Mobile Programming

Dr. Nader Mahmoud

Lecturer at Computer Science department

Course Content

- Chapter 1: Getting Started with Android Programming
- Chapter 2: Using Android Studio for Android Development
- Chapter 3: Activities, Fragments, and Intents
- Chapter 4: Getting to know the Android User Interface
- Chapter 5: Designing Your User Interface with Views
- Chapter 6: Displaying Pictures and Menus with Views
- Chapter 7: Data Persistence
- Chapter 8: Content Providers
- Chapter 9: Messaging
- Chapter 10: Location-Based Services
- Chapter 11: Networking
- Chapter 12: Developing Android Services

Sharing Data in Android

- A content provider behaves very much like a database—you can query it, edit its content, and add or delete content
- A content provider can use different ways to store its data. The data can be stored in a database, in files, or even over a network

Sharing Data in Android

Android ships with many useful content providers, including the following:

- **Browser**—Stores data such as browser bookmarks, browser history, and so on
- **CallLog**—Stores data such as missed calls, call details, and so on
- **Contacts**—Stores contact details
- **MediaStore**—Stores media files such as audio, video, and images
- **Settings**—Stores the device’s settings and preferences

Besides the many built-in content providers, you can also create your own content providers

Sharing Data in Android

- To query a content provider, you specify the query strin
in the form of a Uniform Resource Identifier (URI):
- *<standard_prefix>://<authority>/<data_path>/<id>*

1. *standard prefix*: content providers is always **content://**
2. *Authority*: name of the content provider
3. *data path*: specifies the data requested
(content://contacts/people)
4. *Id*: specifies the specific record requested

(content://contacts/people/2)

for contact number 2 in the Contacts content provider

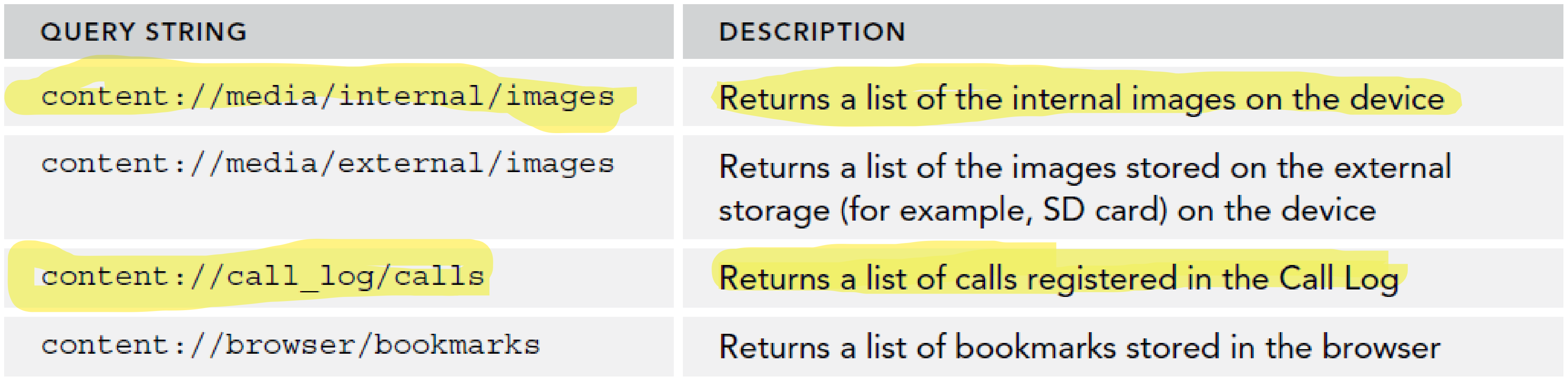
[illegible]

table

Sharing Data in Android

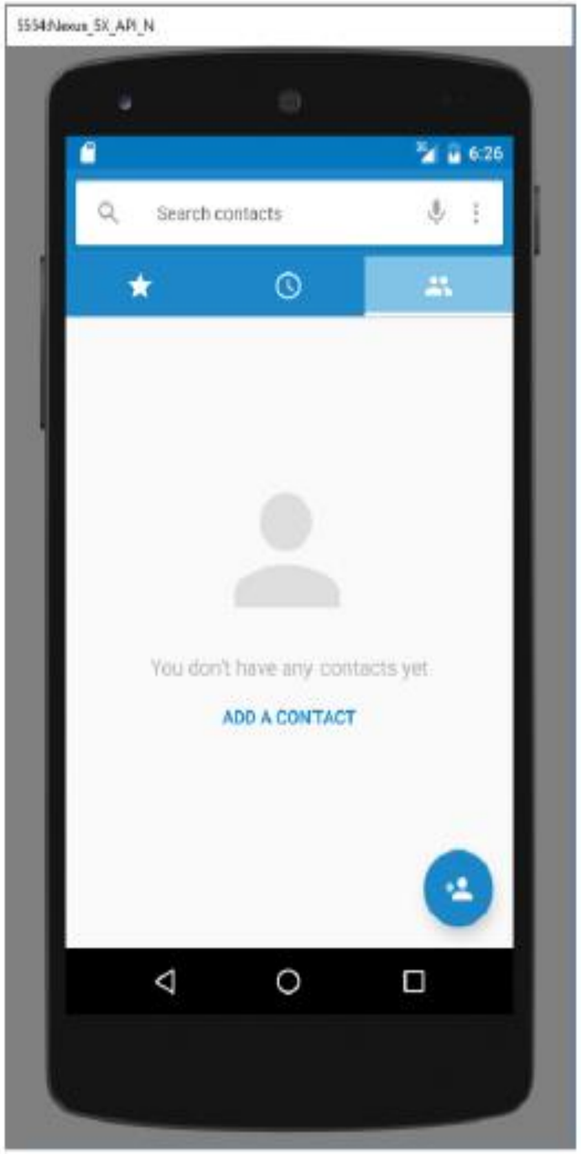
- To query a content provider, you specify the query string in the form of a Uniform Resource Identifier (URI):
- *<standard_prefix>://<authority>/<data_path>/<id>*

