Android Database Example

In Android programming, storing data into a database is not a common practice, as we have other and most suitable methods to store our data, such as the [SharedPreferences](http://developer.android.com/reference/android/content/SharedPreferences.html) way.

However, we do have a very strong tool, that can help our Android Application communicate with a database, and this is [SQLite](http://developer.android.com/reference/android/database/sqlite/package-summary.html). SQLite is an Open Source Database for structured data in relational databases. It is embedded in Android, to you don’t have to do anything special to set up or administer an SQLite server.

This example shows how to work with an SQLite database and how to make the basic actions. We will do this by following the DataModel and DataHandling structure. We will use a data access object (DAO) to manage the handling of the database connection and for accessing and modifying the data. By dealing with the database entries as Data Objects, it will be easier for us to present the sql queries results on our user interface.

Let’s start! For our example will use the following tools in a Windows 64-bit or an OS X platform:

* JDK 1.7
* Eclipse 4.2 Juno
* Android SDK 4.4.2

take a closer look:

Начало формы

Want to create a kick-ass Android App?

|  |
| --- |
| Subscribe to our newsletter and download the Android UI Design mini-book right now!  With this book, you will delve into the fundamentals of Android UI design. You will understand user input, views and layouts, as well as adapters and fragments. Furthermore, you will learn how to add multimedia to an app and also leverage themes and styles! |

Конец формы

1. Create a New Android Application Project

**Tip**  
You may skip project creation and jump directly to the [**beginning of the example**](https://examples.javacodegeeks.com/android/core/database/android-database-example/#code) below.

Open Eclipse IDE and go to File → New → Project → Android Application Project.

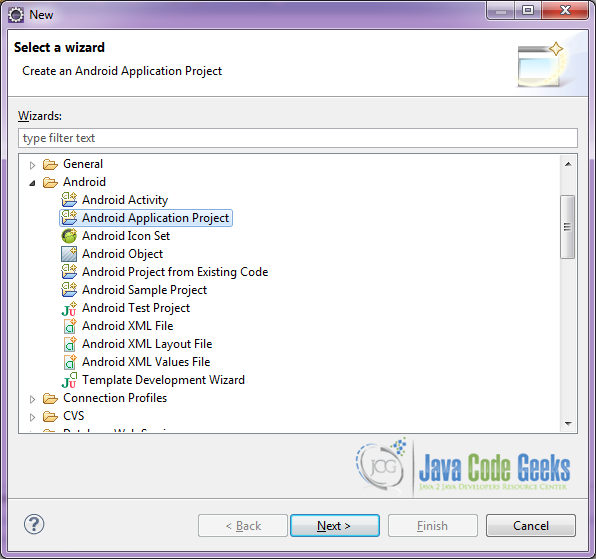
[](http://examples.javacodegeeks.com/wp-content/uploads/2014/08/AndroidExample1.png)

Figure 1. Create a new Android project

Specify the name of the application, the project and the package and then click Next.

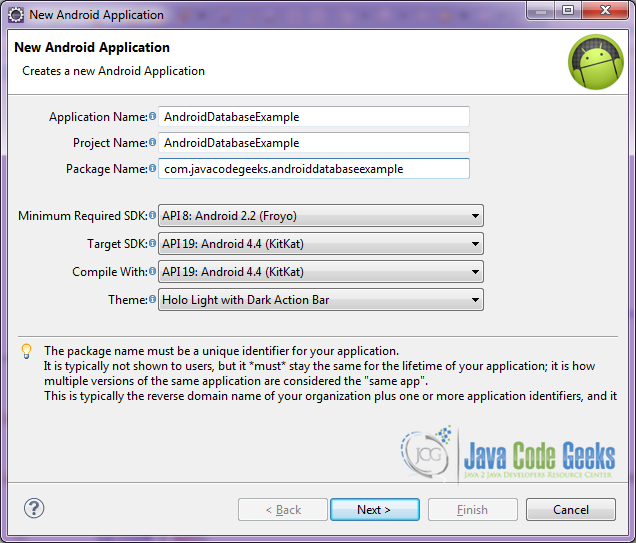
[](http://examples.javacodegeeks.com/wp-content/uploads/2014/10/AndroidDatabaseExample-1.jpg)

Figure 2. Create a new Android project name

In the next window, the “Create Activity” option should be checked. The new created activity will be the main activity of your project. Then press Next button.

