Experiment 07: Write a program to implement content provider.

Aim: Write a program to implement content provider

Tools: Android Studio

Theory:

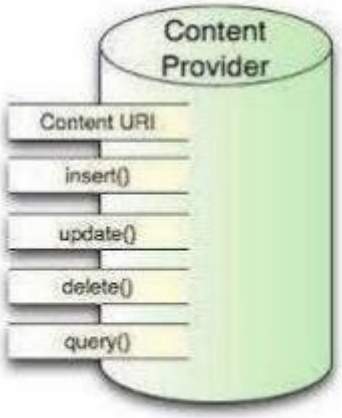
A Content provider is used to share and access data from a central repository. Usually, android applications keep data hidden from the other applications but sometimes, it is useful to share data across them. So, content provider is a suitable reason to share data with other applications, based on a standard interface. There are many ways to store data in a content provider but in most cases SQL Database is used.

Creating a Content Provider

To create a content provider in android applications we should follow below steps:

- We need to create a content provider class that extends the ContentProvider base class.
- We need to define our content provider URI to access the content.
- The ContentProvider class defines a six abstract methods (insert(), update(), delete(), query(), getType()) which we need to implement all these methods as a part of our subclass.
- We need to register our content provider in AndroidManifest.xml using <provider> tag.

Following are the list of methods which need to implement as a part of ContentProvider class.



ContentProvider

Write a program to implement content provider.
Code:

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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:layout_width="match_parent"
android:layout_height="match_parent"
android:background="@color/black"
tools:context=".MainActivity">
<LinearLayout
android:id="@+id/linearLayout"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_centerVertical="true"
android:orientation="vertical"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="0.13"
tools:ignore="MissingConstraints">
<TextView
android:id="@+id/textView1"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="40dp"
android:layout_marginBottom="70dp"
android:text="@string/heading"
android:textAlignment="center"
android:textAppearance="@style/TextAppearance.AppCompat.Large"
android:textColor="@color/teal_200"
android:textSize="36sp"
android:textStyle="bold" />
<EditText
android:id="@+id/textName"
android:layout_width="370dp"
android:layout_height="69dp"
android:layout_marginStart="20dp"
android:layout_marginEnd="20dp"
android:layout_marginBottom="40dp"
android:hint="@string/hintText"
android:textColor="#FFFFFF"
android:textSize="22sp" />
<Button
android:id="@+id/insertButton"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_marginStart="20dp"
android:layout_marginTop="10dp"
android:layout_marginEnd="20dp"
android:layout_marginBottom="20dp"
android:backgroundTint="#F44336"
android:onClick="onClickAddDetails"
android:text="@string/insertButtontext"
android:textAlignment="center"
android:textAppearance="@style/TextAppearance.AppCompat.Display1"
android:textColor="#FFFFFF"
android:textSize="30sp"
android:textStyle="bold"
app:rippleColor="#4CAF50" />
<Button
android:id="@+id/loadButton"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_marginStart="20dp"
android:layout_marginTop="10dp"
android:layout_marginEnd="20dp"
android:layout_marginBottom="20dp"
android:backgroundTint="#F44336"
android:onClick="onClickShowDetails"
android:text="@string/loadButtonText"
android:textAlignment="center"
android:textAppearance="@style/TextAppearance.AppCompat.Display1"
android:textColor="#FFFFFF"
android:textSize="28sp"
android:textStyle="bold" />
```

```
<TextView
android:id="@+id/res"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginStart="20dp"
android:layout_marginEnd="20dp"
android:clickable="false"
android:ems="10"
android:textColor="@android:color/holo
_green_dark"
android:textSize="18sp"
android:textStyle="bold" />
</LinearLayout>
</androidx.constraintlayout.widget.Cons
traintLayout>
```

MainActivity.java

```
package com.example.exp7;
import
androidx.appcompat.app.AppCompatAct
ivity;
import android.annotation.SuppressLint;
import android.os.Bundle;
import
androidx.appcompat.app.AppCompatAct
ivity;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import
androidx.appcompat.app.AppCompatAct
ivity;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import
android.view.inputmethod.InputMethod
Manager;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
public class MainActivity extends
AppCompatActivity {
Uri CONTENT_URI =
Uri.parse("content://com.demo.user.prov
ider/users");
@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(getCurr
entFocus().getWindowToken(), 0);
return true;}
public void onClickAddDetails(View
view) {
// class to add values in the database
ContentValues values = new
ContentValues();
// fetching text from user
values.put(MyContentProvider.name,
((EditText)
findViewById(R.id.textName)).getText()
.toString());
// inserting into database through content
URI
getContentResolver().insert(MyContentP
rovider.CONTENT_URI, values);
// displaying a toast message
Toast.makeText(getBaseContext(), "New
Record Inserted",
Toast.LENGTH_LONG).show();
}@SuppressLint("Range")
public void onClickShowDetails(View
view) {// inserting complete table details
in this text field
TextView resultView= (TextView)
findViewById(R.id.res);
// creating a cursor object of the
// content URI
Cursor cursor =
getContentResolver().query(Uri.parse("c
```

