Android Programming Pushing the Limits

Erik Hellman

google.com/+ErikHellman

Spotify

February 3, 2014

“Write a book. It’ll be great!”

Two months later…

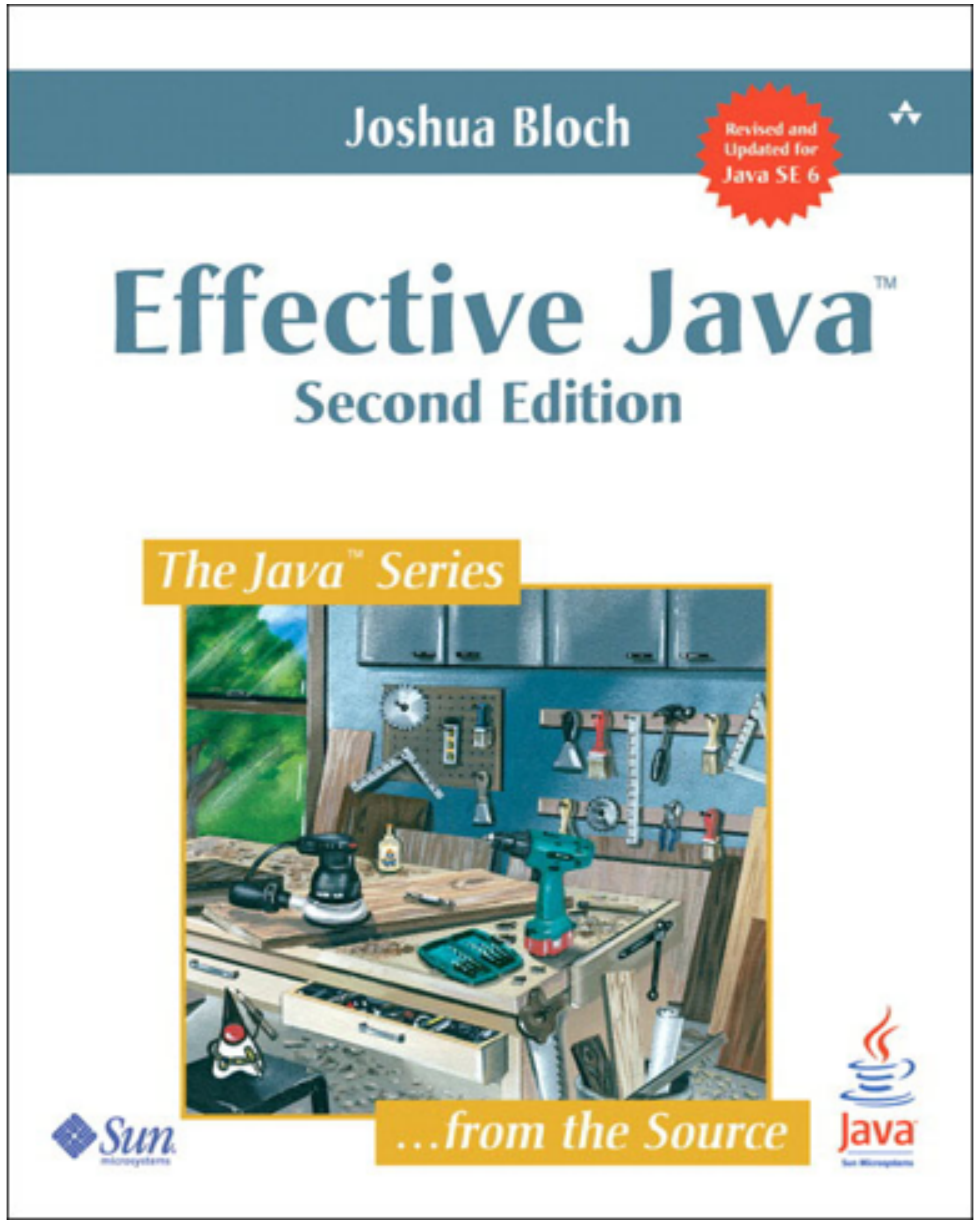
Finally…

www.wiley.com/go/ptl/androidprogramming

ANDROID
studio

Effective Java for Android

The only other book you'll need…



Static factory method (Item 1)

```
public final class MyData {
    private MyData next;
    private static final Object sPoolSync = new Object();
    private static MyData sPool;

    private static int sPoolSize = 0;
    private static final int MAX_POOL_SIZE = 50;

    private MyData() {
    }

    public static MyData obtain() {
        synchronized (sPoolSync) {
            if (sPool != null) {
                MyData data = sPool;
                sPool = data.next;
                data.next = null;
                sPoolSize--;
                return data;
            }
        }
        return new MyData();
    }
```

```
    public void recycle() {
        synchronized (sPoolSync) {
            f(sPoolSize < MAX_POOL_SIZE) {
                next = sPool;
                sPool = this;
                sPoolSize++;
            }
        }
    }

}
```

Rotation gesture

```
private static double angle(MotionEvent event) {
    MotionEvent.PointerCoords[] coords = new MotionEvent.PointerCoords[]
            {new MotionEvent.PointerCoords(), new MotionEvent.PointerCoords()};

    event.getPointerCoords(0, coords[0]);
    event.getPointerCoords(1, coords[1]);
    double deltaX = (event.getX(0) - event.getX(1));
    double deltaY = (event.getY(0) - event.getY(1));

    return Math.atan2(deltaX, deltaY);
}

private static double clamp(double value, double min, double max) {
    if (value < min) {
        return min;
    } else if (value > max) {
        return max;
    } else {
        return value;
    }
}
```

