

Software Engineering Large Practical

Working with databases in Android

Stephen Gilmore

School of Informatics, University of Edinburgh

October 16th, 2013

A database example

In this lecture we will look at an example Android application which creates a database of TODO notes, with reminders of things which need to be done, and descriptions of these.

The application is due to Lars Vogel (<http://www.vogella.com>).

A database example

The screenshot shows an IDE with two windows. The left window, titled 'Package Explorer', displays the project structure for 'de.vogella.android.todos.database'. It contains two main classes: 'TodoDatabaseHelper.java' and 'TodoDbAdapter.java'. 'TodoDatabaseHelper.java' includes constants for 'DATABASE_CREATE', 'DATABASE_NAME', and 'DATABASE_VERSION', and methods for 'onCreate(SQLiteDatabase) : void' and 'onUpgrade(SQLiteDatabase, int, int) : void'. 'TodoDbAdapter.java' includes constants for 'DATABASE_TABLE', 'KEY_CATEGORY', 'KEY_DESCRIPTION', 'KEY_ROWID', and 'KEY_SUMMARY', and methods for 'context', 'database', 'dbHelper', 'TodoDbAdapter(Context)', 'close() : void', 'createContentValues(String, String, String) : ContentValues', 'createTodo(String, String, String) : long', 'deleteTodo(long) : boolean', 'fetchAllTodos() : Cursor', 'fetchTodo(long) : Cursor', 'open() : TodoDbAdapter', and 'updateTodo(long, String, String, String) : boolean'. The right window, titled 'TodosOverview.java', shows the code for 'TodoDbAdapter'. It includes imports for 'android.content.ContentValues', 'android.content.Context', 'android.database.Cursor', 'android.database.SQLException', and 'android.database.sqlite.SQLiteDatabase'. The class 'TodoDbAdapter' has fields for 'DATABASE_TABLE', 'KEY_CATEGORY', 'KEY_DESCRIPTION', 'KEY_ROWID', 'KEY_SUMMARY', 'context', and 'database'. It has a constructor 'TodoDbAdapter(Context context)' that initializes 'this.context = context'. It has a method 'open()' that returns a new 'TodoDbAdapter' instance with a new 'dbHelper' and 'database'. It has a method 'close()' that calls 'dbHelper.close()' and 'database.close()'.

```
Package Explorer
```

- de.vogella.android.todos.database
 - TodoDatabaseHelper.java
 - TodoDatabaseHelper
 - DATABASE_CREATE
 - DATABASE_NAME
 - DATABASE_VERSION
 - TodoDatabaseHelper(Context)
 - onCreate(SQLiteDatabase) : void
 - onUpgrade(SQLiteDatabase, int, int) : void
 - TodoDbAdapter.java
 - TodoDbAdapter
 - DATABASE_TABLE
 - KEY_CATEGORY
 - KEY_DESCRIPTION
 - KEY_ROWID
 - KEY_SUMMARY
 - context
 - database
 - dbHelper
 - TodoDbAdapter(Context)
 - close() : void
 - createContentValues(String, String, String) : ContentValues
 - createTodo(String, String, String) : long
 - deleteTodo(long) : boolean
 - fetchAllTodos() : Cursor
 - fetchTodo(long) : Cursor
 - open() : TodoDbAdapter
 - updateTodo(long, String, String, String) : boolean

```
TodosOverview.java
```

```
package de.vogella.android.todos;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;

public class TodoDbAdapter {

    // Database fields
    public static final String DATABASE_TABLE = "todos";
    public static final String KEY_CATEGORY = "category";
    public static final String KEY_DESCRIPTION = "description";
    private static final String KEY_ROWID = "rowid";
    private Context context;
    private SQLiteDatabase database;
    private TodoDatabaseHelper dbHelper;

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

    public TodoDbAdapter open() {
        dbHelper = new TodoDatabaseHelper(context);
        database = dbHelper.getWritableDatabase();
        return this;
    }

    public void close() {
        dbHelper.close();
        database.close();
    }
}
```

