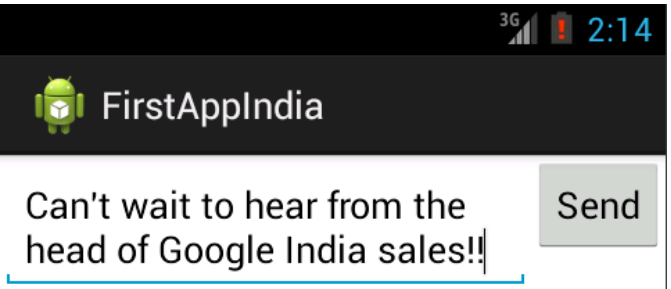
Lecture 7: User Interface (Navigation) + Version Control

Today's agenda

- Recap
- Navigation
- Version Control

Tutorial Recap

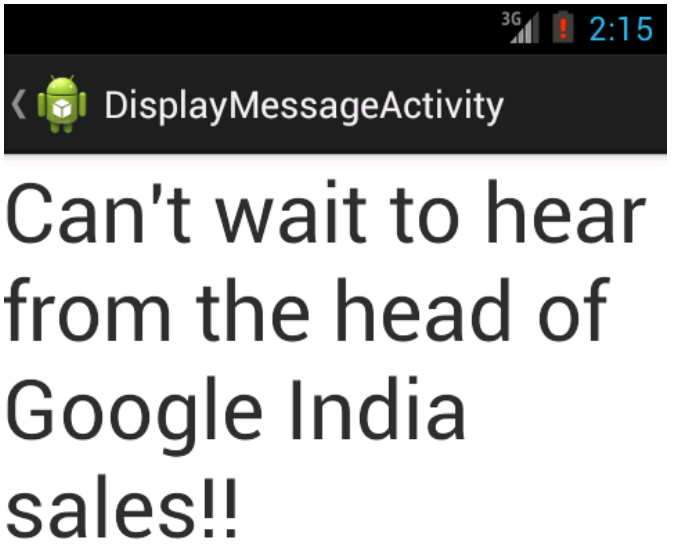
Activity 1



Button

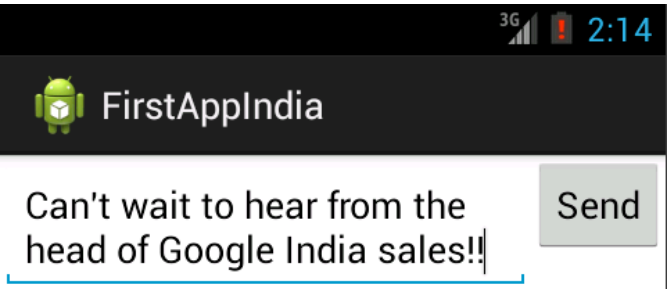
Text field

Activity 2



Tutorial Recap

Activity 1



Text field

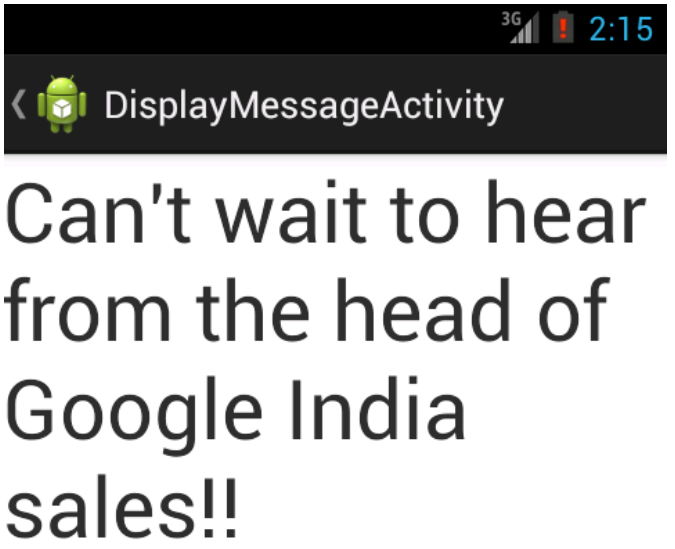
Button

Click Event

Intent

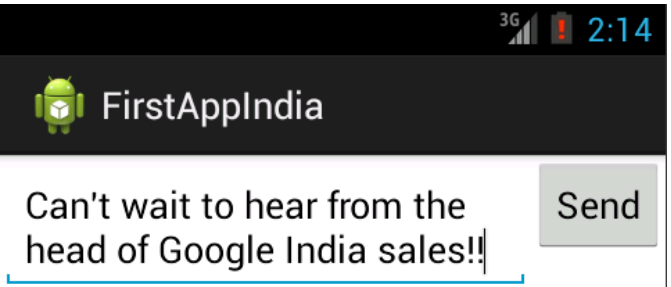
(Contains message object)

Activity 2



Tutorial Recap: Activities

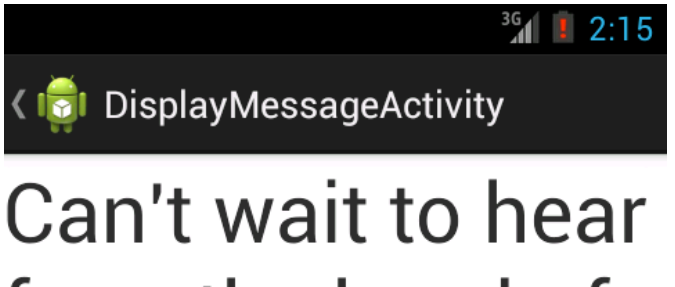
Activity 1



Activity 1.xml

Activity 1.java

Activity 2

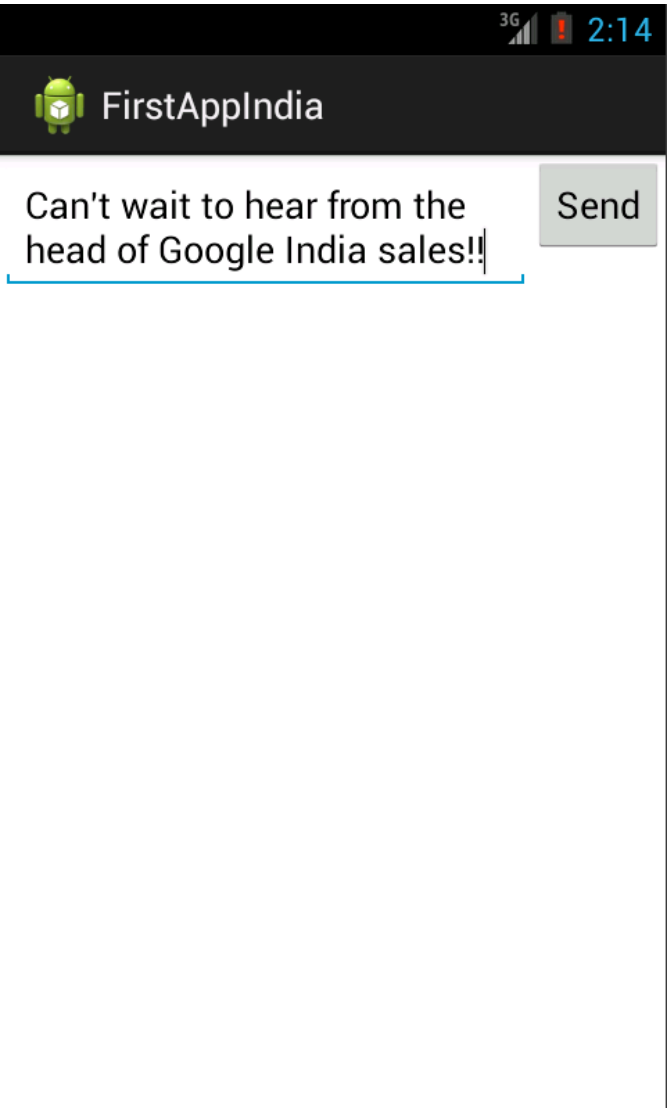


Activity 2.xml

Activity 2.java

Tutorial Recap: Buttons

Activity 1



Button

Activity 1.xml:

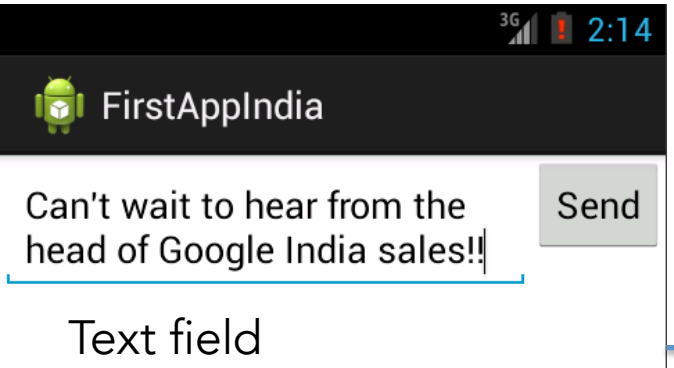
- add button, give "directions" on what to do with a click event (name of a click event method that determines behavior)

Activity 1.java:

- Contains our behavior function that came from the xml file

Tutorial Recap: Intent

Activity 1



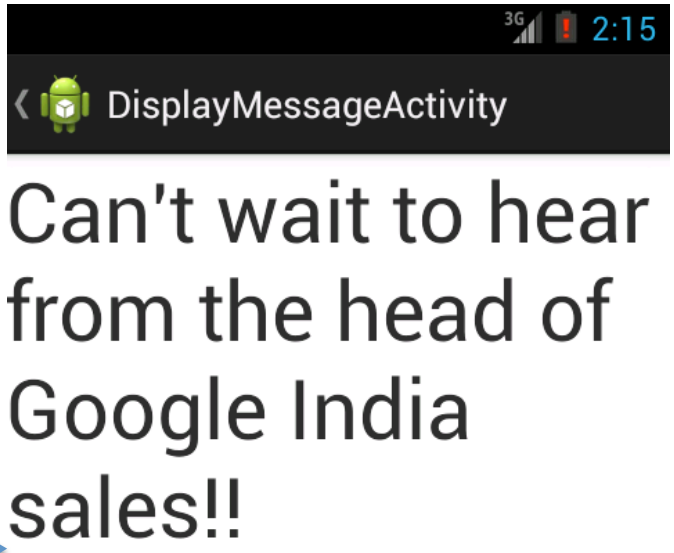
Text field

Button Click Event

Intent

(Contains message object)

Activity 2



Click Event Method (in Activity 1.java)

- Create the Intent Object
- Add the message to it
- Start the activity in the intent object (activity 2)

onCreate Method (in Activity 2.java)

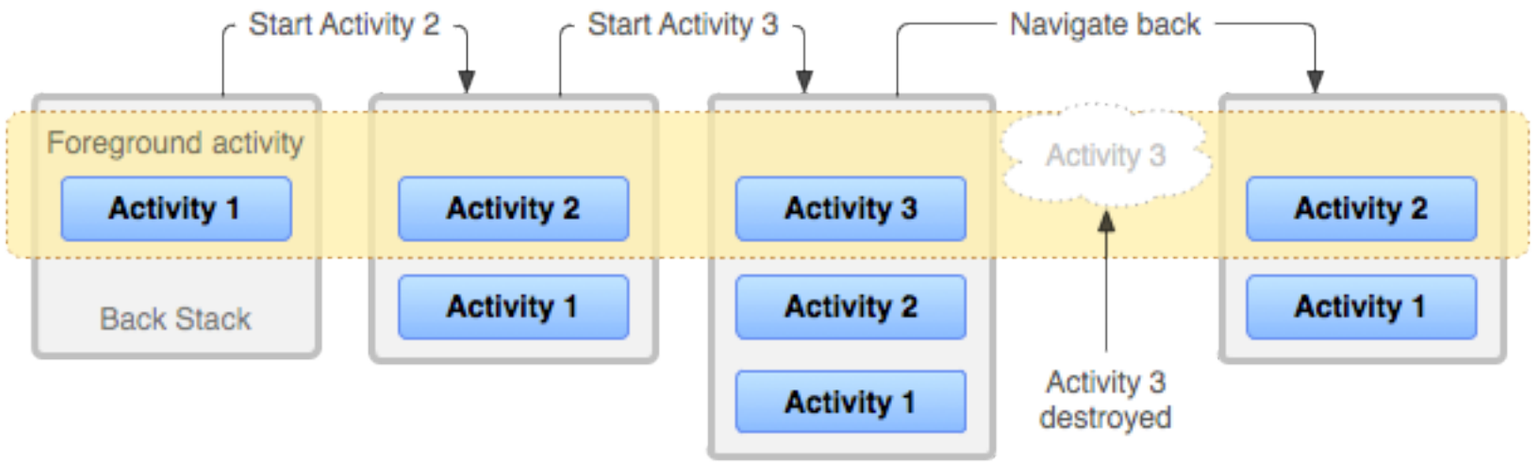
- get the intent object
- get the message from the intent object

)

Overview

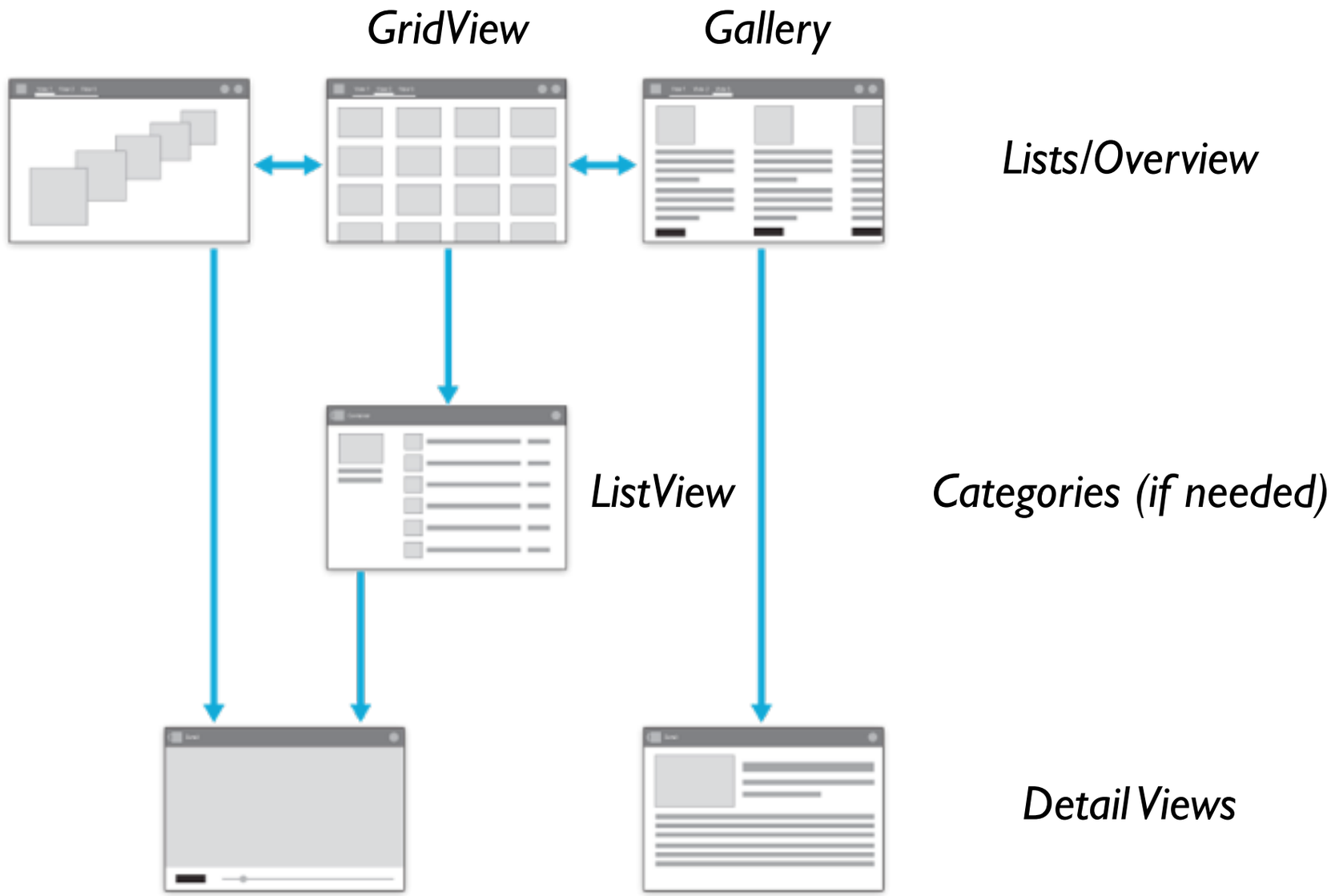
- Models of app navigation
  - The activity stack
  - ListView/GridView
  - Menus
  - Dialogs and Notifications
- Using navigation in your app

The Activity Stack



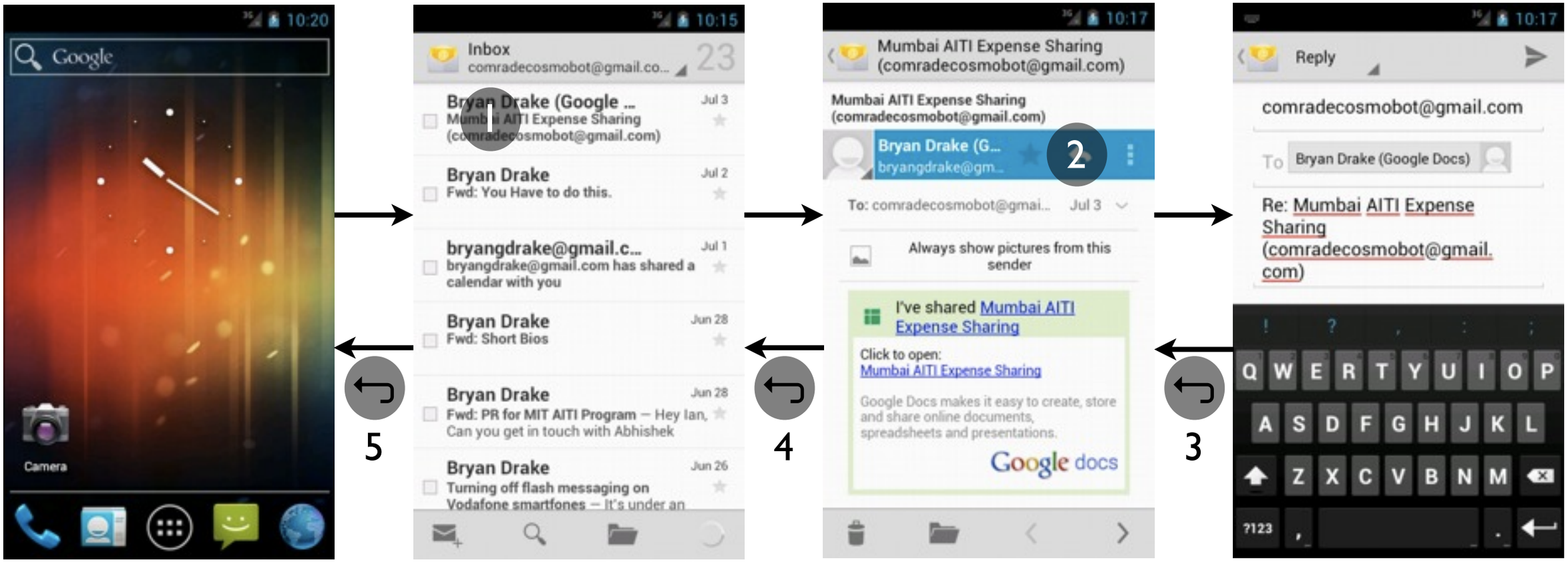
*The activity stack is usually used to separate **modal** actions (i.e. different modes: "reading mail" → "writing reply")*

How to use the Activity Stack



Generally: Navigate from lists/overview to details!