Rotation gesture

```
private Double mPreviousAngle = null;
private double mRotation = 0;

public boolean onTouch(View v, MotionEvent event) {
    if (event.getPointerCount() == 2) {
        double currentAngle = angle(event);
        if (mPreviousAngle != null) {
            mRotation -= Math.toDegrees(clamp(mPreviousAngle - currentAngle,
                                              -MAX_ANGLE, MAX_ANGLE));
        }
        mPreviousAngle = currentAngle;
    } else {
        mPreviousAngle = null;
    }
}
```

Playing nice with the main thread

```
public void onFrequentEventHappend(Bundle data) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            // handle event...
        }
    });
}
```

Playing nice with the main thread

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mMainHandler = new Handler(getMainLooper(), this);
}

@Override
public boolean handleMessage(Message msg) {
    switch (msg.what) {
        // Will run on the main thread!
        case HANDLE_EVENT:
            handleEventOnMainThread((Bundle) msg.obj);
            break;
    }
    return true;
}

public void onFrequentEventHappend(Bundle data) {
    mMainHandler.obtainMessage(HANDLE_EVENT, 0, 0, data).sendToTarget();
}

private void handleEventOnMainThread(Bundle data) {
    // Handle event on main thread
}
```

Enabling and disabling components

```
<activity
    android:name="se.hellsoft.jfokusdemos.FreeAppActivity"
    android:enabled="true"
    android:label=“@string/app_name_light" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name="se.hellsoft.jfokusdemos.PremiumAppActivity"
    android:enabled="false"
    android:label=“@string/app_name_premium” >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Enabling and disabling components

```
private void toggleComponentEnabled(Context, ComponentName componentName) {
    PackageManager packageManager = getPackageManager();

    int state = packageManager.getComponentEnabledSetting(componentName);
    int newState = state == PackageManager.COMPONENT_ENABLED_STATE_ENABLED
            ? PackageManager.COMPONENT_ENABLED_STATE_DISABLED
            : PackageManager.COMPONENT_ENABLED_STATE_ENABLED;

    packageManager.setComponentEnabledSetting(componentName, newState, 0);
}
```

Auto-starting the right way

```
<receiver
    android:name="se.hellsoft.jfokusdemos.MyReceiver"
    android:enabled="false"
    android:exported="true" >
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
    </intent-filter>
</receiver>
```

Auto-starting the right way

```
<receiver
    android:name="se.hellsoft.jfokusdemos.MyReceiver"
    android:enabled="false"
    android:exported="true" >
    <intent-filter>
        <action android:name="android.intent.action.USER_PRESENT" />
    </intent-filter>
</receiver>
```

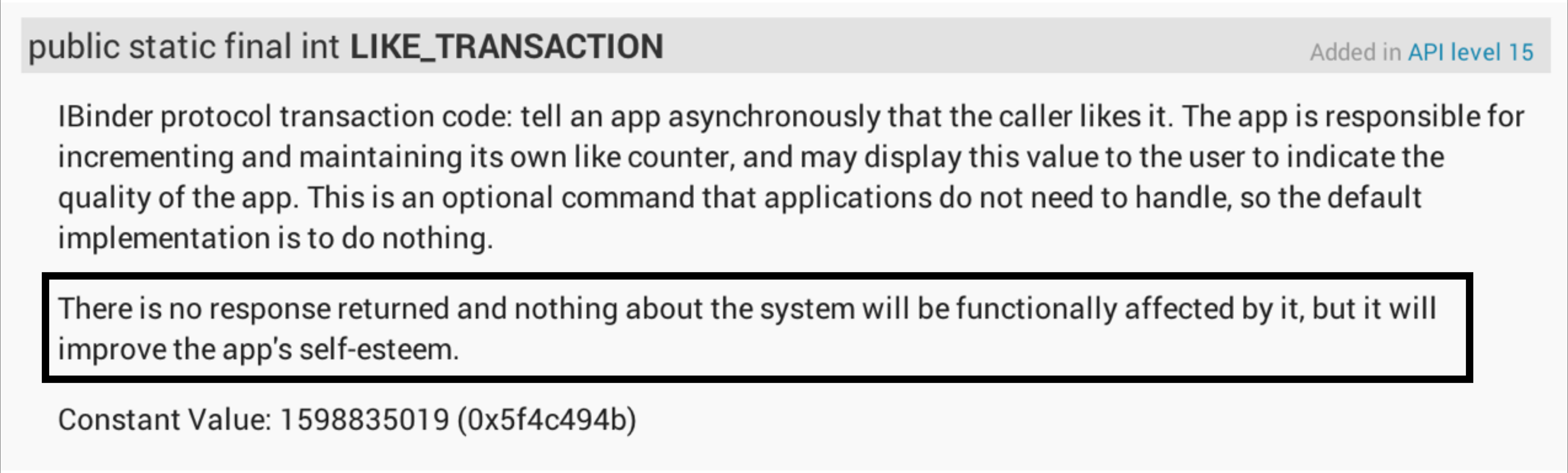
Custom Binder operations

```
public String performCustomBinderTransacttion(IBinder binder, String arg0,
                                              int arg1, float arg2) throws RemoteException {
    Parcel request = Parcel.obtain();
    Parcel response = Parcel.obtain();

    // Populate request data...
    request.writeString(arg0);
    request.writeInt(arg1);
    request.writeFloat(arg2);

    // Perform transaction
    binder.transact(IBinder.FIRST_CALL_TRANSACTION, request, response, 0);

    // Read the result from the response…
    String result = response.readString();

    // Recycle the objects
    request.recycle();
    response.recycle();

    return result;
}
```

