G54MDP
Mobile Device Programming

Lecture 12 – Databases and Content Providers

Logical Data Storage on Android

- File-based abstractions
  - Shared Preferences
    - Simple key value pairs
  - File-based storage
    - Internal Data Storage
      - Soldered RAM
      - Internal APK resources, temporary files
    - External Data Storage
      - SD Card
      - Large media files
  - SQLite Database
    - Structured data, small binary files
- Network
  - Shared contact lists, backups
  - SyncAdapter

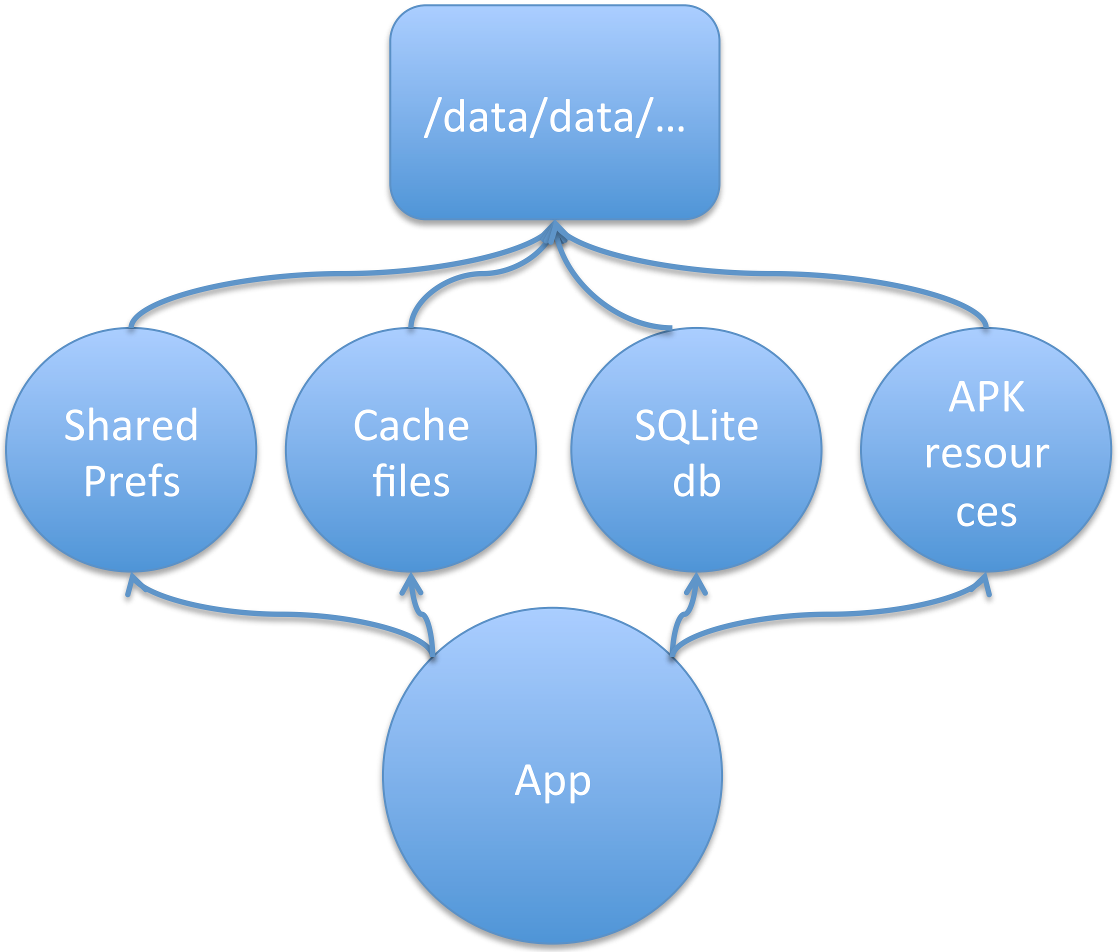
Cursors

- Provides random access to results of a query
- Fairly self explanatory object
  - Enables you to step over all the rows returned by a query
  - Has a close() method to close the query when you are finished
    - don’t wait for it to be garbage collected
  - “Connect” a cursor to a CursorAdapter and ListView

