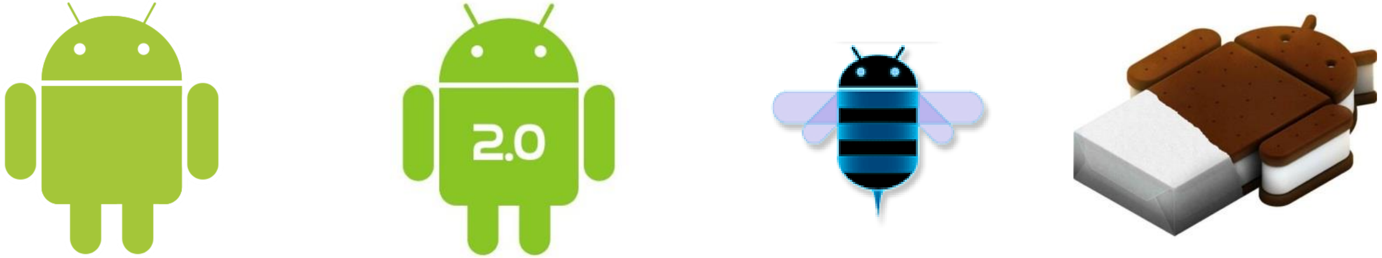


Chapter 10: Data management

outline

- Data Management in Android
  - Preferences
  - Text Files
  - XML Files
  - SQLite Database
  - Content Provider

MANAGING

Preferences: Key/Value pairs of data

Direct File I/O: Read/write files onboard or on SD cards. Remember to req uest permission for writing, for instance, on SD card

Database Tables: SQL Lite

Application Direct Access: Read only access from res assets/raw directories

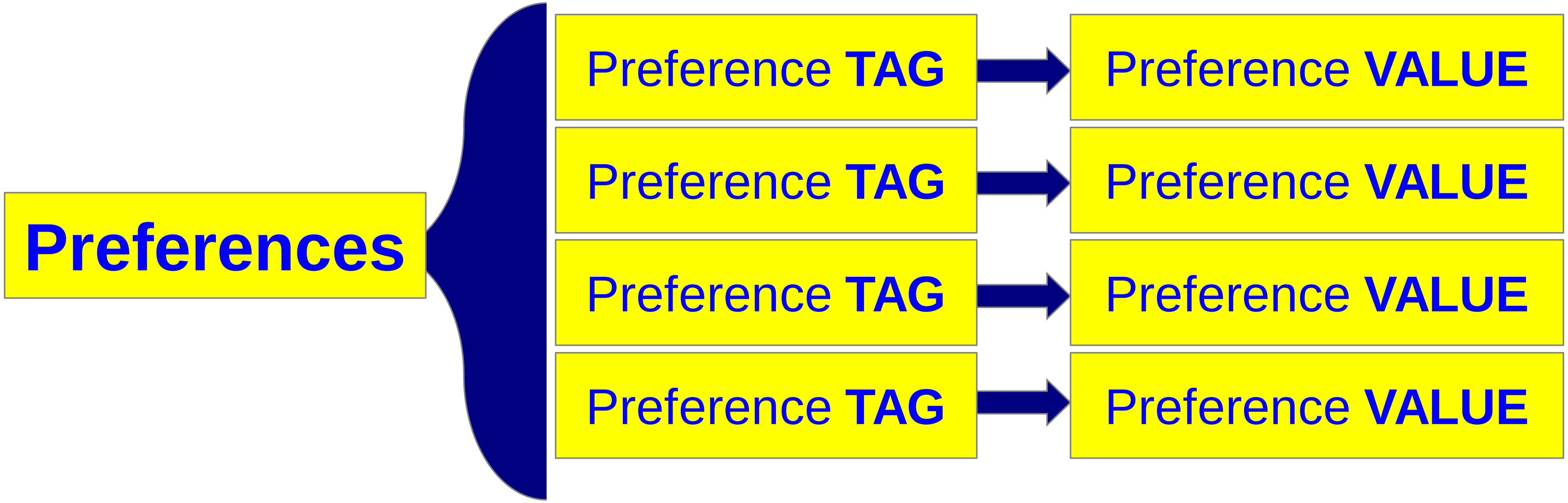
Increase functionality:

Content Providers: expose data to other applications

Services: background processes that run detached from any view

Preference

- Preferences are a convenient way to store configuration parameters
- Structured with a key-value mode