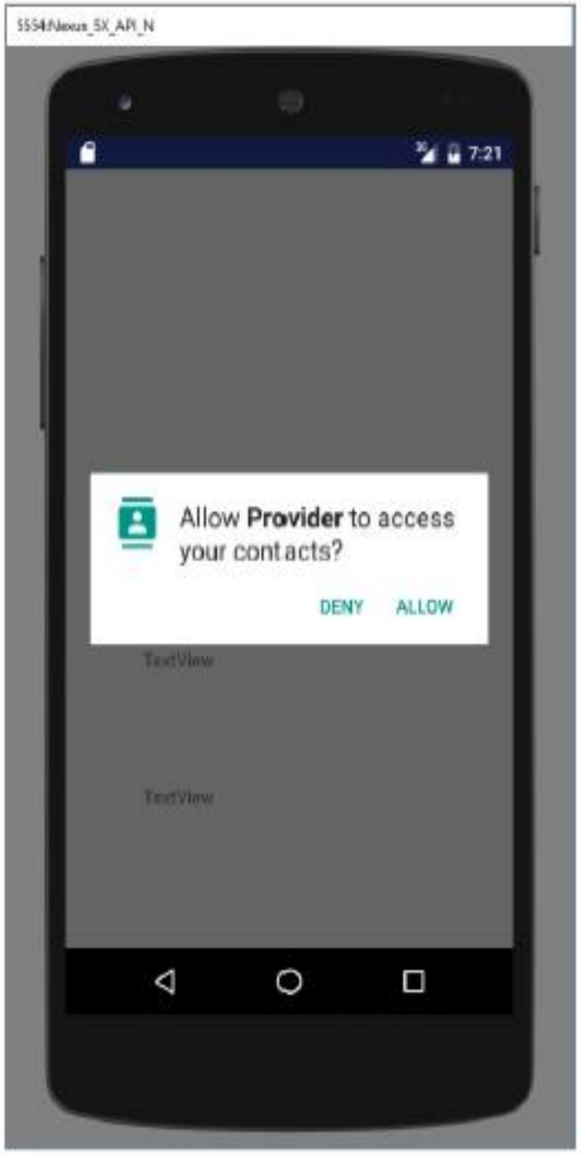
TABLE 8-1: Example Query Strings



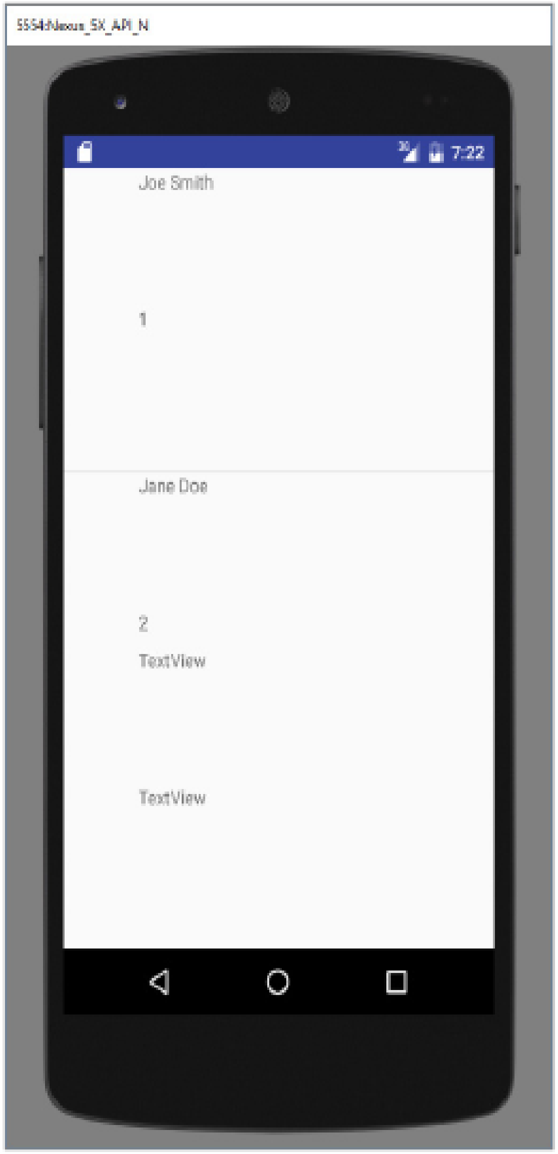
QUERY STRING	DESCRIPTION
`content://media/internal/images`	Returns a list of the internal images on the device
`content://media/external/images`	Returns a list of the images stored on the external storage (for example, SD card) on the device
`content://call_log/calls`	Returns a list of calls registered in the Call Log
`content://browser/bookmarks`	Returns a list of bookmarks stored in the browser

Using a Content Provider





Allow **Provider** to access your contacts?