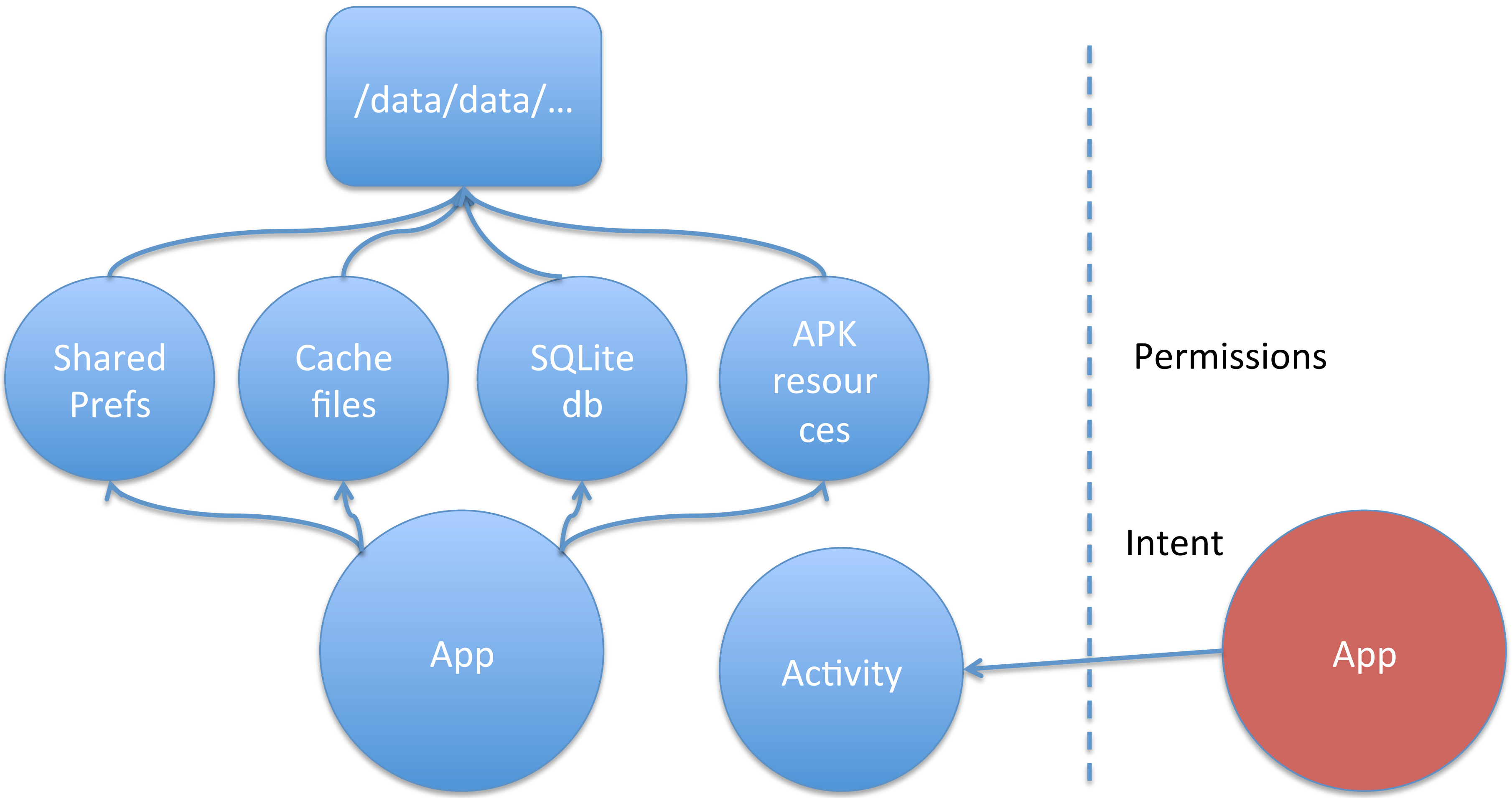
Database Abstraction

- Good software architecture
  - Separation of data model from presentation / views
- Abstraction of database architecture
  - Easier to update storage code
  - Expose column indices as static class variables
    - c.getInt(0) -> c.getInt(DBHelper.NAME)
  - Helper methods keep database internals from "leaking" into other classes
    - Return a Collection of results rather than a Cursor
      - Use Cursor internally in DBHelper class
  - SQL injection
    - Sanitise user input

