School of Mathematical & Computer Sciences

Department of Computer Science

Course F20MC: Mobile Communication and Programming

"Coursework 2: Android Content Provider"

Beni Iyaka - H00181266
Bi34@hw.ac.uk

Professor: Talal Shaikh

December 3, 2015
Dubai

Table of Contents

Introduction

Content providers are one of the primary building blocks of Android applications, providing **content** to applications. They encapsulate data and provide it to applications through the single ContentResolver interface. (Johnson 2012)

Android devices come in all kinds of sizes, with all sorts of features, and at all sorts of prices. Each version of Android is named after a dessert. With Android, you're in control of your mobile experience. (Android)

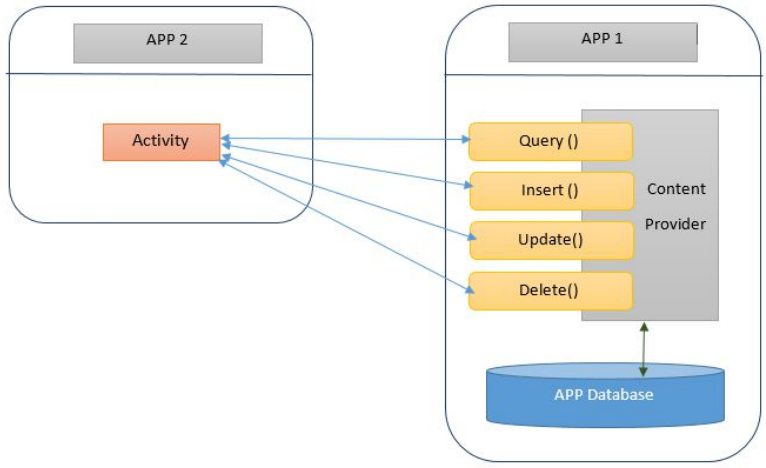


Figure 1 (Android)

The product I have created is a content provider application that stores countries details; and these details can also be accessed by other applications on the android phone that are linked to it.

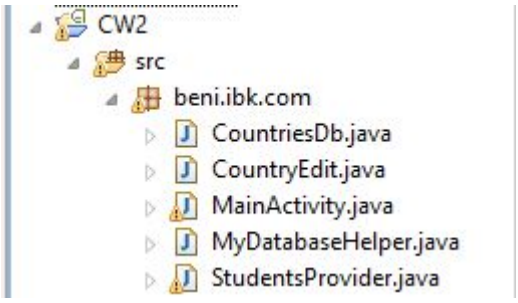
The country details being:

- Continent
- Country code
- Capital city
- Population density
- Surface area
- Official language
- Number of provinces.

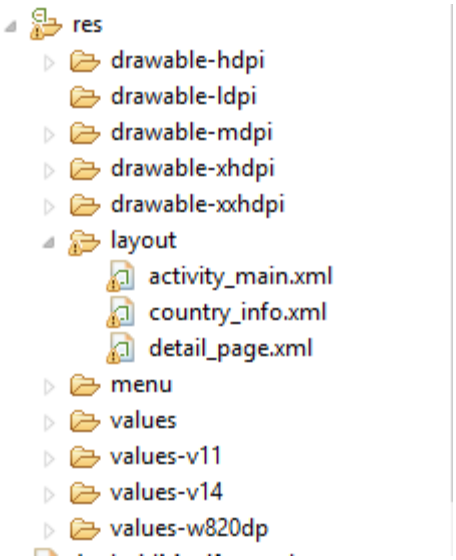
Summary of technology used:

The technology used in terms of programming the application was the following:

