



Google
Developers



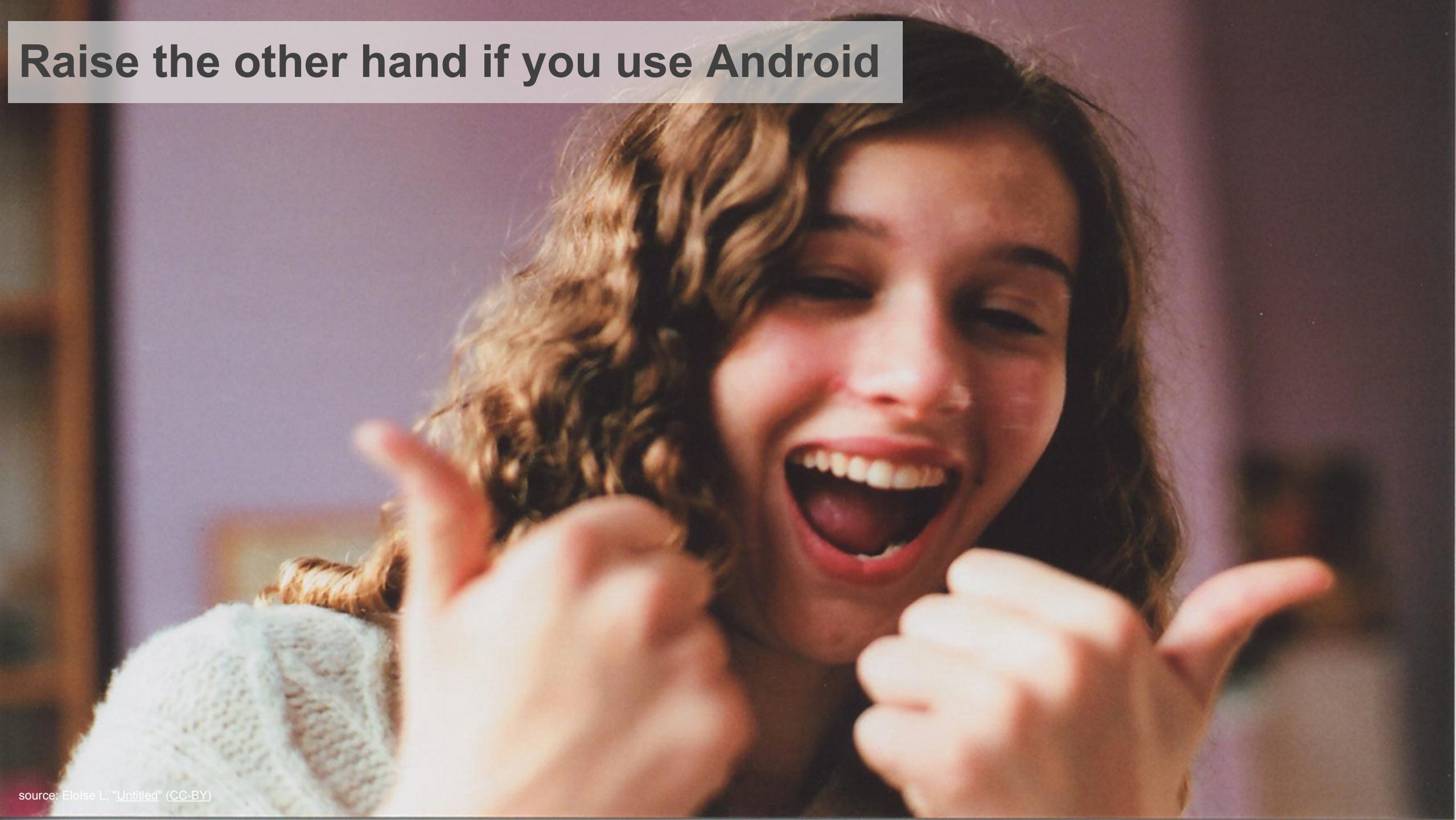
Writing Efficient Drive Apps for Android

Claudio Cherubino / Alain Vongsouvanh
Google Drive Developer Relations

Raise your hand if you use Google Drive



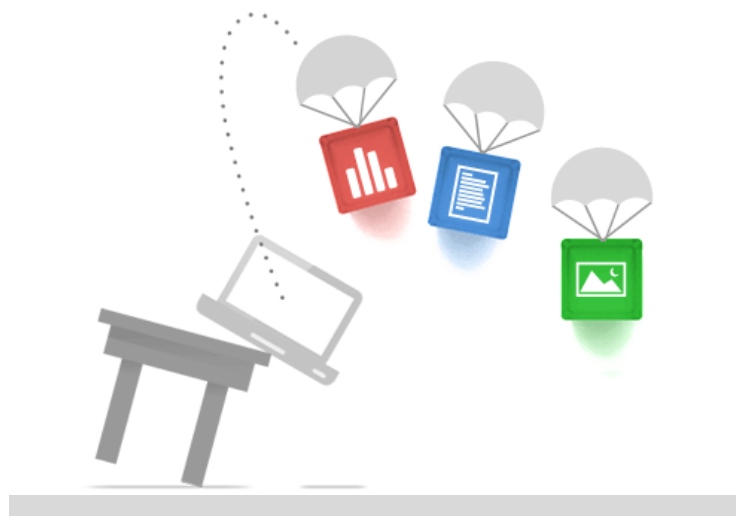
Raise the other hand if you use Android



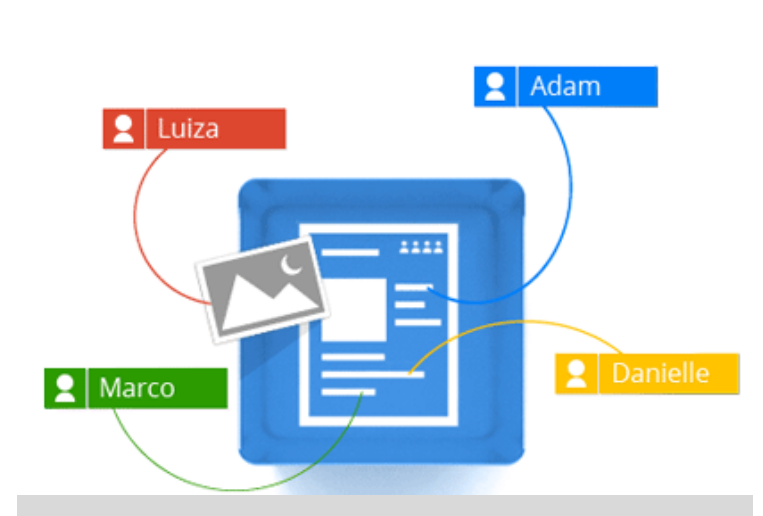
Google Drive



Access everywhere



Store files in a safe place



Go beyond storage. Collaborate.



Google Drive in your browser

The screenshot shows a web browser window with the address bar displaying `https://drive.google.com/?tab=mo&authuser=0#search/Grand+Canyon`. The Google search bar contains the text "Grand Canyon". The Drive interface includes a sidebar with "My Drive", "Shared with me", "Starred", "Recent", and "More" options. The main content area displays search results for "Grand Canyon".