```
ontent://com.demo.user.provider/user
s"), null, null, null, null);
// iteration of the cursor
// to print whole table
if(cursor.moveToFirst()) {
StringBuilder strBuild=new
StringBuilder();
while (!cursor.isAfterLast()) {
strBuild.append("\n"+cursor.getString(cu
rsor.getColumnIndex("id"))+ "-"+
cursor.getString(cursor.getColumnIndex(
"name")));
cursor.moveToNext(); }
resultView.setText(strBuild); }
else {
resultView.setText("No Records
Found");
} } }
```

mycontentprovider.java

```
package com.example.exp7;
import android.content.ContentProvider;
import android.content.ContentValues;
import android.database.Cursor;
import android.net.Uri;
import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
import android.content.UriMatcher;
import android.database.Cursor;
import
android.database.sqlite.SQLiteDatabase;
import
android.database.sqlite.SQLiteException
;
import
android.database.sqlite.SQLiteOpenHelp
er;
import
android.database.sqlite.SQLiteQueryBuil
der;
import android.net.Uri;
import java.util.HashMap;
public class MyContentProvider extends
ContentProvider {
public MyContentProvider() {}
static final String PROVIDER_NAME =
"com.demo.user.provider";
// defining content URI
static final String URL = "content://" +
PROVIDER_NAME + "/users";
// parsing the content URI
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 {
// to match the content URI
// every time user access table under
content provider
uriMatcher = new
UriMatcher(UriMatcher.NO_MATCH);
// to access whole table
uriMatcher.addURI(PROVIDER_NAM
E, "users", uriCode);
// to access a particular row
// of the table
uriMatcher.addURI(PROVIDER_NAM
E, "users/*", uriCode);
}@Override
public int delete(Uri uri, String selection,
String[] selectionArgs) {
// Implement this to handle requests to
delete one or more rows.
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().notify
Change(uri, null);
return count;
}@Override
public String getType(Uri uri) {
// TODO: Implement this to handle
```

```
requests for the MIME type of the
data // at the given URI.
switch (uriMatcher.match(uri)) {
case uriCode:
return "vnd.android.cursor.dir/users";
default:
throw new
IllegalArgumentException("Unsupported
URI: " + uri); }}
@Override
public Uri insert(Uri uri, ContentValues
values) {
// TODO: Implement this to handle
requests to insert a new row.
long rowID = db.insert(TABLE_NAME,
"", values);
if (rowID > 0) {
Uri _uri =
ContentUris.withAppendedId(CONTEN
T_URI, rowID);
getContext().getContentResolver().notify
Change(_uri, null);
return _uri;}
throw new SQLiteException("Failed to
add a record into " + uri);
}@Override
public boolean onCreate() {
// TODO: Implement this to initialize
your content provider on
startup.
Context context = getContext();
DatabaseHelper dbHelper = new
DatabaseHelper(context);
db = dbHelper.getWritableDatabase();
if (db != null) {
return true;}
return false;}
@Override
public Cursor query(Uri uri, String[]
projection, String selection,
String[] selectionArgs, String sortOrder)
{
// TODO: Implement this to handle query
requests from clients.
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().getCon
tentResolver(), uri);
return c;
}@Override
public int update(Uri uri, ContentValues
values, String selection,
String[] selectionArgs) {
// TODO: Implement this to handle
requests to update one or more
rows.
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().notify
Change(uri, null);
return count;}
private SQLiteDatabase db;
// declaring name of the database
static final String DATABASE_NAME
= "UserDB";
// declaring table name of the database
static final String TABLE_NAME =
"Users";
// declaring version of the database
static final int DATABASE_VERSION
= 1;
```

```
// sql query to create the table
static final String
CREATE_DB_TABLE = " CREATE
TABLE " + TABLE_NAME
+ " (id INTEGER PRIMARY KEY
AUTOINCREMENT, "
+ " name TEXT NOT NULL);";
// creating a database
private static class DatabaseHelper
extends SQLiteOpenHelper {
// defining a constructor
DatabaseHelper(Context context) {
super(context, DATABASE_NAME,
null, DATABASE_VERSION);}
// creating a table in the database
@Override
public void onCreate(SQLiteDatabase
db) {
db.execSQL(CREATE_DB_TABLE);
}@Override
public void onUpgrade(SQLiteDatabase
db, int oldVersion, int
newVersion) {
// sql query to drop a table
// having similar name
db.execSQL("DROP TABLE IF
EXISTS " + TABLE_NAME);
onCreate(db);
}}}
```

Implementation:

Result and Discussion: We successfully implemented a to implement content provider in Android Studio.

Learning Outcomes: The student should have the ability to execute a simple program to implement content provider in Android Studio

Course Outcomes: Upon completion of the course students will be able to implement content provider in Android Studio

Conclusion: This experiment aimed to implement a content provider in an Android application. A content provider allows sharing and accessing data from a central repository, making it a suitable way to share data across applications. The experiment outlined the steps involved in creating a content provider, including creating a class, defining a URI, implementing methods, and registering the content provider. Overall, this experiment demonstrated the process of implementing a content provider, which can be a useful way to share data across applications in Android.

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				