



Data Retrieval and Storage

Sparky Rhode

Android Developer Programs Engineer

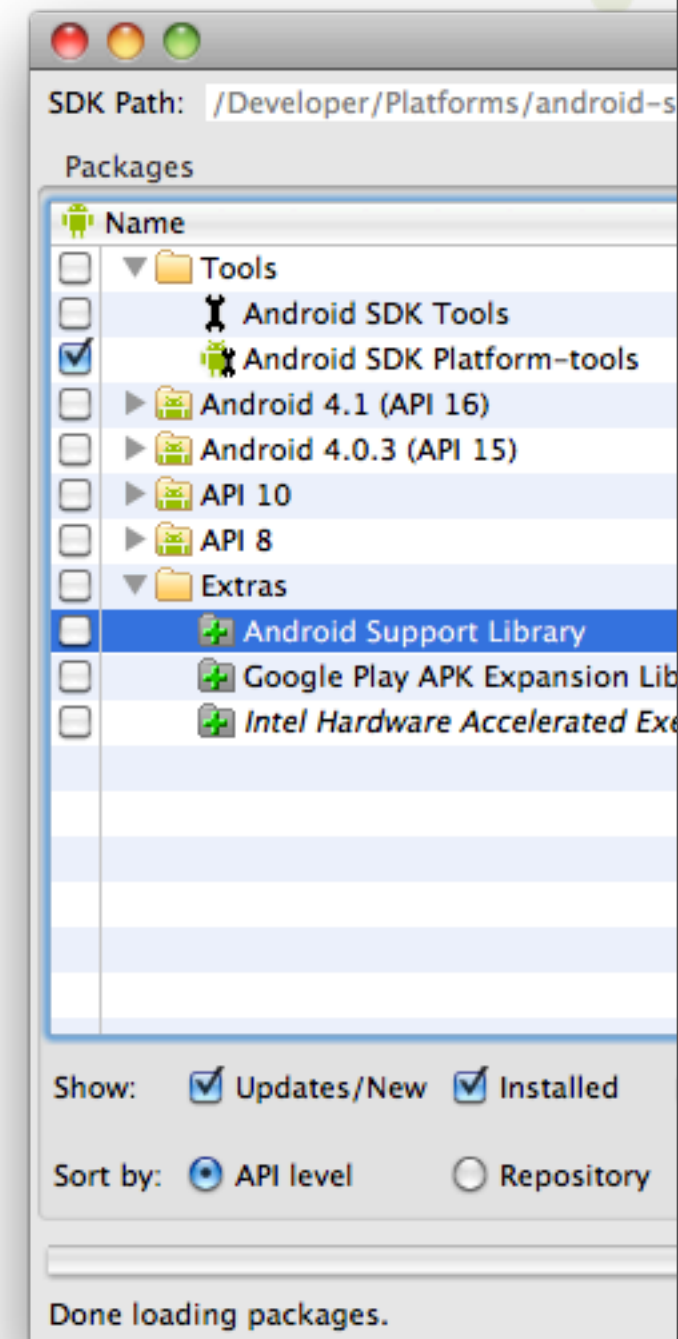
August 15, 2012



Herzliche Glückwünsche zur Mariä Himmelfahrt

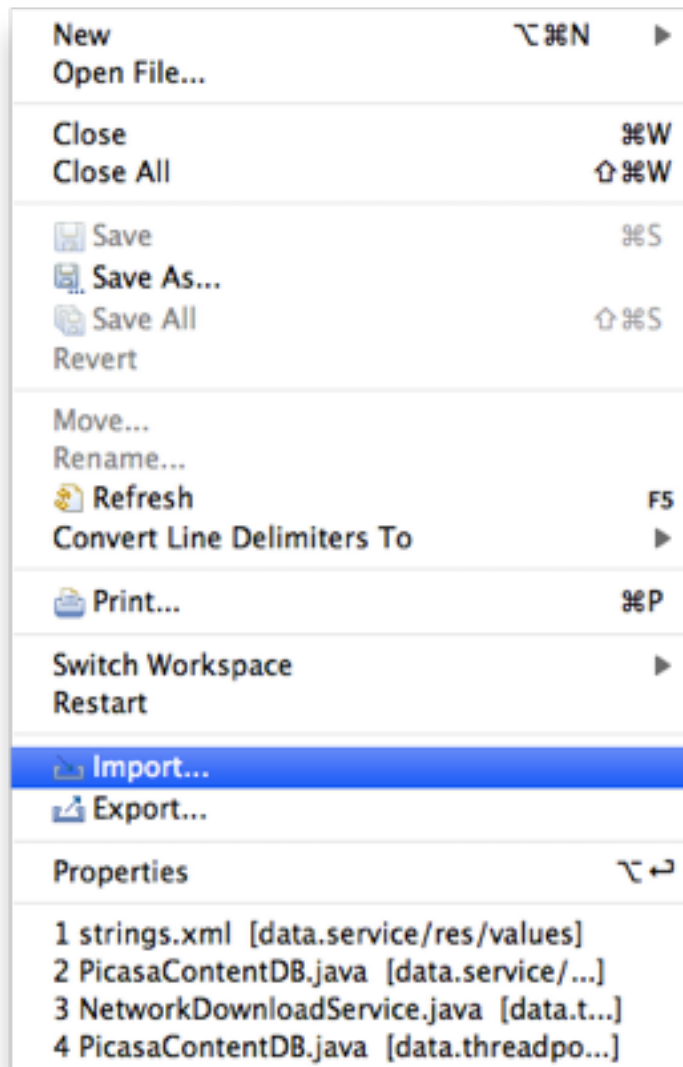
Before We Begin

- Make sure that you have
 - API level 16
 - With the support library installed
 - Java Compiler Compliance set to 1.6
- Get the Code
 - `git clone http://code.google.com/p/main-egac-2012/`