Custom Binder operations

```
public class CustomBinder extends Binder {
 @Override
 protected boolean onTransact(int code, Parcel request, Parcel response,
                              int flags) throws RemoteException {
        // Read the data in the request
        String arg0 = request.readString();
        int arg1 = request.readInt();
        float arg2 = request.readFloat();

        String result = buildResult(arg0, arg1, arg2);

        // Write the result to the response Parcel
        response.writeString(result);

        // Return true on success
        return true;
    }

    private String buildResult(String arg0, int arg1, float arg2) {
        String result = null;
        // TODO Build the result
        return result;
    }
}
```

…and everyone should be nice to their apps…



public static final int **LIKE_TRANSACTION**

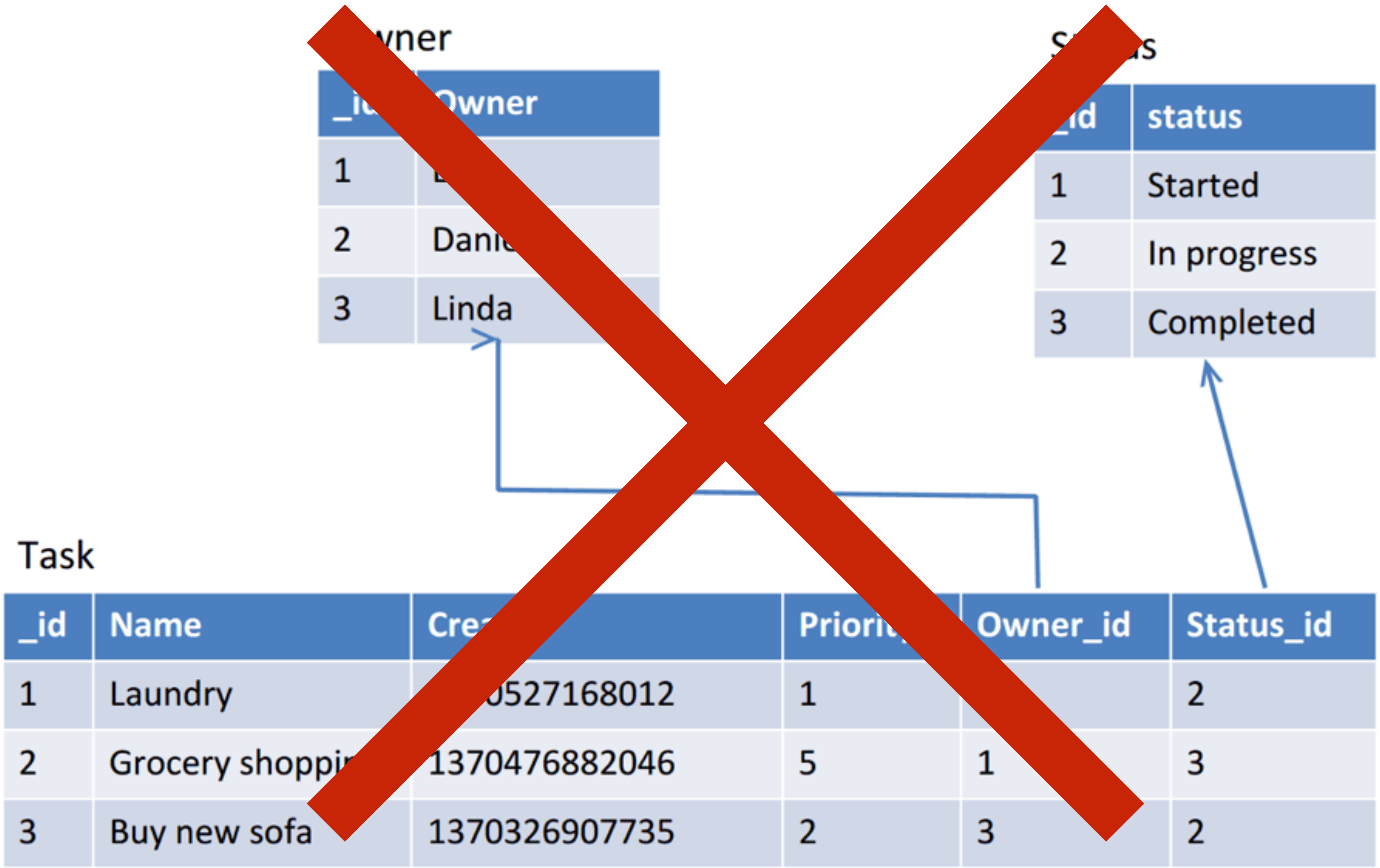
Added in API level 15

IBinder protocol transaction code: tell an app asynchronously that the caller likes it. The app is responsible for incrementing and maintaining its own like counter, and may display this value to the user to indicate the quality of the app. This is an optional command that applications do not need to handle, so the default implementation is to do nothing.

There is no response returned and nothing about the system will be functionally affected by it, but it will improve the app's self-esteem.

Constant Value: 1598835019 (0x5f4c494b)