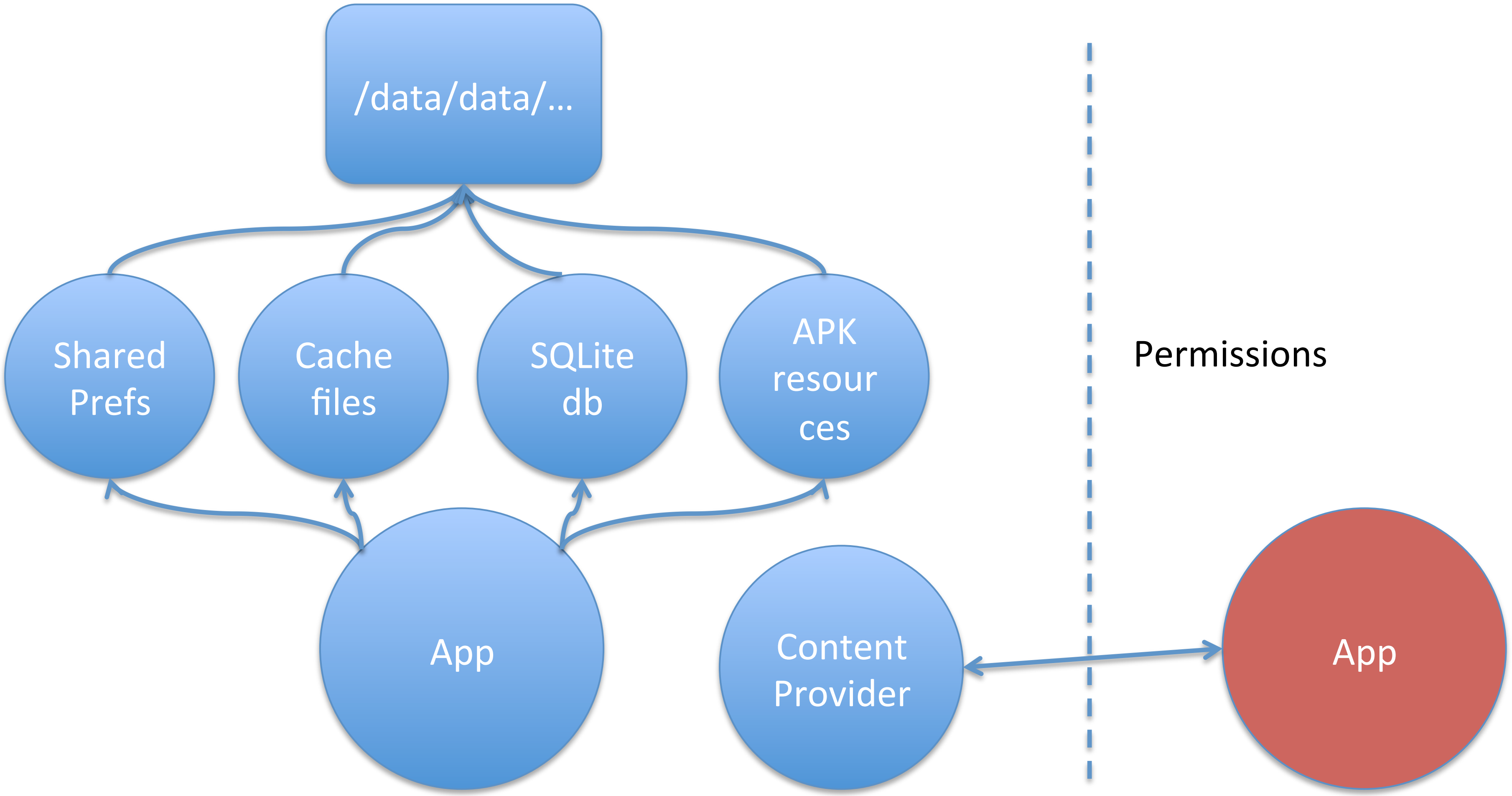
Sharing Data



Sharing Data – is this good enough?



Sharing Data – if not



ContentProvider

- Access to data is restricted to the app that owns it
  - Remember where the database file is?
  - If we want other apps to access our data, or we want to access other apps' data
  - …we need to provide or make use of a ContentProvider
    - Component number **3**
    - Exposes data / content to other applications in a structured manner
    - Fundamentally IPC via Binder with a strict interface

System ContentProviders

- ContentProviders manage data for:
  - Browser
    - Bookmarks, history
  - Call log
    - Telephone usage
  - Contacts
    - Contact data
    - WhatsApp?
  - Media
    - Media database
  - UserDictionary
    - Database for predictive spelling
  - …
- Again, recall common mobile capabilities

