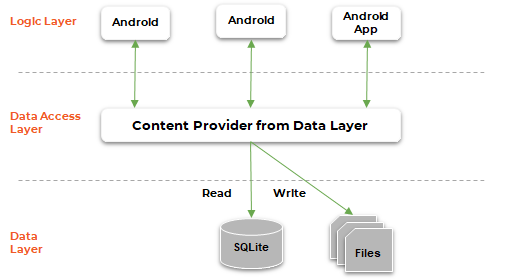
**Lab 22: Android Content Provider using Kotlin**

# **Introduction**

A 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.



**Let’s get Started:**

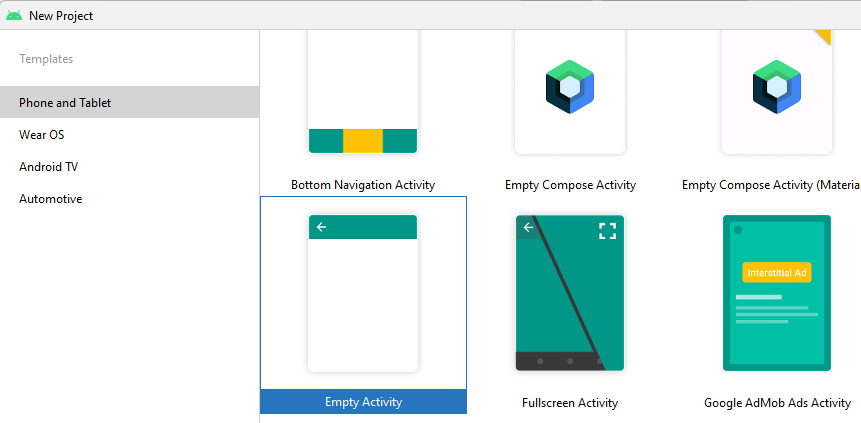
In this experiment we will develop an Android App to demonstrate the use of Android Content Provider.

**Step 1: Create a New Project in Android Studio as shown below**

Graphical user interface, text, application

Description automatically generated

**Step 2: Select Empty Activity as shown below**



**Step 3: Provide a Project Name as shown below**

**Graphical user interface, text, application, email

Description automatically generated**

**Step 4: Update MainActivity.kt as per the code given below**

**package** com.example.democontentproviderkotlin  
**import** android.content.ContentValues  
**import** android.content.Context  
**import** android.net.Uri  
**import** android.os.Bundle  
**import** android.view.MotionEvent  
**import** android.view.View  
**import** android.view.inputmethod.InputMethodManager  
**import** android.widget.EditText  
**import** android.widget.TextView  
**import** android.widget.Toast  
**import** androidx.appcompat.app.AppCompatActivity  
**import** com.example.democontentproviderkotlin.MyContentProvider  
  
  
**class** MainActivity : AppCompatActivity() {  
 **override fun** onCreate(savedInstanceState: Bundle?) {  
 **super**.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
 }  
  
 **override fun** onTouchEvent(event: MotionEvent?): Boolean {  
 **val** imm = getSystemService(Context.*INPUT\_METHOD\_SERVICE*) **as** InputMethodManager  
 imm.hideSoftInputFromWindow(*currentFocus*!!.*windowToken*, 0)  
 **return true** }  
  
 **fun** onClickAddDetails(view: View?) {  
  
 *// class to add values in the database* **val** values = ContentValues()  
  
 *// fetching text from user* values.put(MyContentProvider.**name**, (findViewById<View>(R.id.*textName*) **as** EditText).*text*.toString())  
  
 *// inserting into database through content URI  
 contentResolver*.insert(MyContentProvider.**CONTENT\_URI**, values)  
  
 *// displaying a toast message* Toast.makeText(*baseContext*, **"New Record Inserted"**, Toast.*LENGTH\_LONG*).show()  
 }  
  
 **fun** onClickShowDetails(view: View?) {  
 *// inserting complete table details in this text field* **val** resultView = findViewById<View>(R.id.*res*) **as** TextView  
  
 *// creating a cursor object of the  
 // content URI* **val** cursor = *contentResolver*.query(Uri.parse(**"content://com.demo.user.provider/users"**), **null**, **null**, **null**, **null**)  
  
 *// iteration of the cursor  
 // to print whole table* **if** (cursor!!.moveToFirst()) {  
 **val** strBuild = StringBuilder()  
 **while** (!cursor.*isAfterLast*) {  
 strBuild.append(**"""  
   