Database design for apps



Owner

_id	Owner
1	L[illegible]
2	Dani[illegible]
3	Linda

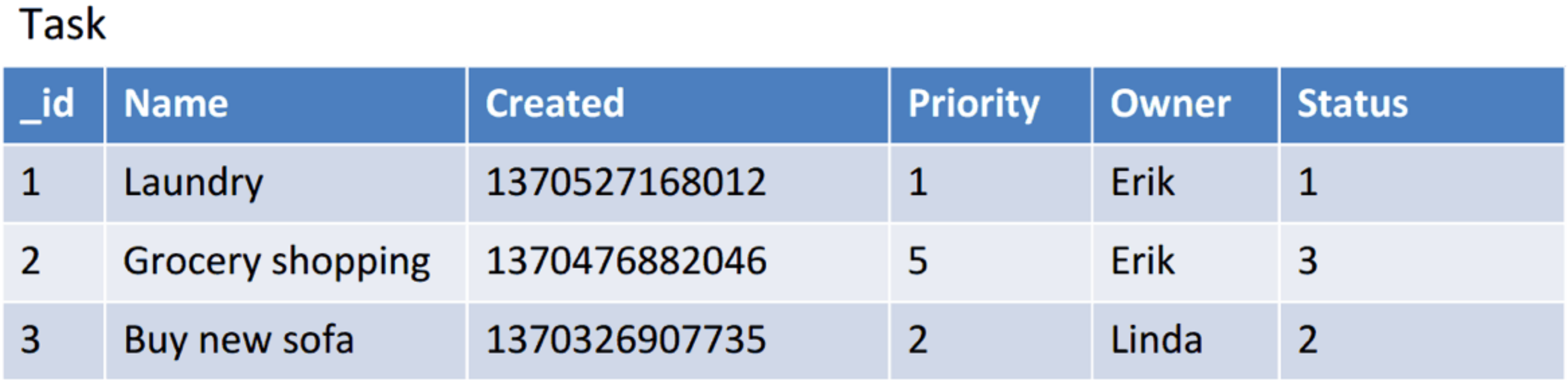
Status

_id	status
1	Started
2	In progress
3	Completed

Task

_id	Name	Created	Priority	Owner_id	Status_id
1	Laundry	[illegible]0527168012	1	[illegible]	2
2	Grocery shoppi[illegible]	1370476882046	5	1	3
3	Buy new sofa	1370326907735	2	3	2

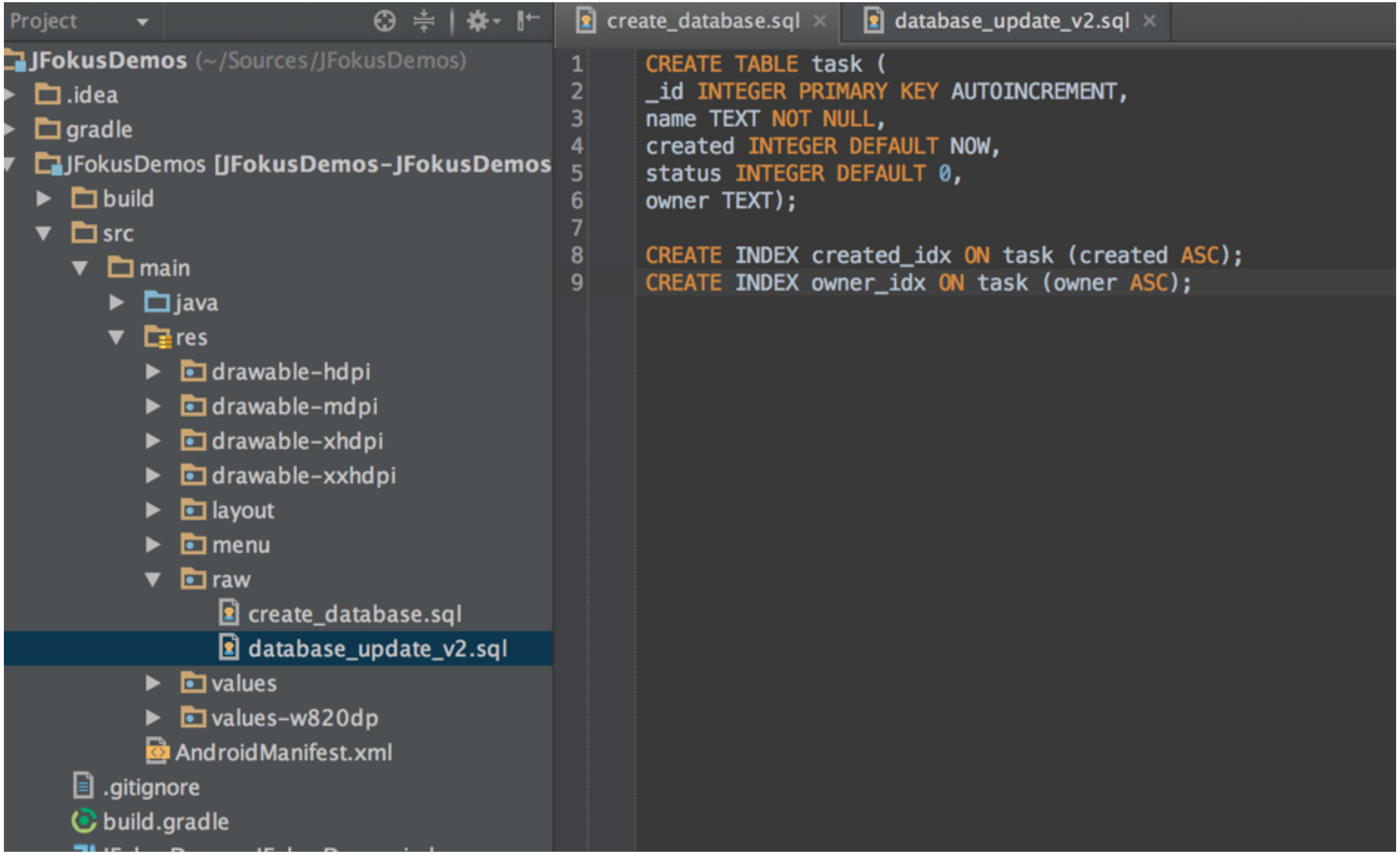
Database design for apps



Task

_id	Name	Created	Priority	Owner	Status
1	Laundry	1370527168012	1	Erik	1
2	Grocery shopping	1370476882046	5	Erik	3
3	Buy new sofa	1370326907735	2	Linda	2