Preferences

Preference **TAG** → Preference **VALUE**

Preference **TAG** → Preference **VALUE**

Preference **TAG** → Preference **VALUE**

Preference **TAG** → Preference **VALUE**

Preferences

- Preferences could be either private or shared
  - Shared means that other applications could potentially read such preferences
  - Private means that they could be restricted at
    - Application level
    - Activity level

Preferences

- Shared preferences

```
getSharedPreferences(String name, Context.MODE_WORLD_READABLE);
getSharedPreferences(String name, Context.MODE_WORLD_WRITABLE);
```

- Private at application level

```
getSharedPreferences(String name, Context.MODE_PRIVATE);
```

- Private at activity level

```
getPreferences(int mode);
```

PREFERENCE

```
public void onCreate(Bundle savedInstanceState) {
    Super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    SharedPreferences pref = getSharedPreferences(MY_TAG,
       Context.MODE_PRIVATE);
    String myData = pref.getString(MY_KEY, “No pref”);
    TextView myView = (TextView)findViewById(R.id.myTextView);
    myView.setText(myData);
}
```

Preferences

- How to edit preferences?
- You need to a SharedPreferences.Editor

```
SharedPreferences.Editor editor = pref.edit();
editor.putString("mydata", et.getText().toString());
editor.commit();
```

- Be sure to commit operations at the end

Preferences

- Could be defined via XML
- Some specializations to ease the process
  - CheckBoxPreference
  - EditTextPreference
  - ListPreference
  - RingtonePreference
- Create a class that extends PreferenceActivity and call

```
addPreferencesFromResource(R.xml.mypreferences);
```

FileSystem

- Linux architecture
- User privileges
  - Quite limited
- Onboard data
  - Application's reserved data
- External data
  - SD card (/mnt/sdcard)

I/O

- Onboard
  - Write to a designated place for each application
  - **Where?** /data/data/<package>/files
  - **How?** Use standard java I/O classes
- SD card
  - **Where?** Environment.getExternalStorageDirectory()
  - **How?** Use standard java I/O classes
  - **Permissions?** android.permission.WRITE_EXTERNAL_STORAGE

how?

- Raw Text File
  - Place it under res/raw/ directory
  - Fill it with the text you like
  - Cannot edit it
  - Populate a TextView with it’s content inside the code

```
TextView tv = (TextView)findViewById(R.id.tv_main);
tv.setText(streamToString(R.raw.myfile));
```

```
private String streamToString(int id) {
    InputStream file = getResources().openRawResource(id);
    StringBuffer data = new StringBuffer();
    DataInputStream dataIO = new DataInputStream(file);
    String line = null;
    try {
        while ((line = dataIO.readLine()) != null)
            data.append(line + "\n");
        dataIO.close();
        file.close();
    } catch (IOException e) { }
    return data.toString();
   }
```

how?

- XML File
  - Place it under res/xml/ directory
  - Start the file with

    <?xml version=“1.0” encoding=“utf-8”?>
  - Add whatever you want with <mytag>value</mytag>

example

- We want to visualize all the grades of this class
- Our XML file is like this:

```
<student
    name="Student's name"
    class="Laboratorio di Applicazioni Mobili"
    year="2012"
    grade="30L" />
```

code example

```
XmlResourceParser grades = getResources().getXml(R.xml.myxmlfile);
LinearLayout ll = (LinearLayout)findViewById(R.id.myLL);   int tag = -1;
while (tag != XmlResourceParser.END_DOCUMENT) {
if (tag == XmlResourceParser.START_TAG) {
    String name = grades.getName();
        if (name.equals("student")) {
                TextView tv = new TextView(this);
                LayoutParams lp = new LayoutParams(LayoutParams.FILL_PARENT,
                            LayoutParams.WRAP_CONTENT);
            tv.setLayoutParams(lp);
                String toWrite = grades.getAttributeValue(null, "name") + …;
                tv.setText(toWrite);     ll.addView(tv);
        }    }
        try {       tag = grades.next();     } catch (Exception e) { }
    }
```

SQL

- General purpose solution
  - Lightweight database based on SQL
- Standard SQL syntax

```
SELECT name FROM table WHERE name = “Luca”
```

- Android gives a standard interface to SQL tables of other apps
- For application tables no content providers are needed

how?