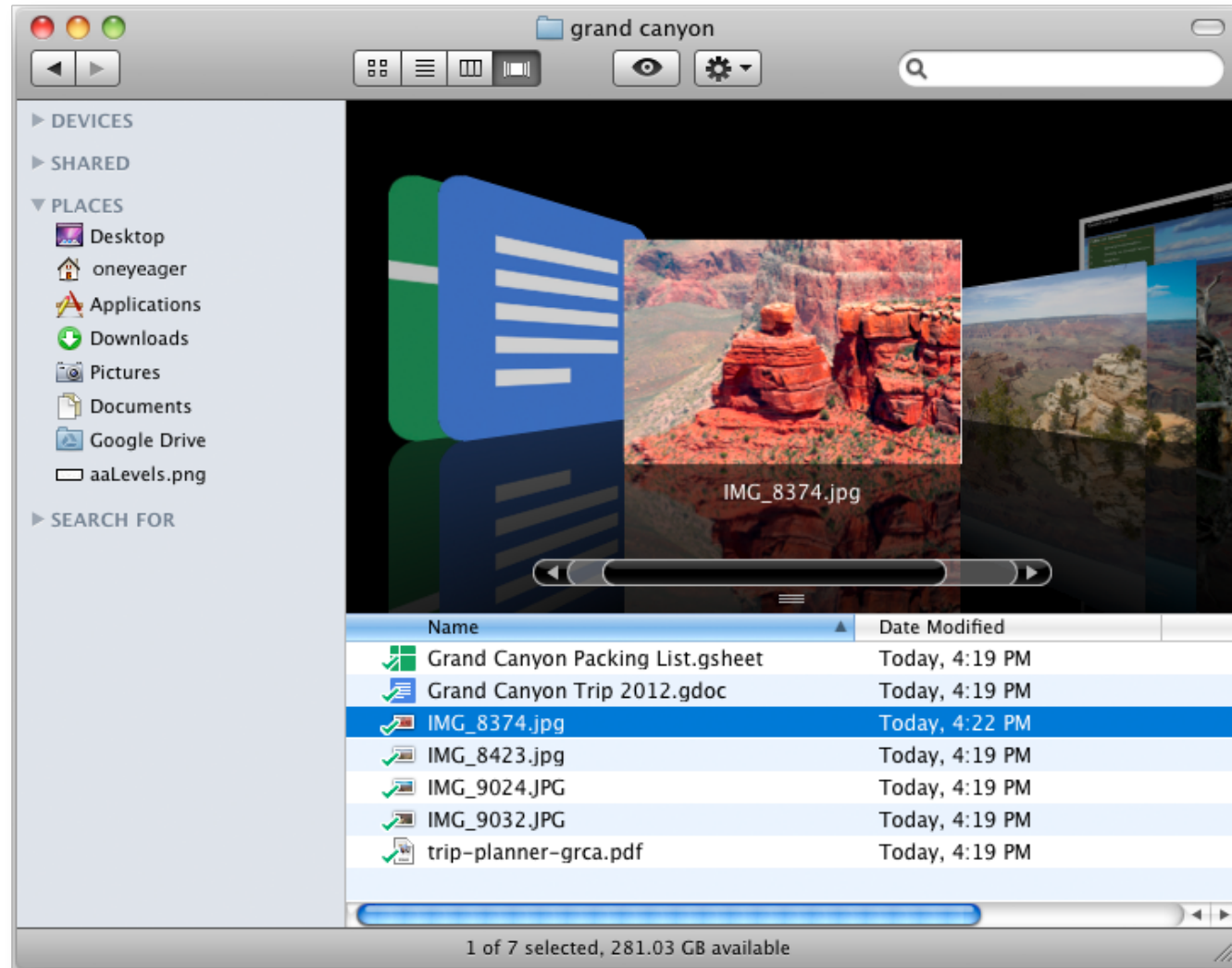
Search Results:

- Document:** "grand canyon" with a thumbnail image of the canyon. The text includes: "Get ready for the trip of a life time!", "The South Rim has an airport and rail service and is close to Interstate 40 and to the transportation centers of Williams and Flagstaff, Arizona. We will be meeting up with our group in Flagstaff and then traveling by night coach to the south entrance. Please make sure to have all of your supplies ready for pick up the night before!", and a "Itinerary" section with dates: Monday: Fly to Arizona, prep for walk; Tuesday: Enter the park - 3 hours; Wednesday: Sunday Hike! Full hike map below.
- Table:** A checklist table with columns: Name, Purchased, Owner, Notes.

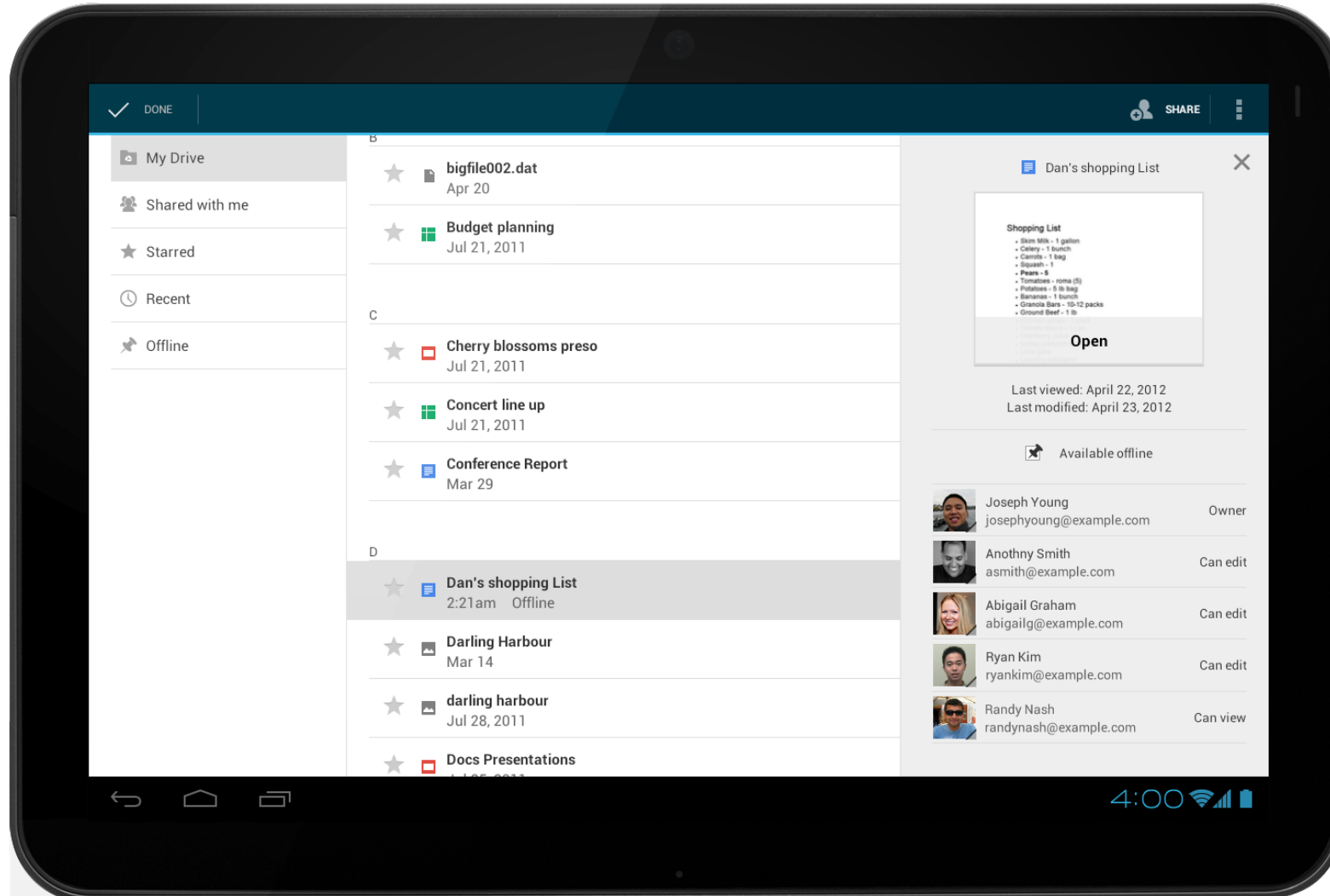
Name	Purchased	Owner	Notes
Toothbrush	Yes	Mark	http://www.amazon.com/dp/B000A9ZD00
Toothpaste	Yes	Mandy	Have this
Small mirror	Yes	Mark	I think this is in the first ar
Moisture lotion	No	Mark	
Chap stick with sunscreen	No	Mark	Make sure to get WorldSt
Soap	No	Mark	
Shampoo - biodegradable	Yes	Mandy	Have this
Camera and Film	Yes	Mandy	
Batteries	Yes	Mandy	
Sunglasses	Yes	Mark	
Eyeglass retainers / chums	No	Mark	http://www.amazon.com/dp/B000A9ZD00
Swimsuit	Yes	Mark	

- Photos:** Several landscape photographs of the Grand Canyon.
- Table of Contents:** A document thumbnail showing a "Table of Contents" with numbered links to various sections like "General Information", "Getting to Grand Canyon", "Checklist", "Itinerary", "Map", "Hiking", "Camping", "Weather", "Safety", "FAQ", "Glossary", "Index", "About Us", "Contact Us", "Privacy Policy", "Terms of Service", "Disclaimer", "Feedback".