Managing SQL statements



```sql
CREATE TABLE task (
_id INTEGER PRIMARY KEY AUTOINCREMENT,
name TEXT NOT NULL,
created INTEGER DEFAULT NOW,
status INTEGER DEFAULT 0,
owner TEXT);

CREATE INDEX created_idx ON task (created ASC);
CREATE INDEX owner_idx ON task (owner ASC);
```

Database transactions

```
private Uri doInsert(Uri uri, ContentValues values, SQLiteDatabase database) {
    Uri result = null;

    switch (mUriMatcher.match(uri)) {
        case ALL_TASKS:
            long id = database.insert(TASK_TABLE, "", values);
            if (id == –1)
                throw new SQLException("Error inserting data!");

            result = Uri.withAppendedPath(uri, String.valueOf(id));
    }

    return result;
}

@Override
public Uri insert(Uri uri, ContentValues values) {
    SQLiteDatabase database = mOpenHelper.getWritableDatabase();
    Uri result = doInsert(uri, values, database);
    return result;
}
```

Database transactions

```
@Override
public int bulkInsert(Uri uri, ContentValues[] contentValueses) {
    SQLiteDatabase database = mOpenHelper.getWritableDatabase();
    int count = 0;

    try {
        database.beginTransaction();
        for (ContentValues values : contentValueses) {
            Uri resultUri = doInsert(uri, values, database);
            if (resultUri != null) {
                count++;
            } else {
                count = 0;
                throw new SQLException("Error in bulk insert");
            }
        }
        database.setTransactionSuccessful();
    } finally {
        database.endTransaction();
    }

    return count;
}
```

Database transactions

```
@Override
public ContentProviderResult[] applyBatch(ArrayList<ContentProviderOperation> operations)
        throws OperationApplicationException {
    SQLiteDatabase database = mOpenHelper.getWritableDatabase();
    ContentProviderResult[] result = new ContentProviderResult[operations.size()];

    try {
        database.beginTransaction();
        for (int i = 0; i < operations.size(); i++) {
            ContentProviderOperation operation = operations.get(i);
            result[i] = operation.apply(this, result, i);
        }
        database.setTransactionSuccessful();
    } finally {
        database.endTransaction();
    }

    return result;
}
```

Database transactions

```
public void deleteWithTransactions(Context context, int[] ids) {
    ArrayList<ContentProviderOperation> operations
            = new ArrayList<ContentProviderOperation>();

    for (int id : ids) {
        ContentProviderOperation operation = ContentProviderOperation.
                newUpdate(Constants.TASK_URI).
                withSelection("_id = ?", new int[] {id}).build();
        operations.add(operation);
    }

    context.getContentResolver().applyBatch(Constant.AUTHORITY, operations);
}
```

Verifying calling applications

```
private String getPackageNameForCaller(Context context) {
    PackageManager packageManager = context.getPackageManager();

    int callingUid = Binder.getCallingUid();
    String[] packages = packageManager.getPackagesForUid(callingUid);

    if(packages != null && packages.length > 0) {
        return packages[0]; // Return the first matching package…
    }

    return null;
}
```

Working with JNI

```
public class NativeSorting {
    static {
        System.loadLibrary("sorting_jni");
    }

    public NativeSorting() { }

    public void sortIntegers(int[] ints) {
        nativeSort(ints);
    }

    private native void nativeSort(int[] ints);
}
```

Working with JNI

```
$ javah -classpath ../../build/classes/release/ -d jni/ \
  com.aptl.jnidemo.NativeSorting
```

Working with JNI

```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include <jni.h>
/* Header for class com_aptl_jnidemo_NativeSorting */

#ifndef _Included_com_aptl_jnidemo_NativeSorting
#define _Included_com_aptl_jnidemo_NativeSorting
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class: com_aptl_jnidemo_NativeSorting
 * Method: nativeSort
 * Signature: ([F)V
 */
JNIEXPORT void JNICALL Java_com_aptl_jnidemo_NativeSorting_nativeSort
 (JNIEnv *, jobject, jintArray);

#ifdef __cplusplus
}

#endif
#endif
```

Working with JNI

```
#include <jni.h>
#include <android/log.h>
#include "com_aptl_jnidemo_NativeSorting.h"

void quicksort(int *arr, int start, int end);

JNIEXPORT jint JNI_OnLoad(JavaVM *vm, void *reserved) {
 return JNI_VERSION_1_6;
}

JNIEXPORT void JNICALL Java_com_aptl_jnidemo_NativeSorting_nativeSort(JNIEnv *env, jobject obj,
                                                                      jintArray data) {
 jint* array = (*env)->GetIntArrayElements(env, data, 0);
 jint length = (*env)->GetArrayLength(env, data);

 quicksort(array, 0, length);

 (*env)->ReleaseIntArrayElements(env, data, array, 0);
}

void quicksort(int *arr, int start, int end)
{
 // Left out as an exercise for your upcoming interview at Spotify...
}
```

Working with JNI

```
public void sortIntegersWithCallback(int[] ints, Callback callback) {
    nativeSortWithCallback(ints, callback);
}

private native void nativeSortWithCallback(int[] ints, Callback callback);

public interface Callback {
    void onSorted(int[] sorted);
}
```

Working with JNI

```
#include <jni.h>
#include <android/log.h>
#include <pthread.h>
#include "com_aptl_jnidemo_NativeSorting.h"

JavaVM *g_vm;

struct thread_args {
 int* data;
 int data_size;
 jobject callback;
};

void quicksort(int *arr, int start, int end);
void background_sorting(void* args);

JNIEXPORT jint JNI_OnLoad(JavaVM *vm, void *reserved) {
 g_vm = vm;
 return JNI_VERSION_1_6;
}
```