Joe Smith

1

Jane Doe

2

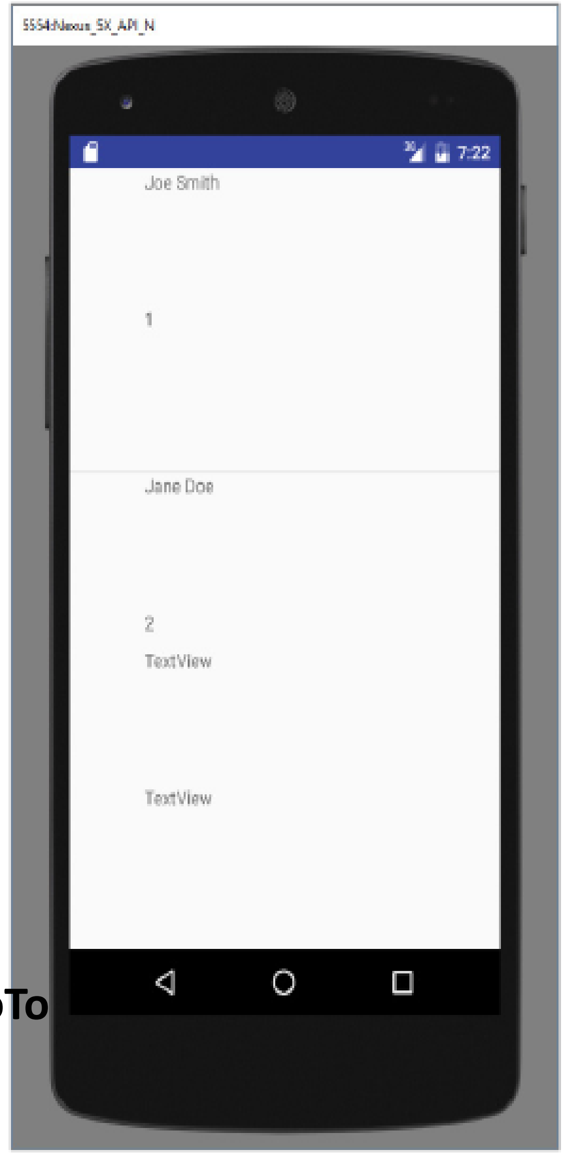
TextView

TextView

Using a Content Provider

activity_main.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android=
"http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/activity_main"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="com.jfdimarzio.provider.MainActivity">
<TextView
android:text="TextView"
android:layout_width="0dp"
android:layout_height="60dp"
android:id="@+id/contactName"
app:layout_constraintLeft_toLeftOf="@+id/activity_main"
android:layout_marginStart="app:layout_constraintBottom_toTo
pOf="@+id/contactID"
android:layout_marginBottom="40dp"
tools:layout_constraintBottom_creator="1" />
```


5554:Nexus_5X_API_N

7:22

Joe Smith

1

Jane Doe

2

TextView

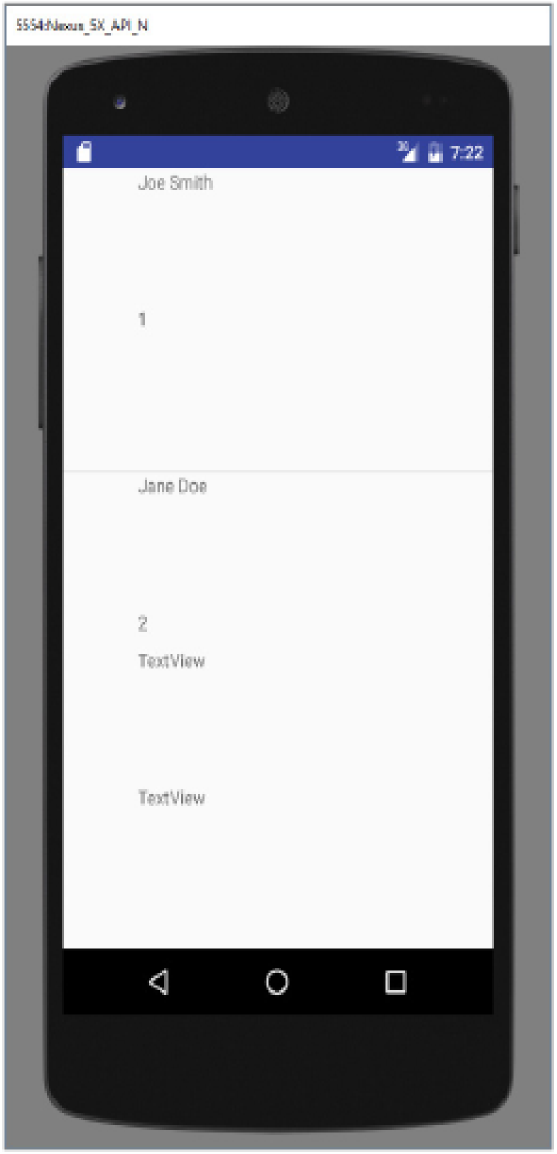
TextView

Using a Content Provider

activity_main.xml file.

```
<TextView
android:text="TextView"
android:layout_width="0dp"
android:layout_height="64dp"
android:id="@+id/contactID"
app:layout_constraintBottom_toBottomOf="@+id/activity_main"
android:layout_marginBottom="56dp"
tools:layout_constraintBottom_creator="1" />

<ListView
android:layout_height="0dp"
android:id="@android:id/list"
android:layout_width="wrap_content"
app:layout_constraintBottom_toTopOf="@+id/contactName"
android:layout_marginBottom="5dp"
tools:layout_constraintBottom_creator="1" />
</android.support.constraint.ConstraintLayout>
```



Joe Smith

1

Jane Doe

2

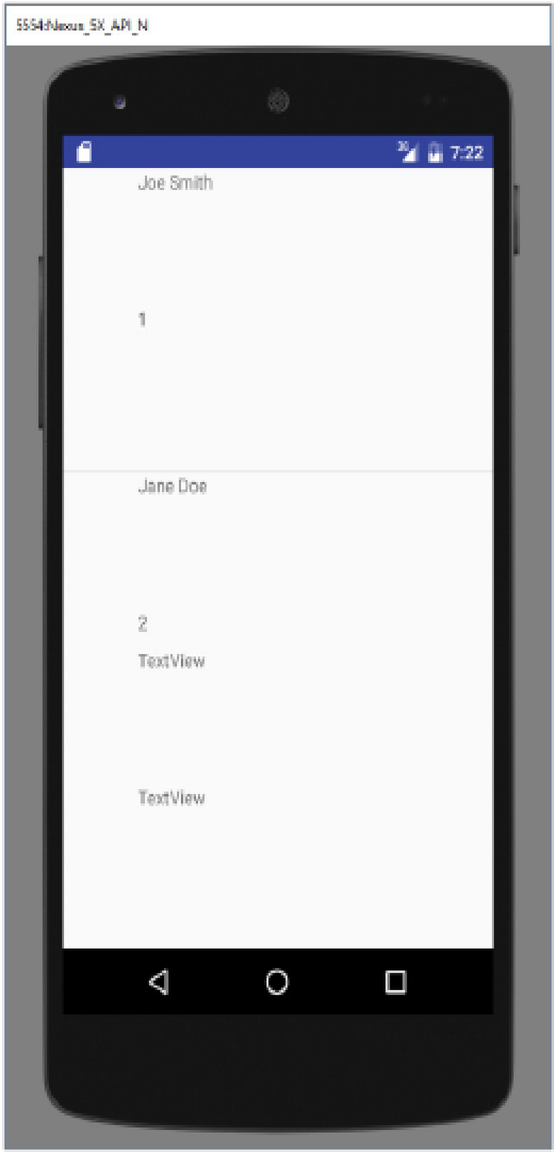
TextView

TextView

Using a Content Provider

MainActivity.java

```
import android.Manifest;
import android.app.ListActivity;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.net.Uri;
import android.provider.ContactsContract;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v4.content.CursorLoader;
import android.support.v4.widget.CursorAdapter;
import android.support.v4.widget.SimpleCursorAdapter;
import android.os.Bundle;
import android.widget.Toast;
public class MainActivity extends ListActivity {
final private int REQUEST_READ_CONTACTS = 123;
@Override
```



Joe Smith

1

Jane Doe

2

TextView

TextView

Using a Content Provider

MainActivity.java

```
protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        if (ContextCompat.checkSelfPermission(this,Manifest.permission.READ_CONTACTS)
        != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this,
              new String[]{Manifest.permission.READ_CONTACTS},
               REQUEST_READ_CONTACTS);
        } else{
        ListContacts();
    }
}
```

Using a Content Provider

MainActivity.java

```
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[]
grantResults) {

        switch (requestCode) {
        case REQUEST_READ_CONTACTS:
        if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                ListContacts();
        } else {
                Toast.makeText(MainActivity.this, "Permission Denied",
        Toast.LENGTH_SHORT).show();
        }
        break;
        default:
                super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        }
}
```

Using a Content Provider

To execute query

MainActivity.java