Google Drive on your desktop



Google Drive on Android

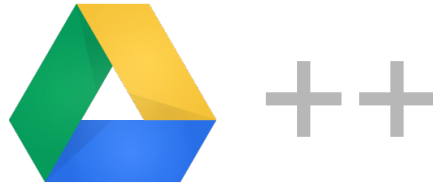


Drive SDK opportunity



Extensive Reach

Put your app in front of millions of users with billions of files



Effortless Integration

Get the best of Google Drive's sharing capabilities, storage capacity and user identity management so you can focus on your app



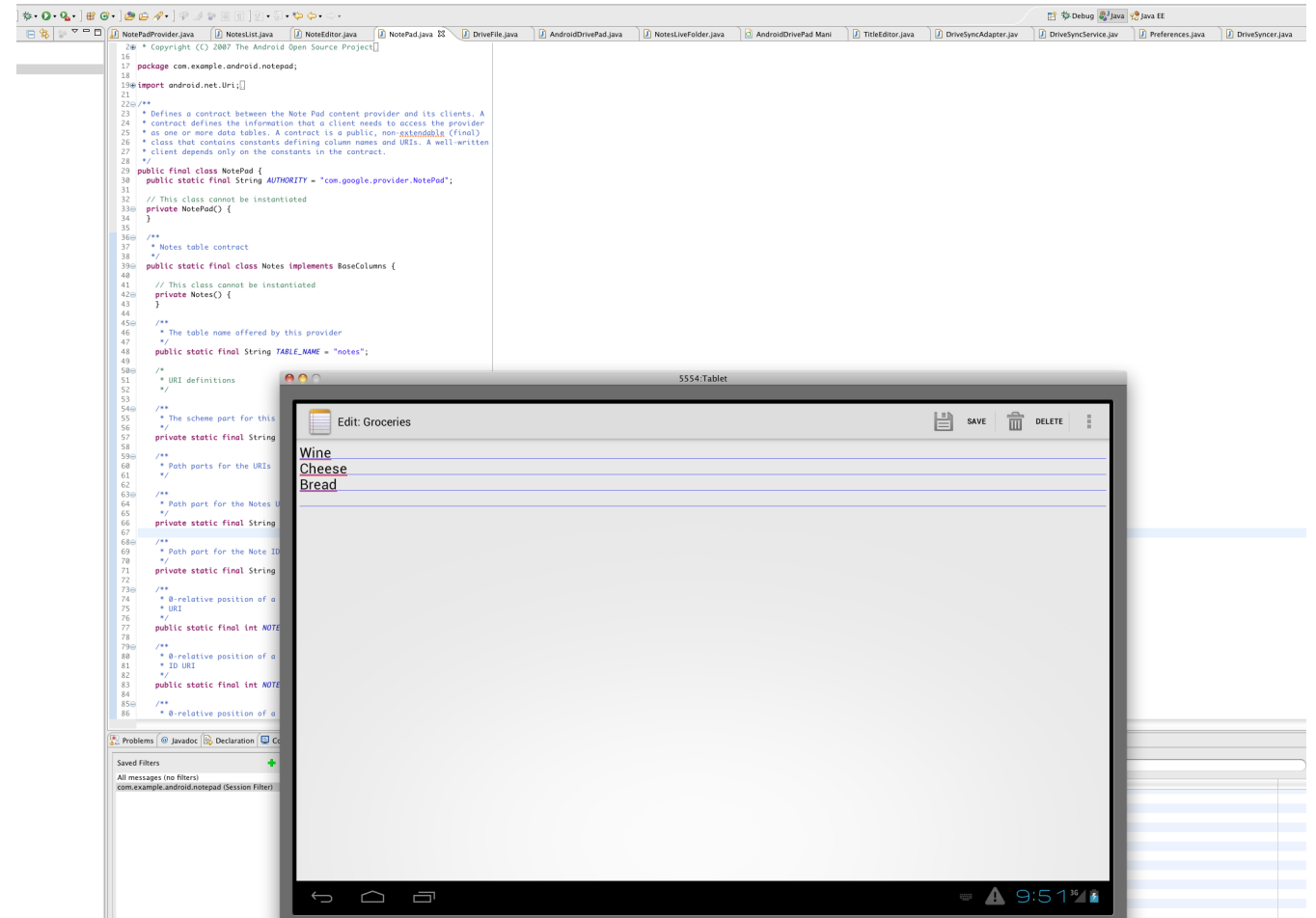
Drive Apps

Open files and docs with your app directly from Drive.
Full integration - search, sharing and revisions



Android Notepad

- Simple notepad application
- Uses SQLite to store notes on the device
- The app you write right after Hello World!
- Perfect starting point for our app



Integrate Notepad with Google Drive

- Perform Authorization
- Unblock the UI thread
- Write a Drive Sync Adapter
- Interact with the Drive app for Android
- Optimize Efficiency



Integrate Notepad with Google Drive

- Perform Authorization
- Unblock the UI thread
- Write a Drive Sync Adapter
- Interact with the Drive app for Android
- Optimize Efficiency



Authorization in Android: Getting an account

```
import com.google.android.gms.common.AccountPicker;
// ...
class MyActivity extends Activity {
    private static final int CHOOSE_ACCOUNT = 0;
    // ...
    private void chooseAccount(Account savedAccount) {
        startActivityForResult(AccountPicker.newChooseAccountIntent(savedAccount, null,
            new String[]() {"com.google"}, false, null, null, null),
            CHOOSE_ACCOUNT);
    }
    // ...
}
```

JAVA



Authorization in Android: Getting an account

```
import com.google.android.gms.common.AccountPicker;
// ...
class MyActivity extends Activity {
    private static final int CHOOSE_ACCOUNT = 0;
    // ...
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == CHOOSE_ACCOUNT && resultCode == RESULT_OK && data != null) {
            String accountName = data.getStringExtra(AccountManager.KEY_ACCOUNT_NAME);
            // TODO: Do something with the account.
        }
    }
    // ...
}
```

JAVA



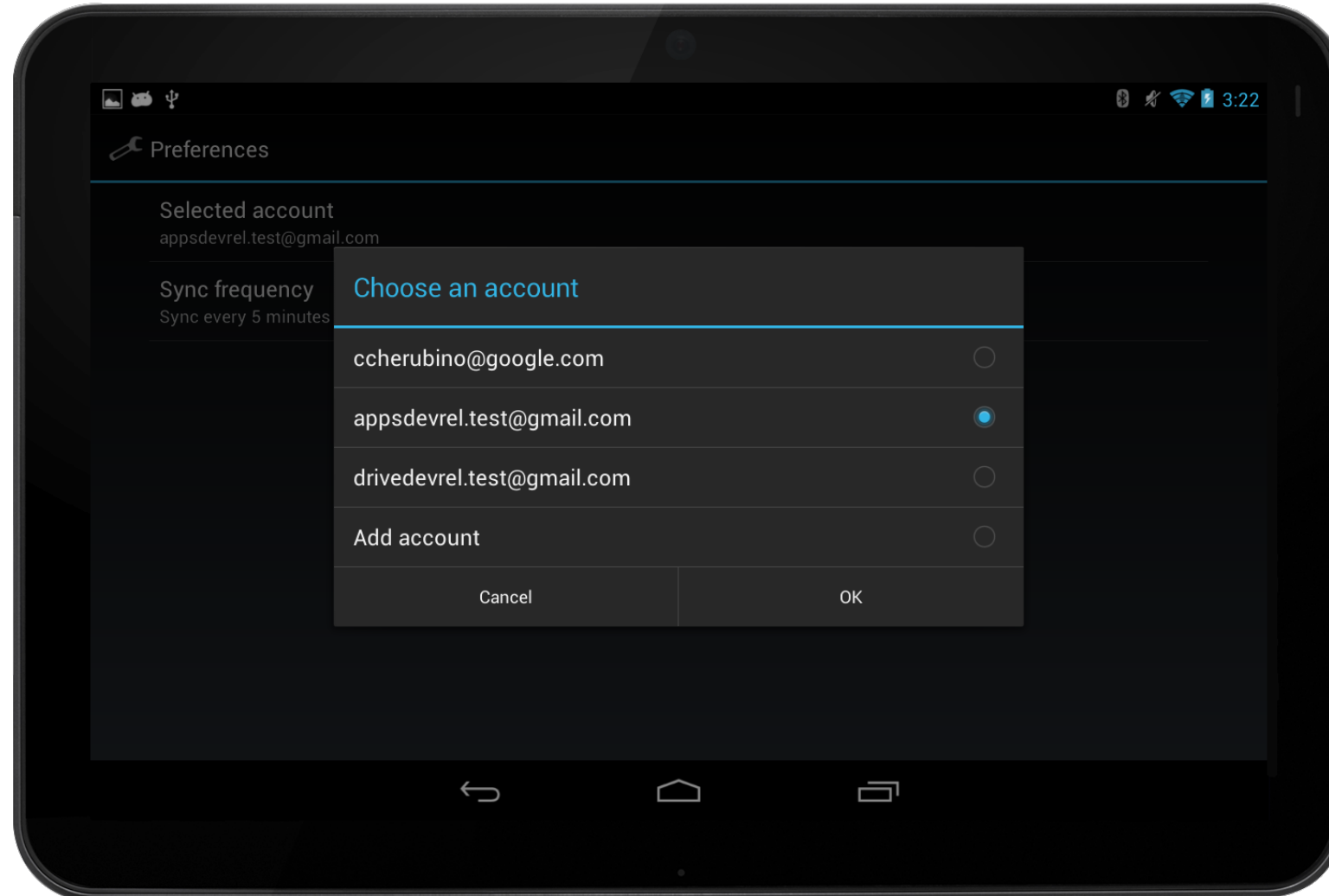
Authorization in Android: Getting an account

```
import com.google.android.gms.common.AccountPicker;
// ...
class MyActivity extends Activity {
    private static final int CHOOSE_ACCOUNT = 0;
    // ...
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == CHOOSE_ACCOUNT && resultCode == RESULT_OK && data != null) {
            String accountName = data.getStringExtra(AccountManager.KEY_ACCOUNT_NAME);
            // TODO: Do something with the account.
        }
    }
    // ...
}
```

