Android Veritabanı İşlemleri

Bu makalede, Android ile veritabanı uygulamaları geliştirirken yapılması gerekenler anlatılacaktır. Öncelikle kısaca ihtiyaç duyulan yapılardan bahsedelim:

1. **SQLite** Veritabanı
2. **SQLiteOpenHelper abstract** sınıfı
3. **SQLiteDatabase** sınıfı
4. Veri ekleme ve güncelleme için **ContentValues** sınıfı
5. Select işlemleri için **Cursor** sınıfı

SQLite Veritabanı Nedir?

Açık kaynak kodlu bir veritabanıdır. **SQL** sentaksı, transaction'lar ve prepared statement'lar gibi standart ilişkisel veritabanı özelliklerini destekler. Koşma zamanında yaklaşık **250 KByte** gibi küçük bir memory alanına ihtiyaç duyduğu için gömülü sistemlerde, telefonlarda rahatça kullanılabilmektedir.

Bu veritabanı her Android cihazında gömülü olarak gelir. Bu nedenle Android cihazı veritabanının kurulmasına ihtiyaç duymaz. Android'te veritabanı kullanan her uygulama kendi **private** veritabanına sahip olduğu için veritabanında yönetici paneli bulunmaz. Bize düşen görev sadece sql işlemleri yapmaktır, diğer tüm işleri Android kendisi düzenler.

SQLite veritabanına erişmek demek dosya sistemine erişmek demek olduğu için bu işlem yavaş olabilir. Bundan dolayı veritabanı işlemlerinin **asekron** yapılması tavsiye edilir.

Oluşturulan veritabanı **DATA/data/APP_NAME/databases/FILENAME** şeklinde tutulur. Burada;

DATA: Environment.getDataDirectory() metodunun dönderdiği değerdir. Genelde dönderdiği değer /data/ dır.
APP_NAME: Uygulamanızın paket adı. **Örn:** mucayufa.android.database
databases: databases olarak direkt kullanılır.
FILENAME: Veritabanının adı. **Örn:** softwarevol

Sonuç olarak yukarıdaki örnek değerlere göre veritabanının tutulduğu yer: **/data/data/mucayufa.android.database/databases/softwarevol**

Not: Oluşan veritabanının içeriğini görmek için yapılması gereken adımların anlatıldığı makale için **tıklayınız** (/tutorial/Android-Emulator-Yerine-Cihaz-Kullanilirken-Olusan-Veritabanini-Bilgisayara-Kopyalamak)

SQLiteOpenHelper Abstract Sınıfı

Android uygulamasında veritabanı **yaratmak** ve **upgrade** etmek için bu sınıf kullanılır. Bu sınıfta bulunan **onCreate()** ve **onUpgrade()** metodları **abstract** tanımlanmıştır. Bundan dolayı biz bu sınıfı direkt kullanamayız. Bunun yerine bu sınıfı **extends** eden bir sınıf tanımlamak gereklidir. Bu iki metod parametre olarak **SQLiteDatabase** sınıfını alır.

Ayrıca **getReadableDatabase()** ve **getWriteableDatabase()** isminde iki metod daha vardır. Bu metodlardan ilki okuma modunda **SQLiteDatabase** nesnesi dönderirken, ikincisi yazma modunda **SQLiteDatabase** nesnesi dönderir.

SQLiteDatabase Sınıfı

open, query, update ve close işlemlerini sağlayan sınıftır. Daha özel olarak ifade etmek istersek, **insert(), rawQuery(), update()** ve **delete()** metodlarını içerir. Ayrıca **execSQL()** metodu sayesinde **SQL** ifadelerini direkt çalıştırmayı sağlar.

insert() ve **update()** metodlarının parametrelerinden biri **ContentValues** sınıfıdır. Bu sınıf key/values şeklinde değerlerlerin belirtilmesini sağlar. Burada **key** tablo sütunu adını gösterirken, **value** ise o sütünun değerini gösterir.

Cursor Sınıfı

Bir query **Cursor** nesnesi dönderir. Query sonucunda dönen her bir satır bir **Cursor** nesnesi tarafından temsil edilir. Bu sayede, Android query sonuçlarını verimli bir şekilde bufferlayabildiği için tüm veriyi memory'e yüklemeye ihtiyaç duymaz.