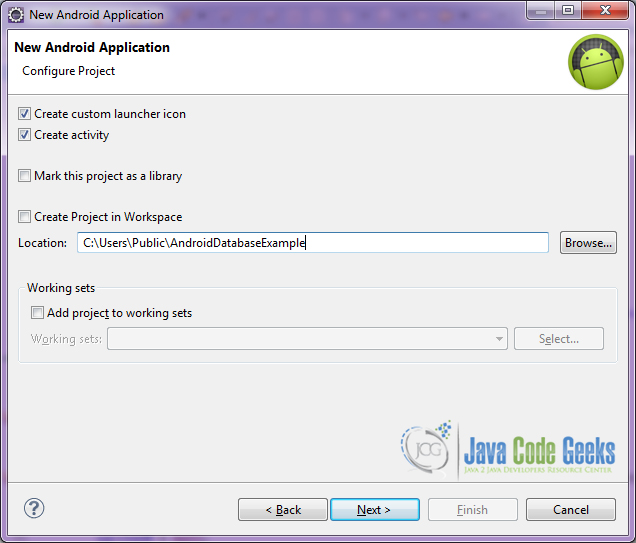
[](http://examples.javacodegeeks.com/wp-content/uploads/2014/10/AndroidDatabaseExample-2.jpg)

Figure 3. Configure the project

In “Configure Launcher Icon” window you should choose the icon you want to have in your app. We will use the default icon of android, so click Next.

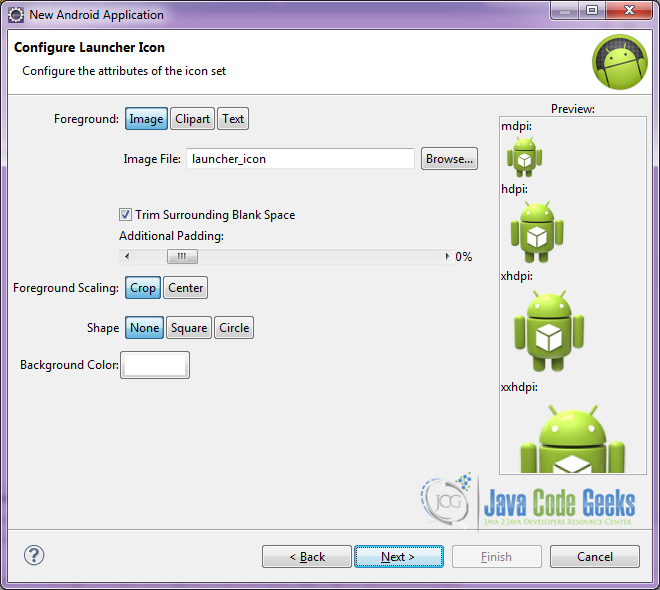
[](http://examples.javacodegeeks.com/wp-content/uploads/2014/08/AndroidExample4.png)

Figure 4. Configure the launcher icon

Select the “Blank Activity” option and press Next.

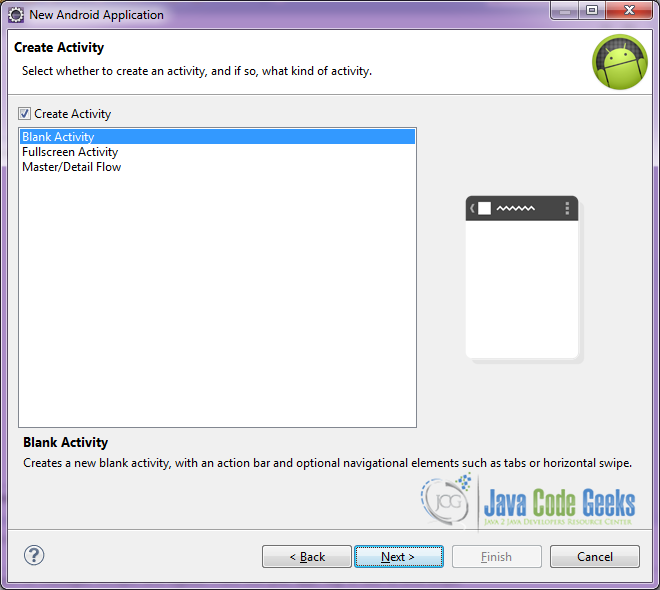
[](http://examples.javacodegeeks.com/wp-content/uploads/2014/08/AndroidExample5.png)

Figure 5. Create the activity and select its type

You have to specify a name for the new Activity and a name for the layout description of your app. The .xml file for the layout will automatically be created in the res/layout folder. It will also be created a fragment layout xml, that we are not going to use in this project and you can remove it if you want. Then press Finish.