- A database to store information
- Useful for structured informations
- Create a DBHelper which extends SQLiteOpenHelper
- Fill it with methods for managing the database
  - Better to use constants like
    - TABLE_GRADES
    - COLUMN_NAME
    - ….

example

- Our database will look like this:
  - grade table:
    - id: integer, primary key, auto increment
    - firstName: text, not null
    - lastName: text, not null
    - class: text, not null
    - grade: integer, not null

better to use constants

- Useful for query definition
- Our constants?

```
private static final String DB_NAME = “grades.db”;
private static final int DB_VERSION = 1;
public static final String TABLE_GRADES = “grades”;
public static final String COL_ID = “id”;
public static final String COL_FIRSTNAME = “firstName”;
public static final String COL_LASTNAME = “lastName”;
public static final String COL_CLASS = “class”;
public static final String COL_GRADE = “grade”;
```

creation code

- Constructor: call the superconstructor

```
Public mySQLiteHelper(Context context) {
     super(context, DB_NAME, null, DB_VERSION);
}
```

- onCreate(SQLiteDatabase db): create the tables

```
String sql_grade = “create table “ + TABLE_GRADES + “( “ +
     COL_ID + “ integer primary key autoincrement, “ +
     COL_FIRSTNAME + “ text not null, “ +
     COL_LASTNAME + “ text not null, “ +
     COL_CLASS + “ text not null, “ +
     COL_GRADE + “ text not null “);”;
db.execSQL(sql_grade);
```

insert code

- Create a public method, like insertDb(…)

```
mySQLiteHelper sql = new mySQLiteHelper(InsertActivity.this);
SQLiteDatabase db = mySQLiteHelper.getWritableDatabase();
ContentValues cv = new ContentValues();
cv.put(mySQLiteHelper.COL_FIRSTNAME, firstName);
cv.put(mySQLiteHelper.COL_LASTNAME, flastName);
cv.put(mySQLiteHelper.COL_FIRSTNAME, firstName);
cv.put(mySQLiteHelper.COL_FIRSTNAME, firstName);

long id = db.insert(mySQLiteHelper.TABLE_GRADES, null, values);
```

delete code

- Create a public method, like deleteDb(…)
- The delete method returns the number of rows affected
- Example:

```
db.delete(mySQLiteHelper.TABLE_GRADES, “id = ?”, new   String[]
{Integer.toString(id_to_delete)});
```

update code

- Create a public method, like updateDb(…)

```
ContentValues cv = new ContentValues();
values.put(mySQLiteHelper.FIRSTNAME, firstName);
values.put(mySQLiteHelper.LASTNAME, lastName);

db.update(mySQLiteHelper.TABLE_GRADES, values, “id = ?”, new  String[]
{Integer.toString(id_to_update));
```

search code

- Create a public method, like getFromDb(…)

```
Cursor gradeCursor = db.query(mySQLiteHelper.TABLE_GRADES,
     new String[]{mySQLiteHelper.COL_GRADE}, mySQLiteHelper.COL_ID + “ = “
     + id_to_search_for, null, null, null, null);
```

data handlers

- A Cursor stores data given by a DB query
- Some methods:
  - getCount()
  - moveTo{First,Next,Last,Position,Previous}()
  - close()
- You need to look inside the Cursor to see query’s results

```
while (gradeCursor.moveToNext()) {
      Log.v(“GRADES”,gradeCursor.getString(0));
}
```

methods

- Manipulating the cursor
  - cursor.moveToFirst()
  - while (cursor.moveToNext())
  - for (cursor.moveToFirst(); !cursor.isAfterLast(); cursor.moveToNext())
- Get column numbers from names
  - int nameColumn = cursor.getColumnIndex(People.NAME);
  - int phoneColumn = cursor.getColumnIndex(People.NUMBER);
- Get Data from column
  - String name = cursor.getString(nameColumn);
  - String number = cursor.getString(phoneColumn);

METHODS

- **Manipulate the cursor (row pointer)**
  - cursor.moveToFirst()
  - while (cursor.moveToNext()) { /* code */ }
  - for (cursor.moveToFirst(); !cursor.isAfterLast(); cur.moveToNext) { /* code */ }
- **Get column numbers from names**
  - int nameColumn = cursor.getColumnIndex(People.NAME);
  - int phoneColumn = cursor.getColumnIndex(People.NUMBER);
- **Get Data from column**
  - String name = cursor.getString(nameColumn);
  - String number = cursor.getString(phoneColumn);