How to use the Activity Stack



Home *List* *Detail (View)* *Detail (Reply)*

Also useful for temporary states (e.g. replying to e-mail)

Using the Activity Stack

NavigationExamples

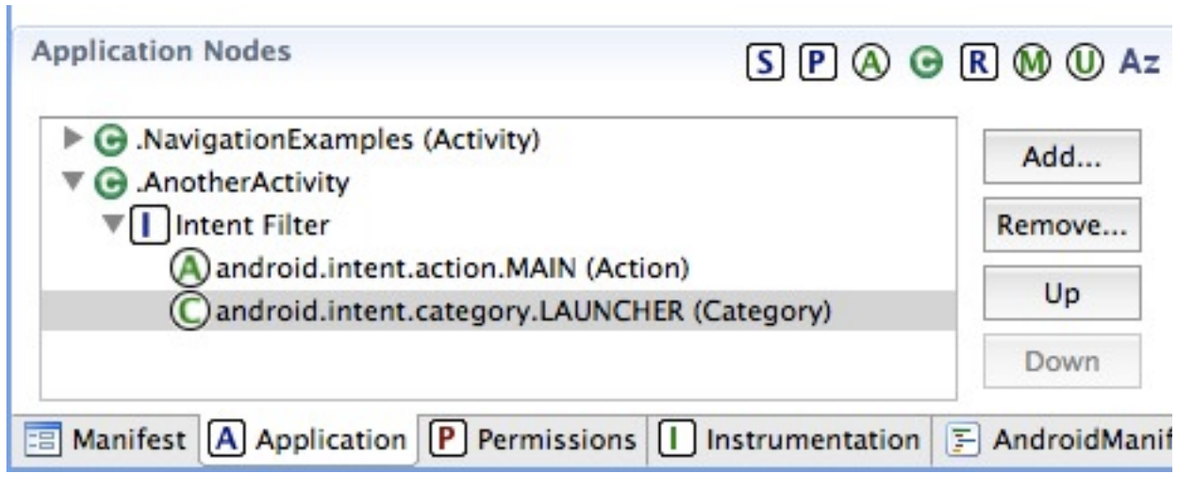
AnotherActivity

The Code:
onClickMe()

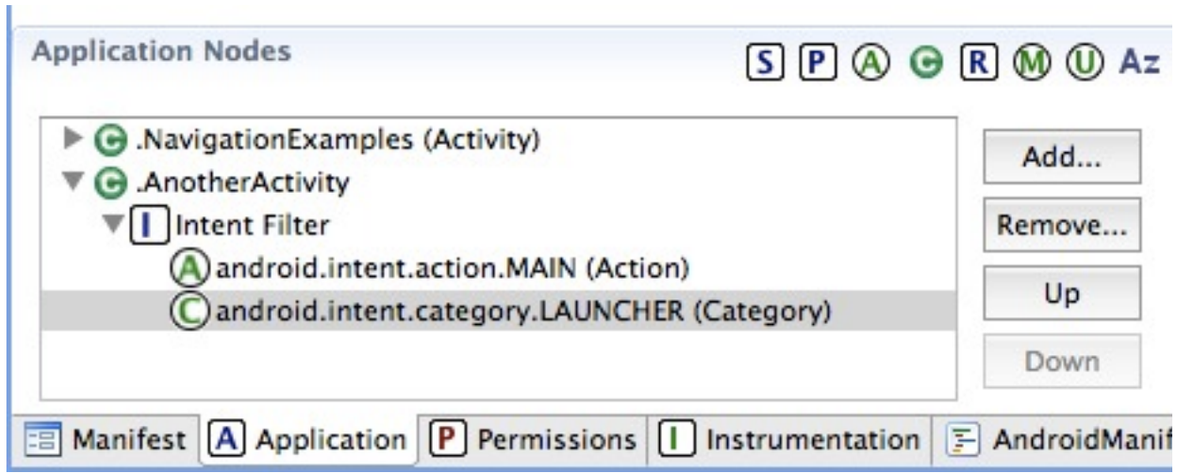
```
public class NavigationExamples extends Activity {
    public void onClickMe(View button) {
        /* Create an Intent to start AnotherActivity */
        Intent startAnotherActivity =
          new Intent(NavigationExamples.this,
                     AnotherActivity.class);

        /* Start the activity. */
        startActivity(startAnotherActivity);
    }
}
```