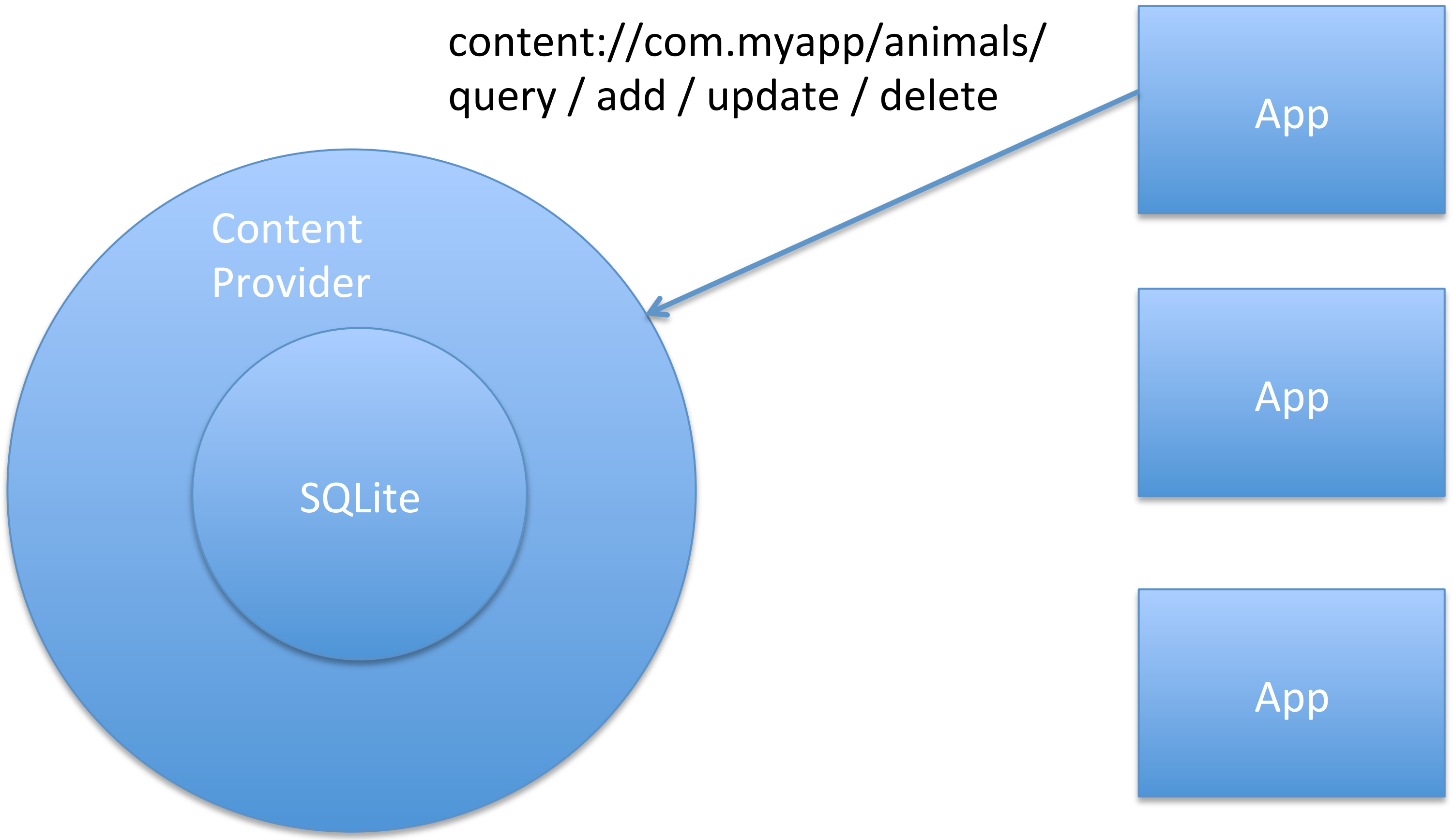
Content Providers

- Either create a new one (by sub-classing ContentProvider)
- Or add / query data via an existing native ContentProvider
- Assuming that spawning an Activity via Intent is not sufficient
  - Querying complex data
  - Requiring close coupling of application to data

Data Model

- ContentProviders enforce a specific data model
- Very similar to a relational database table
  - A collection of records
  - Support for reading and writing
  - Support typical database operations
- Records are stored in rows, with each column providing different data fields
  - Each record has a numeric id (in the field _ID) that uniquely identifies it
- Tables exposed via URI
  - Abstraction again
  - Most of the "work" is specifying the abstraction

Data Model


content://com.myapp/animals/
query / add / update / delete
Content Provider
SQLite
App
App
App

Querying a ContentProvider

- ContentResolver
  - – Manages and supports ContentProvider access
  - – Enables ContentProviders to be used across multiple applications
  - – Provides additional services such as change notification
    - Can *observe* a ContentProvider to be informed of real-time modifications
      - – A new MP3 has been added to the library
- ContentResolver cr = getContentResolver();

Querying a ContentProvider

- ContentProviders identify data sets through URIs
  - content://authority/path/id
- content
  - Data managed by a ContentProvider
- authority
  - ID for the ContentProvider (i.e. fully qualified class name, com.example.martindata)
- path
  - 0 or more segments indicating the subset of data to be requested
    - e.g. table name, or something more readable / abstracted
- id
  - Specific record (row) being accessed

Querying a ContentProvider

- URI for searching Contacts
  - ContactsContract.Contacts.CONTENT_URI = “content://com.android.contacts/contacts/”
- ContentResolver.query(…)
  - Returns a Cursor instance for accessing results
- Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)
- Cursor c = cr.query(ContactsContract.Contacts.CONTENT_URI, new String[] { ContactsContract.Contacts.DISPLAY_NAME }, null, null, null);

Contacts

- To access / modify Contacts, requires a Permission
  - android.permission.READ_CONTACTS
  - android.permission.WRITE_CONTACTS
- Contacts has three components
  - Data
    - Rows (mime-typed) that can hold personal information
  - RawContacts
    - A contact for a given person from a given system
      - Gmail contact, Facebook contact etc
      - Associated with Data entries
  - Contacts
    - Aggregated RawContacts
      - Single view to a person

Modifying a ContentProvider

- Uri insert(Uri url, ContentValues values)
- int update(Uri uri, ContentValues values, String where, String[] selectionArgs)
- Uri
  - The table that we wish to update / insert
- ContentValues
  - Values for the new row
  - Key/value pairs
    - Key is the column name
- where
  - SQL WHERE clause