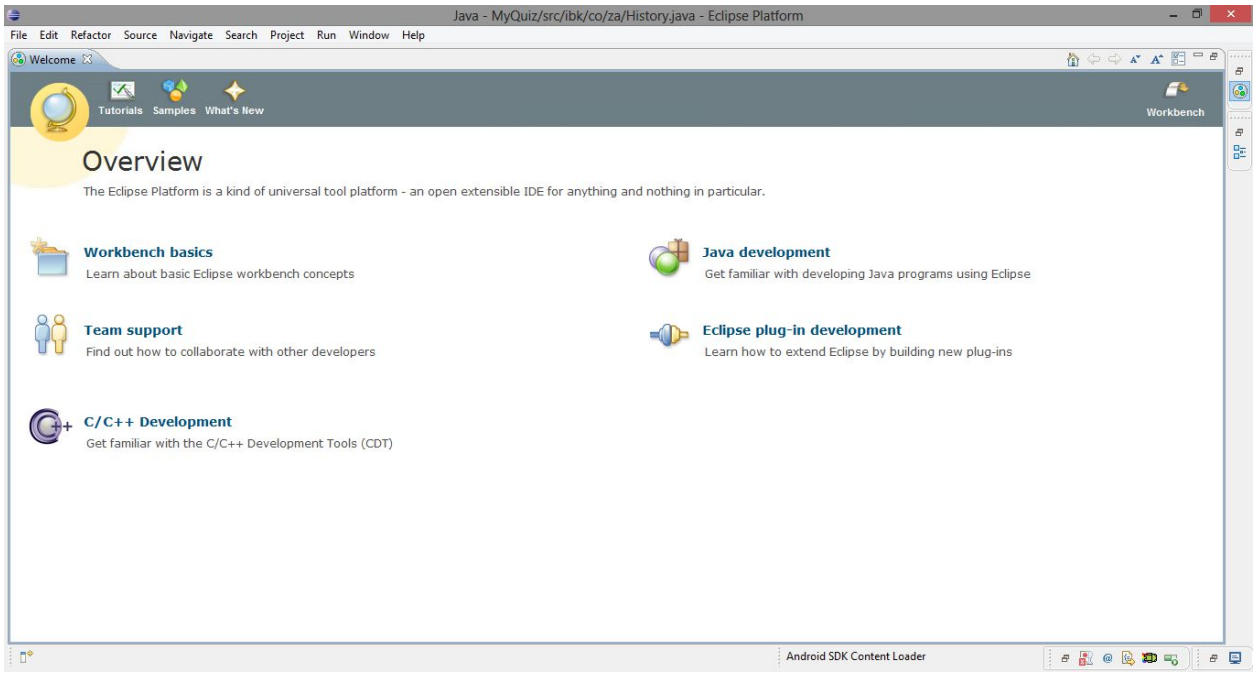
- Java: this was used to create class and some programming style.



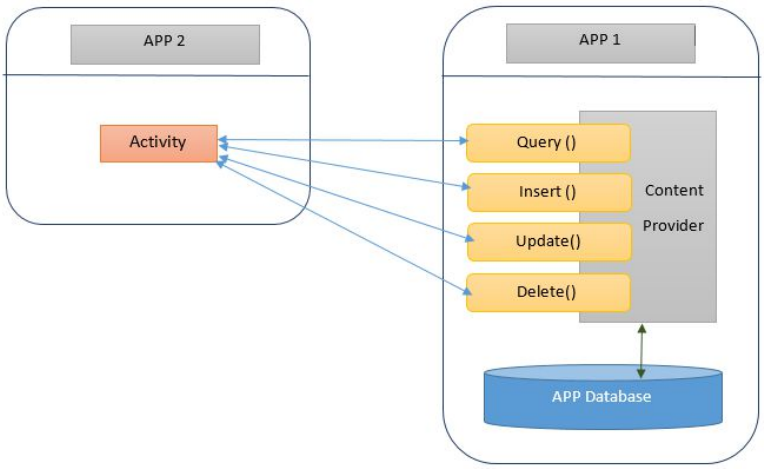
- XML: this was used to set up the style of which I want my application to have.



- Eclipse: this is the programming application I used to program the content provider application.



