
- Stop animations
- Unregister Broadcast Receivers
- Lightweight code

onDestroy()
- Close databases
- Release remaining resources
- Stop background thread



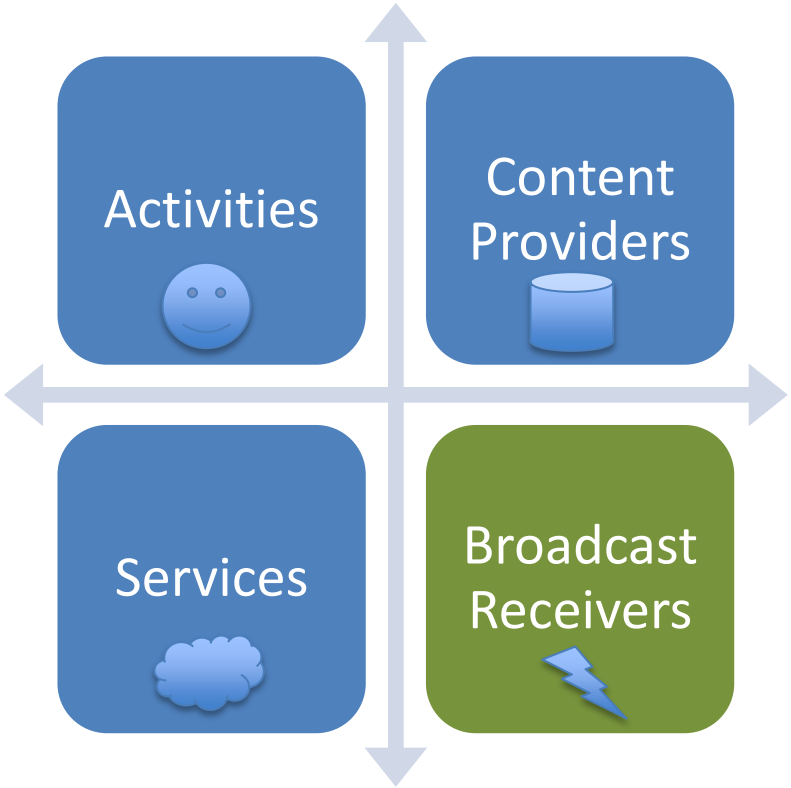
onStart()
- Register Broadcast Receiver
- Reinitialize states

onRestart()
- Open server connections
- Allocate resources

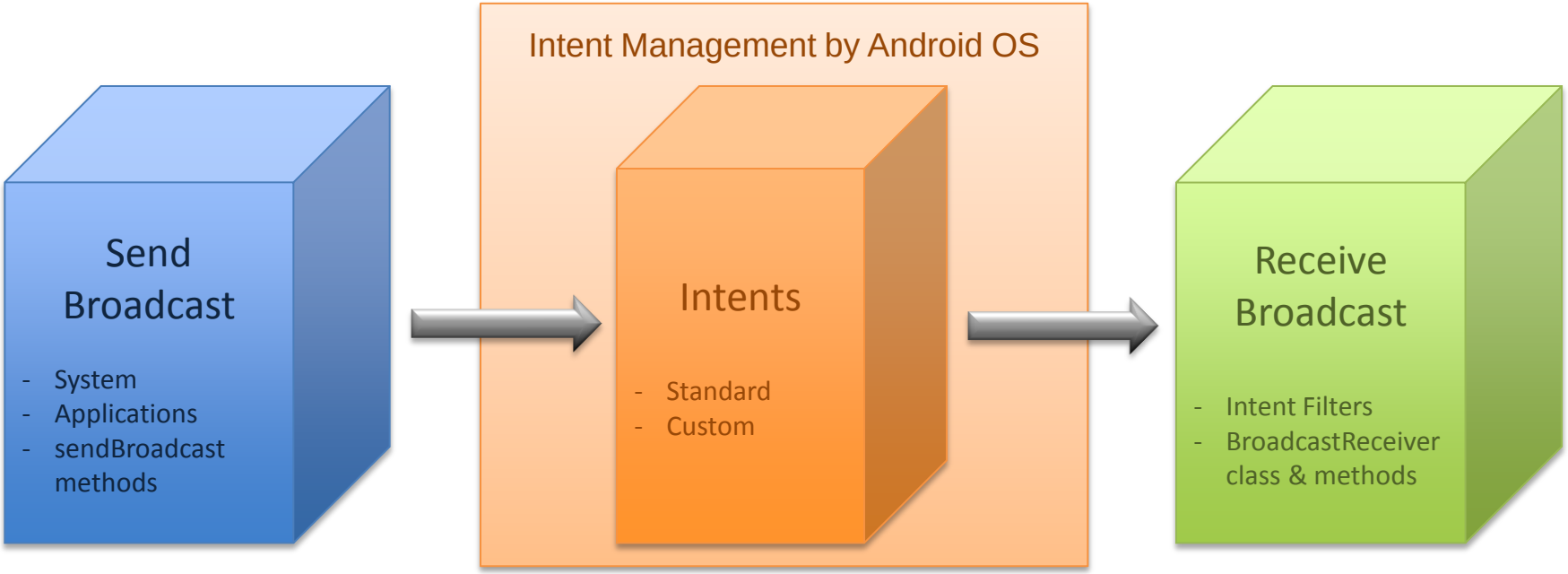
onStop()
- Close server connections
- Free resources
- Unregister Broadcast Receiver

Implementing Lifecycle Callbacks
http://developer.android.com/guide/components/activities.html#ImplementingLifecycleCallbacks

Broadcast Receivers



Sending and Receiving Broadcasts



http://developer.android.com/reference/android/content/Context.html#sendBroadcast(android.content.Intent)

http://developer.android.com/reference/android/content/Intent.html

http://developer.android.com/reference/android/content/BroadcastReceiver.html

Sending a Broadcast

Variations on sendBroadcast()

AsUser: Allows you to specify the user the broadcast will be sent to.

Ordered: Intent is delivered to BroadcastReceivers one at a time.

Sticky: Intent stays around after the broadcast is complete

receiverPermission: Parameter that allows an optional required permission to be enforced.

Simple sendBroadcast() example for a custom intent

```
Intent intent = new Intent();                          // Instantiate Intent object
intent.setAction("com.donkcowan.specialaction");       // Set action using a custom intent
sendBroadcast(intent);                                 // Send broadcast with intent
```

Key Objects Used with Content Providers