JAVA



Authorization in Android: Getting an account



Authorization in Android: Getting an access token

// Must run outside of the UI thread.

```
private String getAccessToken(Context context, Account account) {
```

```
    try {
```

```
        return GoogleAuthUtil.getToken(context, account.name, "oauth2:" + DriveScopes.DRIVE_FILE);
```

```
    } catch (UserRecoverableAuthException e) {
```

```
        // Start the Approval Screen Intent, if not run from an Activity, add the Intent.FLAG_ACTIVITY_NEW_TASK flag.
```

```
        context.startActivityForResult(e.getIntent());
```

```
    }
```

```
    return null;
```

```
}
```

JAVA



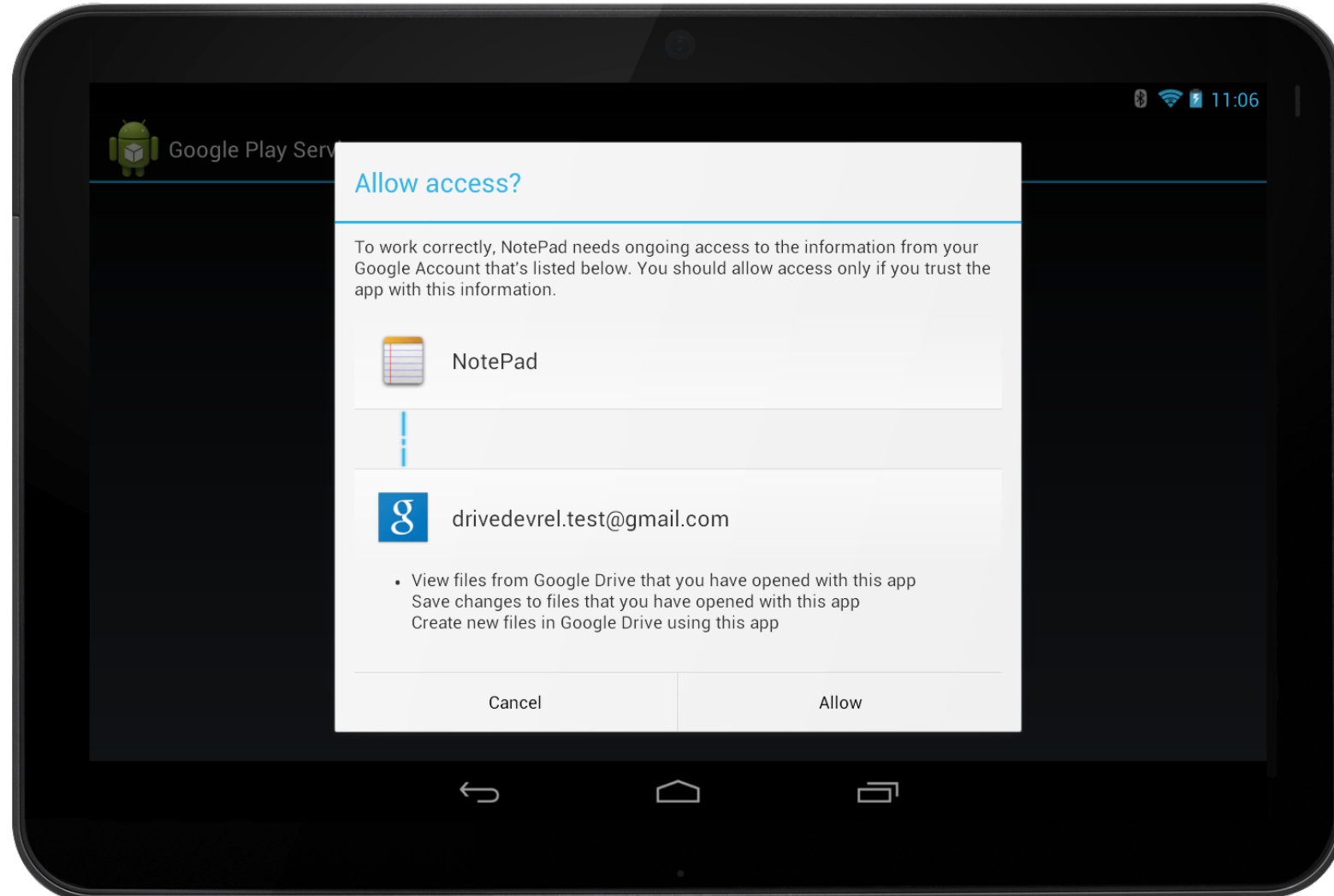
Authorization in Android: Getting an access token

```
// Must run outside of the UI thread.  
private String getAccessToken(Context context, Account account) {  
    try {  
        return GoogleAuthUtil.getToken(context, account.name, "oauth2:" + DriveScopes.DRIVE_FILE);  
    } catch (UserRecoverableAuthException e) {  
        // Start the Approval Screen Intent.  
        context.startActivity(e.getIntent());  
    }  
    return null;  
}
```

JAVA



Authorization in Android: Getting an access token



Authorization in Android: Additional Information

If you want to know more about Authorization in Android, please attend the "Building Android Applications that Use Web APIs" session (<http://goo.gl/rsdh2>)

- Register your Android app on the APIs Console
- Full implementation of the `GoogleAuthUtil.getToken` call in an `AsyncTask`
- Catch errors from `AccountPicker` and the Authorization Grant Screen
- Catch expired or invalid tokens



Integrate Notepad with Google Drive

- Perform Authorization
- Unblock the UI thread
- Write a Drive Sync Adapter
- Interact with the Drive app for Android
- Optimize Efficiency



From synchronous...