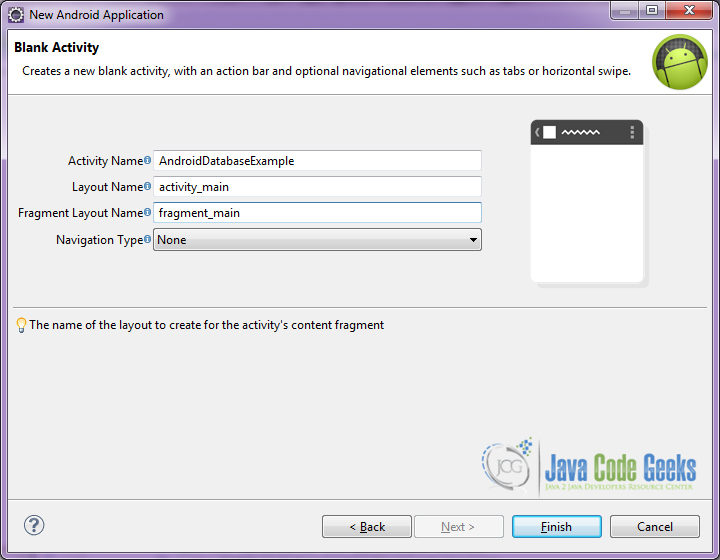
[](http://examples.javacodegeeks.com/wp-content/uploads/2014/10/AndroidDatabaseExample-3.jpg)

Figure 6. Create a new blank activity

Here you can see, how will the structure of the project become when finished:

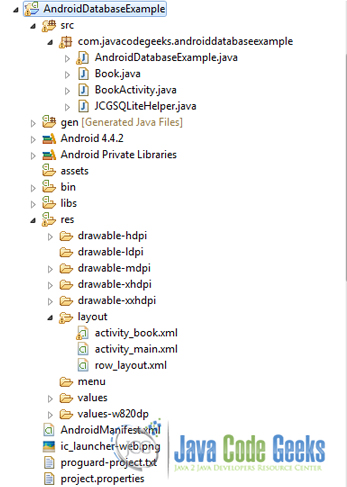
[](http://examples.javacodegeeks.com/wp-content/uploads/2014/10/AndroidDatabaseExample-4.jpg)

Figure 7. The tree of the project

2. Creating the layout of the main AndroidDatabaseExample

We are going to make a very simple layout xml for the AndroidDatabaseExample.class, that only consists of a LinearLayoutthat contains the one ListView.

Open res/layout/activity\_main.xml, go to the respective xml tab and paste the following:

*activity\_main.xml*

<LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>"

    xmlns:tools="<http://schemas.android.com/tools>"

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent"

    android:background="#ffffff"

    android:orientation="vertical" >

    <ListView

        android:id="@android:id/list"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content" >

    </ListView>

</LinearLayout>

3. Creating the Database and Data Model of the Activity AndroidDatabaseExample

Open src/com.javacodegeeks.androidcursorexample/Book.java file and paste the code below.

*Book.java*

package com.javacodegeeks.androiddatabaseexample;

public class Book {

    private int id;

    private String title;

    private String author;

    public Book(){}

    public Book(String title, String author) {

        super();

        this.title = title;

        this.author = author;

    }

    public int getId() {

        return id;

    }

    public void setId(int id) {

        this.id = id;

    }

    public String getTitle() {

        return title;

    }

    public void setTitle(String title) {

        this.title = title;

    }

    public String getAuthor() {

        return author;

    }

    public void setAuthor(String author) {

        this.author = author;

    }

    @Override

    public String toString() {

        return "Book [id=" + id + ", title=" + title + ", author=" + author

                + "]";

    }

}

The Book class is our model and contains the data we will save in the database and show in the user interface.

Now, open src/com.javacodegeeks.androidcursorexample/JCGSQLiteHelper.java file and paste the code below.