```
protected void ListContacts(){
    Uri allContacts = Uri.parse("content://contacts/people");
    Cursor c;
    CursorLoader cursorLoader = new
    CursorLoader(this,allContacts,null,null,null,null);
    c = cursorLoader.loadInBackground();
    String[] columns = new String[]{
        ContactsContract.Contacts.DISPLAY_NAME,ContactsContract.Contacts._ID
    };
    int[] views = new int[]{R.id.contactName, R.id.contactID};
    SimpleCursorAdapter adapter;
    adapter = new SimpleCursorAdapter( this, R.layout.activity_main, c, columns,
    views, CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
    this.setListAdapter(adapter);
    }
}
```

The contract between the contacts provider and applications. Contains definitions for the supported URIs and columns.

to observe the new contact if I have added it

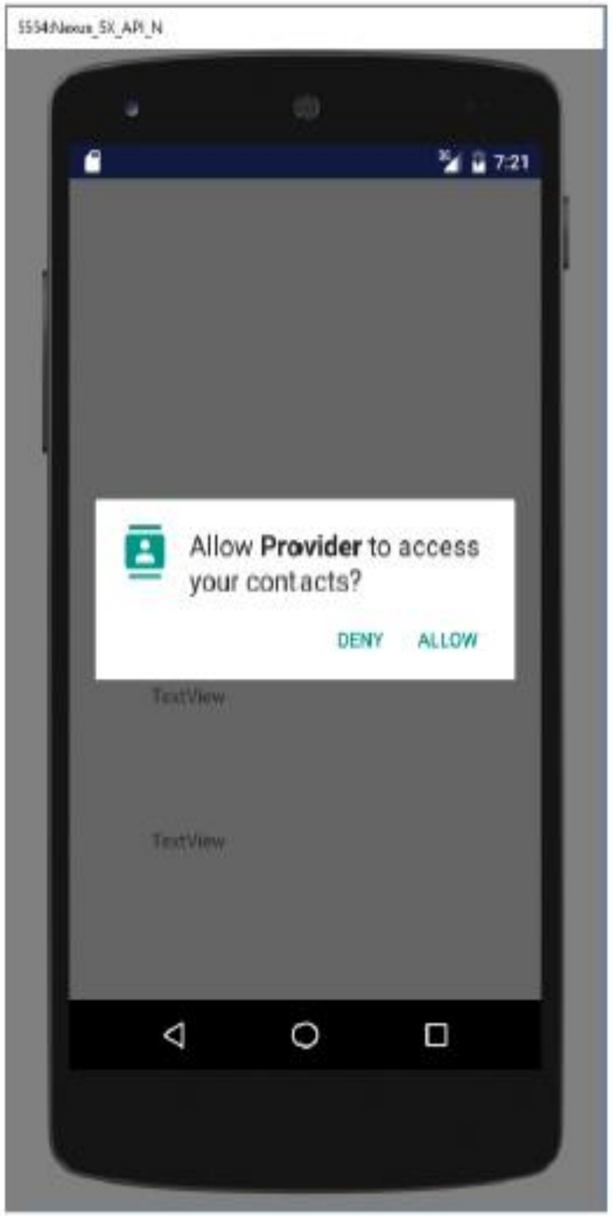
The SimpleCursorAdapter object maps a cursor to TextViews (or ImageViews) defined in your XML file (activity_main.xml). It maps the data (as represented by columns) to views (as represented by views)

Using a Content Provider

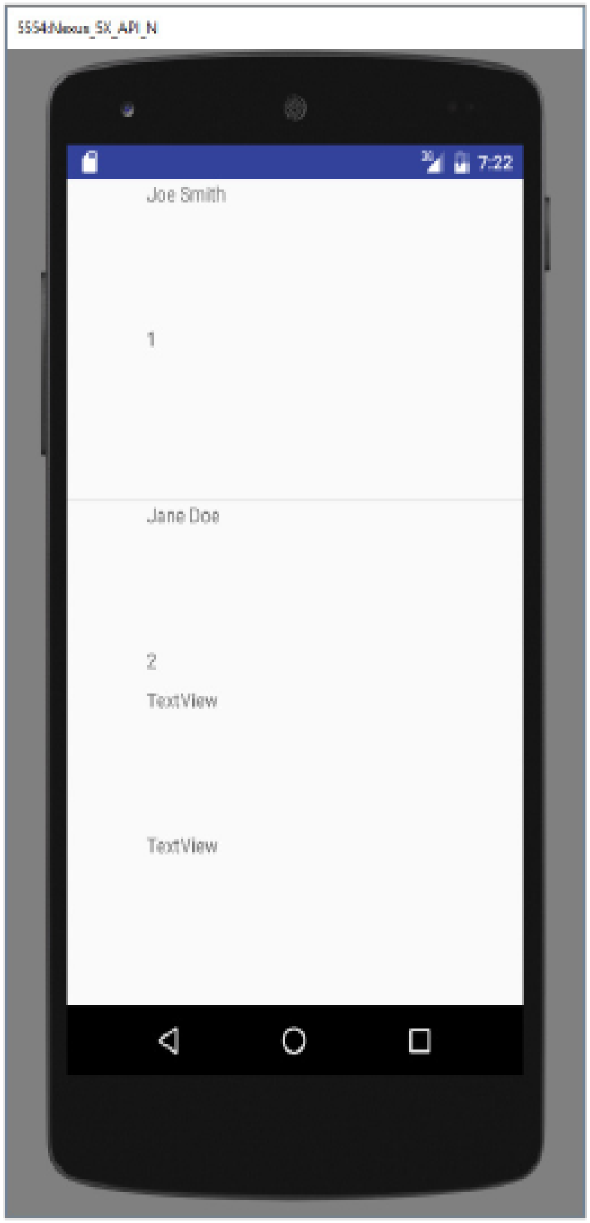
Manifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.jfdimarzio.provider">
<uses-permission android:name="android.permission.READ_CONTACTS"/>
    <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
    </activity>
    </application>
</manifest>
```

Using a Content Provider



Allow **Provider** to access your contacts?



Joe Smith

1

Jane Doe

2

TextView

TextView

Predefined Query String Constants

Besides using the query URI, you can use a list of predefined query string constants in Android to specify the URI for the different data types. For example, besides using the query content: //contacts/people, you can rewrite this statement:

```
Uri allContacts = Uri.parse("content://contacts/people");
```

using one of the predefined constants in Android, as follows:

```
Uri allContacts = ContactsContract.Contacts.CONTENT_URI;
```

Predefined Query String Constants

Prints the ID and name of each contact stored in the Contacts application

```
private void PrintContacts(Cursor c)
{
        if (c.moveToFirst()) {
                do{
                                String contactID = c.getString(c.getColumnIndex(ContactsContract.Contacts._ID));
                                String contactDisplayName =
                                c.getString(c.getColumnIndex( ContactsContract.Contacts.DISPLAY_NAME));
                                Log.v("Content Providers", contactID + ", " + contactDisplayName);
                } while (c.moveToNext());
        }
}
```

The PrintContacts() method prints the following in the logcat window:

```
12-13 08:32:50.471: V/Content Providers(12346): 1, Wei-Meng Lee
12-13 08:32:50.471: V/Content Providers(12346): 2, Linda Chen
12-13 08:32:50.471: V/Content Providers(12346): 3, Joanna Yip
```

Predefined Query String Constants

```
private void PrintContacts(Cursor c)
{
	if (c.moveToFirst()) {
		do{
			String contactID = c.getString(c.getColumnIndex(ContactsContract.Contacts._ID));
			String contactDisplayName =
			c.getString(c.getColumnIndex( ContactsContract.Contacts.DISPLAY_NAME));
			Log.v("Content Providers", contactID + ", " + contactDisplayName);
			//---get phone number---
			Cursor phoneCursor = getContentResolver().query(
			ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
			ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = " +contactID, null,
			null);
		while (phoneCursor.moveToNext()) {
			Log.v("Content Providers", phoneCursor.getString(phoneCursor.getColumnIndex(
						ContactsContract.CommonDataKinds.Phone.NUMBER)));
		}
		phoneCursor.close();
		} while (c.moveToNext());
	}
}
```

The getContentResolver() method returns a ContentResolver object, which helps to resolve a content URI with the appropriate content provider.