Cursor sınıfı içerisinde bulunan **getCount()** metodu sayesinde, query işlemi sonucunda ne kadar satır döndüğünü bulabiliriz.

Satırlar arasında hareket edebilmek için **moveToFirst()** ve **moveToNext()** metodları kullanılmaktadır. **isAfterLast()** metodu ile de son satıra gelinip gelinmediği kontrol edilir.

Ayrıca **SQLite** veritabanında veri tipleri **NULL, INTEGER, REAL, TEXT, BLOB** olduğu için getInt(**int** columnIndex), getDouble(**int** columnIndex), getBlob(**int** columnIndex), getFloat(**int** columnIndex), metodları ile de query'den dönen satırlardaki sütun değerleri alınır.

Gerekli işlemler yapıldıktan sonra **Cursor** nesnesi **close()** metodu ile kapatılmalıdır.

Basit Bir Android Veritabanı Uygulaması

Bu uygulamada, kullanıcının kayıt olması ve kayıtlı kullanıcıların bilgilerinin çekilmesi işlemlerinin yapıldığı butonlar ve text field'ler olacak. Kullanıcı, kullanıcı adı ve şifre kısımlarını doldurduktan sonra kayıt butonuna tıkladığında kayıt olacak, **Tüm Verileri Getir** butonuna tıkladığında ise kayıtlı kullanıcıların kullanıcı adları getirilecek. Şimdi sırasıyla bu uygulamamızı yapalım:

DatabaseHelper Sınıfı

```
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;
import softwarevol.android.database.simple.tables.User;

import java.util.ArrayList;
import java.util.List;


public class DatabaseHelper extends SQLiteOpenHelper {
    private static DatabaseHelper sInstance = null;
    // Logcat tag
    private static final String LOG = "DatabaseHelper";

    // Database Versiyonu
    private static final int DATABASE_VERSION = 1;

    // Database Adi
    private static final String DATABASE_NAME = "users";

    // Table Adlari
    private static final String TABLE_USER = "user";

    //User tablosunun sütunlari
    private static final String USER_ID = "id";
    private static final String USER_NAME = "userName";
    private static final String USER_PASSWORD = "password";


    // Table Create Statements
    // User table
    private static final String CREATE_TABLE_USER = "CREATE TABLE "
            + TABLE_USER + "(" + USER_ID + " INTEGER PRIMARY KEY," + USER_NAME
            + " TEXT," + USER_PASSWORD + " TEXT)";

    /**
     * Bu siniftan sadece tek bir tane nesne olusmasini saglar.
     * Bu sayede memory leak meydana gelmez.
     *
     * @param context
     * @return DatabaseHelper nesnesi
     */
    public static DatabaseHelper getInstance(Context context) {
        if (sInstance == null) {
            sInstance = new DatabaseHelper(context.getApplicationContext());
        }
        return sInstance;
    }

    private DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {

        // creating required tables
        db.execSQL(CREATE_TABLE_USER);

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // on upgrade drop older tables
```

```
68          db.execSQL("DROP TABLE IF EXISTS " + TABLE_USER);
69          // create new tables
70          onCreate(db);
71      }
72
73      /**
74       * Veritabanini kapatir
75       */
76      public void closeDB() {
77          SQLiteDatabase db = this.getReadableDatabase();
78          if (db != null && db.isOpen())
79              db.close();
80      }
81
82      /**
83       * Yeni kullanici eklemeyi saglar
84       *
85       * @param user eklenecek kullanici
86       * @return eklenen kullanicinin id'si doner, hata durumunda -1 doner
87       */
88      public long createUser(User user) {
89          SQLiteDatabase db = this.getWritableDatabase();
90
91          ContentValues values = new ContentValues();
92          values.put(USER_ID, user.getID());
93          values.put(USER_NAME, user.getUserName());
94          values.put(USER_PASSWORD, user.getPassword());
95
96          // insert row
97          return db.insert(TABLE_USER, null, values);
98
99      }
100
101     /**
102      * id'ye gore kullanici getirir.
103      *
104      * @param userId belirtilen id degerine gore kullaniciyi getirir
105      * @return user eger userId ile belirtilen id'ye sahip kullanici varsa donderir,
106      * aksi taktirde null doner
107      */
108     public User getUser(long userId) {
109         SQLiteDatabase db = this.getReadableDatabase();
110
111         String selectQuery = "SELECT  * FROM " + TABLE_USER + " WHERE "
112                 + USER_ID + " = " + userId;
113
114         Log.e(LOG, selectQuery);
115
116         Cursor c = db.rawQuery(selectQuery, null);
117
118         if (c != null) {
119             c.moveToFirst();
120             User user = new User();
121             user.setID(c.getInt(c.getColumnIndex(USER_ID)));
122             user.setUserName((c.getString(c.getColumnIndex(USER_NAME))));
123             user.setPassword(c.getString(c.getColumnIndex(USER_PASSWORD)));
124             return user;
125
126         } else {
127             return null;
128         }
129
130     }
131
132     /**
133      * Tum kullanicilari getirir
134      *
135      * @return Kayitli kullanicilar
136      */
137     public List<User> getAllUsers() {
138         List<User> users = new ArrayList<User>();
139         String selectQuery = "SELECT  * FROM " + TABLE_USER;
140         Log.e(LOG, selectQuery);
141         SQLiteDatabase db = this.getReadableDatabase();
142         Cursor c = db.rawQuery(selectQuery, null);
143
144         // looping through all rows and adding to list
145         if (c.moveToFirst()) {
146             do {
147                 User user = new User();
148                 user.setID(c.getInt(c.getColumnIndex(USER_ID)));
149                 user.setUserName((c.getString(c.getColumnIndex(USER_NAME))));
150                 user.setPassword(c.getString(c.getColumnIndex(USER_PASSWORD)));
151
```

```
                // adding to user list
                users.add(user);
            } while (c.moveToNext());
        }

        return users;
    }

    /**
     * Kullaniciyi gunceller
     *
     * @param user guncellenecek kullanici nesnesi
     * @return etkilenen satir sayisi
     */
    public int updateUser(User user) {
        SQLiteDatabase db = this.getWritableDatabase();

        ContentValues values = new ContentValues();
        values.put(USER_ID, user.getID());
        values.put(USER_NAME, user.getUserName());
        values.put(USER_PASSWORD, user.getPassword());

        // updating row
        return db.update(TABLE_USER, values, USER_ID + " = ?",
                new String[]{String.valueOf(user.getID())});
    }

    /**
     * Kullaniciyi siler
     *
     * @param userId silinecek kullanici id'si
     */
    public void deleteUser(Integer userId) {
        SQLiteDatabase db = this.getWritableDatabase();
        db.delete(TABLE_USER, USER_ID + " = ?",
                new String[]{String.valueOf(userId)});
    }
}
```

Veritabanını oluşturmaktan ve upgrade etmekten sorumludur. Ayrıca arama, güncelleme, silme ve veritabanini kapatma işlemlerinden sorumludur.

DATABASE_VERSION değişkeninin kullanılmasının nedeni, yeni versiyon olduğu zaman önceki versiyonu kullanan kişilerin yeni versiyona geçmelerini sağlar. Constructor içerisinde kullanılan bu integer değişkenin kullanılması zorunludur.

onCreate() metodunda yaratılacak tabloların create query'leri execute edilir. Bu metod ilk kez çağrıldıktan sonra tekrar tekrar çağrılmaz. Veritabanı silindiği takdirde tekrar çağrılır. İlk çağrılma ise **getWritableDatabase()** veya **getReadableDatabase()** metodları çağrıldığında olur.

onUpgrade() metodu veritabanı upgrade olduğu zaman çağrılır. Örneğin Google Play Store'da uygulamanızı güncellediğiniz zaman, kullanıcılar sizin bu uygulamanızı güncellerse bu metod çağrılır. Bu metod içerisinde var olan tablo(lar) varsa silinip tekrar yaratılır, yoksa yaratılır.