 ${**cursor.getString(cursor.getColumnIndex(**"id"**))**}-${**cursor.getString(cursor.getColumnIndex(**"name"**))**}  
 """**.*trimIndent*())  
 cursor.moveToNext()  
 }  
 resultView.*text* = strBuild  
 } **else** {  
 resultView.*text* = **"No Records Found"** }  
 }  
}

**Step 5: Update activity\_main.xml for Vertical Orientation as per the code given below**

*<?***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="#168BC34A"  
 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\_constraintBottom\_toTopOf="@+id/imageView"  
 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="@android:color/holo\_green\_dark"  
 android:textSize="36sp"  
 android:textStyle="bold"** />  
  
 <**EditText  
 android:id="@+id/textName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="20dp"  
 android:layout\_marginEnd="20dp"  
 android:layout\_marginBottom="40dp"  
 android:hint="@string/hintText"** />  
  
 <**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:background="#4CAF50"  
 android:onClick="onClickAddDetails"  
 android:text="@string/insertButtontext"  
 android:textAlignment="center"  
 android:textAppearance="@style/TextAppearance.AppCompat.Display1"  
 android:textColor="#FFFFFF"  
 android:textStyle="bold"** />  
  
 <**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:background="#4CAF50"  
 android:onClick="onClickShowDetails"  
 android:text="@string/loadButtonText"  
 android:textAlignment="center"  
 android:textAppearance="@style/TextAppearance.AppCompat.Display1"  
 android:textColor="#FFFFFF"  
 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**>  
  
 <**ImageView  
 android:id="@+id/imageView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"** />  
  
</**androidx.constraintlayout.widget.ConstraintLayout**>

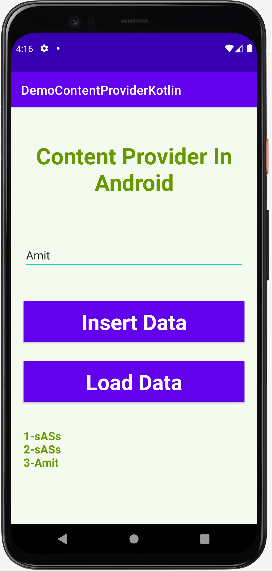
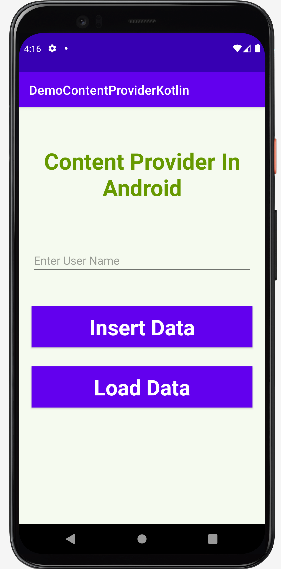
**Step 6: Add code in String.xml**

<**resources**>  
 <**string name="app\_name"**>DemoContentProviderKotlin</**string**>  
 <**string name="hintText"**>Enter User Name</**string**>  
 <**string name="heading"**>Content Provider In Android</**string**>  
 <**string name="insertButtontext"**>Insert Data</**string**>  
 <**string name="loadButtonText"**>Load Data</**string**>  
</**resources**>

**Step 7: Create MyContentProvider.kt and Define authority (it can be anything for example “com.demo.user.provider”)**

**package** com.example.democontentproviderkotlin  
**import** android.content.\*  
**import** android.database.Cursor  
**import** android.database.sqlite.SQLiteDatabase  
**import** android.database.sqlite.SQLiteException  
**import** android.database.sqlite.SQLiteOpenHelper  
**import** android.database.sqlite.SQLiteQueryBuilder  
**import** android.net.Uri  
  
  
**class** MyContentProvider : ContentProvider() {  
 **companion object** {  
 *// defining authority so that other application can access it* **const val PROVIDER\_NAME** = **"com.demo.user.provider"** *// defining content URI* **const val URL** = **"content://$PROVIDER\_NAME/users"** *// parsing the content URI* **val CONTENT\_URI** = Uri.parse(**URL**)  
 **const val id** = **"id"  
 const val name** = **"name"  
 const val uriCode** = 1  
 **var uriMatcher**: UriMatcher? = **null  
 private val values**: HashMap<String, String>? = **null** *// declaring name of the database* **const val DATABASE\_NAME** = **"UserDB"** *// declaring table name of the database* **const val TABLE\_NAME** = **"Users"** *// declaring version of the database* **const val DATABASE\_VERSION** = 1  
  
 *// sql query to create the table* **const val CREATE\_DB\_TABLE** =  
 (**" CREATE TABLE "** + **TABLE\_NAME** + **" (id INTEGER PRIMARY KEY AUTOINCREMENT, "** + **" name TEXT NOT NULL);"**)  
  