Importing Projects into Eclipse

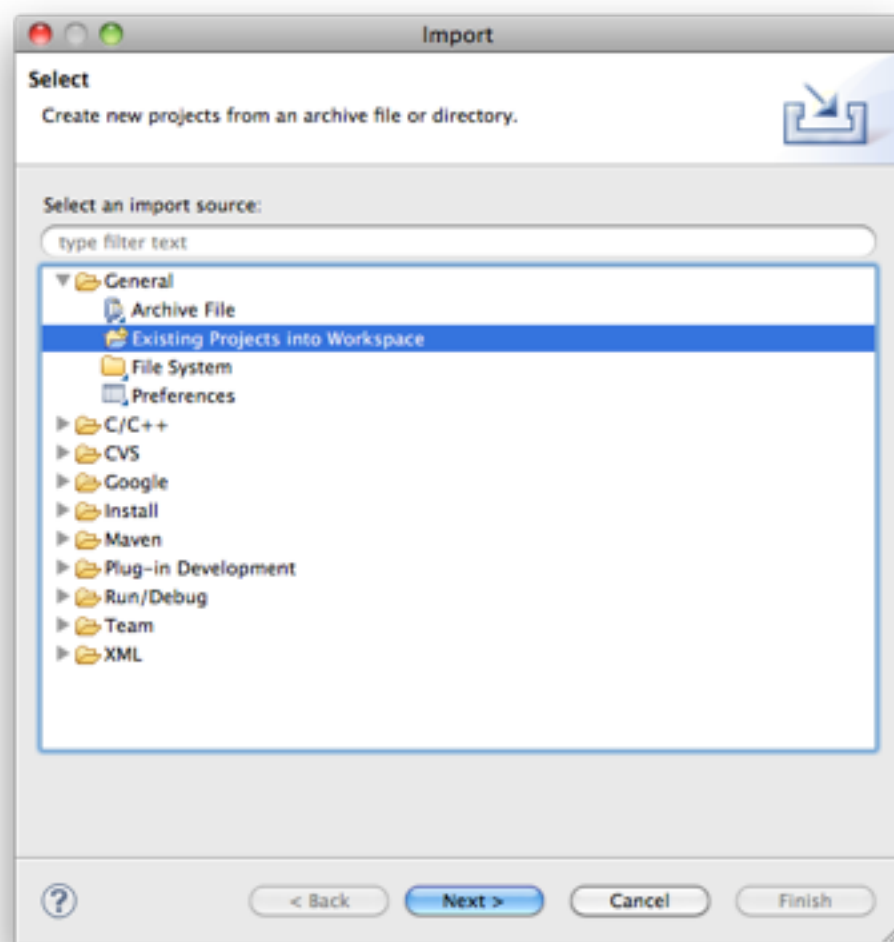
■ Select File-Import



Importing Projects into Eclipse



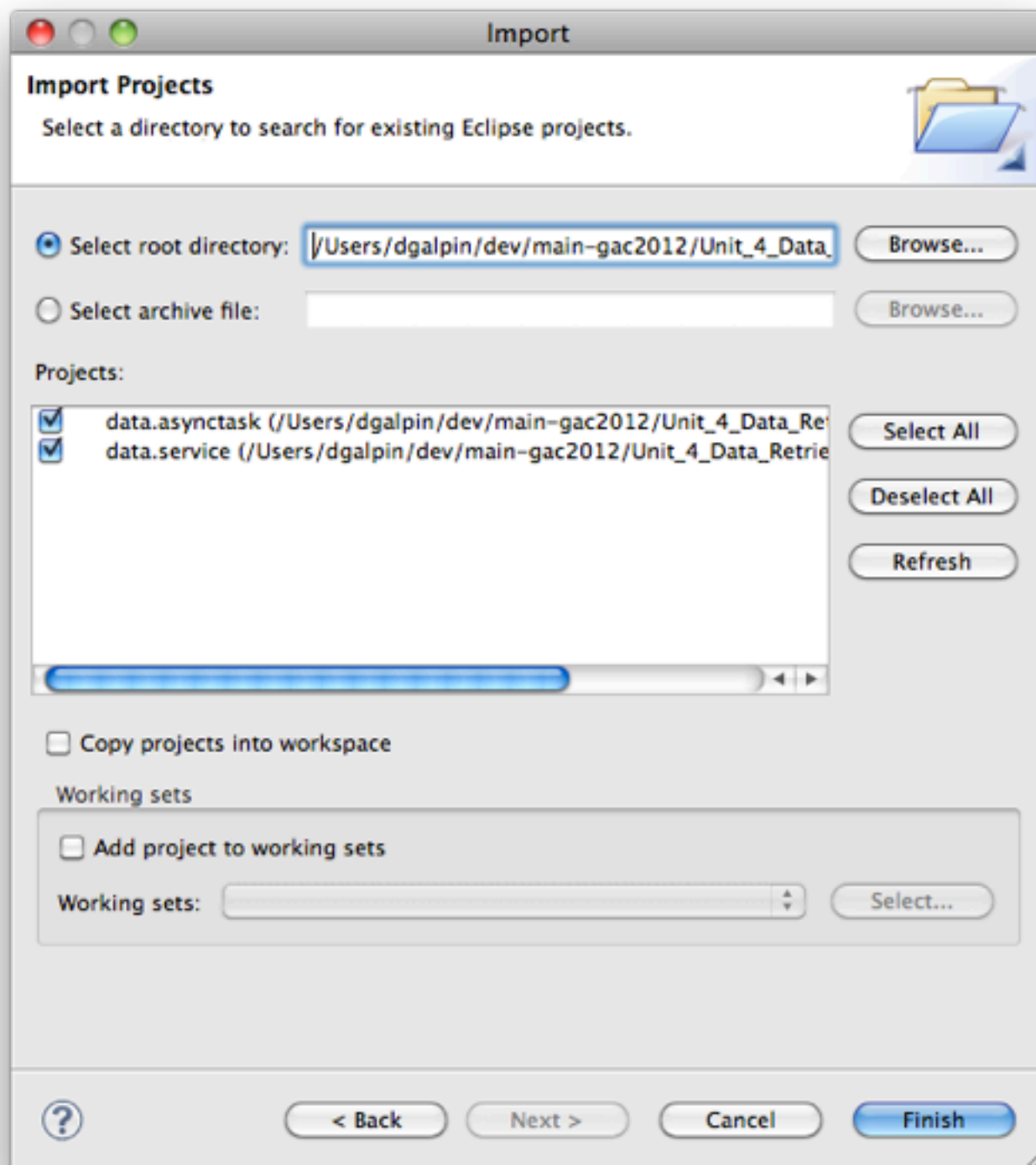
- Select **General**
- Existing Projects into Workspace



Importing Projects into Eclipse



- Browse to Unit_4_Data_Retrieval
- Import Projects





Data Retrieval and Storage?

Retrieval

- Files
- Ports
- Users
- IPC



Storage

- Files
- Databases
- CAM
- Cloud



Universalities

- Responsive
- Thrifty
- Smart



Agenda



- Part 1: Storing and Retrieving Locally
- Part 2: Retrieving from the Internet
- Part 3: Providing Locally



Part 1: Storing and Retrieving Locally

Files

- Long stream
- No defined structure
- Best read in order
- Standard API



Shared Preferences



- Key-Value pairs
- Simple values
- Ideal for saving application state

SQL Databases



- Complex, structured, repeating data
- Random access
- Searching, filtering, reporting, combining
- Familiar API

LRU Cache



- Similar to Shared Preferences
- High speed, low size
- Short lived



Files

Wednesday, August 15, 2012

You are all familiar with Java, so I won't go into the specifics of reading and writing Java files. You can learn that anywhere, and there's not a lot specific to Android.