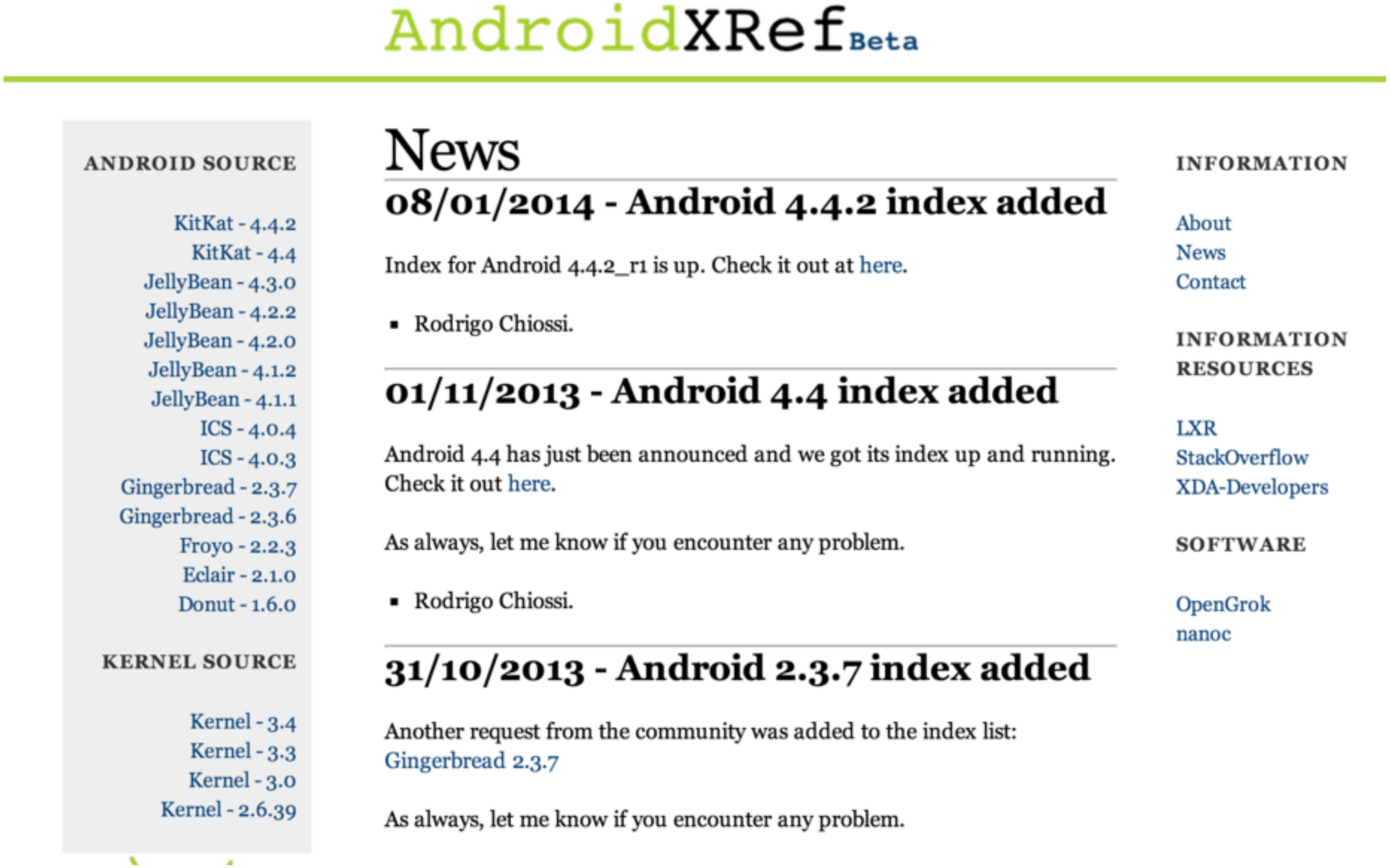
Working with JNI

```
JNIEXPORT void JNICALL
Java_com_aptl_jnidemo_NativeSorting_nativeSortWithCallback
 (JNIEnv *env, jobject obj, jintArray data, jobject callback) {
 jint* array;
 jint length;
 pthread_t thread;

 struct thread_args* myThreadData = malloc(sizeof(struct thread_args));

 array = (*env)->GetIntArrayElements(env, data, 0);
 length = (*env)->GetArrayLength(env, data);

 myThreadData->data = array;
 myThreadData->data_size = length;
 myThreadData->callback = (*env)->NewGlobalRef(env, callback);

 (*env)->ReleaseIntArrayElements(env, data, array, JNI_COMMIT);

 pthread_create(&thread, NULL, (void*)background_sorting, (void*) myThreadData);
}
```

Working with JNI

```
void background_sorting(void* arg) {
 struct thread_args *data = (struct thread_args *) arg;
 JNIEnv* env = NULL;
 jclass callbackClass;
 jmethodID callbackMethodId;
 jintArray result;

 quicksort(data->data, 0, data->data_size);

 (*g_vm)->AttachCurrentThread(g_vm, &env, NULL);

 result = (*env)->NewIntArray(env, data->data_size);
 (*env)->SetIntArrayRegion(env, result, 0, data->data_size, data->data);

 callbackClass = (*env)->GetObjectClass(env, data->callback);
 callbackMethodId = (*env)->GetMethodID(env, callbackClass, "onSorted", "([I)V");
 (*env)->CallVoidMethod(env, data->callback, callbackMethodId, result);

 free(data->data);
 free(data);

 (*env)->DeleteGlobalRef(env, data->callback);
 (*g_vm)->DetachCurrentThread(g_vm);
}
```

Calling the hidden APIs



AndroidXRef Beta

ANDROID SOURCE

- KitKat - 4.4.2
- KitKat - 4.4
- JellyBean - 4.3.0
- JellyBean - 4.2.2
- JellyBean - 4.2.0
- JellyBean - 4.1.2
- JellyBean - 4.1.1
- ICS - 4.0.4
- ICS - 4.0.3
- Gingerbread - 2.3.7
- Gingerbread - 2.3.6
- Froyo - 2.2.3
- Eclair - 2.1.0
- Donut - 1.6.0

KERNEL SOURCE

- Kernel - 3.4
- Kernel - 3.3
- Kernel - 3.0
- Kernel - 2.6.39

News

08/01/2014 - Android 4.4.2 index added

Index for Android 4.4.2_r1 is up. Check it out at here.