Using the Activity Stack: Demo

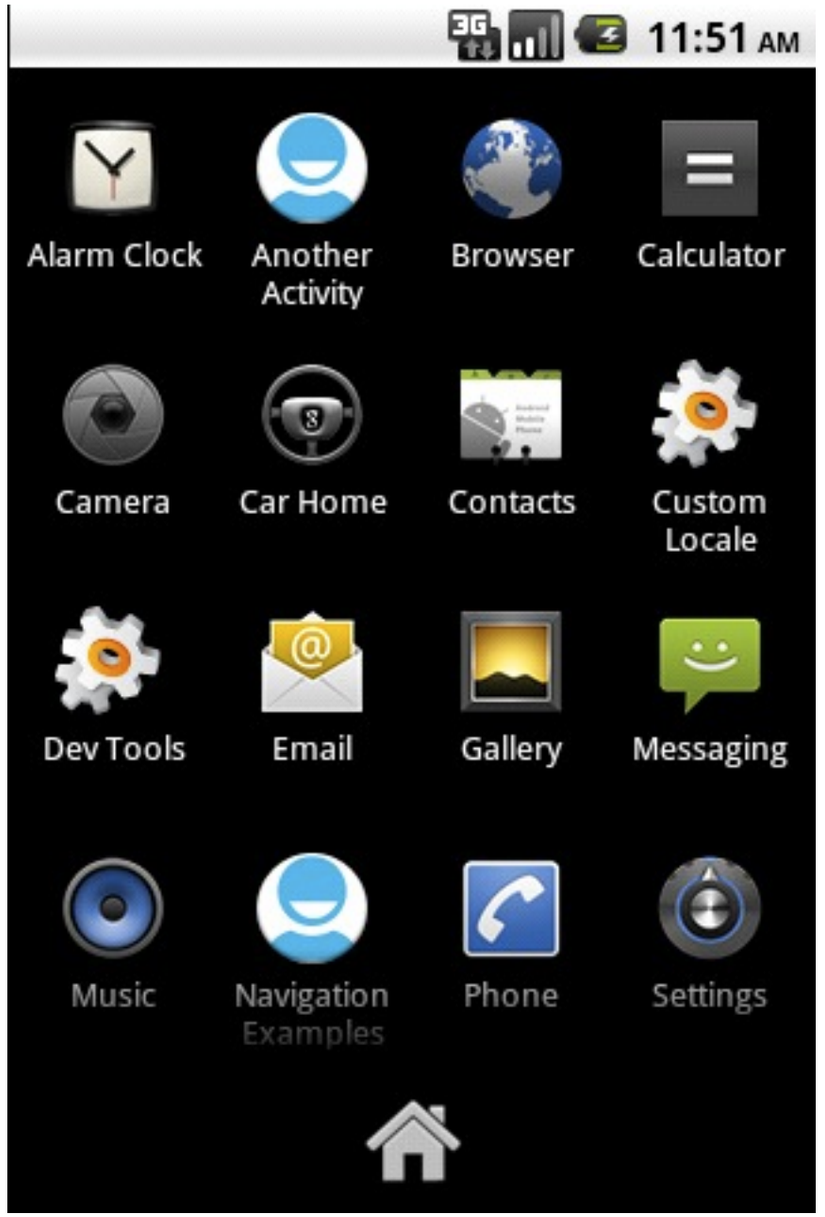
Aside: The Manifest and Intent Filters



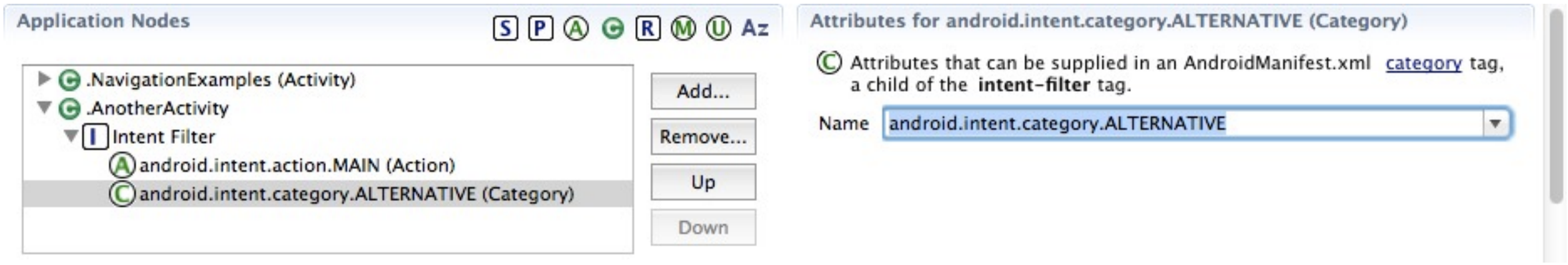
Aside: The Manifest and Intent Filters



This will appear as a separate launcher! →



Aside: The Manifest and Intent Filters

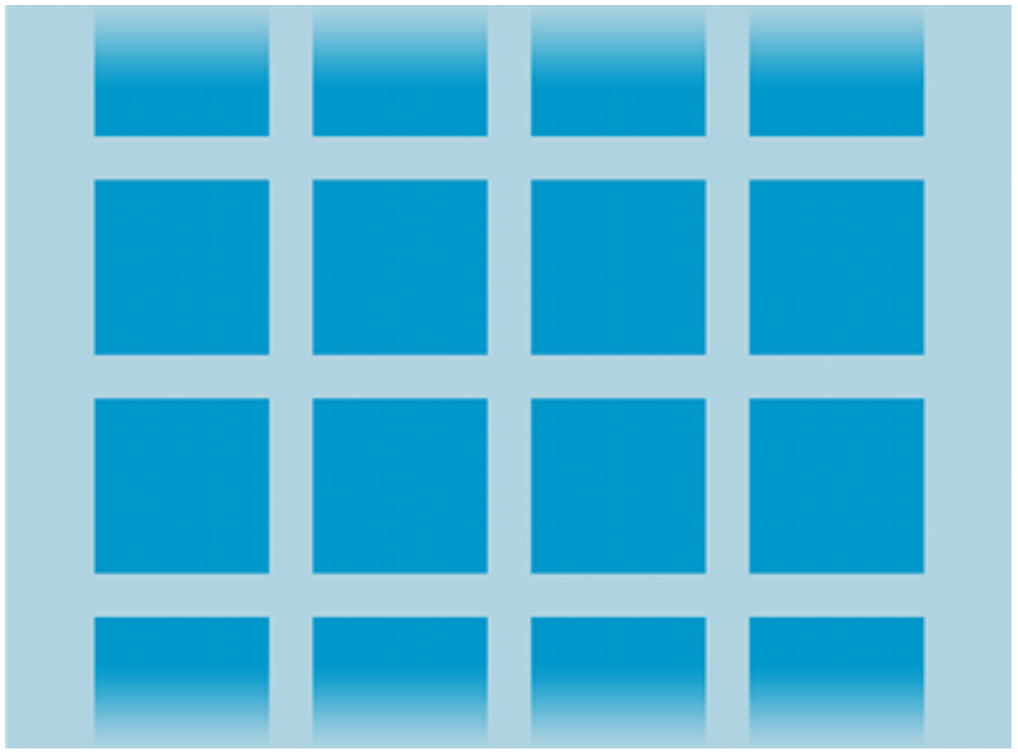


android.intent.category.ALTERNATIVE makes it an "alternative" activity (which won't appear in the menu)

ListView and GridView

ListView and GridView

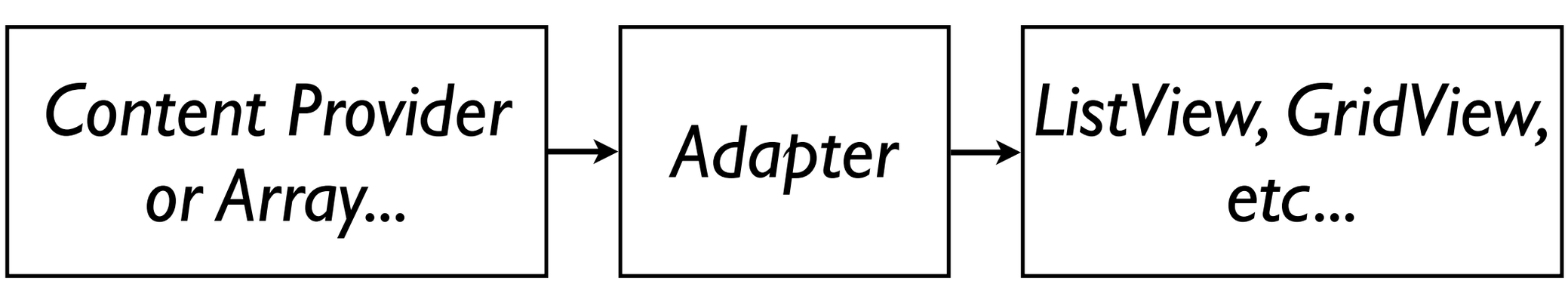
ListView



GridView

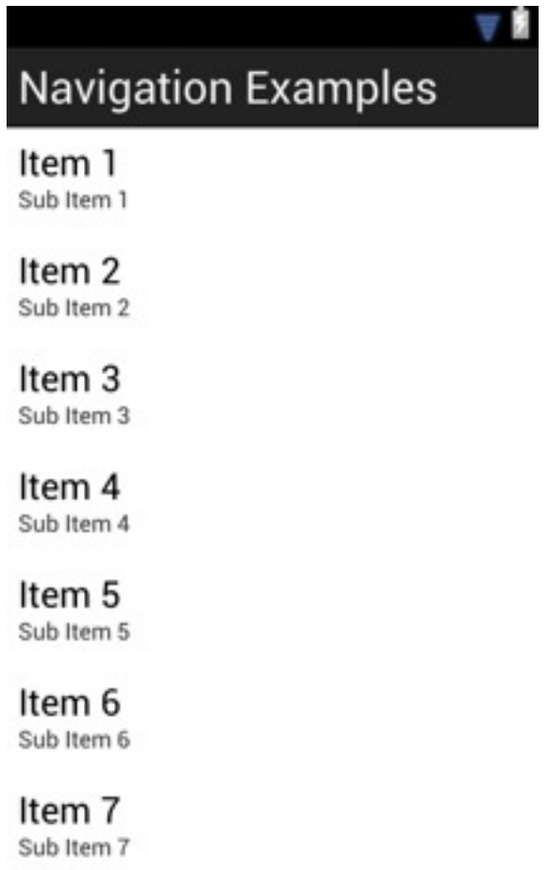
Useful to organize lists of items for further detail (e.g. lists of nearby restaurants, lists of train times...)

Adapters



Adapters "wrap" a data source (database, array, etc.) for use in a View (ListView, GridView, Spinner, etc.)

Using a ListView



NavigationExamples

The Code:
onCreate()

```
public class NavigationExamples extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        /* Initialize the Activity */
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_navigation_examples);

        /* Set up the ListView's adapter */
        ListView myListView =
          (ListView) findViewById(R.id.myListView);
        ArrayAdapter<String> stateList =
          new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1,
            getResources().getStringArray(R.array.states));
        myListView.setAdapter(stateList);

        /* ... */
```

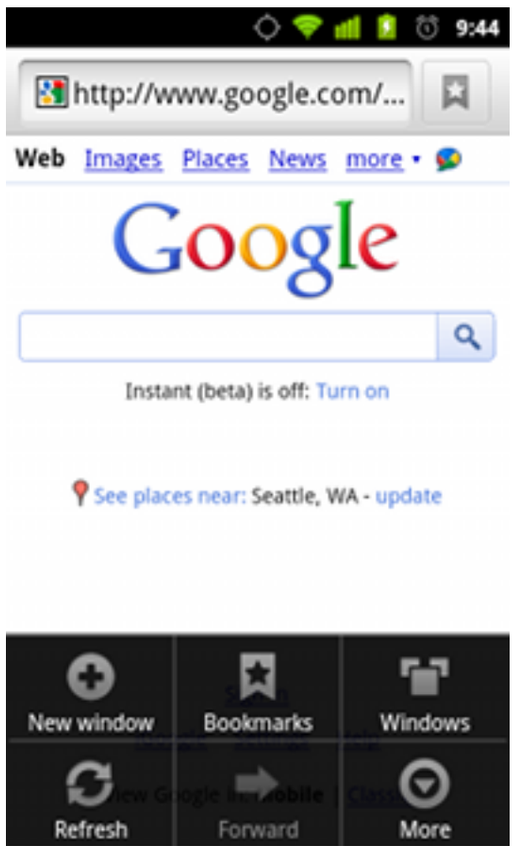
The Code:
onCreate()

```
/* ... */

/* Listen for clicks on each item in the ListView */
myListView.setOnItemClickListener(
    new OnItemClickListener() {
      public void onItemClick(AdapterView<?> parent,
        View view, int position, long id) {
          /* Start AnotherActivity... */
          Intent anotherActivityIntent =
              new Intent(NavigationExamples.this,
                         AnotherActivity.class);
          startActivity(anotherActivityIntent);
        }
    });
  }
```