Application diagram



GUI screenshots

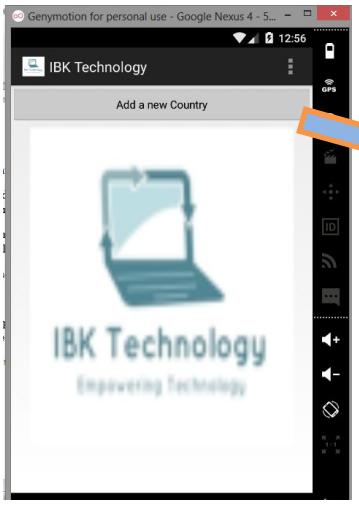
This is the first form of the application, it permits you either add a new country or edit a country that is already in the database.

To add a new country, the user needs to click on “Add new Country and they will be a second form where they can input different details of the country they want to add.

After filling in details, the user then clicks on saves and the country will be saved and displayed in the first form.

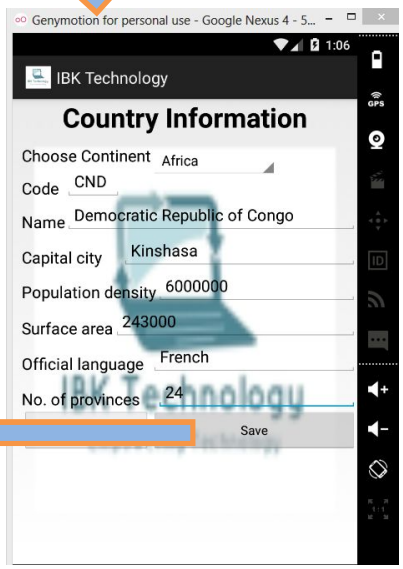
Practically:

1.

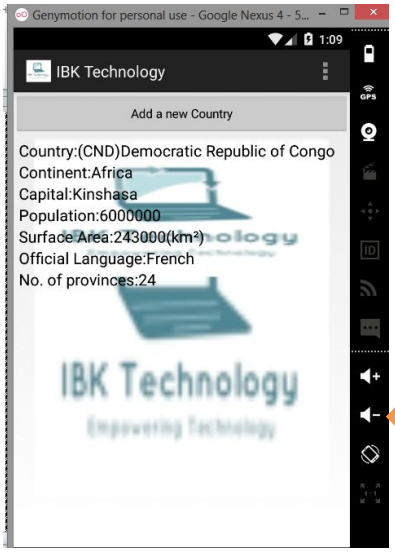


2.

3.



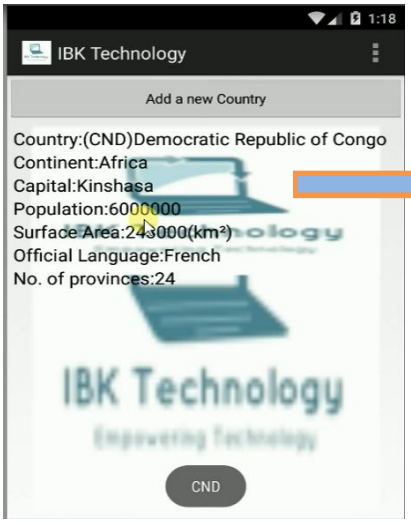
4.



To update a certaing country, the user select the intended country and the second form will appear with details allowing the user to pick which field they want to update.

Practically:

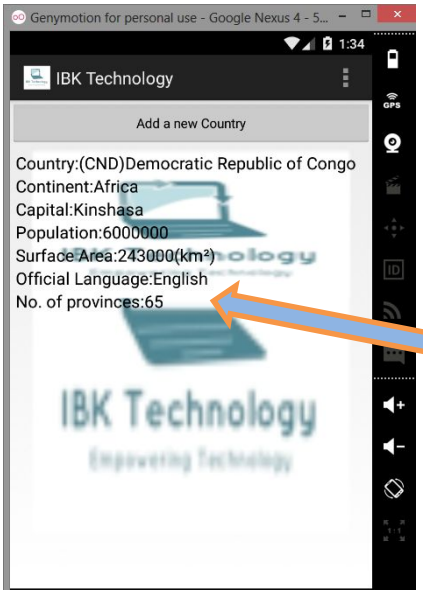
1.