The phone number in another content provider so he can use it by cursor loader like the last example or use it by a Predefined String

Note *To access the phone number of a contact, you need to query against the URI stored in* ***ContactsContract.CommonDataKinds.Phone. CONTENT_URI***

Predefined Query String Constants

```
12-13 08:59:31.881: V/Content Providers(13351): 1, Wei-Meng Lee
12-13 08:59:32.311: V/Content Providers(13351): +651234567
12-13 08:59:32.321: V/Content Providers(13351): 2, Linda Chen
12-13 08:59:32.511: V/Content Providers(13351): +1 876-543-21
12-13 08:59:32.545: V/Content Providers(13351): 3, Joanna Yip
12-13 08:59:32.641: V/Content Providers(13351): +239 846 5522
```

Predefined Query String Constants

Projections

The third parameter for the CursorLoader class controls how many columns are returned by the query. This parameter is known as the *projection*. Earlier, you specified null:

```
Cursor c;
CursorLoader cursorLoader = new CursorLoader(
        this,
        allContacts,    → uri
        null,
        null,
        null ,
        null);
c = cursorLoader.loadInBackground();
```

Predefined Query String Constants

Projections

You can specify the exact columns to return by creating an array containing the name of the column to return, like this:

```
String[] projection = new String[] {ContactsContract.Contacts._ID,
            ContactsContract.Contacts.DISPLAY_NAME,
            ContactsContract.Contacts.HAS_PHONE_NUMBER};
Cursor c;
CursorLoader cursorLoader = new CursorLoader(
      this,
      allContacts,
      projection,
      null,
      null ,
      null);
c = cursorLoader.loadInBackground();
```

Predefined Query String Constants

Filtering $\longrightarrow$ where = Selection

The fourth and fifth parameters for the CursorLoader class enable you to specify a SQL WHERE clause to filter the result of the query.

For example, the following statement retrieves only the people whose name ends with “Lee”:

```
Cursor c;
CursorLoader cursorLoader = new CursorLoader(
        this,
        allContacts,
        projection,
        ContactsContract.Contacts.DISPLAY_NAME + " LIKE '%Lee'",
        null ,
        null);
c = cursorLoader.loadInBackground();
```

any name
ends with lee

ContactsContract.Contacts.CONTACT_ID+"="+5,

Predefined Query String Constants

Filtering

The fourth and fifth parameters for the CursorLoader class enable you to specify a SQL WHERE clause to filter the result of the query.

For example, the following statement retrieves only the people whose name ends with "Lee":

```
Cursor c;
//---Honeycomb and later---
CursorLoader cursorLoader = new CursorLoader(
        this,
        allContacts,
        projection,
        ContactsContract.Contacts.DISPLAY_NAME + " LIKE ?",
        new String[] {"%Lee"},
        null);
c = cursorLoader.loadInBackground();
```

it splits the query into two parts

$\rightarrow$ value of where

Predefined Query String Constants

Sorting

The last parameter of the CursorLoader class enables you to specify a SQL ORDER BY clause to sort the result of the query.

For example, the following statement sorts the contact names in ascending order:

```
Cursor c;
CursorLoader cursorLoader = new CursorLoader(
        this,
        allContacts,
        projection,
        ContactsContract.Contacts.DISPLAY_NAME + " LIKE ?",
        new String[] {"%Lee"},
        ContactsContract.Contacts.DISPLAY_NAME + " ASC");
c = cursorLoader.loadInBackground();
```

Creating Your Own Content Providers

- Creating your own content provider in Android is relatively simple.
- All you need to do is extend the abstract ContentProvider class and override the various methods defined within it.

➤ getType()—Returns the MIME type of the data at the given URI.
➤ onCreate()—Called when the provider is started.
➤ query()—Receives a request from a client. The result is returned as a Cursor object.
➤ insert()—Inserts a new record into the content provider.
➤ delete()—Deletes an existing record from the content provider.
➤ update()—Updates an existing record from the content provider.

Creating Your Own Content Providers

```
public class BooksProvider extends ContentProvider {
    static final String PROVIDER_NAME = "com.jfdimarzio.provider.Books";
    static final Uri CONTENT_URI = Uri.parse("content://"+
        PROVIDER_NAME + "/books");
    static final String _ID = "_id";
    static final String TITLE = "title";
    static final String ISBN = "isbn";
    static final int BOOKS = 1;
    static final int BOOK_ID = 2;
    private static final UriMatcher uriMatcher;
    static{
        uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        uriMatcher.addURI(PROVIDER_NAME, "books", BOOKS);
        uriMatcher.addURI(PROVIDER_NAME, "books/#", BOOK_ID);
    }
```

UriMatcher object to parse the content URI that is passed to the content provider through a ContentResolver

Within your content provider, you are free to choose how you want to store your data—in a traditional file system, XML, a database, or even through web services. For this example, you use the SQLite database approach

Creating Your Own Content Providers

```
//---for database use---
SQLiteDatabase booksDB;
static final String DATABASE_NAME = "Books";
static final String DATABASE_TABLE = "titles";
static final int DATABASE_VERSION = 1;
static final String DATABASE_CREATE = "create table " + DATABASE_TABLE +
" (_id integer primary key autoincrement, “ + "title text not null, isbn text not null);";

private static class DatabaseHelper extends SQLiteOpenHelper{
        DatabaseHelper(Context context) {
                super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }
        public void onCreate(SQLiteDatabase db){
                db.execSQL(DATABASE_CREATE);
        }
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
                Log.w("Provider database", "Upgrading database from version " +
                oldVersion + " to " + newVersion + ", which will destroy all old data");
                db.execSQL("DROP TABLE IF EXISTS titles");
                onCreate(db); } } End of inner class
```

Creating Your Own Content Providers

```
@Override
public boolean onCreate() {
        Context context = getContext();
        DatabaseHelper dbHelper = new DatabaseHelper(context);
        booksDB = dbHelper.getWritableDatabase();
        return (booksDB == null)? false:true;
}
```

Creating Your Own Content Providers

```
@Override
public Uri i
    //---ad
    long ro

    //---if a
    if (row
    {
        Ur                                                        ;
        ge
        re
    }
    throw
}
```

```
 public long insertContact(String name, String
email) {
      ContentValues    initialValues    =    new
ContentValues();
      initialValues.put(KEY_NAME, name);
      initialValues.put(KEY_EMAIL, email);
      return   db.insert(DATABASE_TABLE,   null,
initialValues);
   }
```

you call the notifyChange() method of the **ContentResolver**.
This notifies registered observers that a row was updated.