*JCGSQLiteHelper.java*

package com.javacodegeeks.androiddatabaseexample;

import java.util.LinkedList;

import java.util.List;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

public class JCGSQLiteHelper extends SQLiteOpenHelper {

    // database version

    private static final int database\_VERSION = 1;

    // database name

    private static final String database\_NAME = "BookDB";

    private static final String table\_BOOKS = "books";

    private static final String book\_ID = "id";

    private static final String book\_TITLE = "title";

    private static final String book\_AUTHOR = "author";

    private static final String[] COLUMNS = { book\_ID, book\_TITLE, book\_AUTHOR };

    public JCGSQLiteHelper(Context context) {

        super(context, database\_NAME, null, database\_VERSION);

    }

    @Override

    public void onCreate(SQLiteDatabase db) {

        // SQL statement to create book table

        String CREATE\_BOOK\_TABLE = "CREATE TABLE books ( " + "id INTEGER PRIMARY KEY AUTOINCREMENT, " + "title TEXT, " + "author TEXT )";

        db.execSQL(CREATE\_BOOK\_TABLE);

    }

    @Override

    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

        // drop books table if already exists

        db.execSQL("DROP TABLE IF EXISTS books");

        this.onCreate(db);

    }

    public void createBook(Book book) {

        // get reference of the BookDB database

        SQLiteDatabase db = this.getWritableDatabase();

        // make values to be inserted

        ContentValues values = new ContentValues();

        values.put(book\_TITLE, book.getTitle());

        values.put(book\_AUTHOR, book.getAuthor());

        // insert book

        db.insert(table\_BOOKS, null, values);

        // close database transaction

        db.close();

    }

    public Book readBook(int id) {

        // get reference of the BookDB database

        SQLiteDatabase db = this.getReadableDatabase();

        // get book query

        Cursor cursor = db.query(table\_BOOKS, // a. table

                COLUMNS, " id = ?", new String[] { String.valueOf(id) }, null, null, null, null);

        // if results !=null, parse the first one

        if (cursor != null)

            cursor.moveToFirst();

        Book book = new Book();

        book.setId(Integer.parseInt(cursor.getString(0)));

        book.setTitle(cursor.getString(1));

        book.setAuthor(cursor.getString(2));

        return book;

    }

    public List getAllBooks() {

        List books = new LinkedList();

        // select book query

        String query = "SELECT  \* FROM " + table\_BOOKS;

        // get reference of the BookDB database

        SQLiteDatabase db = this.getWritableDatabase();

        Cursor cursor = db.rawQuery(query, null);

        // parse all results

        Book book = null;

        if (cursor.moveToFirst()) {

            do {

                book = new Book();

                book.setId(Integer.parseInt(cursor.getString(0)));

                book.setTitle(cursor.getString(1));

                book.setAuthor(cursor.getString(2));

                // Add book to books

                books.add(book);

            } while (cursor.moveToNext());

        }

        return books;

    }

    public int updateBook(Book book) {

        // get reference of the BookDB database

        SQLiteDatabase db = this.getWritableDatabase();

        // make values to be inserted

        ContentValues values = new ContentValues();

        values.put("title", book.getTitle()); // get title

        values.put("author", book.getAuthor()); // get author

        // update

        int i = db.update(table\_BOOKS, values, book\_ID + " = ?", new String[] { String.valueOf(book.getId()) });

        db.close();

        return i;

    }

    // Deleting single book

    public void deleteBook(Book book) {

        // get reference of the BookDB database

        SQLiteDatabase db = this.getWritableDatabase();

        // delete book

        db.delete(table\_BOOKS, book\_ID + " = ?", new String[] { String.valueOf(book.getId()) });

        db.close();

    }

}

This class is responsible for all the actions that will take place on the database. Let’s see in detail the code above.

The JCGSQLiteHelper.class contains the basic methods:  
onCreate(SQLiteDatabase db), called when the database is created for the first time.  
onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion), called when the database needs to be upgraded.