2.

3.

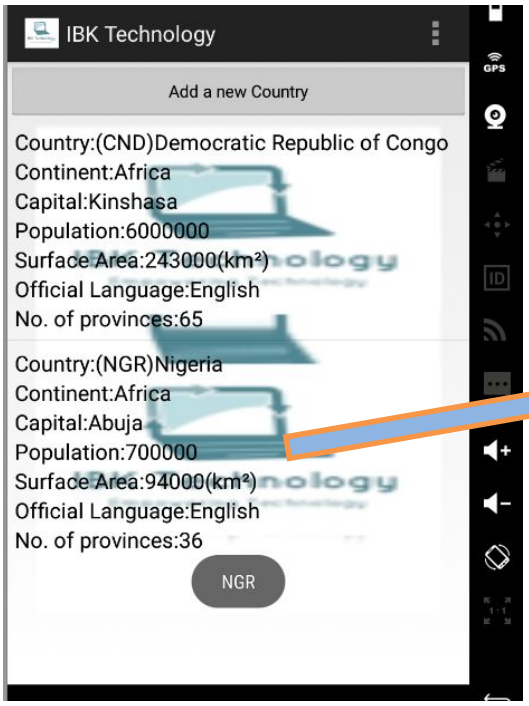
4.



To delete a certaing country, the user select the intended country and the second form will with the delete button enabled and after selecting the delete button, the selected country will be deleted from the database.

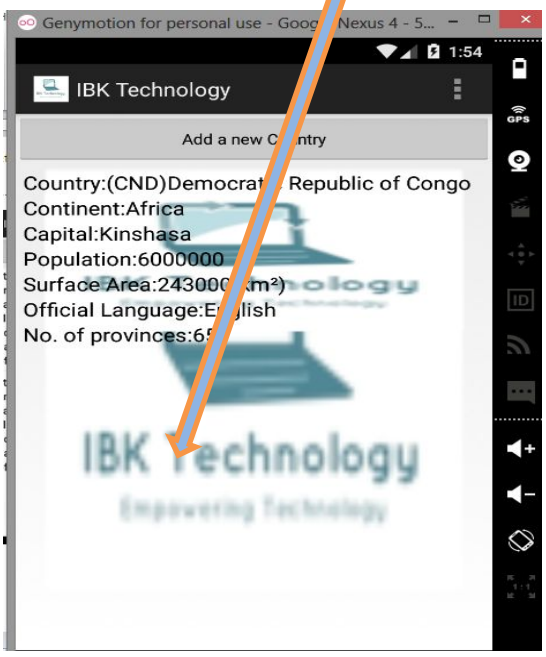
Practically:

1.

