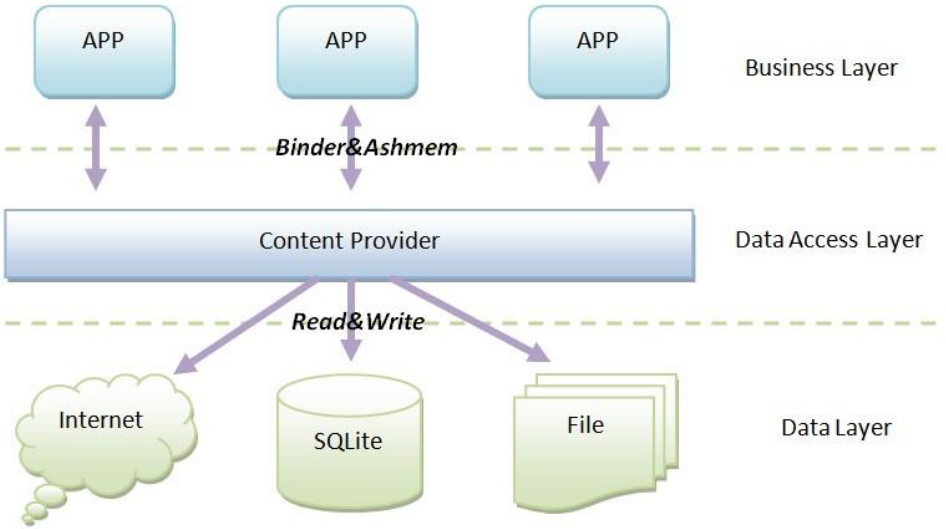
SSE3052: Embedded Systems Practice

Jinkyu jeong
jinkyu@skku.edu
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu

Content Provider

- Component supplies data from one application to others on request
- Such requests are handled by the methods of the ContentResolver class
- A content provider can use different ways to store its data and the data can be stored in a database, in files, or even over a network



Content Provider

- Behaves like a database
  - insert(): inserts new data into the content provider
  - update(): updates data
  - delete(): deletes data
  - query(): returns data to the caller
- In most cases this data is stored in an SQlite database
- Implemented as a subclass of ContentProvider class
  - Must implement a standard set of APIs that enable other applications to perform transaction

```
public class MyApplication extends ContentProvider {
        // Contents
}
```

Content URIs

- To query a content provider, you specify the query string in the form of a URI which has following format:
  - <prefix>://<authority>/<path>/<id>
  - <prefix>
    - Always set to "content"
  - <authority>
    - Specifies the name of the content provider
  - <path>
    - Indicates the type of data
  - <id>
    - Specifies the specific record requested
  - ex) content://contacts/people/5
    - Contact number 5 in the Contacts content provider

How to Create Content Provider?

- Create a Content Provider class that extends the ContentProvider base class
- Define your content provider URI address which will be used to access the content
- Create your own database to keep the content
  - Usually, Android uses SQLite database and framework needs to override onCreate() method which will use SQLite Open Helper method to create or open the provider's database
  - When your application is launched, the onCreate() handler of each of its Content Providers is called on the main application thread

How to Create Content Provider?

- Implement Content Provider queries to perform different database specific operations
- Register your Content Provider in your Manifest file using <provider> tag

Methods in Content Provider Class

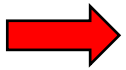
- onCreate() is called when the provider is started
- query() receives a request from a client
  - The result is returned as a Cursor object
- insert() inserts a new record into the content provider
- delete() deletes an existing record from the content provider
- update() updates an existing record from the content provider
- getType() returns the MIME type of the data at the given URI

Cursor

- The interface provides random read-write access to the result set returned by a database query

Example)

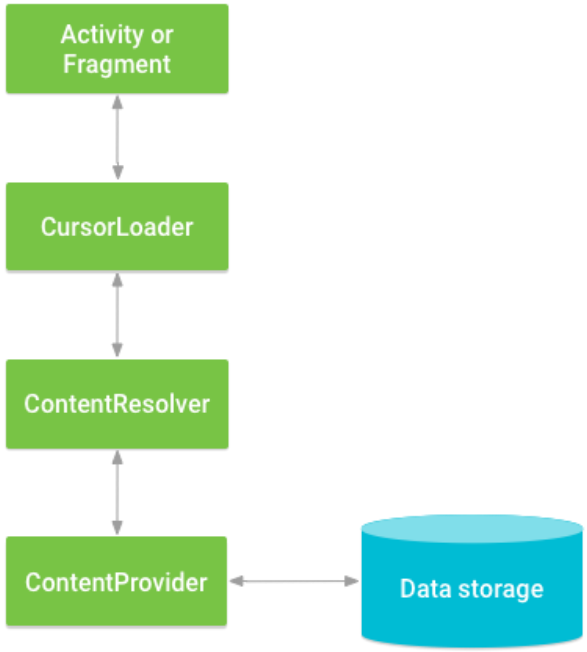
```
Cursor cursor = …;
cursor.moveToFirst();
cursor.getString(cursor.getColumnIndex("name"));
```



