

CS 696 Mobile Phone Application Development  
Fall Semester, 2009  
Doc 5 Content Providers  
Sept 16, 2009

Copyright ©, All rights reserved. 2009 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (<http://www.opencontent.org/opl.shtml>) license defines the copyright on this document.

## References

Google Android Documentation,

<http://developer.android.com/reference/packages.html>

<http://developer.android.com/guide/topics/providers/content-providers.html>

CS 683 Class Notes, <http://www.eli.sdsu.edu/courses/fall08/cs683/notes/index.html>

# Android 1.6

Released Sept 15

Adds

- CDMA

- Multiple screen sizes

- Quick search

- Text-to-speech

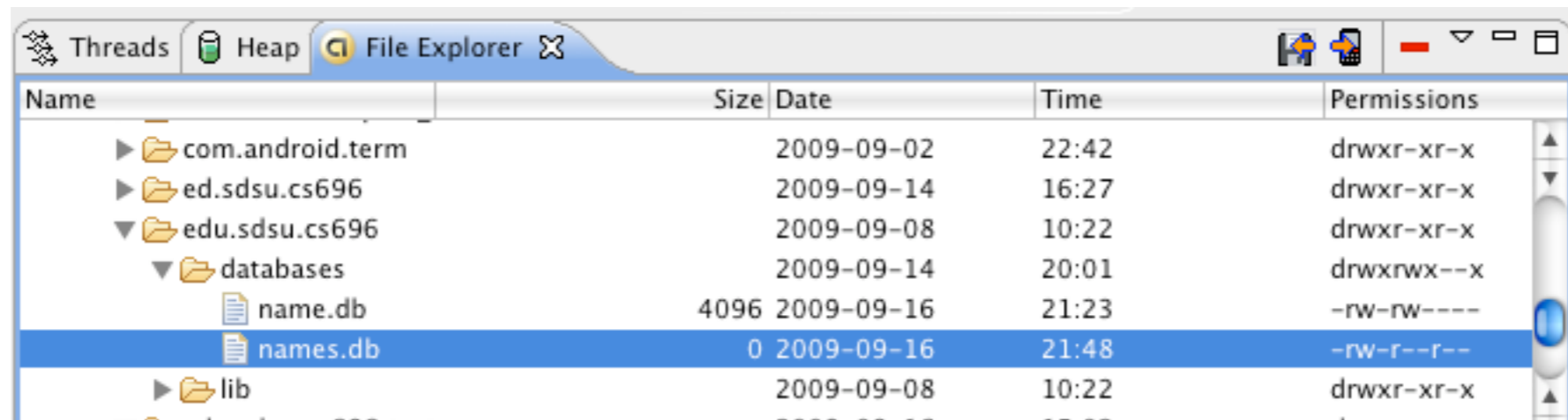
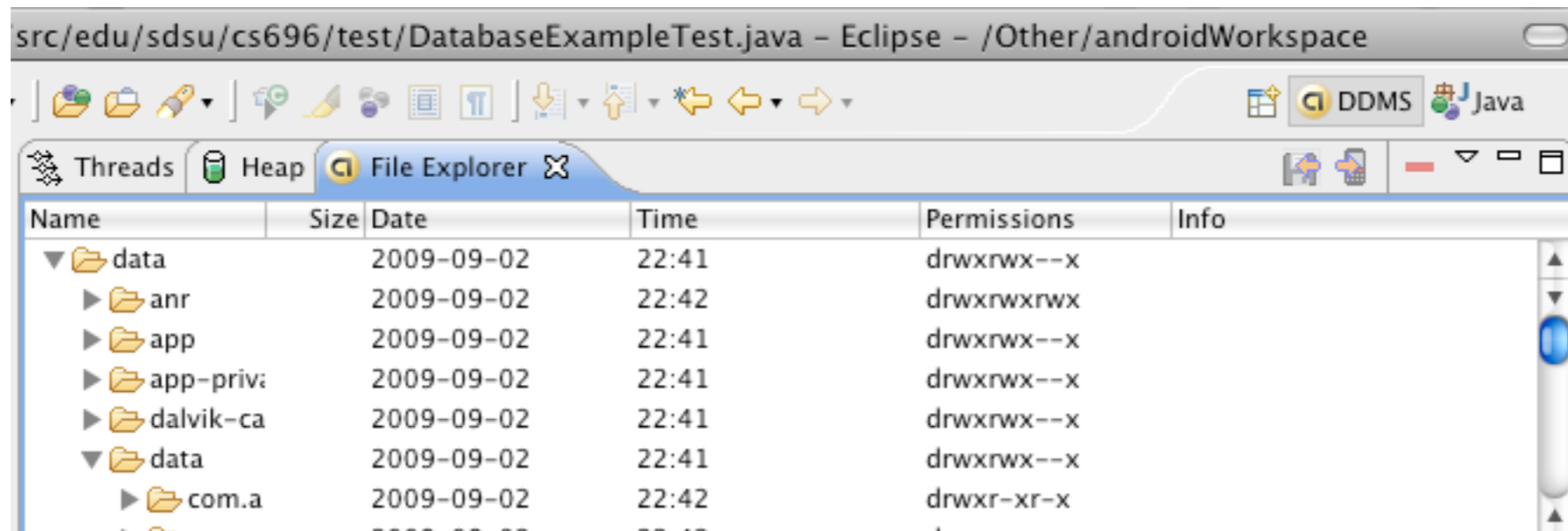
- New Gestures framework