- Rodrigo Chiossi.

01/11/2013 - Android 4.4 index added

Android 4.4 has just been announced and we got its index up and running. Check it out here.

As always, let me know if you encounter any problem.

- Rodrigo Chiossi.

31/10/2013 - Android 2.3.7 index added

Another request from the community was added to the index list: Gingerbread 2.3.7

As always, let me know if you encounter any problem.

INFORMATION

- About
- News
- Contact

INFORMATION RESOURCES

- LXR
- StackOverflow
- XDA-Developers

SOFTWARE

- OpenGrok
- nanoc

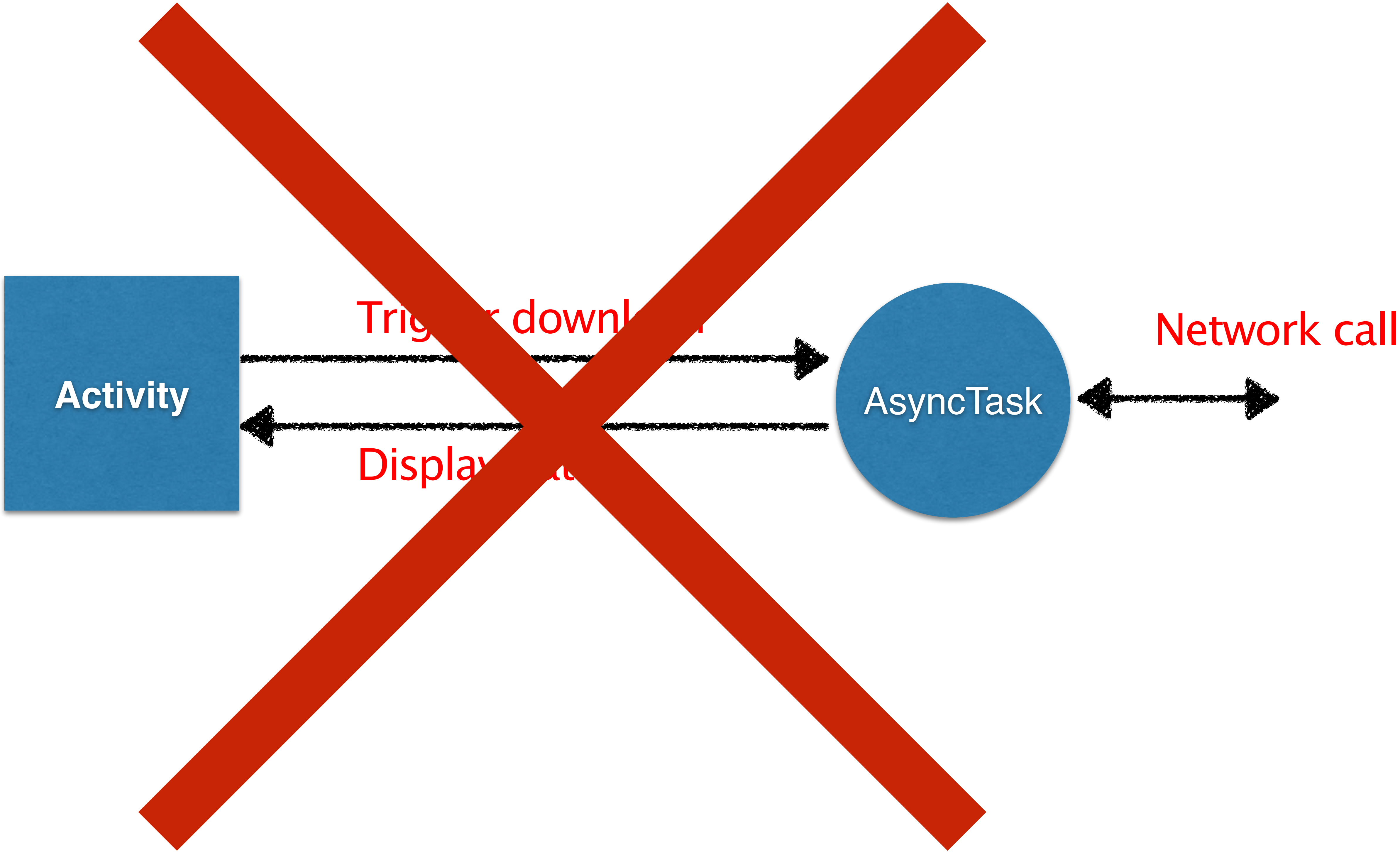
Calling the hidden APIs

```
public static Method getWifiAPMethod(WifiManager wifiManager) {
    try {
        Class clazz = wifiManager.getClass();
        return clazz.getMethod("isWifiApEnabled");
    } catch (NoSuchMethodException e) {
        throw new RuntimeException(e);
    }
}

public static boolean invokeIsWifiAPEnabled(WifiManager wifiManager,
                                            Method isWifiApEnabledMethod) {
    try {
        return (Boolean) isWifiApEnabledMethod.invoke(wifiManager);
    } catch (IllegalAccessException e) {
        throw new RuntimeException(e);
    } catch (InvocationTargetException e) {
        throw new RuntimeException(e);
    }
}
```