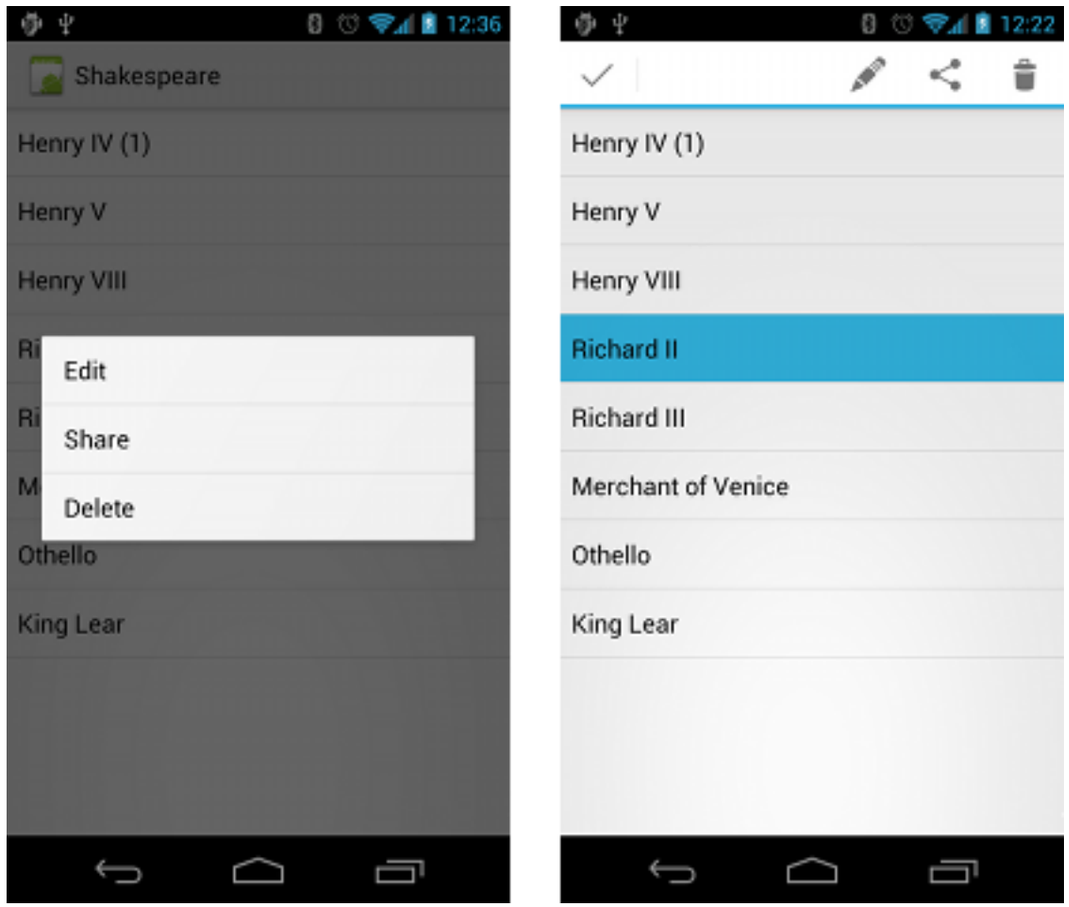
Using a ListView: Demo

Menus

Menus



Option Menu



Floating context menu and contextual action bar

Useful for common modes or actions related to an Activity

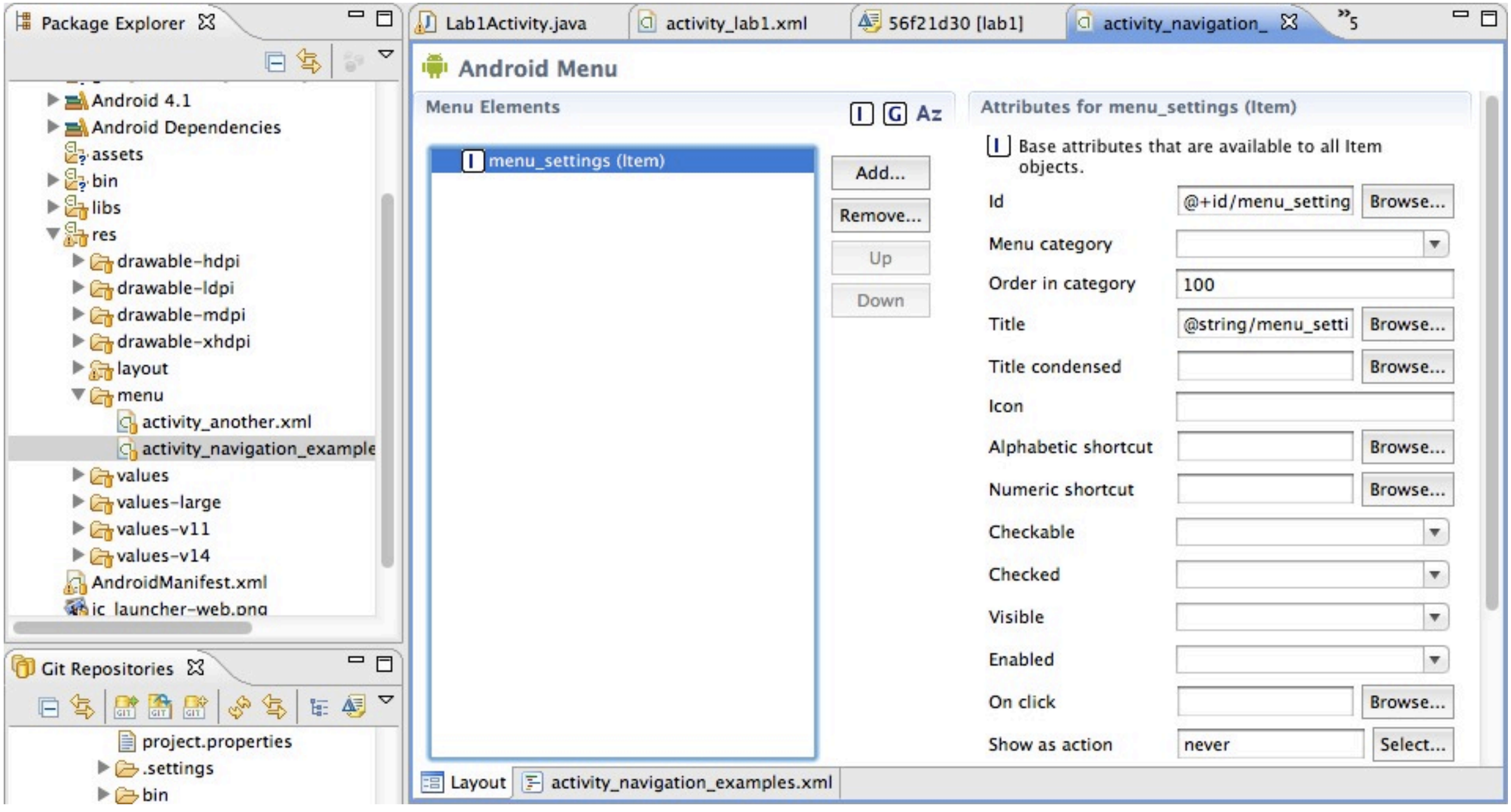
(Context menus are useful for items IN a View)

Option Menus

All Activities have an option menu!

```
public class NavigationExamples extends Activity {
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(
          R.menu.activity_navigation_examples, menu);
        return true;
    }
}
```

Option Menus



The Code:
onOptionsItemSelected()

```
public class NavigationExamples extends Activity {
    public boolean onOptionsItemSelected(MenuItem item) {
        /* Choose your action based on the item selected. */
        switch (item.getItemId()) {
        case R.id.menu_settings:
            /* Make the settings appear. */
            return true;
        default:
            return super.onOptionsItemSelected(item);
        }
    }
}
```

Context Menus

- Very similar to Option Menus
- `onCreateContextMenu()` instead of `onCreateOptionsMenu()`
  - Include `super.onCreateContextMenu()` first
- `onContextItemSelected()` instead of `onOptionsItemSelected()`
- `((AdapterContextMenuInfo) menuItem.getMenuInfo()).id`
  gets the item context in `onOptionsItemSelected`

Using Menus:
Demo

I Taught You That So I Could Teach You This: The Action Bar

The Action Bar

At the top of the screen, this is better than a menu

The Action Bar

At the top of the screen, this is better than a menu

Downside: It's only supported on Android 3.0+ (API Level 11+)

The Action Bar

At the top of the screen, this is better than a menu

Downside: It's only supported on Android 3.0+ (API Level 11+)