# Accessing the database

# Using adb

```
AI pro 17->adb -s emulator-5554 shell
# sqlite3 /data/data/edu.sdsu.cs696/databases/name.db
SQLite version 3.5.9
Enter ".help" for instructions
sqlite> select * from NAMES;
1|Roger
2|Roger
sqlite>
```

# Eclipse DDMS Perspective



# ContentProvider

# Content Provider

Only way to share data across applications

Can

Retrieve

Modify

Create



# Android Native Content Providers

Contacts

MediaStore.Audio

MediaStore.Images

MediaStore.Video

# Basic Idea

Datum mapped to a URI

URI is used to access data by client

content://contacts/people/  
All contact names

content://contacts/people/23  
Contact with `_ID = 21`

# Sample Query

```
myPerson = Uri.parse("content://contacts/people/23");  
Cursor cur = managedQuery(myPerson, null, null, null, null);
```

Second way to create URI

```
Uri myPerson = ContentUris.withAppendedId(People.CONTENT_URI, 23);
```

# CONTENT\_URI

Content URI can be complex

Rather than hardcoding exact URI everywhere

Content Providers expose the base URI as static field

NameProvider.CONTENT\_URI

```
public static final String AUTHORITY = "edu.sdsu.cs696.NameProvider";  
public static final Uri CONTENT_URI = Uri.parse("content://" +  
    AUTHORITY + "/names");
```

# managedQuery

```
public final Cursor managedQuery(Uri uri,  
                                String[] projection,  
                                String selection,  
                                String[] selectionArgs,  
                                String sortOrder)
```

projection

Which columns to return

selection

SQL Where clause with "WHERE"

selectionArgs

Arguments for selection

sortOrder

SQL ORDER BY clause

# managedQuery

Cursors & Activity States

Managed Part

Cursors can be

Active

Deactivated

Consume fewer resources

deactivate(), requery()

Closed

ManagedQuery

Activity will manage changing cursor state

## Second Example

```
// An array specifying which columns to return.
string[] projection = new string[] {
    People._ID,
    People.NAME,
    People.NUMBER,
};

// Get the base URI for People table in Contacts content provider.
Uri mContacts = People.CONTENT_URI;

Cursor managedCursor = managedQuery( mContacts,
    projection, //Which columns to return.
    null,      // WHERE clause--we won't specify.
    People.NAME + " ASC"); // Order-by clause.
```

# Reading Columns

Cursor requires column indexes for access

We don't want to know the column index for each data element

ContentProvider should provide column names as static fields

People.NAME

```
public static final String NAME = "NAME";
```

Use `Cursor.getColumnIndex(String columnName)`



# Example

```
private void getColumnData(Cursor cur){
    if (cur.moveToFirst()) {

        String name;
        String phoneNumber;
        int nameColumn = cur.getColumnIndex(People.NAME);
        int phoneColumn = cur.getColumnIndex(People.NUMBER);

        do {
            // Get the field values
            name = cur.getString(nameColumn);
            phoneNumber = cur.getString(phoneColumn);

            // Do something with the values.
            ...
        } while (cur.moveToNext());
    }
}
```

# Adding a Record

```
ContentValues values = new ContentValues();

values.put(Contacts.People.NAME, "Roger");
values.put(Contacts.People.STARRED, 1);    // 1 = add to favorites

//Add Phone Numbers
Uri uri = getContentResolver().insert(Contacts.People.CONTENT_URI, values);
```