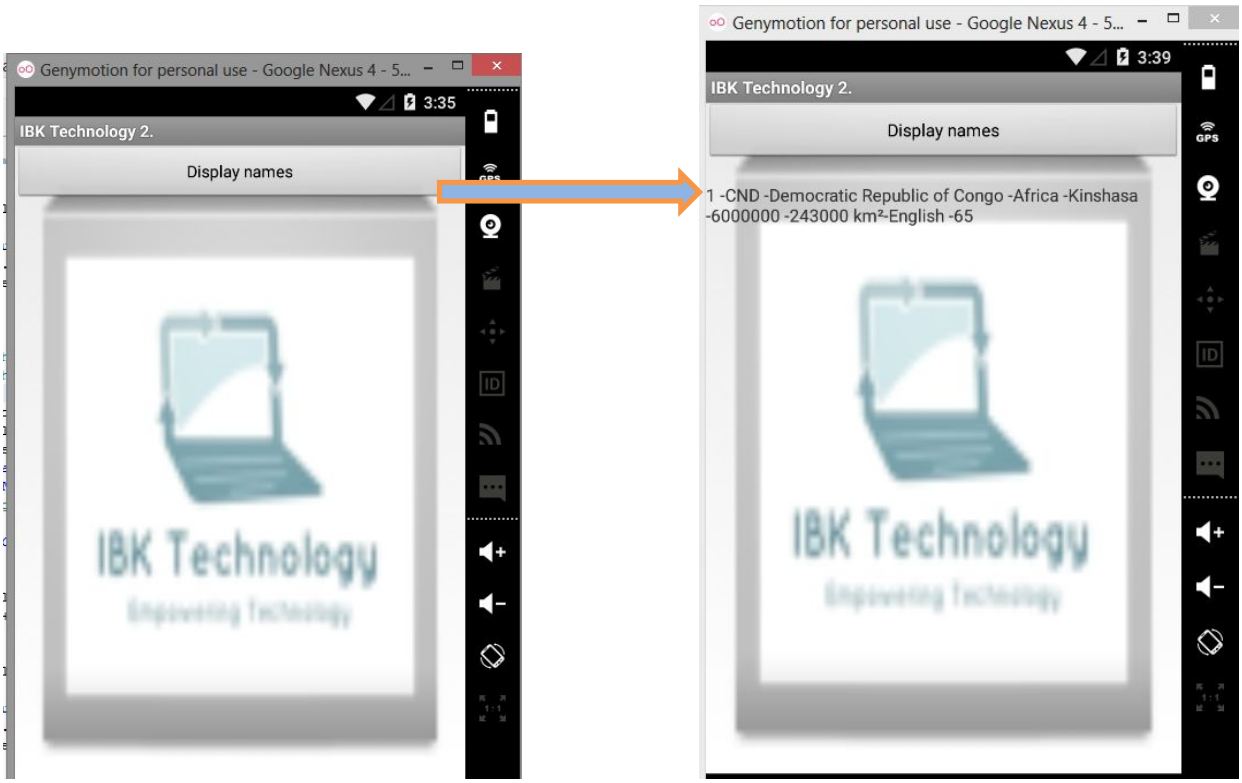


2.

4..



For this content provider, I made a second application that will read the details that was input in the main application.



Content Provider

Specifying the content provider

```
    private MyDatabaseHelper dbHelper;

    private static final int ALL_COUNTRIES = 1;
    private static final int SINGLE_COUNTRY = 2;

    // authority is the symbolic name of my provider
    // To avoid conflicts with other providers, I used
    // Internet domain ownership (in reverse) as the basis of my provider authority.
    private static final String AUTHORITY = "beni.ibk.com.StudentsProvider";

    // create content URIs from the authority by appending path to database table
    public static final Uri CONTENT_URI =
     Uri.parse("content://" + AUTHORITY + "/countries");

    // a content URI pattern matches content URIs using wildcard characters:
    // *: Matches a string of any valid characters of any length.
       // #: Matches a string of numeric characters of any length.
    private static final UriMatcher uriMatcher;
    static {
     uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
     uriMatcher.addURI(AUTHORITY, "countries", ALL_COUNTRIES);
     uriMatcher.addURI(AUTHORITY, "countries/#", SINGLE_COUNTRY);
    }

    // system calls onCreate() when it starts up the provider.
    @Override
    public boolean onCreate() {
     // get access to the database helper
     dbHelper = new MyDatabaseHelper(getContext());
     return false;
    }
```

Return the MIME type to a content URI

```
//Return the MIME type corresponding to a content URI
@Override
public String getType(Uri uri) {

 switch (uriMatcher.match(uri)) {
 case ALL_COUNTRIES:
  return "vnd.android.cursor.dir/beni.ibk.com.StudentsProvider.countries";
 case SINGLE_COUNTRY:
  return "vnd.android.cursor.item/beni.ibk.com.StudentsProvider.countries";
 default:
  throw new IllegalArgumentException("Unsupported URI: " + uri);
 }
}
```