But there is some sample compatibility code in

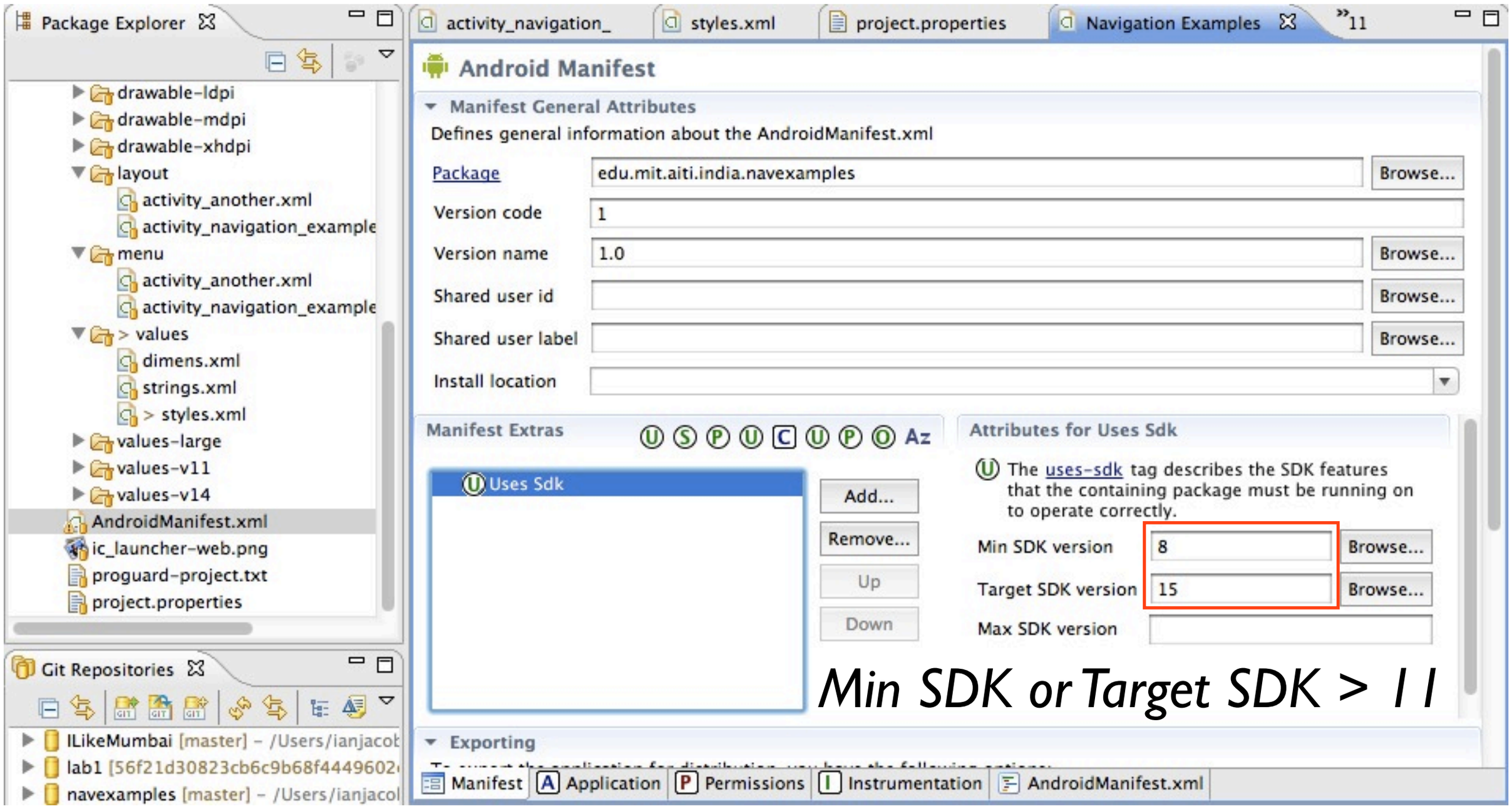
`<sdk>/samples/<android-version>/ActionBarCompat/`

The Action Bar

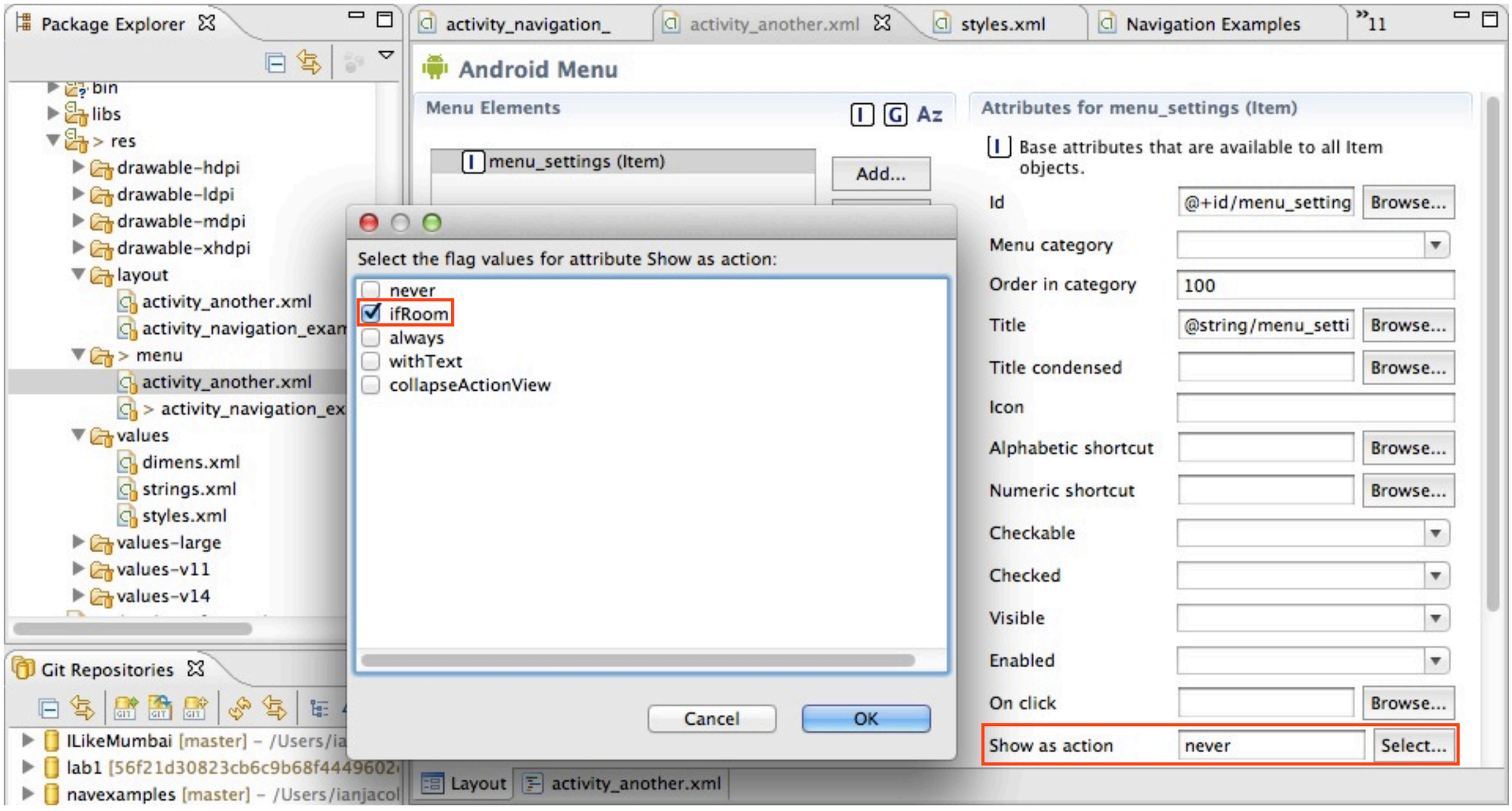
Just like Option Menus!

- Uses `onCreateOptionsMenu()` and `onOptionsItemSelected()`
- Make Action Bar Items using menu resources

Activating the Action Bar



Activating the Action Bar

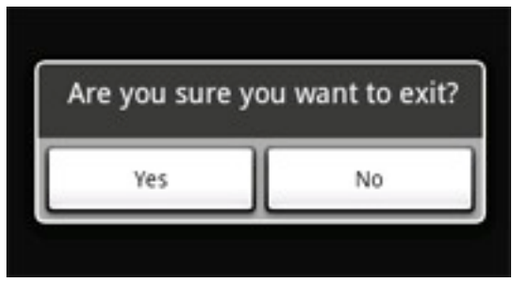


The Action Bar:

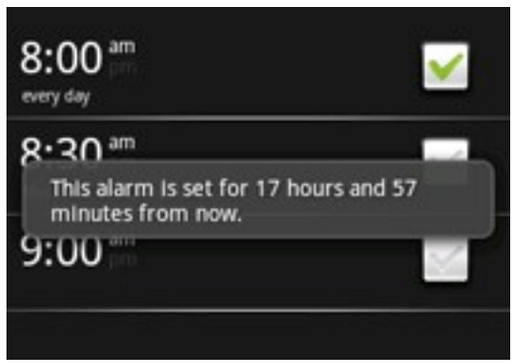
Demo

Dialogs and Notifications

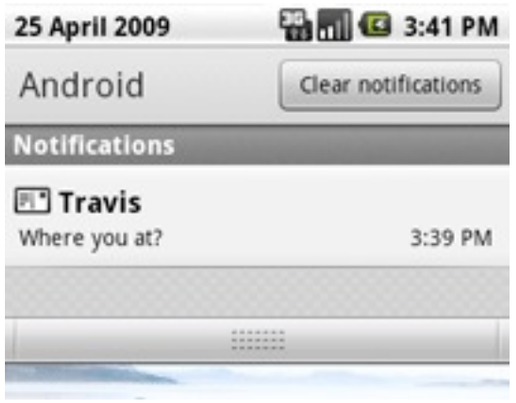
Dialogs and Notifications



AlertDialog



Toast Notification



Status Notification

AlertDialogs ask for simple user input.
Other dialogs are used to show progress of an action.