# **android.content.ContentProvider**

Abstract class

## Primary Abstract Methods

`query(Uri uri, String[] columns, String selection, String[] selectionArgs, String sortOrder)`  
returns data to the caller

`insert(Uri, ContentValues)`  
inserts new data into the content provider

`update(Uri uri, ContentValues values, String selection, String[] selectionArgs)`  
updates existing data in the content provider

`delete(Uri uri, String selection, String[] selectionArgs)`  
deletes data from the content provider

`getType(Uri)`  
returns the MIME type of data in the content provider

# Example

Show

Inserts

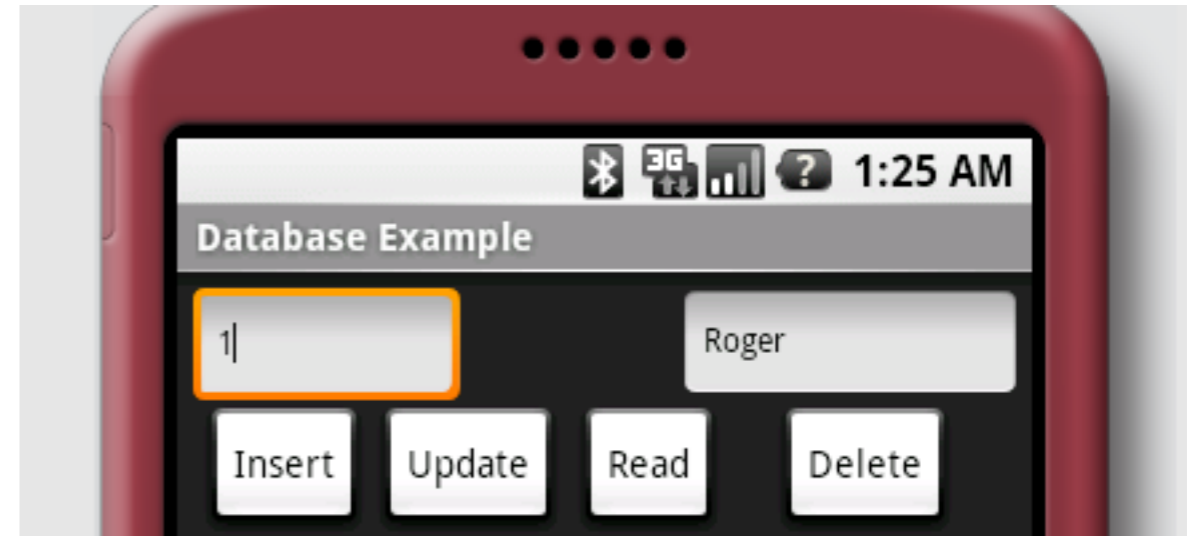
Update

Delete

Query

Same example as last lecture

Uses ContentProvider



# Classes

R.java

Constants, autogenerated

DatabaseHelper.java

Creates tables

Updating of schema

NameProvider.java

ContentProvider

Performs inserts, deletes, updates, queries of data

DatabaseExample.java

Uses NameProvider to access data

# URI

`content://edu.sdsu.cs696.NameProvider/names`

Used to access all name

`content://edu.sdsu.cs696.NameProvider/names/12`

Access name with ID = 12

`edu.sdsu.cs696.NameProvider`

Authority

Unique string to identify your data

Suggested to use full class name

# MIME Type

vnd.sdsu.cursor.dir/vnd.sdsu.name

MIME type for directory (list) of name

vnd.sdsu.cursor.item/vnd.sdsu.name

MIME type for single name

# AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.sdsu.cs696"
    android:versionCode="1"
    android:versionName="1.0.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <provider android:name="NameProvider"
            android:authorities="edu.sdsu.cs696.NameProvider" />
        <activity android:name=".DatabaseExample"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />
                <action android:name="android.intent.action.EDIT" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:mimeType="vnd.sdsu.cursor.item/vnd.sdsu.name" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```



# DatabaseHelper.java

```
public class DatabaseHelper extends SQLiteOpenHelper {
    public static final String DATABASE_NAME = "name5.db";
    private static final int DATABASE_VERSION = 1;

    // Column names
    public static final String ID = "_ID";
    public static final String NAME = "NAME";
    public static final String NAMES_TABLE = "NAMES";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
}
```

# DatabaseHelper.java

```
public void onCreate(SQLiteDatabase nameDb) {
    nameDb.execSQL("CREATE TABLE " + NAMES_TABLE + " ("
        + ID + " INTEGER PRIMARY KEY,"
        + NAME + " TEXT"
        + ");");
    nameDb.execSQL("INSERT INTO NAMES ( name) VALUES ('Roger' );");
}

public void onUpgrade(SQLiteDatabase arg0, int oldVersion, int newVersion) {
}
}
```

# android.content.UriMatcher

Helper class to match URI

`UriMatcher(int code)`

code = code to match for the root URI (???)