For our example, we have extended our data management class, in order to include the methods for inserting, deleting, and updating a book in our database.

|  |
| --- |
| public void createBook(Book book) { |
| SQLiteDatabase db = this.getWritableDatabase(); |
| ContentValues values = new ContentValues(); |
| values.put(book\_TITLE, book.getTitle()); |
| values.put(book\_AUTHOR, book.getAuthor()); |
|  |
| db.insert(table\_BOOKS, null, values); |
| db.close(); |
| } |

In the code snipped above, we get reference of the BookDB database and create an insert query in order to insert the values of the **book\_TITLE** and **book\_AUTHOR** rows. We execute the insert and then close the transaction.

public Book readBook(int id) {

    SQLiteDatabase db = this.getReadableDatabase();

    Cursor cursor = db.query(table\_BOOKS, // a. table

            COLUMNS, " id = ?", new String[] { String.valueOf(id) }, null, null, null, null);

    if (cursor != null)

        cursor.moveToFirst();

    Book book = new Book();

    book.setId(Integer.parseInt(cursor.getString(0)));

    book.setTitle(cursor.getString(1));

    book.setAuthor(cursor.getString(2));

    return book;

}

With the code above, we get reference of the BookDB database and create an search query in order to find a specific book entry. We execute the search and then we return the first of the results as a book Object.

In the same way we get all the books of the database:

public List getAllBooks() {

    List books = new LinkedList();

    // select book query

    String query = "SELECT  \* FROM " + table\_BOOKS;

    // get reference of the BookDB database

    SQLiteDatabase db = this.getWritableDatabase();

    Cursor cursor = db.rawQuery(query, null);

    // parse all results

    Book book = null;

    if (cursor.moveToFirst()) {

        do {

            book = new Book();

            book.setId(Integer.parseInt(cursor.getString(0)));

            book.setTitle(cursor.getString(1));

            book.setAuthor(cursor.getString(2));

            // Add book to books

            books.add(book);

        } while (cursor.moveToNext());

    }

    return books;

}

We have also implemented an update book method:

public int updateBook(Book book) {

    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();

    values.put("title", book.getTitle());

    values.put("author", book.getAuthor());r

    int i = db.update(table\_BOOKS, values, book\_ID + " = ?", new String[] { String.valueOf(book.getId()) });

    db.close();

    return i;

}

In the code snipped above, we get reference of the BookDB database and create an update query in order to change the values of an already added book in the database. We execute the update and then close the transaction.

public void deleteBook(Book book) {

    SQLiteDatabase db = this.getWritableDatabase();

    db.delete(table\_BOOKS, book\_ID + " = ?", new String[] { String.valueOf(book.getId()) });

    db.close();

}

With the above code we delete a book entry from our database, after of course, we take the reference of the BookDB database and create an delete query.

4. Creating the source code of the main AndroidDatabaseExampleActivity

Open src/com.javacodegeeks.androidcursorexample/AndroidDatabaseExample.java file and paste the code below.

*AndroidDatabaseExample.java*

package com.javacodegeeks.androiddatabaseexample;

import java.util.ArrayList;

import java.util.List;

import android.app.ListActivity;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemClickListener;

import android.widget.ArrayAdapter;

public class AndroidDatabaseExample extends ListActivity implements OnItemClickListener {

    JCGSQLiteHelper db = new JCGSQLiteHelper(this);

    List list;

    ArrayAdapter myAdapter;

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

        // drop this database if already exists

        db.onUpgrade(db.getWritableDatabase(), 1, 2);

        db.createBook(new Book("The Great Gatsby", "F. Scott Fitzgerald"));

        db.createBook(new Book("Anna Karenina", "Leo Tolstoy"));

        db.createBook(new Book("The Grapes of Wrath", "John Steinbeck"));

        db.createBook(new Book("Invisible Man", "Ralph Ellison"));

        db.createBook(new Book("Gone with the Wind", "Margaret Mitchell"));

        db.createBook(new Book("Pride and Prejudice", "Jane Austen"));

        db.createBook(new Book("Sense and Sensibility", "Jane Austen"));

        db.createBook(new Book("Mansfield Park", "Jane Austen"));

        db.createBook(new Book("The Color Purple", "Alice Walker"));

        db.createBook(new Book("The Temple of My Familiar", "Alice Walker"));

        db.createBook(new Book("The waves", "Virginia Woolf"));

        db.createBook(new Book("Mrs Dalloway", "Virginia Woolf"));

        db.createBook(new Book("War and Peace", "Leo Tolstoy"));

        // get all books

        list = db.getAllBooks();

        List listTitle = new ArrayList();

        for (int i = 0; i < list.size(); i++) {

            listTitle.add(i, list.get(i).getTitle());

        }

        myAdapter = new ArrayAdapter(this, R.layout.row\_layout, R.id.listText, listTitle);

        getListView().setOnItemClickListener(this);

        setListAdapter(myAdapter);