ToDoDatabaseAdapter

```
TodosOverview.java  TodoDbAdapter.java  TodoDatabaseHelper.j  5
package de.vogella.android.todos.database;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;

public class TodoDbAdapter {

    // Database fields
    public static final String KEY_ROWID = "_id";
    public static final String KEY_CATEGORY = "category";
    public static final String KEY_SUMMARY = "summary";
    public static final String KEY_DESCRIPTION = "description";
    private static final String DATABASE_TABLE = "todo";
    private Context context;
    private SQLiteDatabase database;
    private TodoDatabaseHelper dbHelper;

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

    public TodoDbAdapter open() throws SQLException {
        dbHelper = new TodoDatabaseHelper(context);
        database = dbHelper.getWritableDatabase();
        return this;
    }
}
```

TodoDatabaseHelper (onCreate())

```
package de.vogella.android.todos.database;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class TodoDatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "applicationdata";

    private static final int DATABASE_VERSION = 1;

    // Database creation sql statement
    private static final String DATABASE_CREATE = "create table todo (_id integer primary key autoincrement, "
        + "category text not null, summary text not null, description text not null);";

    public TodoDatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    // Method is called during creation of the database
    @Override
    public void onCreate(SQLiteDatabase database) {
        database.execSQL(DATABASE_CREATE);
    }

    // Method is called during an upgrade of the database, e.g. if you increase
    // the database version
    @Override
    public void onUpgrade(SQLiteDatabase database, int oldVersion,
        int newVersion) {
        Log.w(TodoDatabaseHelper.class.getName(),
            "Upgrading database from version " + oldVersion + " to "
            + newVersion + ", which will destroy all old data");
        database.execSQL("DROP TABLE IF EXISTS todo");
        onCreate(database);
    }
}
```

**Table "todo" has columns
_id, category, summary,
and description**

ToDoDatabaseHelper (onUpgrade())

```
import android.os.Bundle;

public class ToDoDatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "applicationdata";

    private static final int DATABASE_VERSION = 1;

    // Database creation sql statement
    private static final String DATABASE_CREATE = "create table todo (_id integer primary key autoincrement, "
        + "category text not null, summary text not null, description text not null)";

    public ToDoDatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    // Method is called during creation of the database
    @Override
    public void onCreate(SQLiteDatabase database) {
        database.execSQL(DATABASE_CREATE);
    }

    // Method is called during an upgrade of the database, e.g. if you increase
    // the database version
    @Override
    public void onUpgrade(SQLiteDatabase database, int oldVersion,
        int newVersion) {
        Log.w(ToDoDatabaseHelper.class.getName(),
            "Upgrading database from version " + oldVersion + " to "
            + newVersion + ", which will destroy all old data");
        database.execSQL("DROP TABLE IF EXISTS todo");
        onCreate(database);
    }
}
```

ToDoDatabaseAdapter (open(), close())

```
package de.vogella.android.todos.database;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;

public class ToDoDbAdapter {

    // Database fields
    public static final String KEY_ROWID = "_id";
    public static final String KEY_CATEGORY = "category";
    public static final String KEY_SUMMARY = "summary";
    public static final String KEY_DESCRIPTION = "description";
    private static final String DATABASE_TABLE = "todo";
    private Context context;
    private SQLiteDatabase database;
    private ToDoDatabaseHelper dbHelper;

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

    public ToDoDbAdapter open() throws SQLException {
        dbHelper = new ToDoDatabaseHelper(context);
        database = dbHelper.getWritableDatabase();
        return this;
    }

    public void close() {
        dbHelper.close();
    }

    /**
     * Create a new todo If the todo is successfully created return the new
     * rowId for that note, otherwise return a -1 to indicate failure
     */
}
```

The create, update, and delete methods

```
/**
 * Create a new todo If the todo is successfully created return the new
 * rowId for that note, otherwise return a -1 to indicate failure.
 */
public long createTodo(String category, String summary, String description) {
    ContentValues initialValues = createContentValues(category, summary,
        description);

    return database.insert(DATABASE_TABLE, null, initialValues);
}

/**
 * Update the todo
 */
public boolean updateTodo(long rowId, String category, String summary,
    String description) {
    ContentValues updateValues = createContentValues(category, summary,
        description);

    return database.update(DATABASE_TABLE, updateValues, KEY_ROWID + "="
        + rowId, null) > 0;
}

/**
 * Deletes todo
 */
public boolean deleteTodo(long rowId) {
    return database.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId, null) > 0;
}

/**
 * Return a Cursor over the list of all todo in the database
 *
 * @return Cursor over all notes
 */
```

The insert() method

The screenshot shows an IDE window with a Javadoc viewer at the top and a code editor below. The Javadoc viewer displays the signature and details for the `insert()` method in `android.database.sqlite.SQLiteDatabase`. The code editor shows the implementation of `TodoDbAdapter` in `de.vogella.android.todos.database`.