Where's my file?

- Internal storage
- Cache
- Temp
- External storage
- Media
- Shared



Sidebar: Android Security

- Unique **userid** per application
- Private by default



External Storage

- Insecure by design



File-Reading Permissions



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

Files

- Path
- Uri
- Location
- Scanner



Internal

- Permanent
- Cache



External

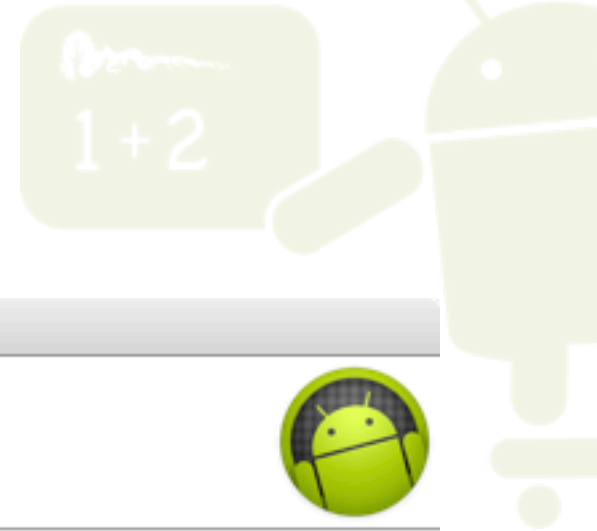
- “Insecure”
- “Unreliable”
- Standard locations





Code Lab 1: Read Local Assets

Step 1: Create a 2-Fragment App



- In Eclipse:
 - File > New > Android Application Project
 - MasterDetail Flow
- NOTE:
Don't make any mistakes.

The screenshot shows the 'New Android App' dialog box in the Eclipse IDE. The title bar says 'New Android App'. Below the title bar, it says 'New Android Application'. A warning icon and text state: 'The prefix 'com.example.' is meant as a placeholder and should not be used'. The dialog contains several input fields and checkboxes:

- Application Name:** 500 dp
- Project Name:** FiveWhat
- Package Name:** com.example.acamp.fivewhat
- Build SDK:** Android 4.1 (API 16) (with a 'Choose...' button)
- Minimum Required SDK:** API 12: Android 3.1 (Honeycomb)
- ☒ Create custom launcher icon
- ☐ Mark this project as a library
- ☒ Create Project in Workspace
- Location:** /Users/sparkyr/Documents/Dev/workspace/FiveWhat (with a 'Browse...' button)

At the bottom, there is a help icon (question mark) and four buttons: '< Back', 'Next >', 'Cancel', and 'Finish'.

Step 2: Make it Dark



- In AndroidManifest.xml:

```
<uses-sdk  
    android:minSdkVersion="16"  
    android:targetSdkVersion="16" />
```

- In /res/values/styles.xml:

- Delete occurrences of **.Light**

```
<style name="AppTheme" parent="android:Theme" />  
or  
<style name="AppTheme" parent="android:Theme.Holo" />
```

Step 3: Add Graphics



- Copy the drawable-* folders from Icon_Set into your project's /res/ folder. Overwrite the existing folders.
- Copy the images folder into your project's /assets/ folder.

Step 4: Read Assets Directory



- Add class ListAssetsTask into the end of DummyContent.

```
public static class ListAssetsTask extends AsyncTask<Context, Void, String[]> {

    Context mContext = null;
    String[] mAssetFiles = null;

    static final String ASSET_IMG_DIR = "images";

    @Override
    protected String[] doInBackground(Context... params) {

        mContext = params[0];

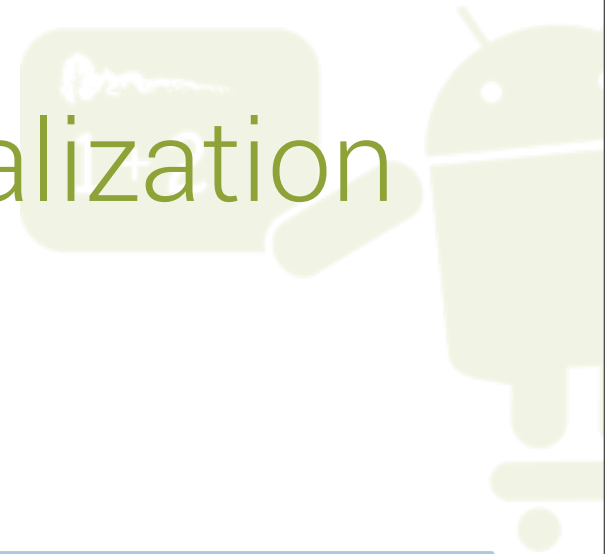
        AssetManager assetManager = mContext.getAssets();

        try {
            mAssetFiles = assetManager.list(ASSET_IMG_DIR);
        } catch (IOException e) {
            Log.e("tag", e.getMessage());
        }

        for (String filename : mAssetFiles) {
            addItem(new DummyItem(filename, ASSET_IMG_DIR + java.io.File.separator + filename));
        }
        return mAssetFiles;
    }

} // ListAssetsTask
```

Step 5: Run ListAssetsTask During Initialization



- Add Init method in DummyContent

```
public static boolean initialized = false;

public static void Init(Context context) {
    if (!initialized) {
        new ListAssetsTask().execute(context);
        initialized = true;
    }
}
```

- Call Init during PicListActivity.onCreate()
- TIP: Use Cmd-Shift-O to organize imports.
- TIP: Now would be a good time to run your app and see if it reads the directory.

Step 6: Add ImageView



- Add an ImageView to fragment_pic_detail.xml
- Wrap it in a LinearLayout
 - Hint: Refactor > Android > Wrap in Container...

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/fragment_pic_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/pic_detail"
        style="?android:attr/textAppearanceLarge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="16dp"
        tools:context=".PicDetailFragment" />

    <ImageView
        android:id="@+id/pic_detail_pic"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:contentDescription="Large image" />

</LinearLayout>
```

Step 7: Populate ImageView



- In `PicDetailFragment.onCreateView()`, decode image from asset file.