`addURI(String authority, String path, int code)`

Add a URI to match, and the code to return when this URI is matched.

`match(Uri uri)`

Try to match against the path in a url.

```
nameUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
```

```
nameUriMatcher.addURI("edu.sdsu.cs696.NameProvider", "names", 1);
```

```
nameUriMatcher.addURI("edu.sdsu.cs696.NameProvider", "names/#", 2);
```

```
int match = nameUriMatcher.match(uri);
```

# NameProvider.java

```
public class NameProvider extends ContentProvider {
    public static final String AUTHORITY = "edu.sdsu.cs696.NameProvider";
    public static final Uri CONTENT_URI = Uri.parse("content://"
        + AUTHORITY + "/names");

    // MIME types
    public static final String CONTENT_TYPE = "vnd.sdsu.cursor.dir/vnd.sdsu.name";
    public static final String CONTENT_ITEM_TYPE = "vnd.sdsu.cursor.item/vnd.sdsu.name";

    private DatabaseHelper nameHelper;
    private static final UriMatcher nameUriMatcher;
    private static final int NAMES = 1;
    private static final int NAMES_ID = 2;
    static {
        nameUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        nameUriMatcher.addURI(AUTHORITY, "names", NAMES);
        nameUriMatcher.addURI(AUTHORITY, "names/#", NAMES_ID);
    }
}
```

# NameProvider.java

```
public boolean onCreate() {  
    nameHelper = (new DatabaseHelper(getContext()));  
    return true;  
}
```

# NameProvider.java

```
public String getType(Uri uri) {  
    int match = nameUriMatcher.match(uri);  
    switch (match) {  
        case NAMES:  
            return CONTENT_TYPE;  
  
        case NAMES_ID:  
            return CONTENT_ITEM_TYPE;  
  
        default:  
            throw new IllegalArgumentException("Unknown URI " + uri);  
    }  
}
```

# NameProvider.java

```
public Uri insert(Uri uri, ContentValues values) {
    if (nameUriMatcher.match(uri) != NAMES) {
        throw new IllegalArgumentException("Unknown URI " + uri);
    }

    SQLiteDatabase db = nameHelper.getWritableDatabase();
    long rowId = db.insert(DatabaseHelper.NAMES_TABLE, DatabaseHelper.NAME,
        values);
    if (rowId > 0) {
        Uri noteUri = ContentUris.withAppendedId(CONTENT_URI, rowId);
        getContext().getContentResolver().notifyChange(noteUri, null);
        return noteUri;
    }

    throw new SQLException("Failed to insert row into " + uri);
}
```

# NameProvider.java

```
public Cursor query(Uri uri, String[] projection, String selection,
    String[] selectionArgs, String sortOrder) {

    String sqlSelection = null;
    switch (nameUriMatcher.match(uri)) {
    case NAMES:
        sqlSelection = selection;
        break;

    case NAMES_ID:
        sqlSelection = DatabaseHelper.NAMES_TABLE + "." + DatabaseHelper.ID
            + " = " + uri.getPathSegments().get(1);
        break;

    default:
        throw new IllegalArgumentException("Unknown URI " + uri);
    }
    SQLiteDatabase db = nameHelper.getWritableDatabase();
    Cursor result = db.query(DatabaseHelper.NAMES_TABLE, projection,
        sqlSelection, selectionArgs, null, null, sortOrder);
    result.setNotificationUri(getContext().getContentResolver(), uri);
    return result;
}
```