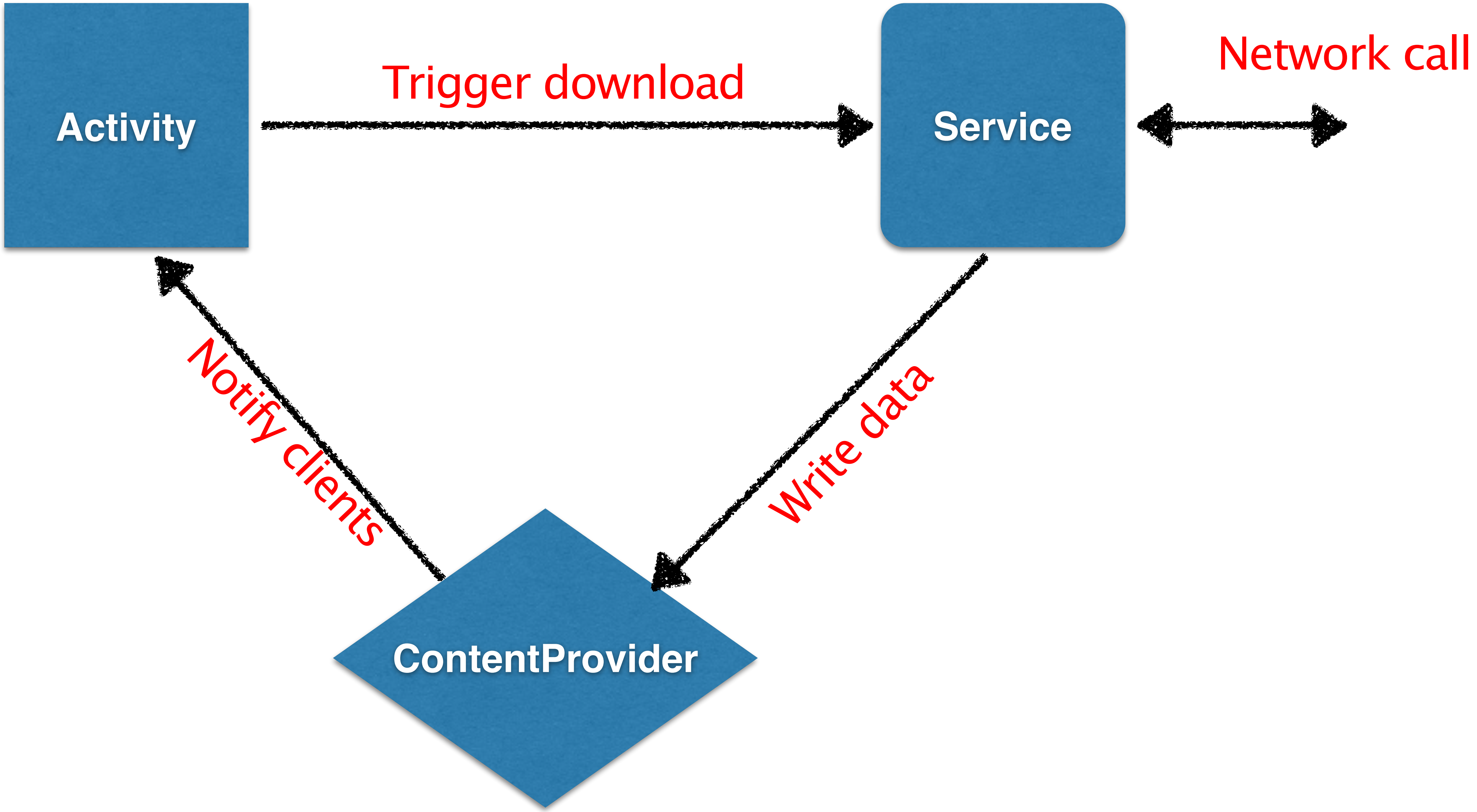
Calling the hidden APIs

```
private static WifiConfiguration getWifiApConfig(Context context) {
    WifiConfiguration wifiConfiguration = null;
    try {
        WifiManager wifiManager =
                (WifiManager) context.getSystemService(Context.WIFI_SERVICE);

        Class clazz = WifiManager.class;
        Method getWifiApConfigurationMethod = clazz.getMethod("getWifiApConfiguration");

        return (WifiConfiguration) getWifiApConfigurationMethod.invoke(wifiManager);
    } catch (NoSuchMethodException e) {
        Log.e(TAG, "Cannot find method", e);
    } catch (IllegalAccessException e) {
        Log.e(TAG, "Cannot call method", e);
    } catch (InvocationTargetException e) {
        Log.e(TAG, "Cannot call method", e);
    }
    return wifiConfiguration;
}
```

Efficient networking on Android



Efficient networking on Android



Poll or push?

```
public void scheduleNetworkPolling() {
    AlarmManager alarmManager = (AlarmManager) getSystemService(ALARM_SERVICE);

    int alarmType = AlarmManager.ELAPSED_REALTIME;
    long interval = AlarmManager.INTERVAL_FIFTEEN_MINUTES;
    long start = System.currentTimeMillis() + interval;

    Intent pollIntent = new Intent(MyNetworkService.ACTION_PERFORM_POLL);

    PendingIntent pendingIntent = PendingIntent.getService(this, 0, pollIntent, 0);
    alarmManager.setInexactRepeating(alarmType, start, interval, pendingIntent);
}
```

www.wiley.com/go/ptl/androidprogramming

Want to join the band?

Check out http://www.spotify.com/jobs
or @Spotifyjobs for more information.