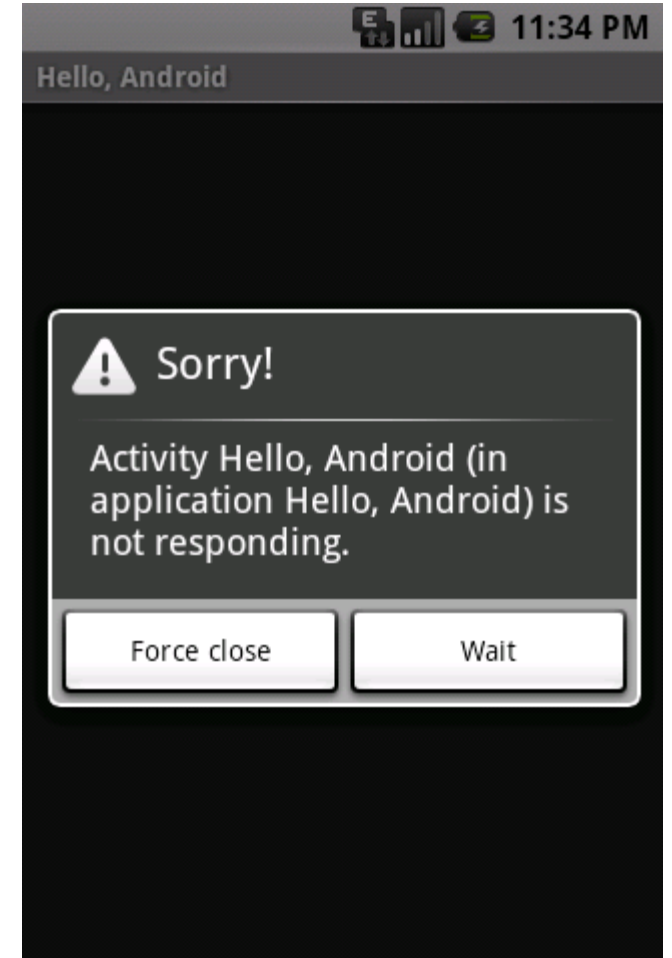
...to asynchronous



source: William Warby, "Heat 1 of the Womens 100m Hurdles Semi-Final" (CC-BY)

Unblock the UI thread

- If the UI thread freezes the user is presented with the "application not responding" (ANR) dialog
- Always run long-running operations in worker threads
- Provide a Handler for child threads to post back to upon completion
- Fortunately, SyncAdapters do all the heavy lifting!



Integrate Notepad with Google Drive

- Perform Authorization
- Unblock the UI thread
- Write a Drive Sync Adapter
- Interact with the Drive app for Android
- Optimize Performance



Drive sync adapter

3 components are involved:

1. Data is stored in the local database by a **ContentProvider**
2. A **SyncAdapter Service** runs in the background and periodically synchronizes data with the cloud
3. An **AbstractThreadedSyncAdapter** implementation contains the sync logic

You may optionally add a settings page to allow users to configure the sync frequency



Drive sync adapter: DriveSyncService

Java

```
public class DriveSyncService extends Service {  
    @Override  
    public void onCreate() {  
        // Instantiate a new static DriveSyncAdapter.  
    }  
  
    @Override  
    public IBinder onBind(Intent intent) {  
        // Return the DriveSyncAdapter instance's Sync Adapter Binder.  
    }  
}
```



Drive sync adapter: DriveSyncService

Java

```
//...
private static final Object sSyncAdapterLock = new Object();
private static DriveSyncAdapter sSyncAdapter = null;
@Override
public void onCreate() {
    synchronized (sSyncAdapterLock) {
        if (sSyncAdapter == null) {
            sSyncAdapter = new DriveSyncAdapter(getApplicationContext(), true);
        }
    }
}
//...
```



Drive sync adapter: DriveSyncService

```
//...  
private static DriveSyncAdapter sSyncAdapter = null;  
@Override  
public IBinder onBind(Intent intent) {  
    return sSyncAdapter.getSyncAdapterBinder();  
}  
//...
```

Java



Drive sync adapter: Service manifest

XML

```
<service android:name="DriveSyncService">
  <intent-filter>
    <action android:name="android.content.SyncAdapter" />
  </intent-filter>
  <meta-data
    android:name="android.content.SyncAdapter"
    android:resource="@xml/syncadapter" />
</service>
```



Drive sync adapter: XML resource

XML

```
<?xml version="1.0" encoding="utf-8"?>  
<sync-adapter xmlns:android="http://schemas.android.com/apk/res/android"  
    android:contentAuthority="com.google.provider.NotePad"  
    android:accountType="com.google"  
    android:userVisible="true"  
    android:supportsUploading="true" />
```



Drive sync adapter: DriveSyncAdapter

Java

```
public class DriveSyncAdapter extends AbstractThreadedSyncAdapter {
    public DriveSyncAdapter(Context context, boolean autoInitialize) {
        super(context, autoInitialize);
    }

    @Override
    public void onPerformSync(Account account, Bundle bundle, String authority,
        ContentProviderClient provider, SyncResult syncResult) {
        // Perform sync logic.
    }
}
```



Drive sync adapter: Enable it!

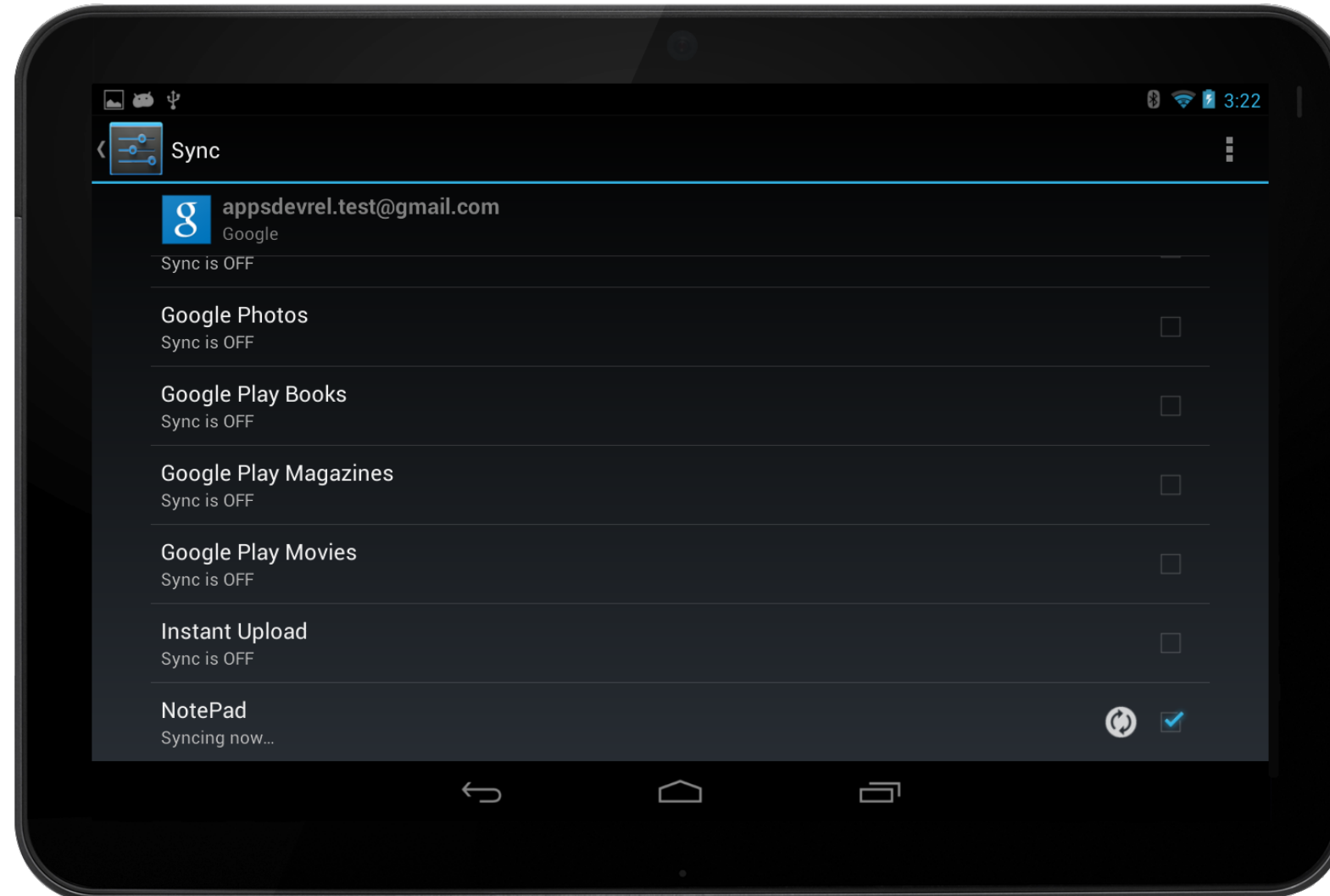
```
private static final String AUTHORITY = "com.google.provider.NotePad";
```

JAVA

```
private void setSyncFrequency(Account account, long syncFrequency) {  
    ContentResolver.setSyncAutomatically(account, AUTHORITY, true);  
    ContentResolver.addPeriodicSync(account, AUTHORITY, new Bundle(), syncFrequency);  
}  
}
```



Drive sync adapter



Let's talk to Drive!