Javadoc for `insert()`:

```
long android.database.sqlite.SQLiteDatabase.insert(String table, String nullColumnHack, ContentValues values)
```

Signature: `public long insert (String table, String nullColumnHack, ContentValues values)`
Since: [API Level 1](#)

Convenience method for inserting a row into the database.

Parameters

table	the table to insert the row into
nullColumnHack	optional; may be null. SQL doesn't allow inserting a completely empty row without naming at least one column name. If your provided values is empty, no column names are known and an empty row can't be inserted. If not set to null, the nullColumnHack parameter provides the name of nullable column name to explicitly insert a NULL into in the case where your values is empty.
values	this map contains the initial column values for the row. The keys should be the column names and the values the column values

Returns

- the row ID of the newly inserted row, or -1 if an error occurred

Code Editor:

```
TodosOverview.java | TodoDbAdapter.java | TodoDatabaseHelper.j | todo_edit.xml | »4
```

TODOs ▸ src ▸ de.vogella.android.todos.database ▸ TodoDbAdapter ▸ createTodo(String, String, String) : long

```
public static final String KEY_DESCRIPTION = "description";
private static final String DATABASE_TABLE = "todo";
private Context context;
private SQLiteDatabase database;
private TodoDatabaseHelper dbHelper;

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

public TodoDbAdapter open() throws SQLException {
    dbHelper = new TodoDatabaseHelper(context);
```

The update() method

The screenshot shows an IDE window with two panes. The top pane displays the Javadoc for the `update()` method of `android.database.sqlite.SQLiteDatabase`. The bottom pane shows the source code of `TodoDbAdapter.java`, with the `updateTodo()` method highlighted. The code in the bottom pane uses the `update()` method to update a row in the database.

Javadoc for `update()`:

```
int android.database.sqlite.SQLiteDatabase.update(String table, ContentValues values, String whereClause, String[] whereArgs)
```

public int update ([String](#) table, [ContentValues](#) values, [String](#) whereClause, [String\[\]](#) whereArgs)
Since: [API Level 1](#)

Convenience method for updating rows in the database.

Parameters

- table** the table to update in
- values** a map from column names to new column values. null is a valid value that will be translated to NULL.
- whereClause** the optional WHERE clause to apply when updating. Passing null will update all rows.