    }

    @Override

    public void onItemClick(AdapterView arg0, View arg1, int arg2, long arg3) {

        // start BookActivity with extras the book id

        Intent intent = new Intent(this, BookActivity.class);

        intent.putExtra("book", list.get(arg2).getId());

        startActivityForResult(intent, 1);

    }

    @Override

    protected void onActivityResult(int requestCode, int resultCode, Intent data) {

        super.onActivityResult(requestCode, resultCode, data);

        // get all books again, because something changed

        list = db.getAllBooks();

        List listTitle = new ArrayList();

        for (int i = 0; i < list.size(); i++) {

            listTitle.add(i, list.get(i).getTitle());

        }

myAdapter = new ArrayAdapter(this, R.layout.row\_layout, R.id.listText, listTitle);

        getListView().setOnItemClickListener(this);

        setListAdapter(myAdapter);

    }

}

Open src/com.javacodegeeks.androidcursorexample/BookActivity.java file and paste the code below.

*BookActivity.java*

package com.javacodegeeks.androiddatabaseexample;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;

public class BookActivity extends Activity {

    TextView bookTitle;

    TextView authorName;

    Book selectedBook;

    JCGSQLiteHelper db;

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_book);

        bookTitle = (TextView) findViewById(R.id.title);

        authorName = (TextView) findViewById(R.id.author);

        // get the intent that we have passed from AndroidDatabaseExample

        Intent intent = getIntent();

        int id = intent.getIntExtra("book", -1);

        // open the database of the application context

        db = new JCGSQLiteHelper(getApplicationContext());

        // read the book with "id" from the database

        selectedBook = db.readBook(id);

        initializeViews();

    }

    public void initializeViews() {

        bookTitle.setText(selectedBook.getTitle());

        authorName.setText(selectedBook.getAuthor());

    }

    public void update(View v) {

        Toast.makeText(getApplicationContext(), "This book is updated.", Toast.LENGTH\_SHORT).show();

        selectedBook.setTitle(((EditText) findViewById(R.id.titleEdit)).getText().toString());

        selectedBook.setAuthor(((EditText) findViewById(R.id.authorEdit)).getText().toString());

        // update book with changes

        db.updateBook(selectedBook);

        finish();

    }

    public void delete(View v) {

        Toast.makeText(getApplicationContext(), "This book is deleted.", Toast.LENGTH\_SHORT).show();

        // delete selected book

        db.deleteBook(selectedBook);

        finish();

    }

}

5. Creating the layout of the BookActivity

We are going to make a simple layout xml for the BookActivity.class, the Activity that can delete and update the book entries in the database.

Open res/layout/activity\_book.xml, go to the respective xml tab and paste the following:

*activity\_book.xml*

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>"

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent"

    android:orientation="vertical"

    android:padding="25dp" >

    <LinearLayout

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:layout\_marginBottom="10dp"

        android:layout\_marginLeft="5dp" >

        <TextView

            android:id="@+id/titleLabel"

            android:layout\_width="wrap\_content"

            android:layout\_height="wrap\_content"

            android:text="Title: "

            android:textAppearance="?android:attr/textAppearanceLarge" />

        <TextView

            android:id="@+id/title"

            android:layout\_width="wrap\_content"

            android:layout\_height="wrap\_content"

            android:text="Book Title"

            android:textAppearance="?android:attr/textAppearanceLarge" />

    </LinearLayout>

    <LinearLayout

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:layout\_marginBottom="10dp"

        android:layout\_marginLeft="5dp" >

        <TextView

            android:id="@+id/authorLabel"

            android:layout\_width="wrap\_content"

            android:layout\_height="wrap\_content"

            android:text="Author: "

            android:textAppearance="?android:attr/textAppearanceLarge" />

        <TextView

            android:id="@+id/author"

            android:layout\_width="wrap\_content"

            android:layout\_height="wrap\_content"

            android:text="Book Title"

            android:textAppearance="?android:attr/textAppearanceLarge" />

    </LinearLayout>

    <Button

        android:id="@+id/delete"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:layout\_marginBottom="30dp"

        android:onClick="delete"

        android:text="Delete Book" />

    <LinearLayout

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:layout\_marginBottom="10dp"

        android:layout\_marginLeft="5dp" >

        <TextView

            android:id="@+id/titleEditLabel"