Creating a Content Provider

- Implement a storage system for the data
  - Structured data / SQLite
    - Values, binary blobs up to 64k
    - Database
  - Large binary blobs
    - Files
  - Photos / media manager
- Implement a ContentProvider
  - Implement required methods
  - query, add, update, insert etc
  - onCreate
  - getType
    - What type of data are we providing?
  - ParcelFileDescripter openFile()
- Tell Android we are a provider
  - Declare in the AndroidManifest

Contract

- Defines metadata pertaining to the provider
- Constant definitions that are exposed to developers via a compiled .jar file
  - Authority
    - Which app is responsible for this data
  - URI
  - Meta-data types
  - Column names
    - Abstraction of database architecture

URI Matching

- All of these methods (except onCreate()) take a URI as the first parameter
  - The object will need to parse it to some extent to know what to return, insert or update
  - Android provides android.content.UriMatcher to simplify this
    - Provides mapping between abstraction of contract class to concrete db implementation
  - Does the calling application want all data from a table, or just a row, or a specific table?
    - Or a "virtual" table

Let’s have a look…

Network

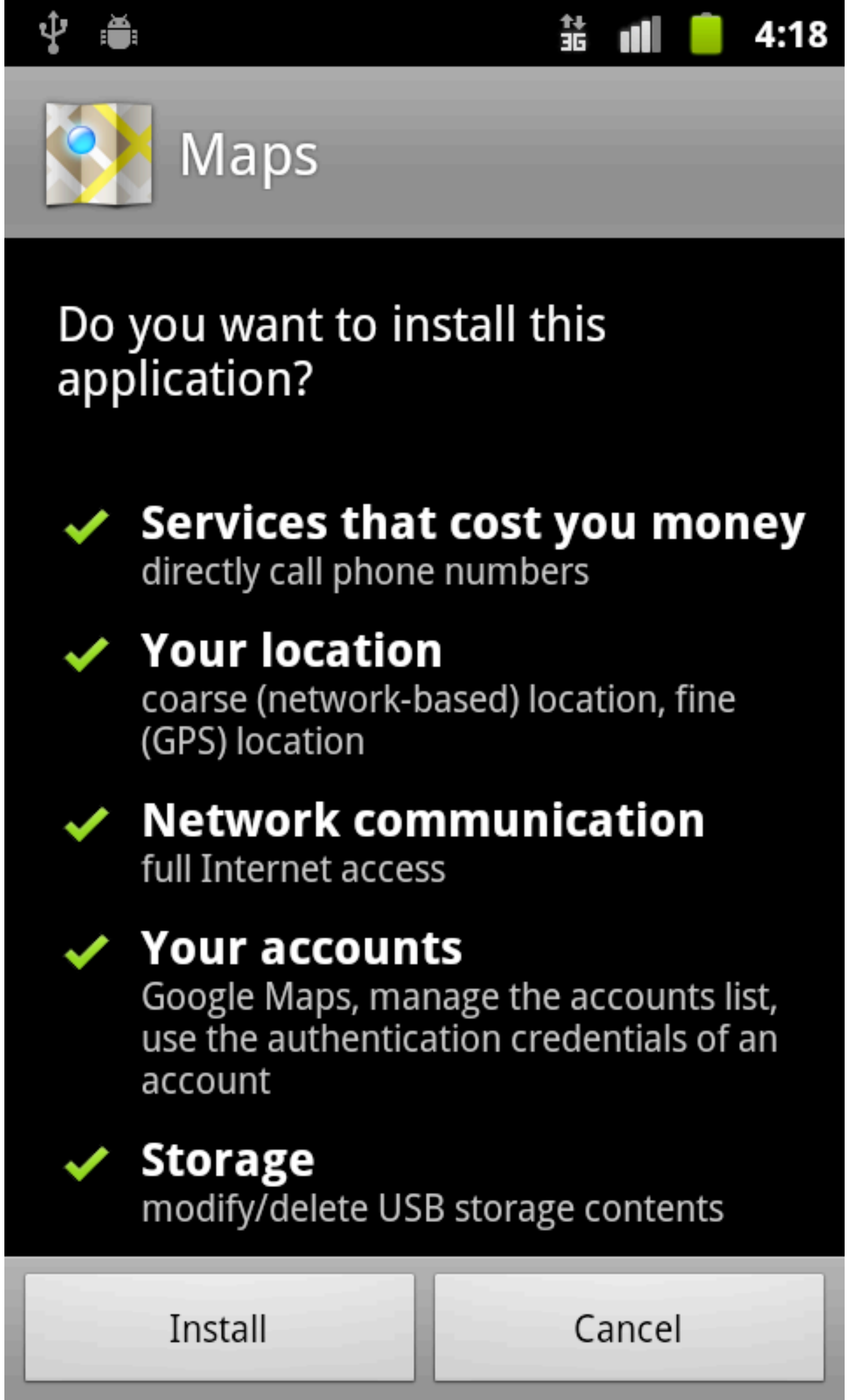
- One last type of data storage
  - Get it off the phone, and into the cloud
- Implement a SyncAdapter
  - Appears in the "Accounts and Sync" menu in the OS
  - Synchronizes a local database / content provider with a remote server
  - Make use of a Service to push data in the background
- http://developer.android.com/training/sync-adapters/creating-sync-adapter.html