```
AssetManager am = getActivity().getAssets();
try {
    Drawable d = Drawable.createFromStream(am.open(mItem.content), null);
    ((ImageView) rootView.findViewById(R.id.pic_detail_pic)).setImageDrawable(d);
} catch (IOException e) {
    ((TextView) rootView.findViewById(R.id.pic_detail))
        .setText("Could not load asset: " + mItem.content);
}
```

- In `DummyContent.toString()`, return id.



Shared Preferences

Storage for Simple Values



- Simple types only
 - Boolean, Float, Int, Long, String, StringSet
- Persistent
- Transactional
- Implicit or Explicit groups
 - Settings are in Implicit group



Code Lab 2: Persist Application State

Step 1: Read Shared Preference



- In class DummyContent:

```
public static final String PREFS_GROUP = "DummyPrefs";  
public static final String LAST_ITEM_KEY = "ItemId";
```

- In PicDetailFragment.onCreate(), check SharedPreferences if no id is passed.

```
if (getArguments().containsKey(ARG_ITEM_ID)) {  
    mItem = DummyContent.ITEM_MAP.get(getArguments().getString(ARG_ITEM_ID));  
} else {  
    SharedPreferences sp =  
        getActivity().getSharedPreferences(DummyContent.PREFS_GROUP,  
            Context.MODE_PRIVATE);  
    String lastItemKey = sp.getString(DummyContent.LAST_ITEM_KEY, "");  
    if (!TextUtils.isEmpty(lastItemKey)) {  
        mItem = DummyContent.ITEM_MAP.get(lastItemKey);  
    }  
}
```

Step 2: Communicate via List Activity



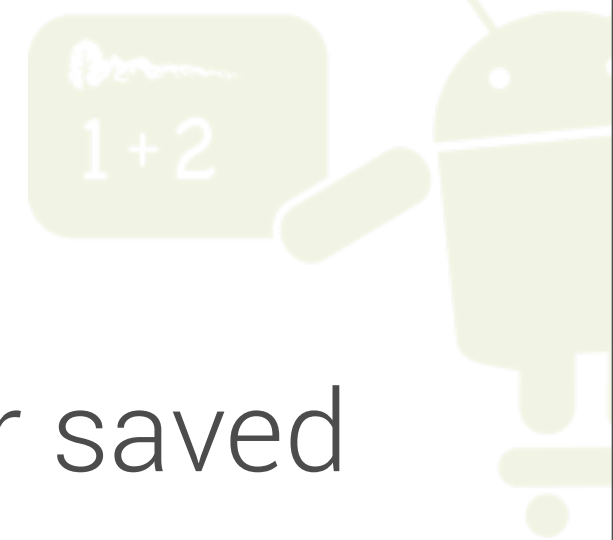
- In class PicListActivity:

```
private String mItemSelected;
```

- In PicListActivity.onItemSelected(), record selection.

```
public void onItemSelected(String id) {  
    mItemSelected = id;  
    if (mTwoPane) {  
    ...
```

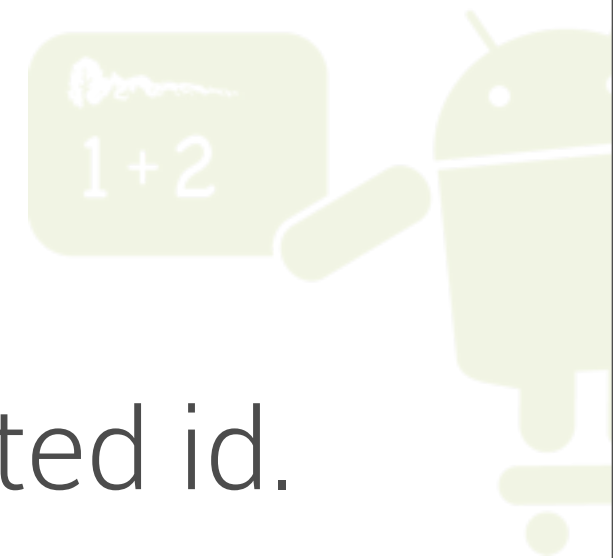
Step 3: Look for Saved Value on Create



- In `PicListActivity.onCreate()`, check for saved value.

```
// If two-pane view, then if first launch, try to restore last item viewed
if (mTwoPane && (savedInstanceState == null)) {
    SharedPreferences sp =
        getSharedPreferences(DummyContent.PREFS_GROUP,
Context.MODE_PRIVATE);
    mItemSelected = sp.getString(DummyContent.LAST_ITEM_KEY, "");
    if (!TextUtils.isEmpty(mItemSelected)) {
        this.onItemSelected(mItemSelected);
    }
}
```

Step 4: Save Value on Stop



- In `PicListActivity.onStop()`, save selected id.

```
@Override
protected void onStop() {
    if (!TextUtils.isEmpty(mItemSelected)) {
        SharedPreferences sp =
            getSharedPreferences(DummyContent.PREFS_GROUP, Context.MODE_PRIVATE);
        Editor spe = sp.edit();
        spe.putString(DummyContent.LAST_ITEM_KEY, mItemSelected);
        spe.commit();
    }
    super.onStop();
}
```

- TIP: Type “onSt” and then ctrl-space for auto-completion.



Part 2: Retrieving from the Internet

Network Data Retrieval

- Asynchronous
- Fault Tolerant
- Serial



HTTP Libraries



- <http://android-developers.blogspot.co.uk/2011/09/androids-http-clients.html>
- ~~Apache HTTP Client~~
- **HttpURLConnection**

Fetching Network Data



```
@Override
protected void onHandleIntent(Intent paramIntent) {
    Process.setThreadPriority(Process.THREAD_PRIORITY_BACKGROUND);
    String str = paramIntent.getDataString();
    URL url;
    try {
        url = new URL(str);
        URLConnection urlConn = url.openConnection();
        if ((urlConn instanceof HttpURLConnection)) {
            HttpURLConnection hUrlConn = (HttpURLConnection) urlConn;
            hUrlConn.setRequestProperty("User-Agent", USER_AGENT);
            hUrlConn.getResponseCode();
        }
    }
    ...
}
```

Wire Formats



- Human-readable
- Be lazy: Use a standard
- XML, JSON, SOAP

SAX vs. XML Pull Parser



- ~~SAX~~
- **XmlPullParser**

Storing the Data - PullParser



```
mImage = new ContentValues();
} else {
    String key;
    if (str1.equalsIgnoreCase(CONTENT)) {
        key = PicasaContentDB.PicasaFeatured.IMAGE_URL;
    } else if (str1.equalsIgnoreCase(THUMBNAIL)) {
        key = PicasaContentDB.PicasaFeatured.IMAGE_THUMB_URL;
    } else continue;
    String value = localXmlPullParser.getAttributeValue(null, "url");
    if (value == null) break;
    mImage.put(key, value);
}
```

Parsing XML with XmlPullParser