 **init** {  
  
 *// to match the content URI  
 // every time user access table under content provider* **uriMatcher** = UriMatcher(UriMatcher.*NO\_MATCH*)  
  
 *// to access whole table* **uriMatcher**!!.addURI(  
 **PROVIDER\_NAME**,  
 **"users"**,  
 **uriCode** )  
  
 *// to access a particular row  
 // of the table* **uriMatcher**!!.addURI(  
 **PROVIDER\_NAME**,  
 **"users/\*"**,  
 **uriCode** )  
 }  
 }  
  
 **override fun** getType(uri: Uri): String? {  
 **return when** (**uriMatcher**!!.match(uri)) {  
 **uriCode** -> **"vnd.android.cursor.dir/users"  
 else** -> **throw** IllegalArgumentException(**"Unsupported URI: $**uri**"**)  
 }  
 }  
  
 *// creating the database* **override fun** onCreate(): Boolean {  
 **val** context = *context* **val** dbHelper =  
 DatabaseHelper(context)  
 **db** = dbHelper.*writableDatabase* **return if** (**db** != **null**) {  
 **true** } **else false** }  
  
 **override fun** query(  
 uri: Uri, projection: Array<String>?, selection: String?,  
 selectionArgs: Array<String>?, sortOrder: String?  
 ): Cursor? {  
 **var** sortOrder = sortOrder  
 **val** qb = SQLiteQueryBuilder()  
 qb.*tables* = **TABLE\_NAME  
 when** (**uriMatcher**!!.match(uri)) {  
 **uriCode** -> qb.*projectionMap* = **values  
 else** -> **throw** IllegalArgumentException(**"Unknown URI $**uri**"**)  
 }  
 **if** (sortOrder == **null** || sortOrder === **""**) {  
 sortOrder = **id** }  
 **val** c = qb.query(  
 **db**, projection, selection, selectionArgs, **null**,  
 **null**, sortOrder  
 )  
 c.setNotificationUri(*context*!!.*contentResolver*, uri)  
 **return** c  
 }  
  
 *// adding data to the database* **override fun** insert(uri: Uri, values: ContentValues?): Uri? {  
 **val** rowID = **db**!!.insert(**TABLE\_NAME**, **""**, values)  
 **if** (rowID > 0) {  
 **val** \_uri =  
 ContentUris.withAppendedId(**CONTENT\_URI**, rowID)  
 *context*!!.*contentResolver*.notifyChange(\_uri, **null**)  
 **return** \_uri  
 }  
 **throw** SQLiteException(**"Failed to add a record into $**uri**"**)  
 }  
  
 **override fun** update(  
 uri: Uri, values: ContentValues?, selection: String?,  
 selectionArgs: Array<String>?  
 ): Int {  
 **var** count = 0  
 count = **when** (**uriMatcher**!!.match(uri)) {  
 **uriCode** -> **db**!!.update(**TABLE\_NAME**, values, selection, selectionArgs)  
 **else** -> **throw** IllegalArgumentException(**"Unknown URI $**uri**"**)  
 }  
 *context*!!.*contentResolver*.notifyChange(uri, **null**)  
 **return** count  
 }  
  
 **override fun** delete(  
 uri: Uri,  
 selection: String?,  
 selectionArgs: Array<String>?  
 ): Int {  
 **var** count = 0  
 count = **when** (**uriMatcher**!!.match(uri)) {  
 **uriCode** -> **db**!!.delete(**TABLE\_NAME**, selection, selectionArgs)  
 **else** -> **throw** IllegalArgumentException(**"Unknown URI $**uri**"**)  
 }  
 *context*!!.*contentResolver*.notifyChange(uri, **null**)  
 **return** count  
 }  
  
 *// creating object of database  
 // to perform query* **private var db**: SQLiteDatabase? = **null** *// creating a database* **private class** DatabaseHelper *// defining a constructor* **internal constructor**(context: Context?) : SQLiteOpenHelper(  
 context,  
 **DATABASE\_NAME**,  
 **null**,  
 **DATABASE\_VERSION** ) {  
 *// creating a table in the database* **override fun** onCreate(db: SQLiteDatabase) {  
 db.execSQL(**CREATE\_DB\_TABLE**)  
 }  
  
 **override fun** onUpgrade(  
 db: SQLiteDatabase,  
 oldVersion: Int,  
 newVersion: Int  
 ) {  
  
 *// sql query to drop a table  
 // having similar name* db.execSQL(**"DROP TABLE IF EXISTS $TABLE\_NAME"**)  
 onCreate(db)  
 }  
 }  
}

**Step 6: Check Output on Android Emulator and it should look like as given below**

****

**Voila!!** We have successfully completed this lab.