Android Security

- Isolation by default
- Linux kernel
  - Filesystem / UID
    - Private, per-application file storage
  - Processes
    - Individual virtual machine instances
    - Native-code controlled by the application sandbox
- Restricted access to the root user
  - Most processes run as normal users
- IPC through specific interfaces
  - Binder: Services Intents, Messages, ContentProviders
  - Intentional lack of APIs for sensitive functionality
    - Direct SIM card access

Permissions

- No access by default
  - Control access to specific mechanisms
- Applications can offer protected access to resources and data with **permissions**
  - Permissions explicitly granted by users
- Permission architecture
  - Applications statically declare permissions
    - Required **of** components interacting with them
      - You must have this permission to interact with me
    - Required **by** components they interact with
      - I will need these permissions
  - Android requires user's consent to specific permissions when an application is installed



4:18

Maps

Do you want to install this application?

✓ **Services that cost you money**
directly call phone numbers

✓ **Your location**
coarse (network-based) location, fine (GPS) location

✓ **Network communication**
full Internet access

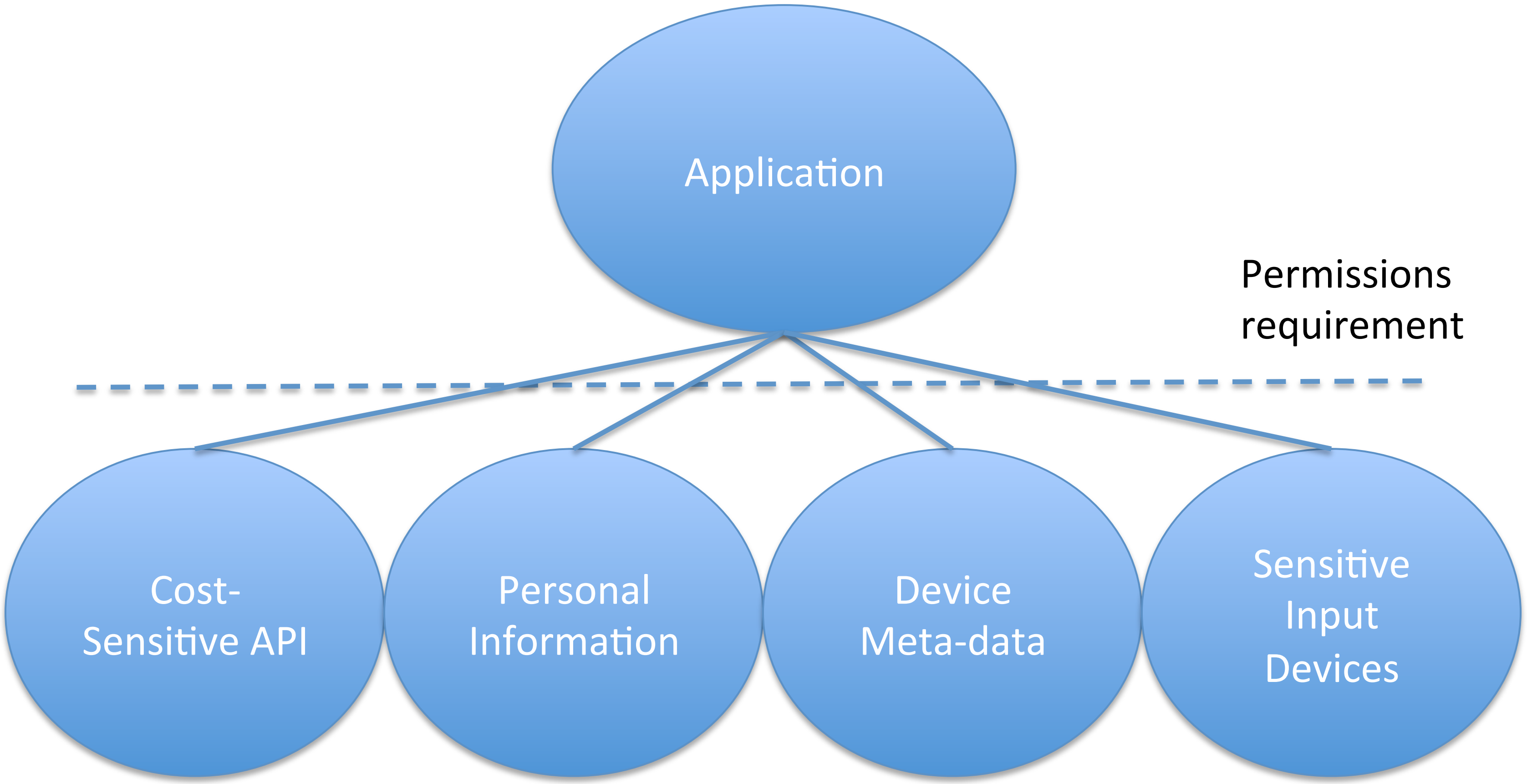
✓ **Your accounts**
Google Maps, manage the accounts list, use the authentication credentials of an account