Creating Your Own Content Providers

```
@Override
public Cursor query(Uri uri, String[] projection, String selection, String[]
selectionArgs, String sortOrder) {
    SQLiteQueryBuilder sqlBuilder = new SQLiteQueryBuilder();
```

```
 //---retrieves a particular contact---
   public Cursor getContact(long rowId) throws SQLException {
      Cursor mCursor =
              db.query(true,  DATABASE_TABLE,  new  String[]{KEY_ROWID,
KEY_NAME,
                           KEY_EMAIL}, KEY_ROWID + "=" + rowId, null,  null,
null, null, null);
      if (mCursor != null) {
         mCursor.moveToFirst();
      }
      return mCursor;
   }
```

```
et(1));
```

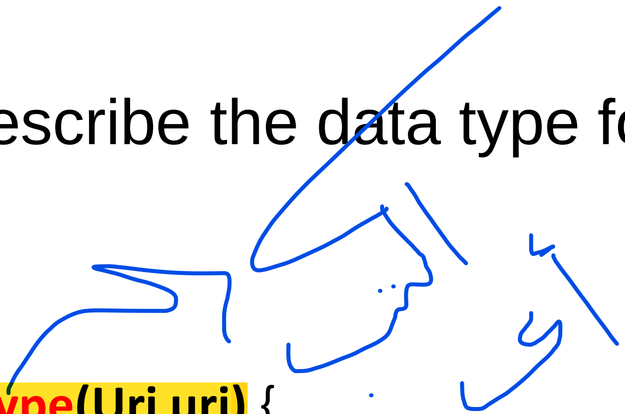
```
    return c;
}
```

Creating Your Own Content Providers

```
@Override
public int delete(Uri arg0, String arg1, String[] arg2) {
    // arg0 = uri
    // arg1 = selection
    // arg2 = selectionArgs
    int count=0;
    switch (uriMatcher.match(arg0)){
        case BOOKS:
                count = booksDB.delete(DATABASE_TABLE,arg1,arg2);
        break;
        case BOOK_ID:
                String id = arg0.getPathSegments().get(1);
                count = booksDB.delete(DATABASE_TABLE,_ID + " = " + id
               +(!TextUtils.isEmpty(arg1) ? " AND (" +arg1 + ')' : ""),arg2);
        break;
        default: throw new IllegalArgumentException("Unknown URI " + arg0);
    }
    getContext().getContentResolver().notifyChange(arg0, null);
    return count;
}
```

Creating Your Own Content Providers

To uniquely describe the data type for your content provider



implementation

```
@Override
public String getType(Uri uri) {
      switch (uriMatcher.match(uri)){
            //---get all books---
            case BOOKS:
                        return "vnd.android.cursor.dir/vnd.learn2develop.books ";
            //---get a particular book---
            case BOOK_ID:
                        return "vnd.android.cursor.item/vnd.learn2develop.books ";
            default:
                        throw new IllegalArgumentException("Unsupported URI: " + uri);
      }
}
```

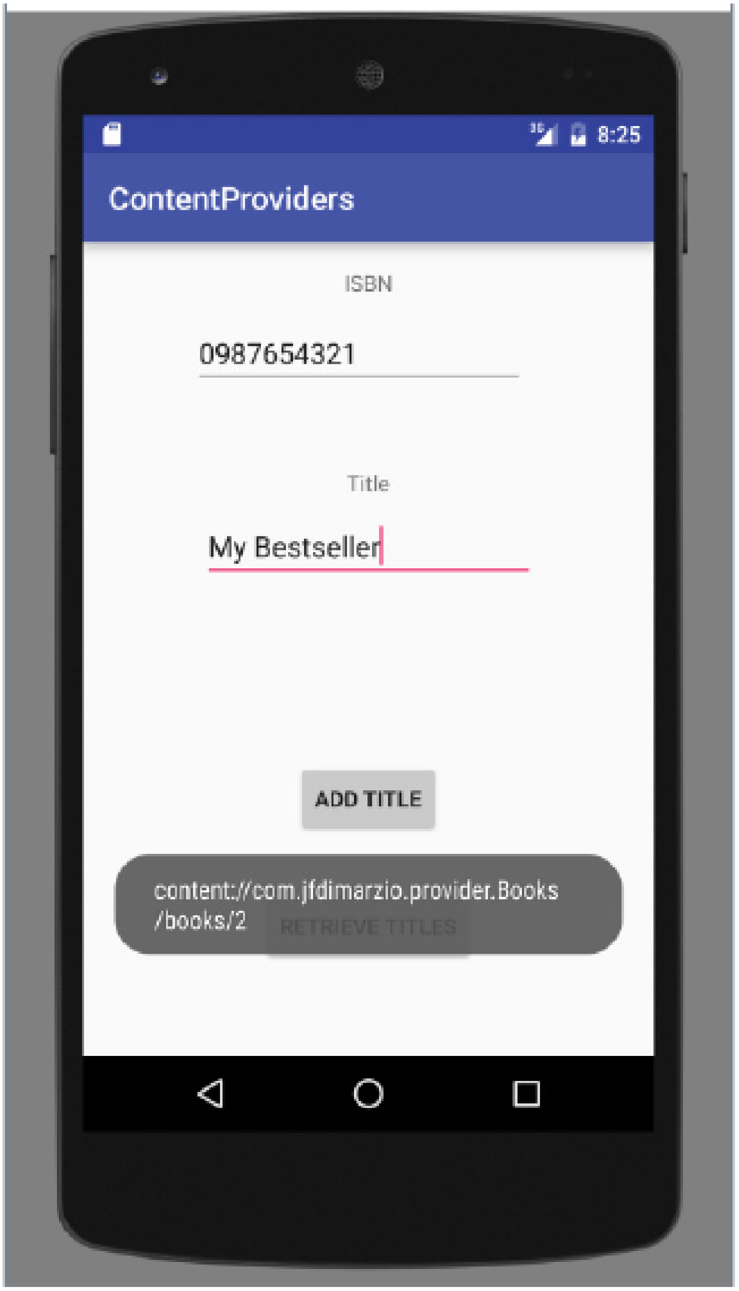
Creating Your Own Content Providers

```
@Override
public int update(Uri uri, ContentValues values, String selection, String[]
selectionArgs) {
	int count = 0;
	switch (uriMatcher.match(uri)){
		case BOOKS:
				count = booksDB.update( DATABASE_TABLE, values, selection,
				selectionArgs);
		break;
		case BOOK_ID:
				count = booksDB.update( DATABASE_TABLE, values, _ID + " = " +
				uri.getPathSegments().get(1) +(!TextUtils.isEmpty(selection) ? " AND
				(" +selection + ')' : ""), selectionArgs);
		break;
		default: throw new IllegalArgumentException("Unknown URI " + uri);
	}
	getContext().getContentResolver().notifyChange(uri, null);
	return count;
	}
}
```