Content Providers

- A system to access shared data
- Similar to a REST web service
- To each Content Provider, one or more URIs are assigned in the form:

content://<authority>/path

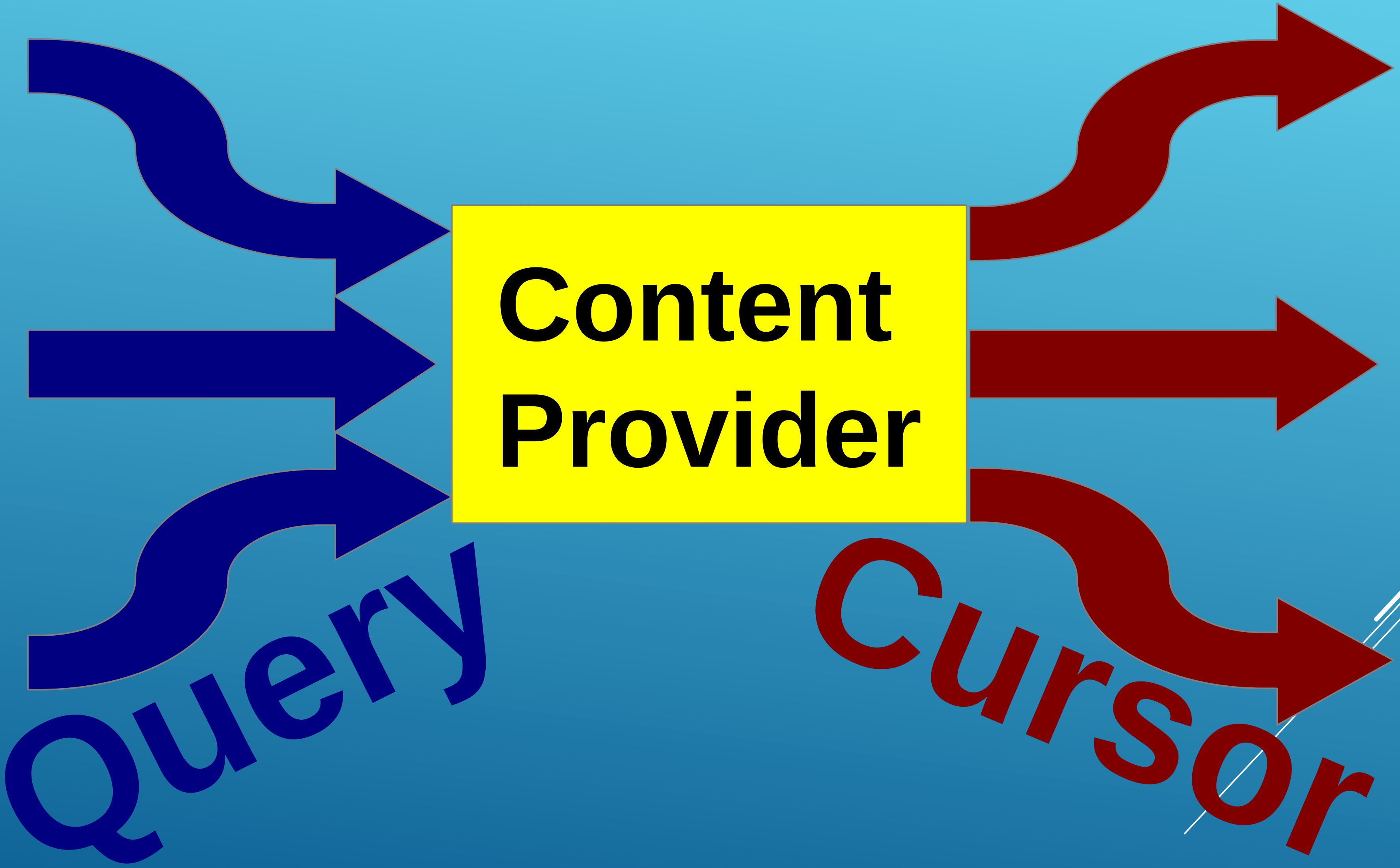
build

- Define the DB
- Create a class that extends android.content.ContentProvider
- Implement query(), insert(), update(), delete()
- Register the ContentProvider in the manifest

use

- Need to get the URI
  - Usually this is declared as public inside the content provider class
- Make a query, maybe adding some where clauses
  - You'll get a Cursor after that
- Navigate the Cursor

CONTENT PROVIDERS



contacts

- Query the contacts content provider
- Contacts information are shared among applications
- You need to request a permission

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

CONTACTS: CODE

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);


    Cursor cursor =
       getContentResolver().query(ContactsContract.Contacts.CONTENT_U
       RI, null, null, null, null);
    while (cursor.moveToNext()) {
       String contactName = cursor.getString(cursor.getColumnIndex(
         ContactsContract.Contacts.DISPLAY_NAME));
    }
    cursor.close();
}
```