Seq.	Name	Number
1	Choo	17
2	Ryu	99
3	Oh	18
4	Choi	26

Accessing a Provider

- Use the ContentResolver object in your application's Context to communicate with the provider as a client
- The ContentResolver object communicates with the provider object, an instance of a class that implements ContentProvider
- The provider object receives data requests from clients, performs the requested action, and returns the results
- This object has methods that call identically-named methods in the provider object, an instance of one of the concrete subclasses of ContentProvider



ContentResolver

- This class provides applications access to the content model
- The ContentResolver methods provide the basic "CRUD" (create, retrieve, update, and delete) functions of persistent storage

Example)

```
Cursor cursor =
getContentResolver.query(Uri.parse("content://…"));
```

query()

- query(Uri uri, String[] projection, String selection,
  String[] selectionArgs, String sortOrder)

query() argument	SELECT keyword/parameter
Uri	FROM table_name
projection	col,col,col,...
selection	WHERE col = value
selectionArgs	(No exact equivalent. Selection arguments replace ? placeholders in the selection clause.)
sortOrder	ORDER BYcol,col,...

Inserting Data

```
// Defines a new Uri object that receives the result of the insertion
Uri newUri;
...

// Defines an object to contain the new values to insert
ContentValues newValues = new ContentValues();

/* Sets the values of each column and inserts the word. The arguments to the "put"
   method are "column name" and "value" */
newValues.put(UserDictionary.Words.APP_ID, "example.user");
newValues.put(UserDictionary.Words.LOCALE, "en_US");
newValues.put(UserDictionary.Words.WORD, "insert");
newValues.put(UserDictionary.Words.FREQUENCY, "100");

newUri = getContentResolver().insert(
    UserDictionary.Words.CONTENT_URI,   // the user dictionary content URI
    newValues                           // the values to insert
);
```

Updating Data

```
// Defines an object to contain the updated values
ContentValues updateValues = new ContentValues();
// Defines selection criteria for the rows you want to update
String selectionClause = UserDictionary.Words.LOCALE +  " LIKE ?";
String[] selectionArgs = {"en_%"};

// Defines a variable to contain the number of updated rows
int rowsUpdated = 0;
...
/* Sets the updated value and updates the selected words. */
updateValues.putNull(UserDictionary.Words.LOCALE);

rowsUpdated = getContentResolver().update(
    UserDictionary.Words.CONTENT_URI,   // the user dictionary content URI
    updateValues,                       // the columns to update
    selectionClause,                    // the column to select on
    selectionArgs                       // the value to compare to
);
```

Deleting Data

```
// Defines selection criteria for the rows you want to delete
String selectionClause = UserDictionary.Words.APP_ID + " LIKE ?";
String[] selectionArgs = {"user"};

// Defines a variable to contain the number of rows deleted
int rowsDeleted = 0;


...

// Deletes the words that match the selection criteria
rowsDeleted = getContentResolver().delete(
    UserDictionary.Words.CONTENT_URI,   // the user dictionary content URI
    selectionClause,                    // the column to select on
    selectionArgs                       // the value to compare to
);
```

Example (activity_main.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Name"
        android:layout_marginLeft="100dp"
        android:layout_marginTop="100dp"/>
    <EditText
        android:id="@+id/txtName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:ems="10"/>
```

```
    <Button
        android:id="@+id/btnAdd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onClickAddDetails"
        android:layout_marginLeft="100dp"
        android:text="Add User"/>

    <Button
        android:id="@+id/btnRetrieve"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onClickShowDetails"
        android:layout_marginLeft="100dp"
        android:text="Show Users"/>
    <TextView
        android:id="@+id/res"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:clickable="false"
        android:ems="10"/>
</LinearLayout>
```

MainActivity.java (1)

```
package com.example.MyApplication;

import …

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        InputMethodManager imm = (InputMethodManager)
                                 getSystemService(Context.INPUT_METHOD_SERVICE);
        imm.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(), 0);
        return true;
    }
    public void onClickAddDetails(View view) {
        ContentValues values = new ContentValues();
        values.put(StudentsProvider.name, ((EditText) findViewById(
                                                    R.id.txtName)).getText().toString());
        getContentResolver().insert(StudentsProvider.CONTENT_URI, values);
        Toast.makeText(getBaseContext(), "New Record Inserted", Toast.LENGTH_LONG).show();
    }
```

MainActivity.java (2)