Creating Your Own Content Providers

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.jfdimarzio.contentproviders">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
        </activity>
        <provider android:name="BooksProvider“
        android:authorities="com.jfdimarzio.provider.Books">
        </provider>
    </application>
</manifest>
```

Using the Content Provider



0987654321

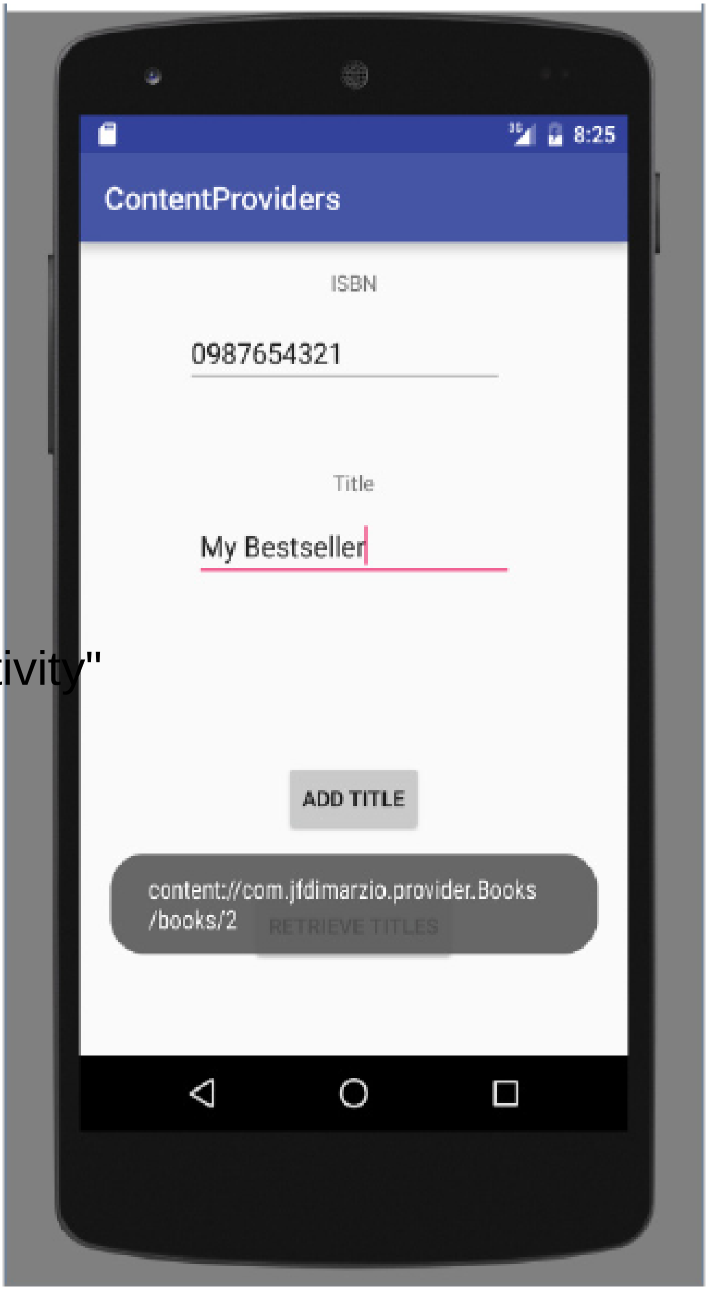
My Bestseller

content://com.jfdimarzio.provider.Books
/books/2

Using the Content Provider

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android=
"http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/activity_main"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="com.jfdimarzio.contentproviders.MainActivity"
>
    <TextView
    android:text="ISBN"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/textView"
    />
```

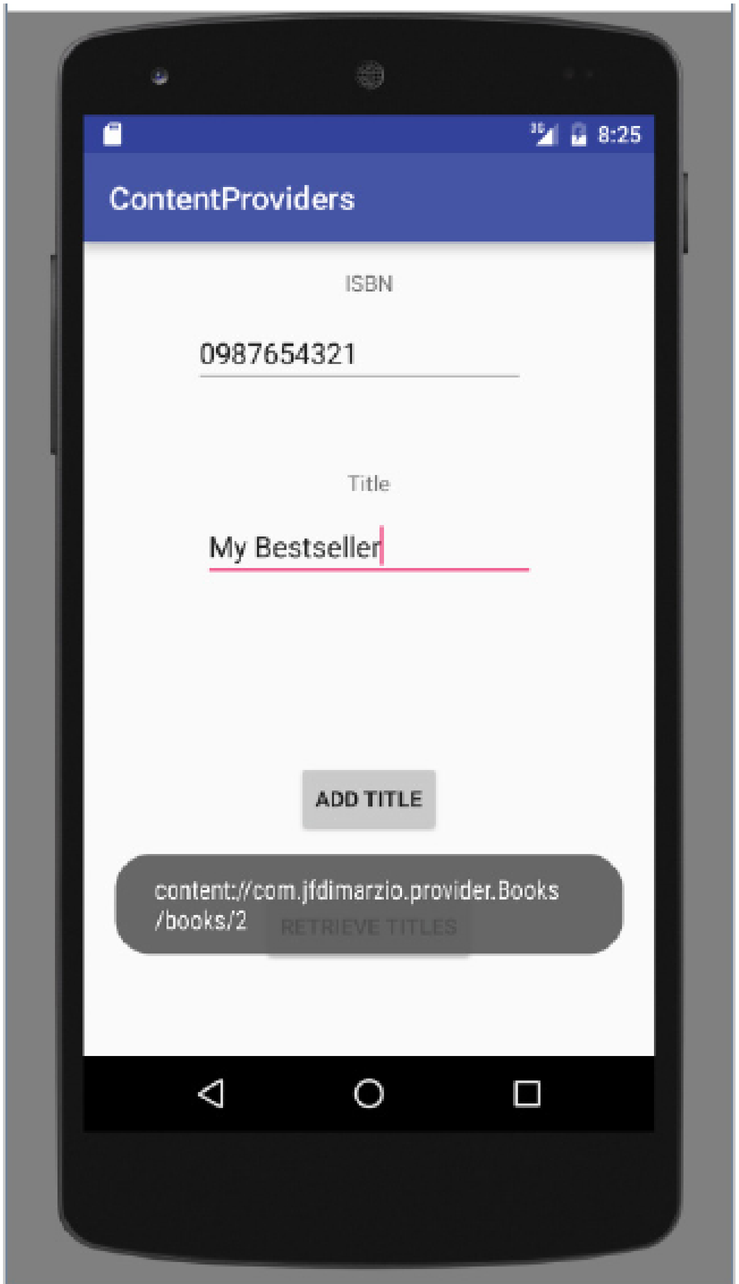


content://com.jfdimarzio.provider.Books/books/2

Using the Content Provider

activity_main.xml

```
<Button
android:text="Retrieve Titles"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/btnRetrieve"
/>
</android.support.constraint.ConstraintLayout>
```