- Retrieve files from the user's Google Drive
- Create new local file
- Sync local files with remote changes
- Handle deleted files



Retrieve files from Drive

```
private void storeAllDriveFiles(Drive service) {  
    Files.List request = service.files().list().setQ("mimeType=text/plain");  
    do {  
        FileList files = request.execute();  
        for (File file : files.getItems()) {  
            String fileId = file.getId();  
            String title = file.getTitle();  
            InputStream content = service.getRequestFactory().buildGetRequest(file.getDownloadUrl()).execute().getContent();  
            insertDriveFile(fileId, title, content);  
        }  
        request.setPageToken(files.getNextPageToken());  
    } while (request.getPageToken() != null && request.getPageToken().length() > 0);  
}
```

JAVA



Retrieve files from Drive

JAVA

```
private void storeAllDriveFiles(Drive service) {
    Files.List request = service.files().list().setQ("mimeType=text/plain");
    do {
        FileList files = request.execute();
        for (File file : files.getItems()) {
            String fileId = file.getId();
            String title = file.getTitle();
            InputStream content = service.getRequestFactory().buildGetRequest(file.getDownloadUrl()).execute().getContent();
            insertDriveFile(fileId, title, content);
        }
        request.setPageToken(files.getNextPageToken());
    } while (request.getPageToken() != null && request.getPageToken().length() > 0);
}
```



Retrieve files from Drive

JAVA

```
private void storeAllDriveFiles(Drive service) {
    Files.List request = service.files().list().setQ("mimeType=text/plain");
    do {
        FileList files = request.execute();
        for (File file : files.getItems()) {
            String fileId = file.getId()
            String title = file.getTitle();
            InputStream content = service.getRequestFactory().buildGetRequest(file.getDownloadUrl()).execute().getContent();
            insertDriveFile(fileId, title, content);
        }
        request.setPageToken(files.getNextPageToken());
    } while (request.getPageToken() != null && request.getPageToken().length() > 0);
}
```



Save files to Drive

```
private void insertLocalFile(Drive service, ContentClientProvider provider, Cursor cursor, Uri noteUri) {  
    File newFile = new File();  
    newFile.setTitle(cursor.getString(COLUMN_INDEX_TITLE));  
    newFile.setMimeType("text/plain");  
    String content = cursor.getString(COLUMN_INDEX_NOTE);  
    File insertedFile = service.files().insert(  
        newFile, ByteArrayContent.fromString("text/plain", content)).execute();  
    ContentValues values = new ContentValues();  
    values.put(COLUMN_NAME_DRIVE_ID, insertedFile.getId());  
    values.put(COLUMN_NAME_MODIFICATION_DATE, insertedFile.getModifiedByMeDate().getValue());  
    values.put(COLUMN_NAME_CREATED_DATE, insertedFile.getCreatedDate().getValue());  
    provider.update(noteUri, values, null, null);  
}
```

JAVA



Save files to Drive

```
private void insertLocalFile(Drive service, ContentClientProvider provider, Cursor cursor, Uri noteUri) {
    File newFile = new File();
    newFile.setTitle(cursor.getString(COLUMN_INDEX_TITLE));
    newFile.setMimeType("text/plain");
    String content = cursor.getString(COLUMN_INDEX_NOTE);
    File insertedFile = service.files().insert(
        newFile, ByteArrayContent.fromString("text/plain", content)).execute();
    ContentValues values = new ContentValues();
    values.put(COLUMN_NAME_DRIVE_ID, insertedFile.getId());
    values.put(COLUMN_NAME_MODIFICATION_DATE, insertedFile.getModifiedByMeDate().getValue());
    values.put(COLUMN_NAME_CREATED_DATE, insertedFile.getCreatedDate().getValue());
    provider.update(noteUri, values, null, null);
}
```

JAVA



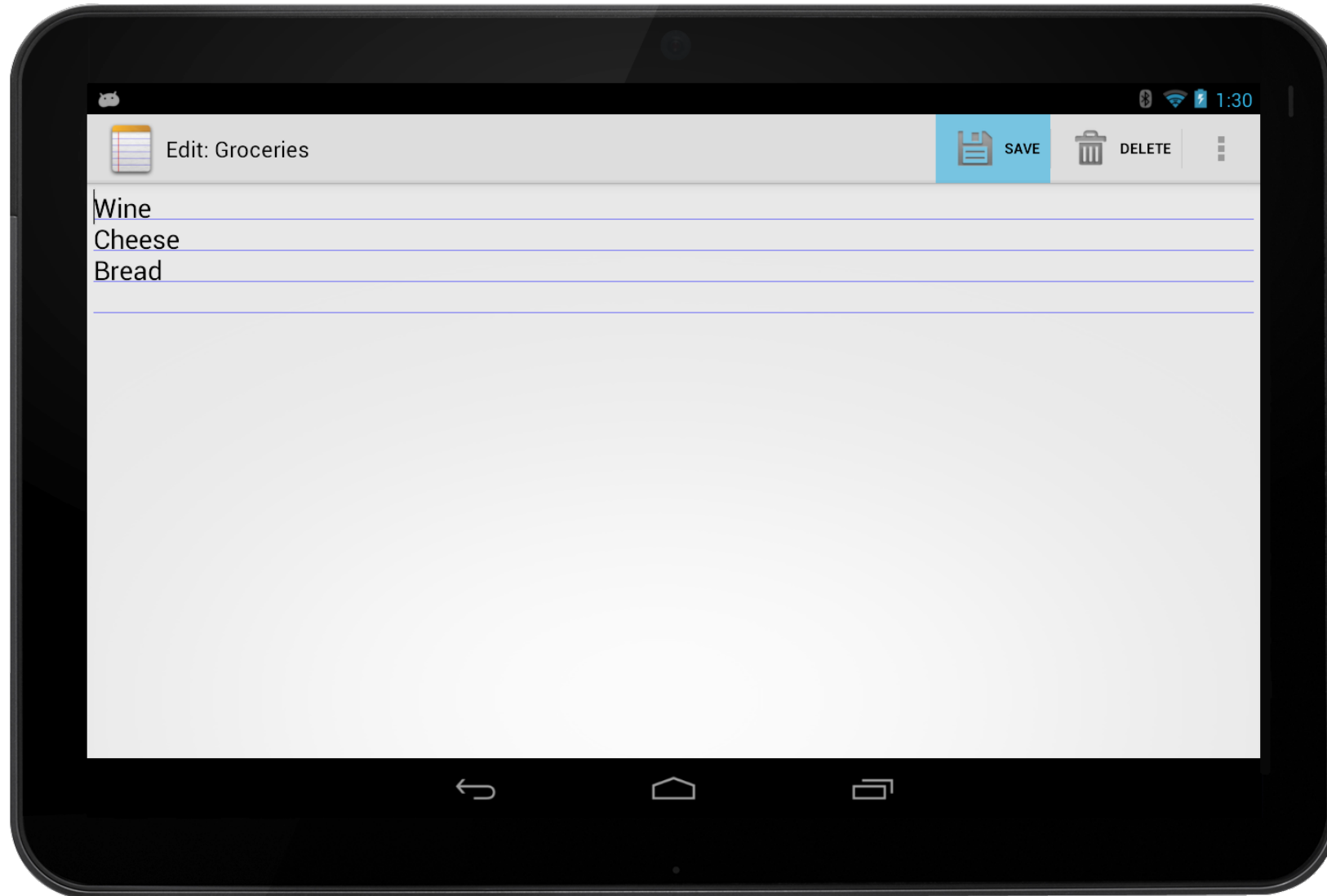
Save files to Drive

```
private void insertLocalFile(Drive service, ContentClientProvider provider, Cursor cursor, Uri noteUri) {  
    File newFile = new File();  
    newFile.setTitle(cursor.getString(COLUMN_INDEX_TITLE));  
    newFile.setMimeType("text/plain");  
    String content = cursor.getString(COLUMN_INDEX_NOTE);  
    File insertedFile = service.files().insert(  
        newFile, ByteArrayContent.fromString("text/plain", content)).execute();  
    ContentValues values = new ContentValues();  
    values.put(COLUMN_NAME_DRIVE_ID, insertedFile.getId());  
    values.put(COLUMN_NAME_MODIFICATION_DATE, insertedFile.getModifiedByMeDate().getValue());  
    values.put(COLUMN_NAME_CREATED_DATE, insertedFile.getCreatedDate().getValue());  
    provider.update(noteUri, values, null, null);  
}
```