Notifications let the user know of a background event (e.g. "2 new e-mails!" or "An update has been downloaded")

Alert Dialogs: Code

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setMessage("You clicked the context menu!")
       .setCancelable(false)
       .setPositiveButton("Yes",
           new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                dialog.cancel();
            }})
       .setNegativeButton("No",
           new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                dialog.cancel();
            }});
AlertDialog dialog = builder.create();
dialog.show();
```

Toast Notifications: Code

```
Toast.makeText(this,
               "This is a toast notification!",
               Toast.LENGTH_SHORT)
     .show();
```

Status Notifications: Code

```
/* Get the NotificationManager service. */
String ns = Context.NOTIFICATION_SERVICE;
NotificationManager mNotificationManager =
    (NotificationManager) getSystemService(ns);

/* Create the notification. */
int icon = R.drawable.ic_launcher;                  // Icon
CharSequence tickerText = "The ticker text";  // Ticker Text
long when = System.currentTimeMillis();             // When to show

Notification notification = new Notification(
    icon, tickerText, when);
```

Status Notifications: Code (Continued)

```
/* Prepare the event info (message) */
Context context = getApplicationContext();
CharSequence contentTitle = "My notification";  // The title
CharSequence contentText = "Hello World!";      // The text
/* This intent to restart is "delayed" for a later date. */
Intent notificationIntent = new Intent(
    this, NavigationExamples.class);
PendingIntent contentIntent = PendingIntent.getActivity(
    this, 0, notificationIntent, 0);

notification.setLatestEventInfo(
    context, contentTitle, contentText, contentIntent);