```
XmlPullParser localXmlPullParser = localXmlPullParserFactory.newPullParser();
localXmlPullParser.setInput(paramInputStream, null);
int i = localXmlPullParser.getEventType(), j = 1;
if (i != 0) return;
this.mImages = new Vector<ContentValues>(NUM_IMAGES);
while (true) {
    int k = localXmlPullParser.next();
    if (Thread.currentThread().isInterrupted()) throw new XmlPullParserException("Cancelled");
    else if (k == XmlPullParser.END_DOCUMENT) break;
    else if (k == XmlPullParser.START_DOCUMENT) continue;
    else if (k == XmlPullParser.START_TAG) {
        String str1 = localXmlPullParser.getName();
        if (str1.equalsIgnoreCase(ITEM)) { mImage = new ContentValues();
        } else {
            String key;
            if (str1.equalsIgnoreCase(CONTENT)) key = PicasaContentDB.PicasaFeatured.IMAGE_URL;
            else if (str1.equalsIgnoreCase(THUMBNAIL)) key =
PicasaContentDB.PicasaFeatured.IMAGE_THUMB_URL;
            else continue;
            String value = localXmlPullParser.getAttributeValue(null, "url");
            if (value == null) break;
            mImage.put(key, value);
        }
    }
    else if ((k == XmlPullParser.END_TAG) && (localXmlPullParser.getName().equalsIgnoreCase(ITEM))
        && (mImage != null)) {
        this.mImages.add(mImage); mImage = null; j++;
    }
}
```

Avoid This!



Avoid This!





Use a Thread

AsyncTask<Params, Progress, Result>



```
private class MyTask extends AsyncTask<Foo, Integer, Bar> {  
    @Override  
    protected Bar doInBackground(Foo... params) {  
        publishProgress(42);  
        return null;  
    }  
  
    @Override  
    protected void onPostExecute(Bar result) {  
        super.onPostExecute(result);  
    }  
  
    @Override  
    protected void onProgressUpdate(Integer... values) {  
        super.onProgressUpdate(values);  
    }  
}  
  
MyTask myTask = new MyTask();  
Bar result = myTask.execute(foo).get();
```



Use a Service with a Thread

Android Services



Android Services



- Have a background lifecycle
- Are a registered component
- Have a Context
- Run in the UI Thread

Declaring a Service in the Manifest



```
<service  
    android:exported="false"  
    android:name="sample.multithreading.NetworkDownloadService" />
```

Deriving from IntentService



```
public class NetworkDownloadService extends IntentService {  
  
    public NetworkDownloadService() {  
        super("PicasaFeaturedService");  
    }  
  
    @Override  
    protected void onHandleIntent(Intent paramIntent) {  
    }  
}
```


Starting the IntentService



```
private static final String PICASA_RSS_URL = "http://picasaweb.google.com/data/feed/base/featured?alt=rss&kind=photo&access=public&slabel=featured&hl=en\_US&imgmax=1600";

Intent localIntent = new Intent(this, NetworkDownloadService.class);
Uri localUri = Uri.parse(PICASA_RSS_URL);
localIntent.setData(localUri);

startService(localIntent);
```

Communicating Status From the Service



```
private class ResponseReceiver extends BroadcastReceiver
{
    private ResponseReceiver() {}

    @Override
    public void onReceive(Context paramContext, Intent paramInt) {
        switch (paramIntent.getIntExtra
(NetworkDownloadService.EXTRA_STATUS,
    NetworkDownloadService.STATE_ACTION_COMPLETE)) {
        case NetworkDownloadService.STATE_ACTION_STARTED:
            setProgressText(R.string.progress_starting_update);
            showProgress(true);
            break;
        ...
    }
}
```

AsyncTask or IntentService



- AsyncTask integrates more
- Service can do more different things



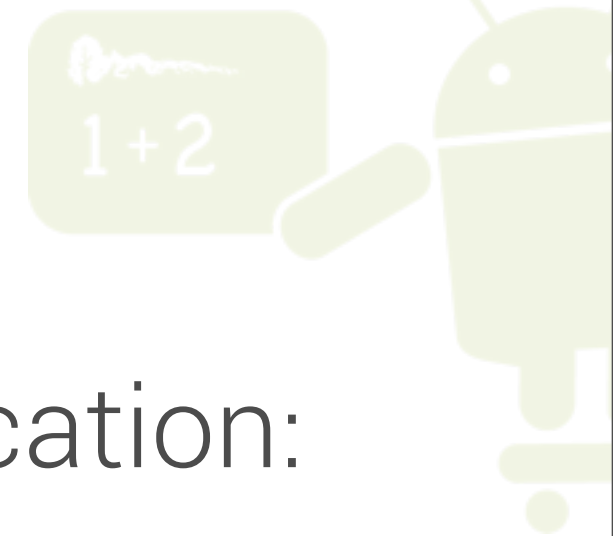
Code Lab 3: Retrieve Data from the Web

Step 1: Declare INTERNET Permission



```
<!--  
Only a n00b forgets to declare internet permission on a feed reader.  
-->  
<uses-permission android:name="android.permission.INTERNET" />
```

Step 1: Declare INTERNET Permission



- In AndroidManifest.xml, above <application:

```
<!--  
Only a n00b forgets to declare internet permission on a feed reader.  
-->  
<uses-permission android:name="android.permission.INTERNET" />
```

Step 2: Copy new Java source code files



Step 2: Copy new Java source code files



- NetworkDownloadService.java
- PicasaPullParser.java

Step 3: Create Constants Class



New Java Class

Java Class
Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

Step 3: Create Constants Class



- Right-click on package
 - com.example.acamp.dip
- New > Class ...

New Java Class

Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

Step 4: Populate Constants



```
public class Constants {  
    static public final boolean LOGV = true;  
    public static final String IMAGE_THUMB_URL = "thumbURL";  
    public static final String IMAGE_URL = "imageURL";  
    public static final int NUM_LINES = 100;  
    public static final String PICASA_RSS_URL = "http://picasaweb...";  
}
```

```
public static void addItem(String key, String content) {  
    addItem(new DummyItem(key, content));  
}
```

Step 4: Populate Constants



```
public class Constants {  
    static public final boolean LOGV = true;  
    public static final String IMAGE_THUMB_URL = "thumbURL";  
    public static final String IMAGE_URL = "imageURL";  
    public static final int NUM_LINES = 100;  
    public static final String PICASA_RSS_URL = "http://picasaweb...";  
}
```

- Add public DummyContent.addItem()

```
public static void addItem(String key, String content) {  
    addItem(new DummyItem(key, content));  
}
```

- [http://picasaweb.google.com/data/feed/base/featured?
alt=rss&kind=photo&access=public&slabel=featured&hl=en_US&img
max=1024&max-results=10](http://picasaweb.google.com/data/feed/base/featured?alt=rss&kind=photo&access=public&slabel=featured&hl=en_US&imgmax=1024&max-results=10)

Step 5: Declare Download Service