✓ **Storage**
modify/delete USB storage contents

Permissions

- Show permissions required at install time
  - Not prompted again regarding permissions at run-time
- Why?
  - Not yet made a commitment (financial, mental) to the application
    - Can compare to other applications
  - Not per session / at run-time
    - “Seamless” switching between Activities / applications
    - Would slow down the user experience
    - Train users to click “ok” repeatedly without considering the implications

Permissions

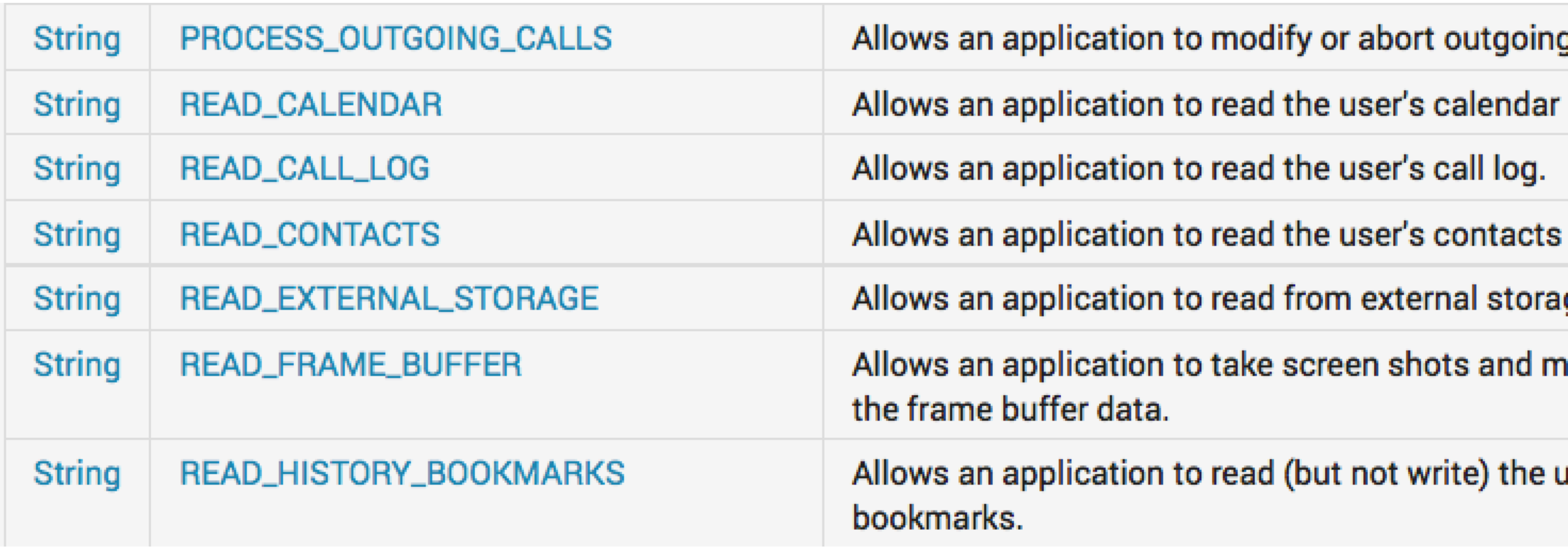


Permissions

- Cost-Sensitive APIs
  - Telephony
  - SMS/MMS
  - Network/Data
  - In-App Billing
  - NFC Access
- Personal Information
  - Contacts, calendar, messages, emails
- Device Meta-data
  - System logs, browser history, network identifiers
- Sensitive Input Devices
  - Interaction with the surrounding environment
  - Camera, microphone, GPS

Permissions

- http://developer.android.com/reference/android/Manifest.permission.html



String	PROCESS_OUTGOING_CALLS	Allows an application to modify or abort outgoing
String	READ_CALENDAR	Allows an application to read the user's calendar
String	READ_CALL_LOG	Allows an application to read the user's call log.
String	READ_CONTACTS	Allows an application to read the user's contacts
String	READ_EXTERNAL_STORAGE	Allows an application to read from external storag
String	READ_FRAME_BUFFER	Allows an application to take screen shots and m the frame buffer data.
String	READ_HISTORY_BOOKMARKS	Allows an application to read (but not write) the u bookmarks.