JAVA



Save files to Drive



Integrate Notepad with Google Drive

- Perform Authorization
- Unblock the UI thread
- Write a Drive Sync Adapter
- Interact with the Drive app for Android
- Optimize Efficiency



Receiving intents from the Drive app

- Export an Activity that supports an intent with the following info:
 - action: [drive.intent.action.DRIVE_OPEN](#)
 - path: [content://com.google.android.drive/open/*resourceId*](#)
 - type: MIME type of the file
- Declare your API Project ID in the Activity's metadata
- List supported MIME types in the [intent-filter](#) element
- The intent will not include the body of the document nor the user account
- Retrieve the file from the Drive API using the [resourceId](#) provided by the intent in the path



Receiving intents from the Drive app

XML

```
<manifest ...>  
  <uses-permission android:name="android.permission.GET_ACCOUNTS" />  
  <application ...>  
    <activity android:name="DriveActivity" android:label="@string/cloud_paint" android:icon="@drawable/app_icon" android:exported="true">  
      <meta-data android:name="com.google.android.apps.drive.APP_ID" android:value="id=1234567890" />  
      <intent-filter>  
        <action android:name="com.google.android.apps.drive.DRIVE_OPEN" />  
        <data android:mimeType="text/plain" />  
        <data android:mimeType="text/html" />  
      </intent-filter>  
    </activity>  
  </application>  
</manifest>
```



Receiving intents from the Drive app

@Override

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    final Intent intent = getIntent();  
    final String action = intent.getAction();  
    if ("drive.intent.action.DRIVE_OPEN".equals(action)) {  
        String fileId = intent.getStringExtra("resourceId");  
        // Prompt the user to choose the account to use and process the file using the Drive API.  
    } else {  
        // Other action.  
    }  
}
```

JAVA



Receiving intents from the Drive app

@Override

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    final Intent intent = getIntent();  
    final String action = intent.getAction();  
    if ("drive.intent.action.DRIVE_OPEN".equals(action)) {  
        String fileId = intent.getStringExtra("resourceId");  
        // Prompt the user to choose the account to use and process the file using the Drive API.  
    } else {  
        // Other action.  
    }  
}
```

JAVA



Receiving intents from the Drive app

@Override

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    final Intent intent = getIntent();  
    final String action = intent.getAction();  
    if ("drive.intent.action.DRIVE_OPEN".equals(action)) {  
        String fileId = intent.getStringExtra("resourceId");  
        // Prompt the user to choose the account to use and process the file using the Drive API.  
    } else {  
        // Other action.  
    }  
}
```

JAVA



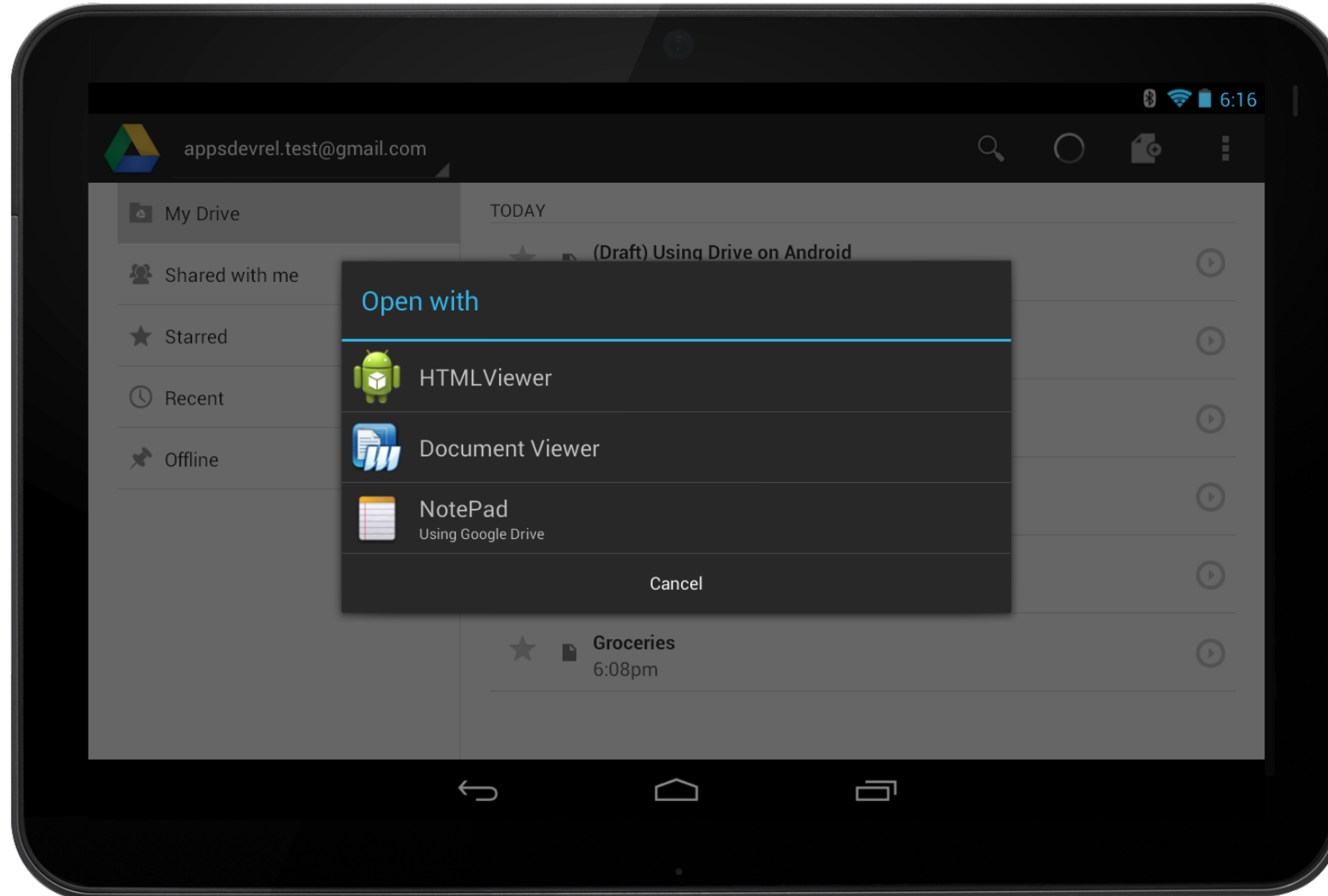
Receiving intents from the Drive app

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    final Intent intent = getIntent();
    final String action = intent.getAction();
    if ("drive.intent.action.DRIVE_OPEN".equals(action)) {
        String fileId = intent.getStringExtra("resourceId");
        // Prompt the user to choose the account to use and process the file using the Drive API.
    } else {
        // Other action.
    }
}
```

JAVA



Receiving intents from the Drive app



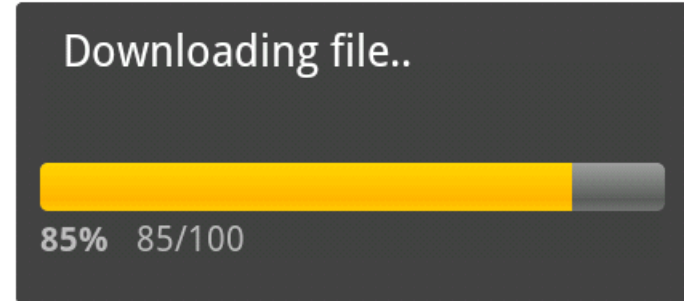
Integrate Notepad with Google Drive

- Perform Authorization
- Unblock the UI thread
- Write a Drive Sync Adapter
- Interact with the Drive app for Android
- Optimize Efficiency



Every byte is sacred

- Drive files can be up to 10GB
- Even with the fastest available 4G, it takes 20+ hours to download a 10GB file!
- Only download files from Drive when it is absolutely necessary
- Use the local database as cache
- When the user requests a file
 - send a request to get its metadata
 - compare the [md5Checksum](#) field in the metadata with the one in the local database (if present)
 - return the local copy if they match or download the document from Drive when they don't



Using Drive file metadata

```
private void pushChangeToDrive(Drive service, Cursor localFile, File driveFile) {
    String localNote = localFile.getString(COLUMN_INDEX_NOTE);
    File updatedFile = null;
    driveFile.setTitle(localFile.getString(COLUMN_INDEX_TITLE));
    if (md5Checksum(localNote).equals(driveFile.getMd5Checksum())) {
        updatedFile = service.files().update(driveFile.getId(), driveFile).execute(); // Only update the metadata.
    } else {
        // Update both content and metadata.
        ByteArrayContent content = ByteArrayContent.fromString("text/plain", localNote);
        updatedFile = service.files().update(driveFile.getId(), driveFile, content).execute();
    }
    // Update local files metadata.
}
```