```
<!--  
Fetches network content in a background task and provides it to the content Views.  
-->  
<service  
    android:name=".NetworkDownloadService"  
    android:exported="false" />  
</application>
```

Step 5: Declare Download Service



- In AndroidManifest.xml, before `</application>`:

```
<!--  
Fetches network content in a background task and provides it to the content Views.  
-->  
<service  
    android:name=".NetworkDownloadService"  
    android:exported="false" />  
</application>
```

Step 6: Image Download Task



```
public final class GetImageTask extends AsyncTask<String, Void, Bitmap> {

    @Override
    protected Bitmap doInBackground(String... params) {
        String picUrlStr = params[0];
        URL picUrl;
        InputStream is;
        Bitmap b = null;

        if (TextUtils.isEmpty(picUrlStr)) {
            return null;
        }

        try {
            picUrl = new URL(picUrlStr);
            is = (InputStream) picUrl.getContent();
            b = BitmapFactory.decodeStream(is);
        } catch (MalformedURLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return b;
    }
}
```

Step 6: Image Download Task



- Implement PicDetailFragment.GetImageTask class

```
public final class GetImageTask extends AsyncTask<String, Void, Bitmap> {  
  
    @Override  
    protected Bitmap doInBackground(String... params) {  
        String picUrlStr = params[0];  
        URL picUrl;  
        InputStream is;  
        Bitmap b = null;  
  
        if (TextUtils.isEmpty(picUrlStr)) {  
            return null;  
        }  
  
        try {  
            picUrl = new URL(picUrlStr);  
            is = (InputStream) picUrl.getContent();  
            b = BitmapFactory.decodeStream(is);  
        } catch (MalformedURLException e) {  
            // TODO Auto-generated catch block  
            e.printStackTrace();  
        } catch (IOException e) {  
            // TODO Auto-generated catch block  
            e.printStackTrace();  
        }  
        return b;  
    }  
}
```


Step 7: Use Image Download Task



```
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    View rootView = inflater.inflate(R.layout.fragment_pic_detail, container, false);
    mImageView = (ImageView) rootView.findViewById(R.id.pic_detail_pic);

    if (mItem != null) {
        if (mImageView != null) {
            Bitmap b = null;
            GetImageTask git = new GetImageTask();

            // The documentation implies that ImageView.setImageBitmap
            // does not run on the UI thread, so we don't need to do this in
            // an AsyncTask.
            try {
                b = git.execute(mItem.content).get();
                mImageView.setImageBitmap(b);
            } catch (Exception e) {
                // Handle exception
            }
        }
    }
    ...
}
```

Step 7: Use Image Download Task



- Use PicDetailFragment.GetImageTask class

```
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    View rootView = inflater.inflate(R.layout.fragment_pic_detail, container, false);
    mImageView = (ImageView) rootView.findViewById(R.id.pic_detail_pic);

    if (mItem != null) {
        if (mImageView != null) {
            Bitmap b = null;
            GetImageTask git = new GetImageTask();

            // The documentation implies that ImageView.setImageBitmap
            // does not run on the UI thread, so we don't need to do this in
            // an AsyncTask.
            try {
                b = git.execute(mItem.content).get();
                mImageView.setImageBitmap(b);
            } catch (Exception e) {
                // Handle exception
            }
        }
    }
    ...
}
```

Step 7: Broadcast Receiver



```
// Set up the broadcast mReceiver so that the ArrayAdapter can update.
IntentFilter filter = new IntentFilter();
filter.addAction(DummyContent.ACTION_UPDATE);

mReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        @SuppressWarnings("unchecked")
        ArrayAdapter<DummyContent.DummyItem> aa = (ArrayAdapter<DummyItem>) getListAdapter();
        aa.notifyDataSetChanged();
    }
};
getActivity().registerReceiver(mReceiver, filter);
```

```
public void onDestroy() {
    getActivity().unregisterReceiver(mReceiver);
    super.onDestroy();
}
```

Step 7: Broadcast Receiver



- Instantiate a BroadcastReceiver in PicListFragment.onCreate()

```
// Set up the broadcast mReceiver so that the ArrayAdapter can update.
IntentFilter filter = new IntentFilter();
filter.addAction(DummyContent.ACTION_UPDATE);

mReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        @SuppressWarnings("unchecked")
        ArrayAdapter<DummyContent.DummyItem> aa = (ArrayAdapter<DummyItem>) getListAdapter();
        aa.notifyDataSetChanged();
    }
};
getActivity().registerReceiver(mReceiver, filter);
```

- Unregister it in onDestroy()

```
public void onDestroy() {
    getActivity().unregisterReceiver(mReceiver);
    super.onDestroy();
}
```

Step 8a: Broadcast Content Updates



```
Intent intent = new Intent();  
intent.setAction(DummyContent.ACTION_UPDATE);  
sendBroadcast(intent);
```

```
public static void Init(Context context) {  
    if (!initialized) {  
        Intent initIntent = new Intent(context, NetworkDownloadService.class);  
        Uri localUri = Uri.parse(Constants.PICASA_RSS_URL);  
        initIntent.setData(localUri);  
        context.startService(initIntent);  
        initialized = true;  
    }  
}
```

Step 8a: Broadcast Content Updates



- Broadcast updates from NetworkDownloadService

```
Intent intent = new Intent();  
intent.setAction(DummyContent.ACTION_UPDATE);  
sendBroadcast(intent);
```

- Invoke NetworkDownloadService from DummyContent

```
public static void Init(Context context) {  
    if (!initialized) {  
        Intent initIntent = new Intent(context, NetworkDownloadService.class);  
        Uri localUri = Uri.parse(Constants.PICASA_RSS_URL);  
        initIntent.setData(localUri);  
        context.startService(initIntent);  
        initialized = true;  
    }  
}
```

Step 8b: Broadcast Content Updates



```
public static final String ACTION_UPDATE = "com.example.acamp.fivewhat.dummy.update";
```

Step 8b: Broadcast Content Updates



- Define Intent action in DummyContent

```
public static final String ACTION_UPDATE = "com.example.acamp.fivewhat.dummy.update";
```




Part 3: Providing Locally

What is a Content Provider?

- Interface
- For sharing content
- Often backed by SQL



SQLite Database



SQLiteOpenHelper



```
@Override
public void onCreate(SQLiteDatabase paramSQLiteDatabase) {
}

@Override
public void onUpgrade(SQLiteDatabase paramSQLiteDatabase,
    int paramInt1, int paramInt2) {
}
```

Why do I need SQL Open Helper?



- Helps manage lifecycle of SQL Database
- Off UI thread

Declaring a Content Provider



In AndroidManifest.xml:

```
<provider
    android:name="sample.multithreading.PicasaContentDB"
    android:exported="false"
    android:authorities="@string/picasa_authority"
/>
```

In values/strings.xml:

```
<resources>
    <string name="picasa_authority"
translatable="false">sample.multithreading.PicasaDB</string>
    <string name="picasa_faves_table" translatable="false">faves</
string>
```

Why do I need a Content Provider?



- Share across apps
- Standard interface
- Contract



Code Lab 4: Provide Data

Step 1: Define the Authority



In res/values/strings.xml:

```
<string name="picasa_authority" translatable="false">com.example.acamp.authority.PicasaProvider</string>  
<string name="picasa_faves_table" translatable="false">faves</string>  
<string name="application_data_table" translatable="false">data</string>
```

Step 2: Declare the Content Provider



In AndroidManifest.xml, before `</application>`:

```
<provider  
    android:name=".PicasaProvider"  
    android:exported="false"  
    android:authorities="@string/picasa_authority"  
</>
```

Step 3: Implement the Content Provider



Copy PicasaProvider.java from
new_source_files into your package directory.

Step 4: Fill the Content Provider



In NetworkDownloadService:

```
// TODO: Uncomment to populate ContentProvider
    Vector<ContentValues> cvv = localPicasaPullParser
        .getImages();

    int size = cvv.size();
    ContentValues[] cvArray = new ContentValues[size];
    cvArray = cvv.toArray(cvArray);

    getContentResolver().bulkInsert(
        PicasaProvider.getUriByType(this,
            PicasaProvider.METADATA_QUERY), cvArray);

    break;
```

Step 5: Instantiate getImages()



In PicasaPullParser:

```
private static ContentValues mImage;  
private Vector<ContentValues> mImages;  
  
public Vector<ContentValues> getImages() {  
    return this.mImages;  
}
```

Also uncomment all the lines using mImage and mImages.

Step 6a: Move Constants to Constants



In Constants:

```
private static final int NUM_LINES = 10;
public static final String PICASA_RSS_URL = "http://picasaweb.google.com/data/
feed/base/featured?
alt=rss&kind=photo&access=public&slabel=featured&hl=en\_US&imgmax=1024&max-
results=10";
public static final String PREFS_GROUP = "FiveDpPrefs";
public static final String LAST_ITEM_KEY = "ItemId";
```

Step 6b: Move Constants to Constants



In PicListActivity:

Reference PREFS_GROUP and LAST_ITEM_KEY via Constants.

Step 7: Change id to long



In PicListFragment:

Change onItemSelected(String id) to (long id)

In PicListActivity:

Change numItemSelected from String to long

Change all occurrences of String id to long id

```
public void onListItemClick(ListView listView, View view, int position, long id) {  
    super.onListItemClick(listView, view, position, id);  
    mCallbacks.onItemSelected(id);  
}
```


Step 8: Get rid of BroadcastReceiver



In PicListFragment:

Delete the BroadcastReceiver and its associated code.

In DummyContent:

Delete ACTION_UPDATE.

Step 9: Implement Cursor Adapter



In PicListFragment:

```
public class PicListFragment extends ListFragment implements
LoaderManager.LoaderCallbacks<Cursor> {
    SimpleCursorAdapter mAdapter;
    ...
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Create an empty adapter we will use to display the loaded data.
        mAdapter = new SimpleCursorAdapter(
            getActivity(),
            android.R.layout.simple_list_item_activated_1,
            null,
            new String[] { PicasaProvider.PicasaFeatured._ID },
            new int[] { android.R.id.text1 },
            0
        );
        setListAdapter(mAdapter);

        // Prepare the loader. Either re-connect with an existing one,
        // or start a new one.
        getLoaderManager().initLoader(0, null, this);
    }
}
```

Step 10: Implement Loader Callbacks



At the bottom of PicListFragment:

```
@Override
public Loader<Cursor> onCreateLoader(int loaderId, Bundle bundle) {
    CursorLoader cl = new CursorLoader(getActivity());
    cl.setUri(PicasaProvider.getUriByType(getActivity(), PicasaProvider.METADATA_QUERY));
    cl.setProjection(PicasaProvider.PicasaFeatured.PROJ_LONG_URL);
    return cl;
}

@Override
public void onLoadFinished(Loader<Cursor> loader, Cursor cursor) {
    mAdapter.swapCursor(cursor);
    setListShown(true);
}

@Override
public void onLoaderReset(Loader<Cursor> loader) {
    mAdapter.swapCursor(null);
}
```

Step 11: Replace mItem with mId



In PicDetailFragment:

Delete DummyContent.DummyItem mItem;
Add long mId = -1;

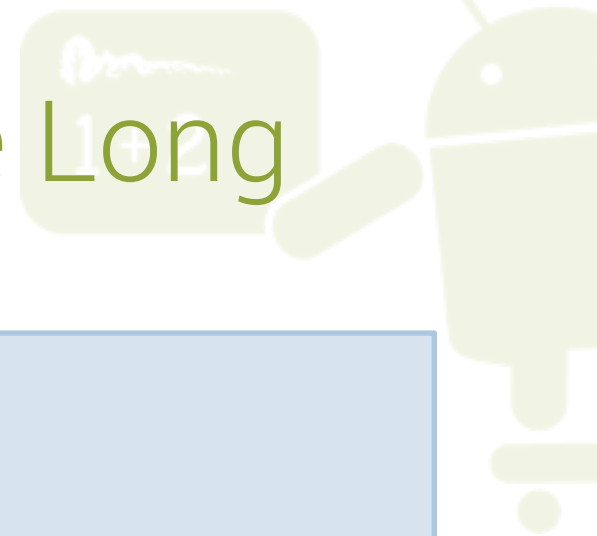
```
long mId = -1;

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if (getArguments().containsKey(ARG_ITEM_ID)) {
        mId = getArguments().getLong(ARG_ITEM_ID, -1);
    } else {
        SharedPreferences sp = getActivity().getSharedPreferences(Constants.PREFS_GROUP, Context.MODE_PRIVATE);
        long lastItemKey = sp.getLong(Constants.LAST_ITEM_KEY, -1);
        if (lastItemKey > 0) {
            mId = lastItemKey;
        }
    }
}

...

try {
    b = git.execute(mId).get();
    mImageView.setImageBitmap(b);
}
```

Step 12: Convert GetImageTask to Take Long



```
public final class GetImageTask extends AsyncTask<Long, Void, Bitmap> {

    @Override
    protected Bitmap doInBackground(Long... params) {
        String picUrlStr = null;
        URL picUrl;
        InputStream is;
        Bitmap b = null;