User Sınıfı

```
public class User {
    private Integer ID;
    private String userName;
    private String password;

    public Integer getID() {
        return ID;
    }

    public void setID(Integer ID) {
        this.ID = ID;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
```

```
}
```

Bu sınıf **User** tablosunu temsil etmek için oluşturulmuştur

MainActivity Sınıfı

```
package softwarevol.android.database.simple;

import android.app.Activity;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import softwarevol.android.database.simple.config.DatabaseHelper;

import java.util.List;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btnRegister = (Button) findViewById(R.id.btnRegister);
        Button btnGet = (Button) findViewById(R.id.btnGet);
        btnRegister.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String userName = ((EditText) findViewById(R.id.reg_userName)).getText().toString();
                String password = ((EditText) findViewById(R.id.reg_password)).getText().toString();
                User user = new User();
                user.setUserName(userName);
                user.setPassword(password);
                DatabaseHelper helper = DatabaseHelper.getInstance(getApplicationContext());
                long result = helper.createUser(user);
                if (result > 0) {
                    TextView resultView = (TextView) findViewById(R.id.resultMessage);
                    resultView.setText("Yeni kullanıcı eklendi");
                    resultView.setText(Environment.getDataDirectory().toString());

                } else {
                    TextView resultView = (TextView) findViewById(R.id.resultMessage);
                    resultView.setText("Kullanıcı eklenirken hata oluştu");
                }
            }
        });
        btnGet.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                DatabaseHelper helper = DatabaseHelper.getInstance(getApplicationContext());
                List<User> userList = helper.getAllUsers();
                String userNames = "";
                for (User user : userList) {
                    userNames += user.getUserName();
                }
                TextView resultView = (TextView) findViewById(R.id.resultMessage);
                resultView.setText(userNames);
            }
        });
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        DatabaseHelper openHelper = DatabaseHelper.getInstance(getApplicationContext());
        if (openHelper != null) {
            openHelper.closeDB();
        }
    }

}
```

AndroidManifest.xml Dosyası

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="softwarevol.android.database.simple"
          android:versionCode="1"
          android:versionName="1.0">
    <application android:label="@string/app_name" android:icon="@drawable/icon">
```

```
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

res/layout/main.xml Dosyası

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Registration Form -->
 <!--FILL_PARENT and MATCH_PARENT are same thing,
  if the version the user is having is 2.2 or higher
  FILL_PARENT would be replaced by MATCH_PARENT automatically.
  So it's better to use FILL_PARENT, to support backward compatibility.-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="vertical"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:padding="10dip"
               >
    <!-- Full Name Label -->
    <TextView android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:textColor="#372c24"
              android:text="@string/form_userName"/>
    <EditText android:id="@+id/reg_userName"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:layout_marginTop="5dip"
              android:singleLine="true"
              android:layout_marginBottom="20dip"/>

    <!-- Password Label -->
    <TextView android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:textColor="#372c24"
              android:text="@string/form_password"/>
    <EditText android:id="@+id/reg_password"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:password="true"
              android:singleLine="true"
              android:layout_marginTop="5dip"/>
    <!-- Register Button -->
    <Button android:id="@+id/btnRegister"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dip"
            android:text="@string/form_button_save"/>
    <!-- Register Button -->
    <Button android:id="@+id/btnGet"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dip"
            android:text="@string/form_button_getAll"/>
    <!-- Link to Login Screen -->
    <TextView android:id="@+id/resultMessage"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:layout_marginTop="40dip"
              android:layout_marginBottom="40dip"
              android:text=""
              android:gravity="center"
              android:textSize="20dip"
              android:textColor="#025f7c"/>
</LinearLayout>
```

Burada dikkat edilmesi gereken husus şudur: Android 2.2 veya daha üst versiyonlarla birlikte gelen **match_parent** yerine, önceki versiyonlarla uyumlu olarak çalışmasını istiyorsanız, **fill_parent** kullanınız.

res/values/strings.xml Dosyası

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Simple Database Example!</string>
    <string name="app_name">Hello, Android Database</string>
    <string name="form_userName">Kullanıcı Adı</string>
    <string name="form_password">Şifre</string>
    <string name="form_button_save">Kayıt</string>
```

```
8       <string name="form_button_getAll">Tüm Kullanıcı Adları</string>
9   </resources>
```

Uygulamayı indirmek için **tıklayınız** (http://www.softwarevol.com/sources/codes/android/helloDatabase.zip)