BLM401 Mobil Cihazlar için ANDROID İşletim Sistemi

SQLite Veritabanı

GİRİŞ (1/4)

SQLite

- açık kaynak kodlu;
- sunucu gerektirmeyen;
- konfigürasyon ayarları gerektirmeyen;
- platformdan bağımsız;
- işlemsel (transactional);
- ilişkisel (relational);
- gömülü

veritabanı metodudur.

GİRİŞ (2/4)

SQLite

- sadece bir dosyadan ibarettir;
- diskte ve hafızada çok az yer kaplar;
- Android içinde hazır gelmektedir.

- Android uygulamalarında saklanması istenen veriler SQLite veritabanı oluşturarak saklanabilir;
- Android' de oluşturulan veritabanları /data/data/ <paket adı>/databases klasöründe durmaktadır;

GİRİŞ (3/4)

- Android, SQL cümlecikleri kurarak veritabanındaki verilere ulaşılmasını sağlayacak kütüphaneler sunmaktadır;

SQLite üzerinde

- veritabanı oluşturma;
- veritabanına kayıt ekleme ve sorgulama;
- kayıt güncelleme;
- silme gibi işlemler yapılabilir.

GİRİŞ (4/4)

- Yukarıda sıralanan işlemler bir örnek proje üzerinde anlatılacaktır;
- Projenin ismi merhabaSQLite olup bir kütüphane uygulamasıdır;
- Bu uygulamayı anlamak için sayfalar 140-158 dikkatle okunmalı ve kodları yazılarak çalıştırılmalıdır.

SQLite

- Embedded RDBMS
- Size – about 257 Kbytes
- Not a client/server architecture
- Accessed via function calls from the application
- Writing (insert, update, delete) locks the database, queries can be done in parallel

SQLite

Datastore – single, cross platform file (like an MS Access DB)

- Definitions
- Tables
- Indicies
- Data

SQLite Data Types

This is quite different than the normal SQL data types so please read:

http://www.sqlite.org/datatype3.html

SQLite Storage classes

- NULL – null value
- INTEGER - signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value
- REAL - a floating point value, 8-byte IEEE floating point number.
- TEXT - text string, stored using the database encoding (UTF-8, UTF-16BE or UTF-16LE).
- BLOB. The value is a blob of data, stored exactly as it was input.

android.database.sqlite

Contains the SQLite database management classes that an application would use to manage its own private database.

android.database.sqlite - Classes

- SQLiteCloseable - An object created from a SQLiteDatabase that can be closed.
- SQLiteCursor - A Cursor implementation that exposes results from a query on a SQLiteDatabase.
- SQLiteDatabase - Exposes methods to manage a SQLite database.

android.database.sqlite - Classes

- SQLiteOpenHelper - A helper class to manage database creation and version management.
- SQLiteProgram - A base class for compiled SQLite programs.
- SQLiteQuery - A SQLite program that represents a query that reads the resulting rows into a CursorWindow.

android.database.sqlite - Classes

- SQLiteQueryBuilder - a convenience class that helps build SQL queries to be sent to SQLiteDatabase objects.
- SQLiteStatement - A pre-compiled statement against a SQLiteDatabase that can be reused.

android.database.sqlite.SQLite Database

- Contains the methods for: creating, opening, closing, inserting, updating, deleting and quering an SQLite database

openOrCreateDatabase( )

- This method will open an existing database or create one in the application data area

```
import android.database.sqlite.SQLiteDatabase;

SQLiteDatabase  myDatabase;

myDatabase = openOrCreateDatabase ("my_sqlite_database.db" ,
                    SQLiteDatabase.CREATE_IF_NECESSARY , null);
```

Creating Tables

- Create a static string containing the SQLite CREATE statement, use the execSQL( ) method to execute it.

```
String  createAuthor = "CREAT TABLE   authors (
                                  id  INTEGER PRIMARY KEY
AUTOINCREMENT,
                                  fname  TEXT,
                                   lname  TEXT);

myDatabase.execSQL(createAuthor);
```

insert( )

- long insert(String table, String nullColumnHack, ContentValues values)

```
import  android.content.ContentValues;

ContentValues  values = new  ContentValues( );
values.put("firstname" , "J.K.");
values.put("lastname" , "Rowling");
long newAuthorID =  myDatabase.insert("tbl_authors" , "" ,
values);
```

update( )

- int update(String table, ContentValues values, String whereClause, String[ ] whereArgs)

```
public void updateBookTitle(Integer bookId,
String newTitle) {
   ContentValues  values = new
ContentValues();
   values.put("title" , newTitle);
   myDatabase.update("tbl_books" ,  values ,
      "id=?" , new String[ ] {bookId.toString() }
);
}
```

delete( )

- int delete(String table, String whereClause, String[] whereArgs)

```
public void deleteBook(Integer  bookId)  {
    myDatabase.delete("tbl_books" , "id=?" ,
        new String[ ]   { bookId.toString( )  } ) ;
}
```

android.database

- http://developer.android.com/reference/android/database/package-summary.html
- Contains classes and interfaces to explore data returned through a content provider.
- The main thing you are going to use here is the Cursor interface to get the data from the resultset that is returned by a query

  http://developer.android.com/reference/android/database/Cursor.html

Queries

- Method of SQLiteDatabase class and performs queries on the DB and returns the results in a Cursor object
- Cursor c = mdb.query(p1,p2,p3,p4,p5,p6,p7)
  - p1 ; Table name (String)
  - p2 ; Columns to return (String array)
  - p3 ; WHERE clause (use null for all, ?s for selection args)
  - p4 ; selection arg values for ?s of WHERE clause
  - p5 ; GROUP BY (null for none) (String)
  - p6 ; HAVING (null unless GROUP BY requires one) (String)
  - p7 ; ORDER BY (null for default ordering)(String)
  - p8 ; LIMIT (null for no limit) (String)

Tutorial

- http://www.screaming-penguin.com/node/7742

(son)

BAŞARILAR …