# NameProvider.java

```
public int delete(Uri uri, String where, String[] whereArgs) {
    SQLiteDatabase db = nameHelper.getWritableDatabase();
    int count;
    switch (nameUriMatcher.match(uri)) {
    case NAMES:
        count = db.delete(DatabaseHelper.NAMES_TABLE, where, whereArgs);
        break;

    case NAMES_ID:
        String nameId = uri.getPathSegments().get(1);
        count = db.delete(DatabaseHelper.NAMES_TABLE,
            DatabaseHelper.ID
                + " = "
                + nameId
                + (!TextUtils.isEmpty(where) ? " AND (" + where
                    + ')' : ""), whereArgs);

        break;

    default:
        throw new IllegalArgumentException("Unknown URI " + uri);
    }

    getContext().getContentResolver().notifyChange(uri, null);
    return count;
}
```

# NameProvider.java

```
public int update(Uri uri, ContentValues values, String where, String[] whereArgs) {
    SQLiteDatabase db = nameHelper.getWritableDatabase();
    int count;
    switch (nameUriMatcher.match(uri)) {
    case NAMES:
        count = db.update(DatabaseHelper.NAMES_TABLE, values, where, whereArgs);
        break;
    case NAMES_ID:
        String nameId = uri.getPathSegments().get(1);
        count = db.update(DatabaseHelper.NAMES_TABLE, values,
            DatabaseHelper.ID
                + " = "
                + nameId
                + (!TextUtils.isEmpty(where) ? " AND (" + where
                    + ')' : ""), whereArgs);

        break;
    default:
        throw new IllegalArgumentException("Unknown URI " + uri);
    }
    getContext().getContentResolver().notifyChange(uri, null);
    return count;
}
```

# NameProvider.java

```
public Uri insert(Uri uri, ContentValues values) {
    if (nameUriMatcher.match(uri) != NAMES) {
        throw new IllegalArgumentException("Unknown URI " + uri);
    }

    SQLiteDatabase db = nameHelper.getWritableDatabase();
    long rowId = db.insert(DatabaseHelper.NAMES_TABLE, DatabaseHelper.NAME,
        values);
    if (rowId > 0) {
        Uri noteUri = ContentUris.withAppendedId(CONTENT_URI, rowId);
        getContext().getContentResolver().notifyChange(noteUri, null);
        return noteUri;
    }

    throw new SQLException("Failed to insert row into " + uri);
}
```

# DatabaseExample.java

```
public class DatabaseExample extends Activity implements View.OnClickListener {
    private EditText databaseldText;
    private EditText nameText;
    private DatabaseHelper namesHelper;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        int[] buttonIds = { R.id.delete, R.id.read, R.id.insert, R.id.update };
        for (int id : buttonIds) {
            Button button = (Button) findViewById(id);
            button.setOnClickListener(this);
        }
        databaseldText = (EditText) this.findViewById(R.id.databaseld);
        nameText = (EditText) this.findViewById(R.id.name);
        namesHelper = (new DatabaseHelper(this));
        displayDatabaseRecord(1);
    }
}
```

# DatabaseExample.java

```
private String getName() {  
    return nameText.getText().toString();  
}
```

```
private String getId() {  
    return databaseldText.getText().toString();  
}
```

```
private int getIntId() {  
    return Integer.parseInt(getId());  
}
```

# DatabaseExample.java

```
private void displayDatabaseRecord(int id) {
    Uri firstName = ContentUris.withAppendedId(
        NameProvider.CONTENT_URI, id);

    Cursor result = managedQuery(firstName, null, "select", null, null);
    int rowCount = result.getCount();
    if (rowCount > 0) {
        result.moveToFirst();
        databaseIdText.setText(String.valueOf(result.getInt(0)));
        nameText.setText(result.getString(1));
    } else {
        nameText.setText("none");
    }
}
```

# DatabaseExample.java

```
public void onClick(View clicked) {
    switch (clicked.getId()) {
        case R.id.delete:
            Uri nameToDelete = ContentUris.withAppendedId( NameProvider.CONTENT_URI, getIntId());
            int rowsDeleted = getContentResolver().delete(nameToDelete, null, null);
            break;
        case R.id.insert:
            ContentValues newName = new ContentValues(1);
            newName.put(DatabaseHelper.NAME, getName());
            Uri recordUri = getContentResolver().insert(NameProvider.CONTENT_URI, newName);
            break;
        case R.id.read:
            displayDatabaseRecord(getIntId());
            break;
        case R.id.update:
            ContentValues updateName = new ContentValues(1);
            updateName.put(DatabaseHelper.NAME, getName());
            int rowsUpdated = getContentResolver().update(
                NameProvider.CONTENT_URI, updateName, DatabaseHelper.ID + " = ?",
                new String[] { getIntId() });
            break;
    }
}
```