/* Send the notification. */
final int HELLO_ID = 1;
mNotificationManager.notify(HELLO_ID, notification);
```

Resources

- Android Developer Site
  - "ListViews" <http://developer.android.com/guide/topics/ui/layout/listview.html>
  - "Menus" <http://developer.android.com/guide/topics/ui/menus.html>
  - "Action Bar" <http://developer.android.com/guide/topics/ui/actionbar.html>
  - "Dialogs" <http://developer.android.com/guide/topics/ui/dialogs.html>
  - "Notifications" <http://developer.android.com/guide/topics/ui/notifiers/index.html>
- Android Design (on the Developer Site)
  - "App Structure" <http://developer.android.com/design/patterns/app-structure.html>
  - "Navigation" <http://developer.android.com/design/patterns/navigation.html>

More Resources

- Android Training (on the Developer Site)
  - "Designing Effective Navigation" <http://developer.android.com/training/design-navigation/index.html>
  - "Implementing Effective Navigation" <http://developer.android.com/training/implementing-navigation/index.html>
- Android Patterns (a list of useful interaction patterns for Android apps) <http://www.androidpatterns.com/>
- Giorgio Venturi has a presentation about Android Design Patterns <http://www.closertag.com/thought-pieces/android-design-patterns/>

Version Control

What is version control?

Version control is a way to manage the history of a project's source code.

Types of version control

- Client/Server (CVS, SVN, SourceSafe...)
- Distributed (Git, Mercurial, bazaar...)

We will use Git in the labs.

Version Control Concepts

- *Revision/Commit* – Saved version of source
- *To commit (verb)* – To save the version
- *Repository* – Where the history is stored

Install EGit (Version Control)

Adding a new software site

There are several different ways to add a software site to the list of sites that are used when browsing available software and checking for updates. You must know the Web Site location (URL) of the site that you want to add. To add the site, use one of the following procedures:

- Add a new site using the Install/Update > Available Software Sites preference page.
  1. Click the **Add...** button.
  2. Type a name into the **Name** text box.
  3. If the software site is located on the web, type the Web Site location (URL) of the site into the **Location** text box. You may also paste or drag and drop a URL from a web browser into this text box.
  4. If the software site is in your local file system (including a CD), click **Local...** to specify the directory location of the site.
  5. If the software site is in your local file system but is packaged as a jar or zip file, click **Archive...** to specify the name of the file.

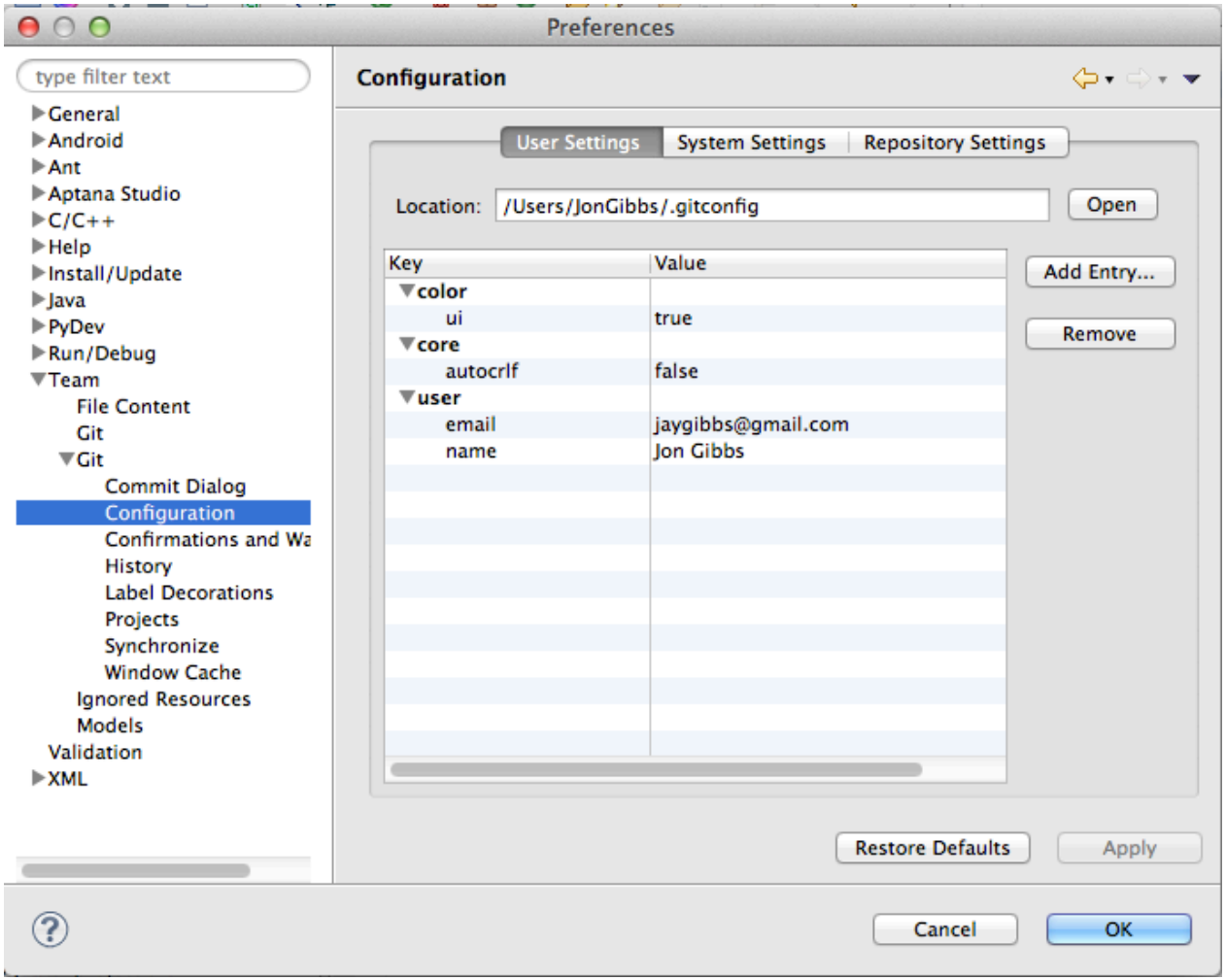
- http://download.eclipse.org/egit/updates

Setting up Egit

Add your information:

- Select Preferences->Team-> Git -> Configuration
- Add your name and email in two separate new entries
- Use user.email and user.name in the “key” field

Setting up Egit



If using windows:

Setting up the Home Directory on Windows

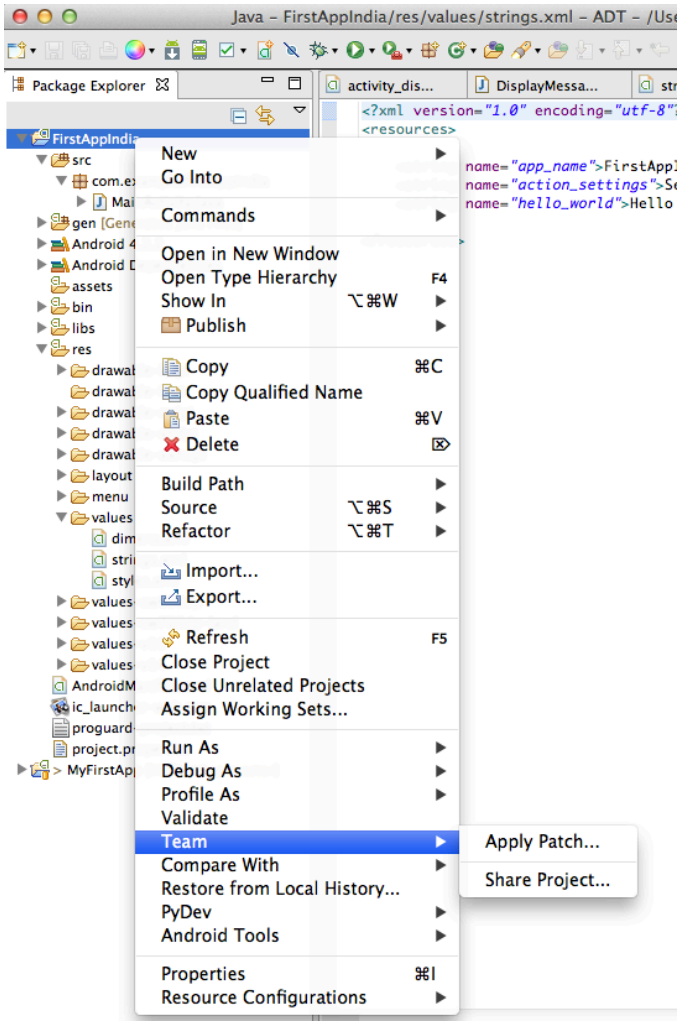
Add the environment variable `HOME` to your environment variables.

1. In Windows 7, type "environment" at the start menu
2. Select "Edit environment variables for your account"
3. Click the "New" button.
4. Enter "HOME" in the name field
5. Enter "%USERPROFILE%" or some other path in the value field.
6. Click OK, and OK again. You have just added the Home directory on Windows.

Using EGit

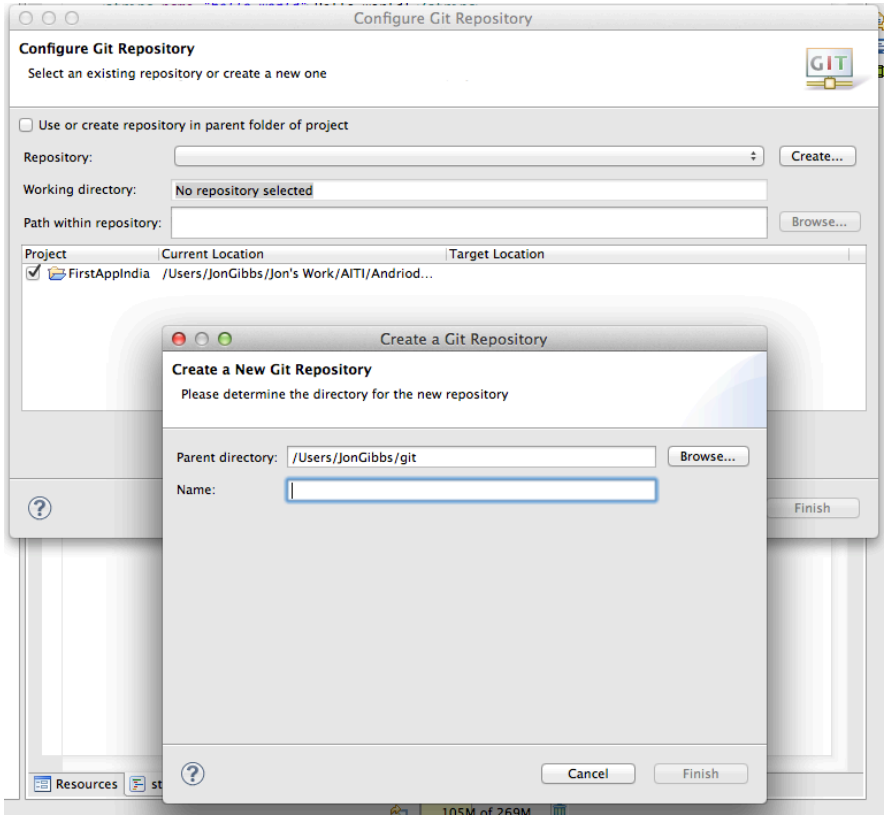
Create a repository

- Right Click on the Project, Select Team -> Share Project



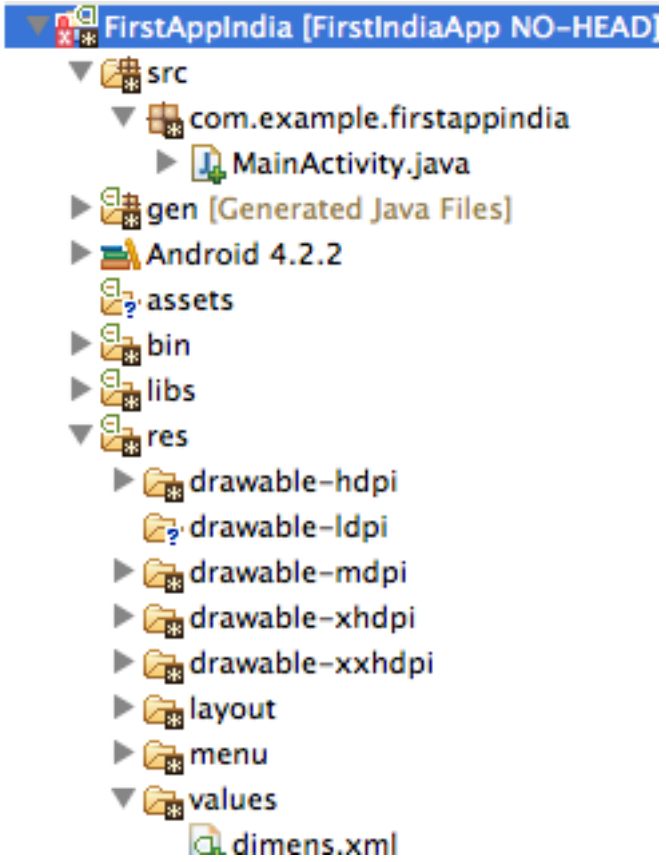
Create a repository

- Select “Git” on the next screen, click next
- Create a new directory OUTSIDE of your working directory
- Click through to “Finish”



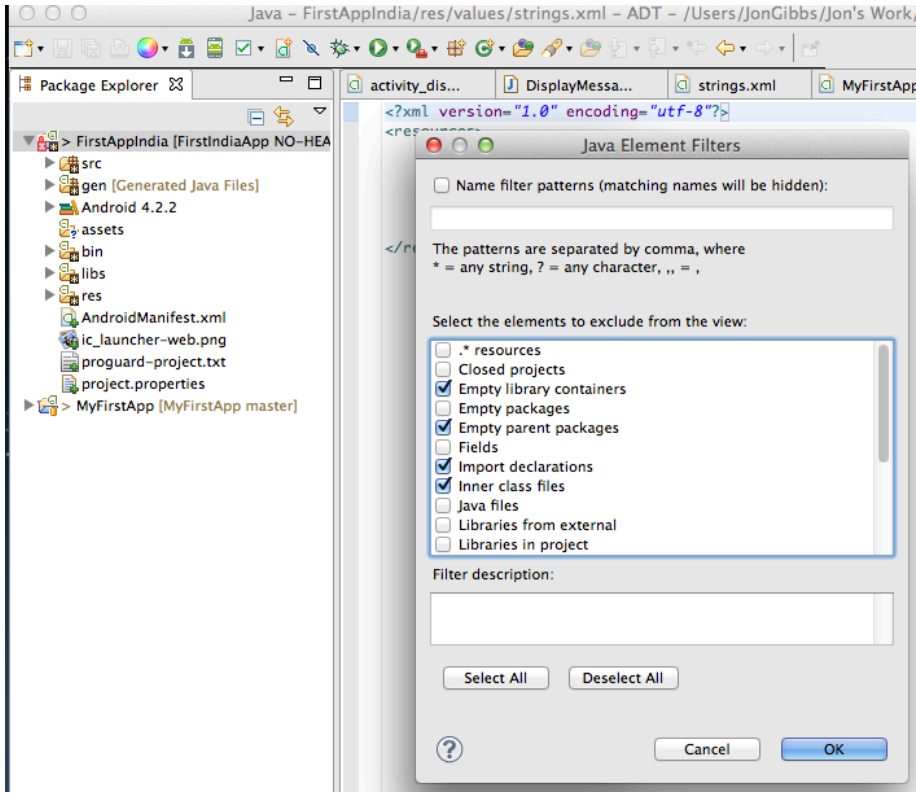
Create a repository

- You should see an error in your app (Don't worry)
- Select Team->Add to Index
- Right click the bin folder and select Team->Ignore



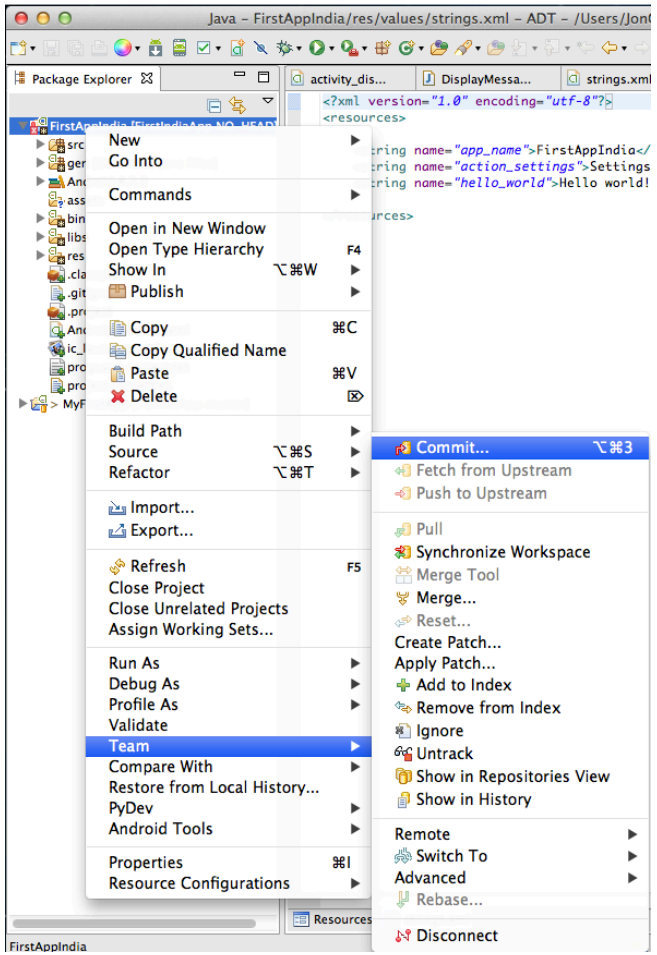
Add the .gitignore file

- Select the down arrow in package explorer -> select filters
- Uncheck the resources box
- Add the .gitignore file to the index



Commit the changes to your repository

- Right click your project and select Team -> Commit



Commit the changes to your repository

- Add a message and commit the files!!
- Quit and Restart Eclipse ADK