Common Permissions

- android.permission.ACCESS_FINE_LOCATION
- android.permission.WRITE_EXTERNAL_STORAGE
- android.permission.INTERNET
- android.permission.WAKE_LOCK

Using Permissions

- Applications can define new permissions in the manifest

  <permission android:name=“android.permission.VIBRATE”

  …

  </>

  - Do we really need a new permission?
  - normal / dangerous / signed
  - “Readable” explanation of the new permission
- Applications can require components interacting with them to have a specified permission, set in the manifest
  - By default all permissions apply to all components hosted by the application
    - Activities, Services etc.
  - Or per component permission requirements

Using Permissions

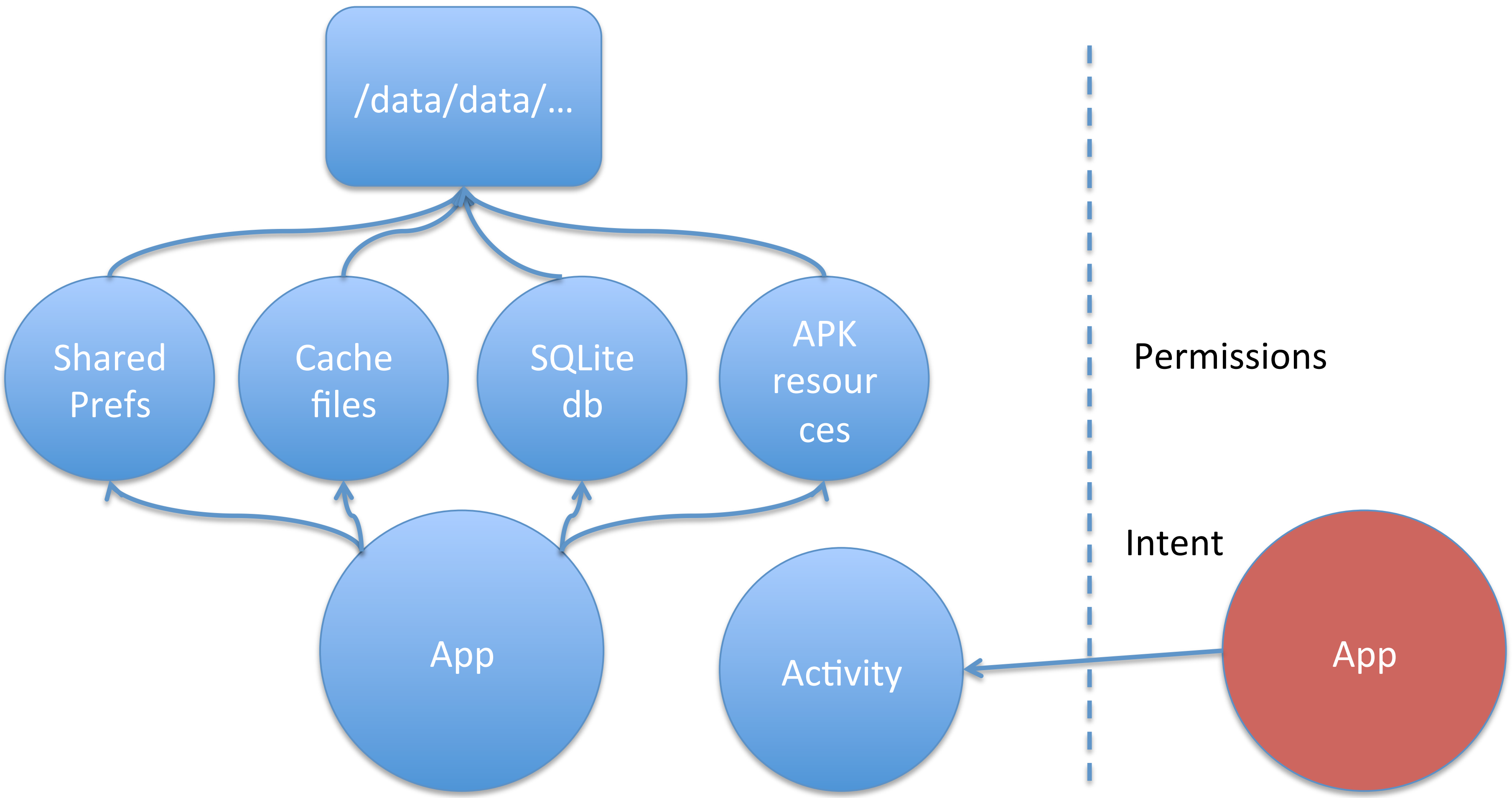
- Specify that an Application **uses** a permission

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

- Specify that an Application **requires** a permission
  - The app must **use** permissions it **requires**

```
 <provider
     android:permission="android.permission.READ_CONTACTS”
     android:authorities="com.example.martincontentprovider.MyProvider”
     android:multiprocess="true”
     android:name="com.example.martincontentprovider.MyProvider">
</provider>
```

Sharing Data – is this good enough?



References

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