Inserting in the provider

```
// The insert() method adds a new row to the appropriate table, using the values
// in the ContentValues argument. If a column name is not in the ContentValues argument,
// I want to provide a default value for it either in my provider code or in
// my database schema.
@Override
public Uri insert(Uri uri, ContentValues values) {

 SQLiteDatabase db = dbHelper.getWritableDatabase();
 switch (uriMatcher.match(uri)) {
 case ALL_COUNTRIES:
  //do nothing
  break;
 default:
  throw new IllegalArgumentException("Unsupported URI: " + uri);
 }
 long id = db.insert(CountriesDb.SQLITE_TABLE, null, values);
 getContext().getContentResolver().notifyChange(uri, null);
 return Uri.parse(CONTENT_URI + "/" + id);
}
```

Checking the query

```
// The query() method must return a Cursor object, or if it fails,
// throw an Exception. As I'm using an SQLite database as a data storage,
// I simply return the Cursor returned by one of the query() methods of the
// SQLiteDatabase class. If the query does not match any rows, I return a
// Cursor instance whose getCount() method returns 0. I return null only
// if an internal error occurred during the query process.
@Override
public Cursor query(Uri uri, String[] projection, String selection,
  String[] selectionArgs, String sortOrder) {

 SQLiteDatabase db = dbHelper.getWritableDatabase();
 SQLiteQueryBuilder queryBuilder = new SQLiteQueryBuilder();
 queryBuilder.setTables(CountriesDb.SQLITE_TABLE);

 switch (uriMatcher.match(uri)) {
 case ALL_COUNTRIES:
  //do nothing
  break;
 case SINGLE_COUNTRY:
  String id = uri.getPathSegments().get(1);
  queryBuilder.appendWhere(CountriesDb.KEY_ROWID + "=" + id);
  break;
 default:
  throw new IllegalArgumentException("Unsupported URI: " + uri);
 }

 Cursor cursor = queryBuilder.query(db, projection, selection,
   selectionArgs, null, null, sortOrder);
 return cursor;

}
```

Deleting data in the provider

```
// The delete() method deletes rows based on the seletion or if an id is
// provided then it deleted a single row. The methods returns the numbers
// of records delete from the database. If you choose not to delete the data
// physically then just update a flag here.
@Override
public int delete(Uri uri, String selection, String[] selectionArgs) {

 SQLiteDatabase db = dbHelper.getWritableDatabase();
 switch (uriMatcher.match(uri)) {
 case ALL_COUNTRIES:
  //do nothing
  break;
 case SINGLE_COUNTRY:
  String id = uri.getPathSegments().get(1);
  selection = CountriesDb.KEY_ROWID + "=" + id
  + (!TextUtils.isEmpty(selection) ?
    " AND (" + selection + ')' : "");
  break;
 default:
  throw new IllegalArgumentException("Unsupported URI: " + uri.toString());
 }
 int deleteCount = db.delete(CountriesDb.SQLITE_TABLE, selection, selectionArgs);
 getContext().getContentResolver().notifyChange(uri, null);
 return deleteCount;
}
```

Updating data in the provider

```
// The update method() is same as delete() which updates multiple rows
// based on the selection or a single row if the row id is provided. The
// update method returns the number of updated rows.
@Override
public int update(Uri uri, ContentValues values, String selection,
  String[] selectionArgs) {
 SQLiteDatabase db = dbHelper.getWritableDatabase();
 switch (uriMatcher.match(uri)) {
 case ALL_COUNTRIES:
  //do nothing
  break;
 case SINGLE_COUNTRY:
  String id = uri.getPathSegments().get(1);
  selection = CountriesDb.KEY_ROWID + "=" + id
  + (!TextUtils.isEmpty(selection) ?
    " AND (" + selection + ')' : "");
  break;
 default:
  throw new IllegalArgumentException("Unsupported URI: " + uri);
 }
 int updateCount = db.update(CountriesDb.SQLITE_TABLE, values, selection, selectionArgs);
 getContext().getContentResolver().notifyChange(uri, null);
 return updateCount;
}
}
```

Bibliography

Android. (n.d.). Retrieved 10 24, 2014, from Android: http://www.android.com

Bolton, M. (2011, 07 06). *What is android?* Retrieved 10 24, 2014, from Techractar: http://www.techractar.com