JAVA



Using Drive file metadata

```
private void pushChangeToDrive(Drive service, Cursor localFile, File driveFile) {  
    String localNote = localFile.getString(COLUMN_INDEX_NOTE);  
    File updatedFile = null;  
    driveFile.setTitle(localFile.getString(COLUMN_INDEX_TITLE));  
    if (md5Checksum(localNote).equals(driveFile.getMd5Checksum())) {  
        updatedFile = service.files().update(driveFile.getId(), driveFile).execute(); // Only update the metadata.  
    } else {  
        // Update both content and metadata.  
        ByteArrayContent content = ByteArrayContent.fromString("text/plain", localNote);  
        updatedFile = service.files().update(driveFile.getId(), driveFile, content).execute();  
    }  
    // Update local files metadata.  
}
```

JAVA



Using Drive file metadata

```
private void pushChangeToDrive(Drive service, Cursor localFile, File driveFile) {
    String localNote = localFile.getString(COLUMN_INDEX_NOTE);
    File updatedFile = null;
    driveFile.setTitle(localFile.getString(COLUMN_INDEX_TITLE));
    if (md5Checksum(localNote).equals(driveFile.getMd5Checksum())) {
        updatedFile = service.files().update(driveFile.getId(), driveFile).execute(); // Only update the metadata.
    } else {
        // Update both content and metadata.
        ByteArrayContent content = ByteArrayContent.fromString("text/plain", localNote);
        updatedFile = service.files().update(driveFile.getId(), driveFile, content).execute();
    }
    // Update local files metadata.
}
```

JAVA



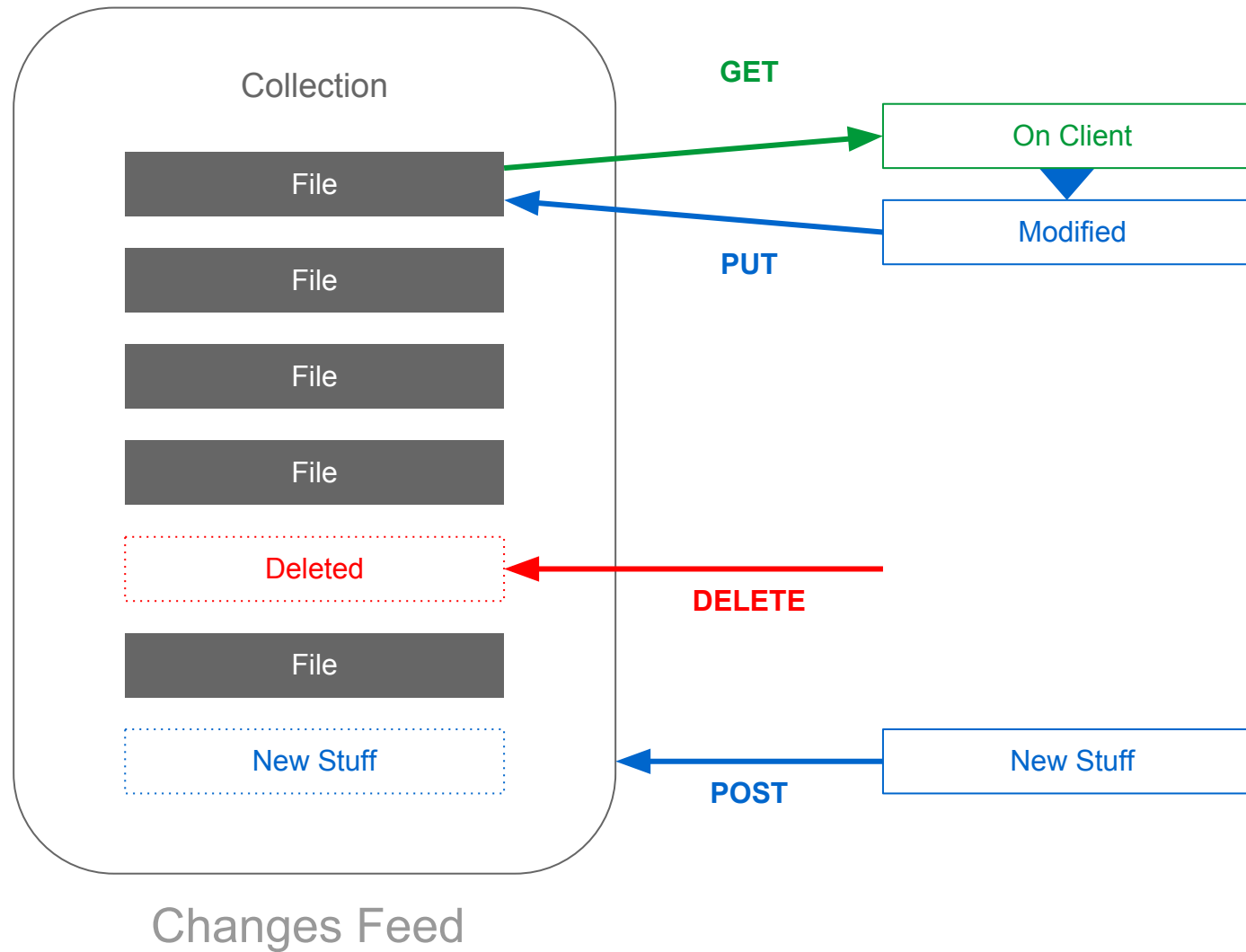
Using Drive file metadata

```
private void pushChangeToDrive(Drive service, Cursor localFile, File driveFile) {
    String localNote = localFile.getString(COLUMN_INDEX_NOTE);
    File updatedFile = null;
    driveFile.setTitle(localFile.getString(COLUMN_INDEX_TITLE));
    if (md5Checksum(localNote).equals(driveFile.getMd5Checksum())) {
        updatedFile = service.files().update(driveFile.getId(), driveFile).execute(); // Only update the metadata.
    } else {
        // Update both content and metadata.
        ByteArrayContent content = ByteArrayContent.fromString("text/plain", localNote);
        updatedFile = service.files().update(driveFile.getId(), driveFile, content).execute();
    }
    // Update local files metadata.
}
```

JAVA



Detecting changes to files



Detecting changes to files

JAVA

```
Changes.List request =
    service.changes().list().setStartChangeId(new BigInteger(Long.toString(startChangeId)));
do {
    ChangeList changes = request.execute();
    for (Change change : changes.getItems()) {
        if (change.getDeleted()) {
            // File with ID "change.getFileId()" has been deleted.
        } else {
            File changedFile = change.getFile(); // Do something with the updated metadata.
        }
    }
    request.setPageToken(changes.getNextPageToken());
} while (request.getPageToken() != null && request.getPageToken().length() > 0);
```



Detecting changes to files

```
Changes.List request =
    service.changes().list().setStartChangeId(new BigInteger(Long.toString(startChangeId)));
do {
    ChangeList changes = request.execute();
    for (Change change : changes.getItems()) {
        if (change.getDeleted()) {
            // File with ID "change.getFileId()" has been deleted.
        } else {
            File changedFile = change.getFile(); // Do something with the updated metadata.
        }
    }
    request.setPageToken(changes.getNextPageToken());
} while (request.getPageToken() != null && request.getPageToken().length() > 0);
```

JAVA



Detecting changes to files

```
Changes.List request =
    service.changes().list().setStartChangeld(new BigInteger(Long.toString(startChangeld)));
do {
    ChangeList changes = request.execute();
    for (Change change : changes.getItems()) {
        if (change.getDeleted()) {
            // File with ID "change.getFileId()" has been deleted.
        } else {
            File changedFile = change.getFile(); // Do something with the updated metadata.
        }
    }
    request.setPageToken(changes.getNextPageToken());
} while (request.getPageToken() != null && request.getPageToken().length() > 0);
```

JAVA





Live demo

<Thank You!>

<https://developers.google.com/drive/>

ccherubino@google.com

<http://plus.claudiocherubino.it>

#ccherubino

alainv@google.com

<http://plus.vongsouvanh.com>





Google
Developers