        // Inherit context and content resolver from containing Activity.
        Context context = getActivity();
        ContentResolver cr = context.getContentResolver();

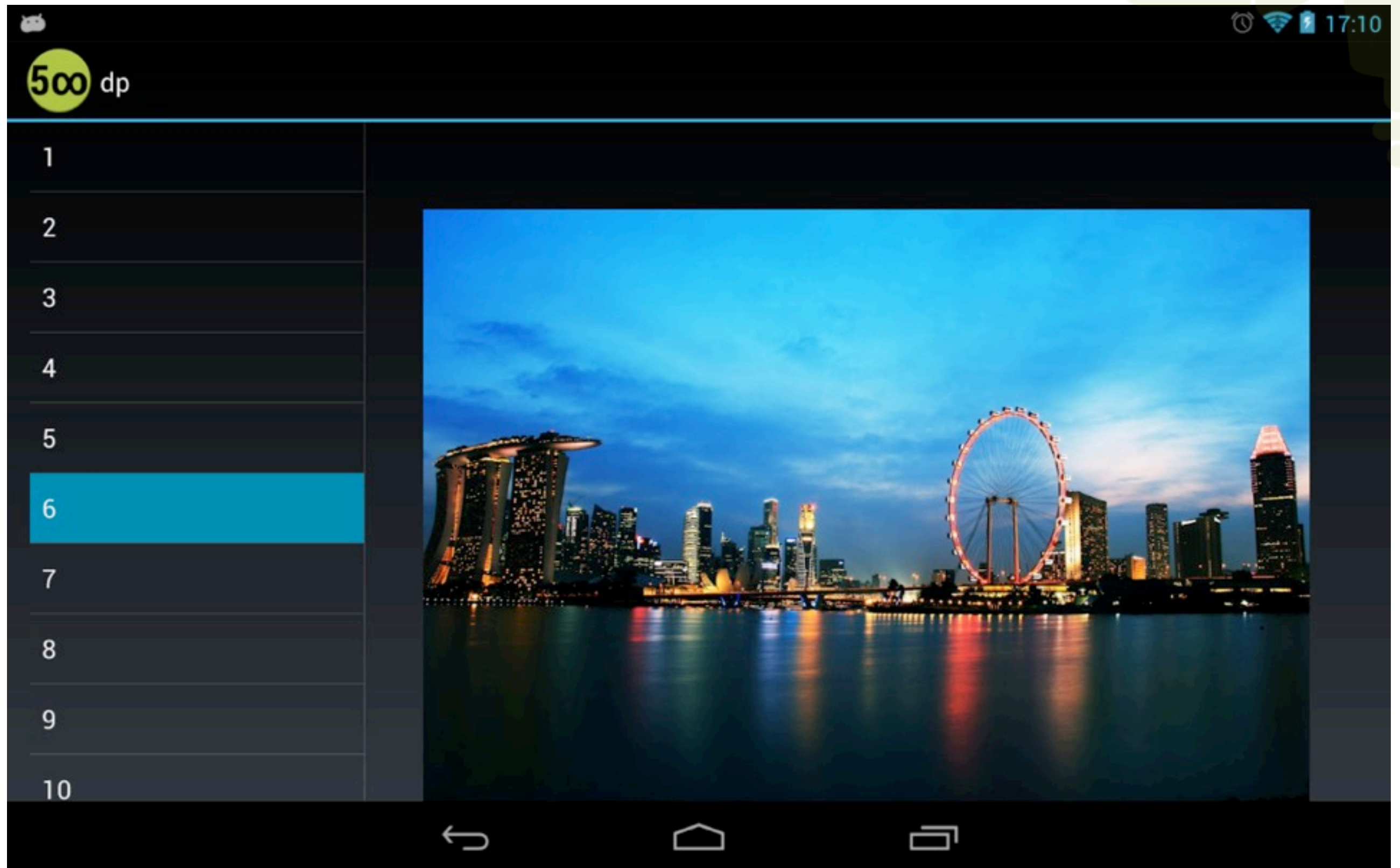
        // Compose a content URI and access the content resolver, specifically
        // seeking the row matching the ID passed in through params.
        Uri singleUri = ContentUris.withAppendedId(PicasaProvider.getUriByType(context, PicasaProvider.METADATA_QUERY),
params[0]);
        Cursor cursor = cr.query(singleUri, PicasaProvider.PicasaFeatured.PROJ_LONG_URL, null, null, null);

        // Fetch the content out of the result cursor.
        // NOTE! Do not just use magic numbers to get the
        // columns. Better to look up by name.
        if ((cursor != null) && (cursor.getCount() > 0)) {
            cursor.moveToFirst();
            picUrlStr = cursor.getString(cursor.getColumnIndex(PicasaFeatured.IMAGE_URL));
        }
        ...
    }
}
```

... and the remainder as before.

CONGRATULATIONS!

1+2



You Made an App With



- Dual-Fragment layout
- Icon in 4 pixel densities
- ListView, TextView, ImageView
- URL to InputStream
- InputStream to Drawable
- Reads files from Assets
- Shared Preferences
- Downloads using HttpURLConnection
- Parses XML using XmlPullParser
- AsyncTask
- IntentService
- BroadcastReceiver
- SQL Database
- ContentProvider
- CursorLoader
- ... and a whole lot of persistence!

Code Lab 3 Extra Credit Ideas

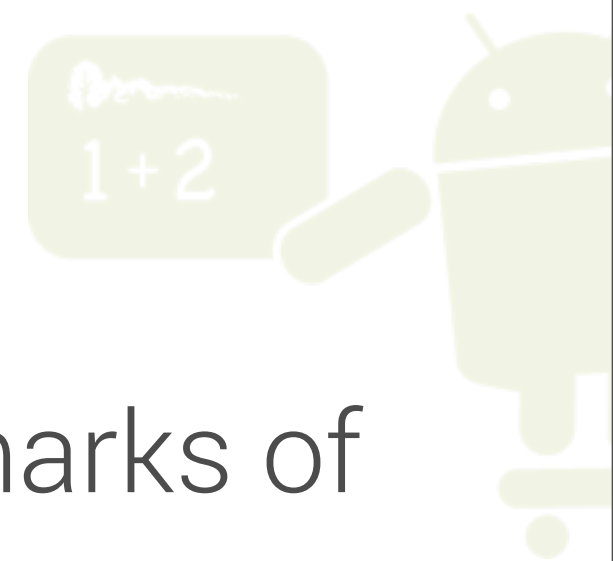


- Scheduled fetching
- Incremental fetching
- Caching
- Push notification



Let's discuss!

Copyrights and Trademarks



- Android, Google are registered trademarks of Google Inc.
- All other trademarks and copyrights are the property of their respective owners.