```
    public void onClickShowDetails(View view) {
        // Retrieve employee records
        TextView resultView= (TextView) findViewById(R.id.res);
        Cursor cursor = getContentResolver().query(Uri.parse(
          "content://com.example.MyApplication.StudentsProvider/users"), null, null, null, null);
        if(cursor.moveToFirst()) {
            StringBuilder strBuild=new StringBuilder();
            while (!cursor.isAfterLast()) {
                strBuild.append("₩n"+cursor.getString(cursor.getColumnIndex("id"))+ "-"+
                                        cursor.getString(cursor.getColumnIndex("name")));
                cursor.moveToNext();
            }
            resultView.setText(strBuild);
        }
        else {
            resultView.setText("No Records Found");
        }
    }
}
```

StudentsProvider.java (1)

```
package com.example.MyApplication;
import …;
public class StudentsProvider extends ContentProvider {
    static final String PROVIDER_NAME = "com.example.MyApplication.StudentsProvider";
    static final String URL = "content://" + PROVIDER_NAME + "/users";
    static final Uri CONTENT_URI = Uri.parse(URL);
    static final String id = "id";
    static final String name = "name";
    static final int uriCode = 1;
    static final UriMatcher uriMatcher;
    private static HashMap<String, String> values;
    static {
        uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        uriMatcher.addURI(PROVIDER_NAME, "users", uriCode);
        uriMatcher.addURI(PROVIDER_NAME, "users/*", uriCode);
    }
    @Override
    public String getType(Uri uri) {
        switch (uriMatcher.match(uri)) {
            case uriCode:
                return "vnd.android.cursor.dir/users";
            default:
                throw new IllegalArgumentException("Unsupported URI: " + uri);
        }
    }
```

StudentsProvider.java (2)

```
@Override
public boolean onCreate() {
    Context context = getContext();
    DatabaseHelper dbHelper = new DatabaseHelper(context);
    db = dbHelper.getWritableDatabase();

    return db != null;
}
@Override
public Cursor query(Uri uri, String[] projection, String selection,
                    String[] selectionArgs, String sortOrder) {
    SQLiteQueryBuilder qb = new SQLiteQueryBuilder();
    qb.setTables(TABLE_NAME);
    switch (uriMatcher.match(uri)) {
        case uriCode:
            qb.setProjectionMap(values);
            break;
        default:
            throw new IllegalArgumentException("Unknown URI " + uri);
    }
    if (sortOrder == null || sortOrder == "") {
        sortOrder = id;
    }
    Cursor c = qb.query(db, projection, selection, selectionArgs, null, null, sortOrder);
    c.setNotificationUri(getContext().getContentResolver(), uri);
    return c;
}
```

StudentsProvider.java (3)

```
@Override
public Uri insert(Uri uri, ContentValues values) {
    long rowID = db.insert(TABLE_NAME, "", values);
    if (rowID > 0) {
        Uri _uri = ContentUris.withAppendedId(CONTENT_URI, rowID);
        getContext().getContentResolver().notifyChange(_uri, null);
        return _uri;
    }
    throw new SQLiteException("Failed to add a record into " + uri);
}
@Override
public int update(Uri uri, ContentValues values, String selection,
                  String[] selectionArgs) {
    int count = 0;
    switch (uriMatcher.match(uri)) {
        case uriCode:
            count = db.update(TABLE_NAME, values, selection, selectionArgs);
            break;
        default:
            throw new IllegalArgumentException("Unknown URI " + uri);
    }
    getContext().getContentResolver().notifyChange(uri, null);
    return count;
}
```

StudentsProvider.java (4)

```
@Override
public int delete(Uri uri, String selection, String[] selectionArgs) {
    int count = 0;
    switch (uriMatcher.match(uri)) {
        case uriCode:
            count = db.delete(TABLE_NAME, selection, selectionArgs);
            break;
        default:
            throw new IllegalArgumentException("Unknown URI " + uri);
    }
    getContext().getContentResolver().notifyChange(uri, null);
    return count;
}
private SQLiteDatabase db;
static final String DATABASE_NAME = "EmpDB";
static final String TABLE_NAME = "Employees";
static final int DATABASE_VERSION = 1;
static final String CREATE_DB_TABLE = " CREATE TABLE " + TABLE_NAME
        + " (id INTEGER PRIMARY KEY AUTOINCREMENT, "
        + " name TEXT NOT NULL);";
```

StudentsProvider.java (5)

```
    private static class DatabaseHelper extends SQLiteOpenHelper {
        DatabaseHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }

        @Override
        public void onCreate(SQLiteDatabase db) {
            db.execSQL(CREATE_DB_TABLE);
        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
            db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
            onCreate(db);
        }
    }
}
```

Exercise

- Make "Delete User" button in example code
  - When the button clicked, the user which has same name with EditText should be deleted
  - Please check if the user is existed or not before invoking delete method
  - Assume that there is no duplicated user
    - Every user have a unique name

Appendix A - SQLite

- SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine
- https://www.tutorialspoint.com/sqlite/index.htm

Appendix B - ContentResolver

- https://developer.android.com/reference/android/content/ContentResolver.html#