            android:layout\_width="wrap\_content"

            android:layout\_height="wrap\_content"

            android:text="Title: "

            android:textAppearance="?android:attr/textAppearanceLarge" />

        <EditText

            android:id="@+id/titleEdit"

            android:layout\_width="wrap\_content"

            android:layout\_height="wrap\_content"

            android:layout\_marginLeft="5dp"

            android:layout\_weight="1"

            android:ems="10" >

            <requestFocus />

        </EditText>

    </LinearLayout>

    <LinearLayout

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:layout\_marginBottom="10dp"

        android:layout\_marginLeft="5dp" >

        <TextView

            android:id="@+id/authorEditLabel"

            android:layout\_width="wrap\_content"

            android:layout\_height="wrap\_content"

            android:text="Author: "

            android:textAppearance="?android:attr/textAppearanceLarge" />

        <EditText

            android:id="@+id/authorEdit"

            android:layout\_width="wrap\_content"

            android:layout\_height="wrap\_content"

            android:layout\_marginLeft="5dp"

            android:layout\_weight="1"

            android:ems="10" >

        </EditText>

    </LinearLayout>

    <Button

        android:id="@+id/update"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:onClick="update"

        android:text="Update Book" />

</LinearLayout>

6. Build, compile and run

When we build, compile and run our project, the main AndroidDatabaseExample should look like this:

[](http://examples.javacodegeeks.com/wp-content/uploads/2014/10/AndroidDatabaseExample-5.jpg)

Figure 8. Figure This is how the main Activity looks like.

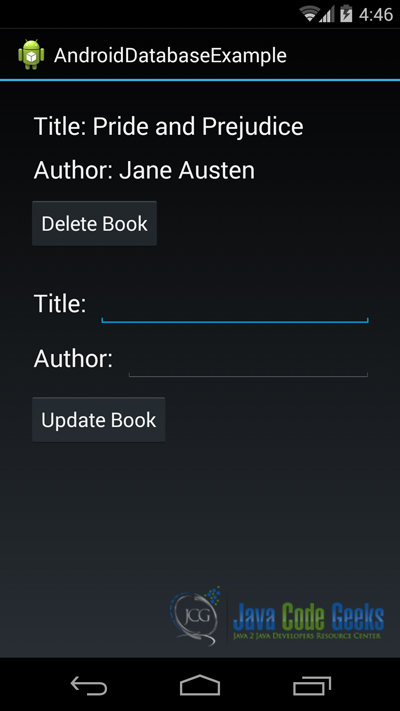
[](http://examples.javacodegeeks.com/wp-content/uploads/2014/10/AndroidDatabaseExample-6.jpg)

Figure 9. This is how the BookActivity looks like.

[](http://examples.javacodegeeks.com/wp-content/uploads/2014/10/AndroidDatabaseExample-7.jpg)

Figure 10. This is our book list, after we have deleted the book in the BookActivity.

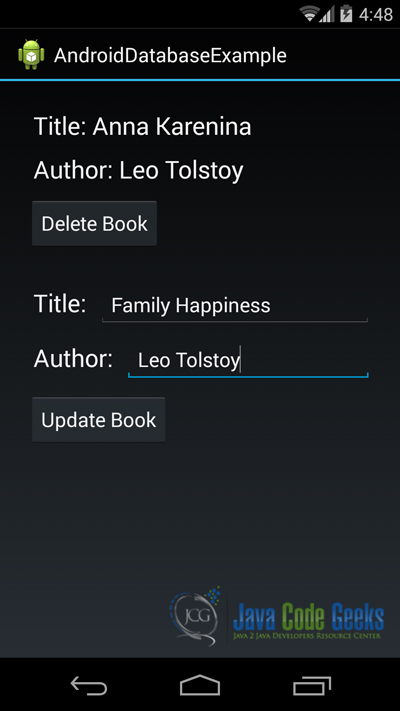
[](http://examples.javacodegeeks.com/wp-content/uploads/2014/10/AndroidDatabaseExample-8.jpg)

Figure 11. In this figure we see how we can update a book in our database.

Download the Eclipse Project

This was an example of Android AndroidDatabaseExample.

**Download**You can download the full source code of this example here: **[AndroidDatabaseExample](http://examples.javacodegeeks.com/wp-content/uploads/2014/10/AndroidDatabaseExample.zip)**