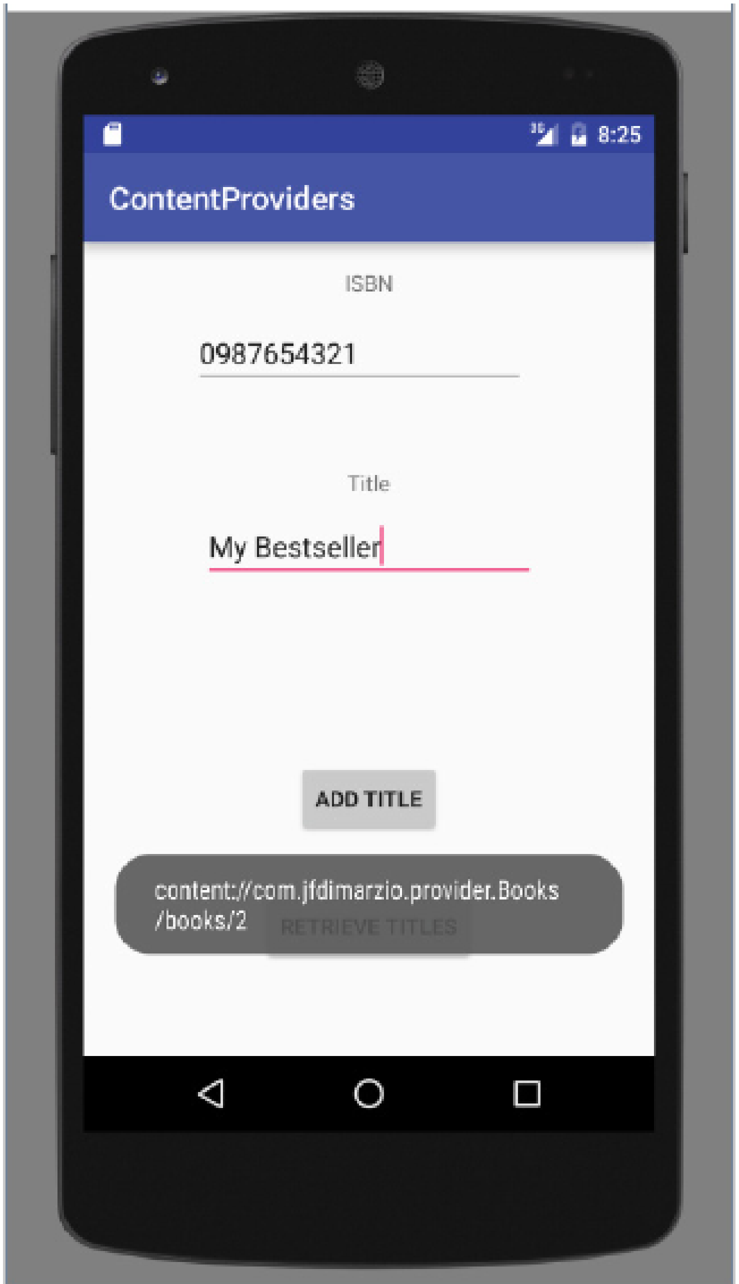


content://com.jfdimarzio.provider.Books
/books/2

Using the Content Provider

activity_main.xml

```
<EditText
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:inputType="text"
android:ems="10"
android:id="@+id/txtISBN"/>
<TextView
android:text="Title"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/textView2"/>
<EditText
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/txtTitle"/>
<Button
android:text="Add Title"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/btnAdd“>
```



0987654321

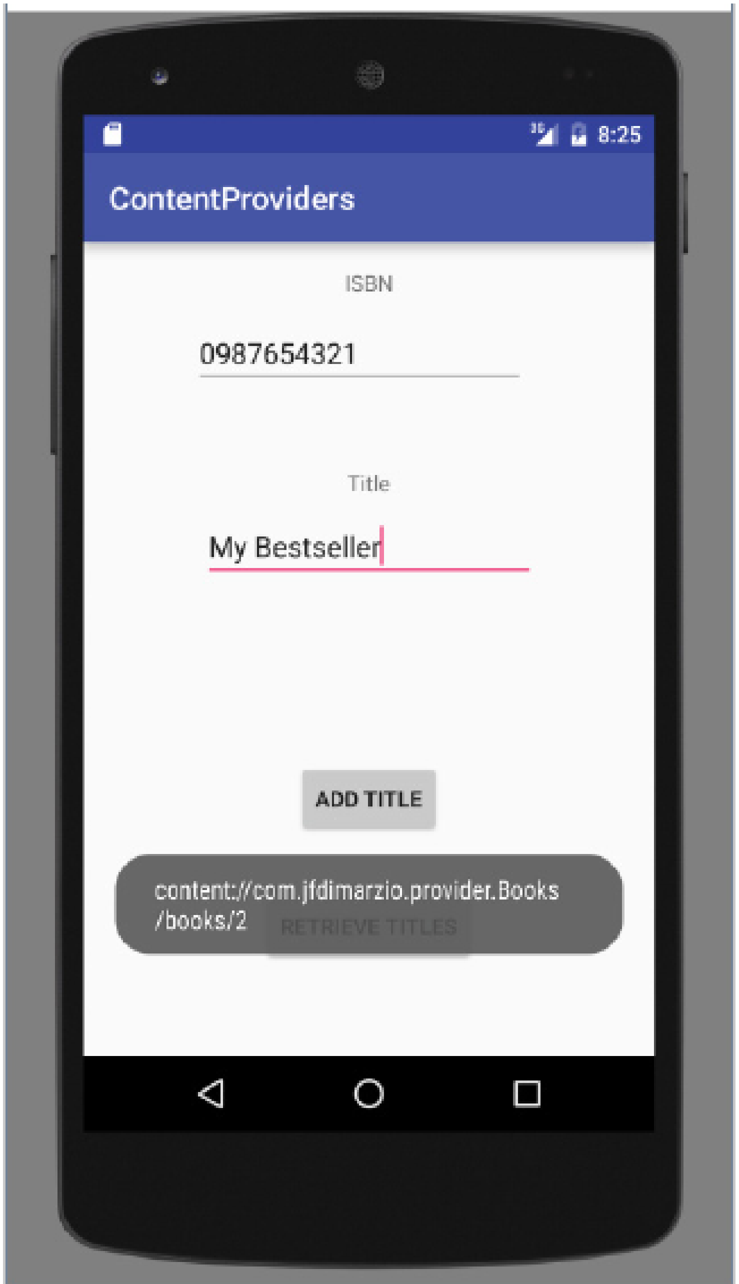
My Bestseller

content://com.jfdimarzio.provider.Books
/books/2

Using the Content Provider

MainActivity.java

```
import android.content.ContentValues;
import android.content.CursorLoader;
import android.database.Cursor;
import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
}
```



ISBN: 0987654321

Title: My Bestseller

content://com.jfdimarzio.provider.Books/books/2

Using the Content Provider

MainActivity.java

```
public void onClickAddTitle(View view) {
//---add a book---
        ContentValues values = new ContentValues();
        values.put(BooksProvider.TITLE, ((EditText)

        findViewById(R.id.txtTitle)).getText().toString());
        values.put(BooksProvider.ISBN, ((EditText)

        findViewById(R.id.txtISBN)).getText().toString());
        Uri uri = getContentResolver().insert( BooksProvider.CONTENT_URI, values);
        Toast.makeText(getBaseContext(),uri.toString(), Toast.LENGTH_LONG).show();
}
```

Using the Content Provider

MainActivity.java

```
public void onClickRetrieveTitles(View view) {
        //---retrieve the titles---
        Uri allTitles = Uri.parse( "content://com.jfdimarzio.provider.Books/books");
        Cursor c;
        CursorLoader cursorLoader = new CursorLoader( this, allTitles, null, null, null, "title
                                                desc");
        c = cursorLoader.loadInBackground();
        if (c.moveToFirst()) {
                do{
                Toast.makeText(this, c.getString(c.getColumnIndex( BooksProvider._ID)) + ", " +
                c.getString(c.getColumnIndex( BooksProvider.TITLE)) + ", " +
                c.getString(c.getColumnIndex(
                BooksProvider.ISBN)),Toast.LENGTH_SHORT).show();
                } while (c.moveToNext());
        }
        }
}
```

End of Lecture