MODULE 10

DATABASE

Françoise Dubisy

TABLE OF CONTENT

Android Database

- Android stores database in private disk space associated to the application (internal storage)
  - Data is secure
  - By default this area is not accessible to other applications
- SQLite

Creating Database Using a SQL Helper

- SQLiteOpenHelper class
  - Use this class to obtain references to database
  - The system performs the potentially long-running operations of creating and updating the database
    - Only when needed
    - Not during app startup

Creating Database Using a SQL Helper

- Create a subclass of **SQLiteOpenHelper**
  - Override the onCreate(), onUpgrade() and onOpen() methods
  - Call getWritableDatabase() or getReadableDatabase() methods
    - **Note:** Because they can be long-running, call getWritableDatabase() or getReadableDatabase() in a background thread, such as with AsyncTask or IntentService

Creating Database Using a SQL Helper

- E.g,

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class BookDBHelper extends SQLiteOpenHelper {
  private static final String SQL_CREATE_ENTRIES =
    "create table book (bookid integer primary key autoincrement, title text, price real)";
  private static final String SQL_DELETE_ENTRIES =
     "drop table if exists book";
  public static final int DATABASE_VERSION = 1;
  public static final String DATABASE_NAME = "BookDB.db";

 public BookDBHelper(Context context) {
     super(context, DATABASE_NAME, null, DATABASE_VERSION);
 }
 public void onCreate(SQLiteDatabase db) {
     db.execSQL(SQL_CREATE_ENTRIES);
 }
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
     db.execSQL(SQL_DELETE_ENTRIES);
     onCreate(db);
 }
}
```

Datatypes In SQLite

- **NULL**
  - The value is a NULL value
- **INTEGER**
  - The value is a signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value
- **REAL**
  - The value is a floating point value, stored as an 8-byte IEEE floating point number
- **TEXT**
  - The value is a 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

Datatypes In SQLite

- **No Boolean Datatype**
  - Boolean values are stored as integers 0 (false) and 1 (true)
- **No Date and Time Datatype**
  - Built-in Date And Time Functions of SQLite are capable of storing dates and times as TEXT, REAL, or INTEGER values:
    - **TEXT** as ISO8601 strings ("YYYY-MM-DD HH:MM:SS.SSS")
    - **REAL** as Julian day numbers, the number of days since noon in Greenwich on November 24, 4714 B.C. according to the proleptic Gregorian calendar
    - **INTEGER** as Unix Time, the number of seconds since 1970-01-01 00:00:00 UTC

Inserting Data into Database

- To access database, instantiate subclass of SQLiteOpenHelper
- Get the data repository (**SQLiteDatabase** object) in write mode
  - Call **getWritableDatabase**()
- Create a new map of values, where column names are the keys
  - Using **ContentValues** object
- Then insert row
  - Call **insert** method on SQLiteDatabase object using ContentValues object

Inserting Data into Database

- E.g,

```
import model.Book;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;

public class BookDAO {

    private Context context;
    private BookDBHelper bookHelper;
    private SQLiteDatabase db;

    public BookDAO (Context context)
    {
        this.context = context;
    }

    public void addBook(Book book)
    {
        bookHelper = new BookDBHelper(context);
        db = bookHelper.getWritableDatabase();

        ContentValues values = new ContentValues();
        values.put("title", book.getTitle());
        values.put("price", book.getPrice());
        db.insert("book", null, values);

        db.close();
        bookHelper.close();
    }
```

Get the data repository in write mode

Insert row into database

Table name

Reading Data from Database

- Get the data repository (**SQLiteDatabase** object) in read mode
  - Call **getReadableDatabase**()
- Use the **query()** method on SQLiteDatabase object
  - Arguments
    - tableName : the table to query
    - projection: the columns to return
    - selection : the columns for the WHERE clause
    - selectionArgs : the values for the WHERE clause
    - group : the columns to group
    - filter: the filter for row groups
    - sortOrder : the columns to sort on
  - Result
    - A **Cursor** object

Reading Data from Database

- Then loop on the cursor to read rows
  - Use Cursor methods
    - **getCount()**
    - **moveToFirst()**
    - **moveToNext()**
    - **isAfterLast()**
- Access current row data using
  - Gettor according of type of row
    - E.g, **getInt, getString**, …

Reading Data from Database

- E.g, selecting a book based on its ID

```
public Book getBookById(Integer id)
{
    bookHelper = new BookDBHelper(context);
    db = bookHelper.getReadableDatabase();
    Book book = null;

    String[] projection = {"bookid" , "title" , "price"}; // The columns to return
    String selection = "bookid = ?";                      // The columns for the WHERE clause
    String[] selectionArgs = { String.valueOf(id) };      // The values for the WHERE clause
    String sortOrder = "title";                           // The sort order

    Cursor cursor = db.query("book", projection, selection, selectionArgs, null, null, sortOrder);

    if (cursor.getCount() != 0)
        {cursor.moveToFirst();
         book = new Book();
         book.set_id(cursor.getInt(cursor.getColumnIndex("bookid")));
         book.setTitle(cursor.getString(cursor.getColumnIndex("title")));
         book.setPrice(cursor.getDouble(cursor.getColumnIndex("price")));
        }

    db.close();
    bookHelper.close();
    return book;
}
```

Get the data repository in read mode

Reading Data from Database

- E.g, accessing all books

```
public ArrayList<Book> getAllBooks()
{
    bookHelper = new BookDBHelper(context);
    db = bookHelper.getReadableDatabase();
    ArrayList<Book> allBooks = new ArrayList<Book>();
    Book book;

    String[] projection = {"bookid" , "title" , "price"}; // The columns to return

    Cursor cursor = db.query("book", projection, null, null, null, null, null);

    if (cursor.getCount() != 0)
        {cursor.moveToFirst();
        while (cursor.isAfterLast() == false)
        { book = new Book();
          book.set_id(cursor.getInt(cursor.getColumnIndex("bookid")));
          book.setTitle(cursor.getString(cursor.getColumnIndex("title")));
          book.setPrice(cursor.getDouble(cursor.getColumnIndex("price")));
          allBooks.add(book);
          cursor.moveToNext();
        }
        }
    db.close();
    bookHelper.close();
    return allBooks;
}
```

Webography

- http://developer.android.com/training/basics/data-storage/databases.html
- http://developer.android.com/guide/topics/ui/layout/listview.html