Returns

- the number of rows affected

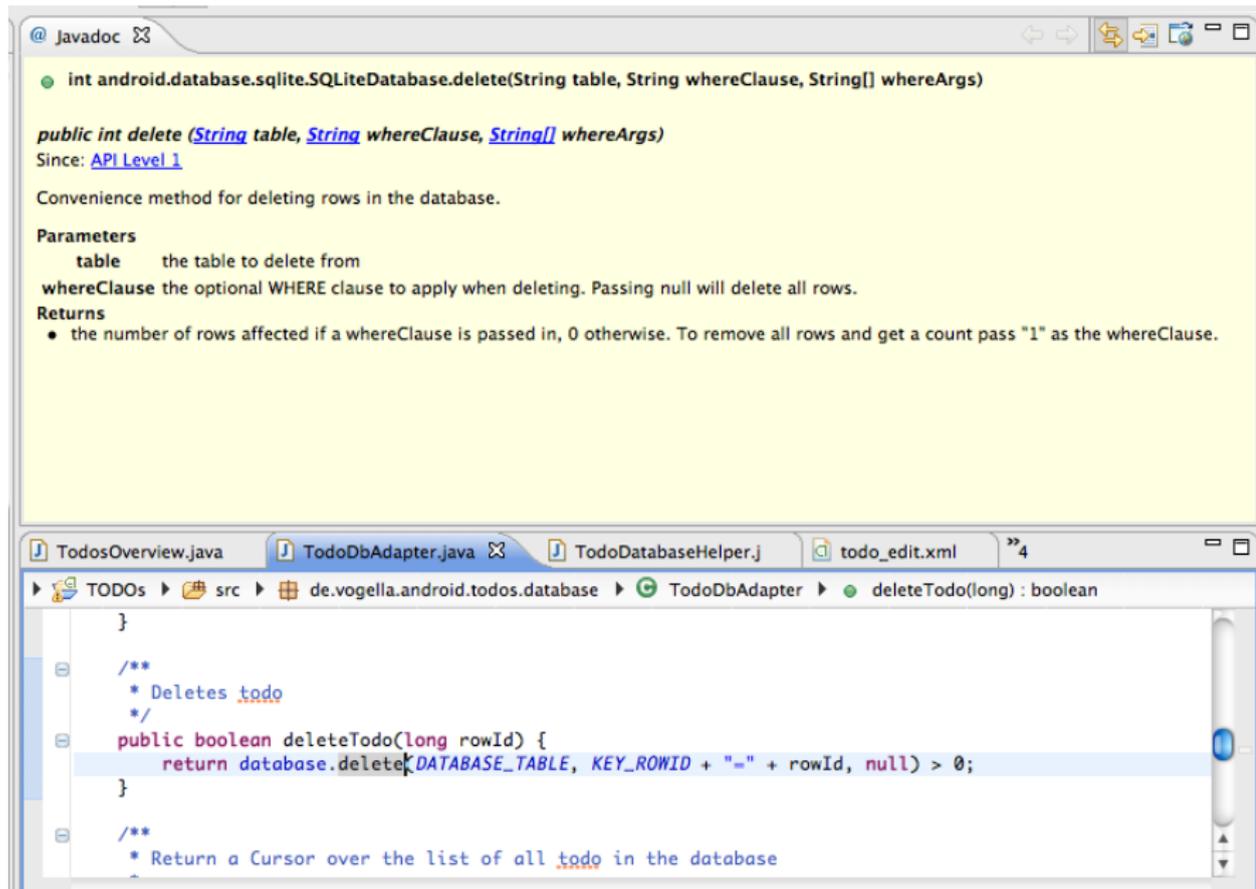
Code in `TodoDbAdapter.java`:

```
return database.update(DATABASE_TABLE, updateValues, KEY_ROWID + "=" + rowId, null) > 0;
```

Comments:

```
/**  
 * Deletes todo  
 */
```

The delete() method



The screenshot shows an IDE window with two panes. The top pane displays the Javadoc for the `delete()` method in `android.database.sqlite.SQLiteDatabase`. The bottom pane shows the implementation of `deleteTodo(long)` in `TodoDbAdapter.java`.

Javadoc for `delete()`:

```
int android.database.sqlite.SQLiteDatabase.delete(String table, String whereClause, String[] whereArgs)
```

Signature: `public int delete (String table, String whereClause, String[] whereArgs)`

Since: [API Level 1](#)

Convenience method for deleting rows in the database.

Parameters

- table** the table to delete from
- whereClause** the optional WHERE clause to apply when deleting. Passing null will delete all rows.

Returns

- the number of rows affected if a whereClause is passed in, 0 otherwise. To remove all rows and get a count pass "1" as the whereClause.

Code Implementation:

```
public boolean deleteTodo(long rowId) {
    return database.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId, null) > 0;
}
```

Fetch data

```
/**
 * Return a Cursor over the list of all todo in the database
 *
 * @return Cursor over all notes
 */
public Cursor fetchAllTodos() {
    return database.query(DATABASE_TABLE, new String[] { KEY_ROWID,
        KEY_CATEGORY, KEY_SUMMARY, KEY_DESCRIPTION }, null, null, null,
        null, null);
}

/**
 * Return a Cursor positioned at the defined todo
 */
public Cursor fetchTodo(long rowId) throws SQLException {
    Cursor mCursor = database.query(true, DATABASE_TABLE, new String[] {
        KEY_ROWID, KEY_CATEGORY, KEY_SUMMARY, KEY_DESCRIPTION },
        KEY_ROWID + "=" + rowId, null, null, null, null, null);
    if (mCursor != null) {
        mCursor.moveToFirst();
    }
    return mCursor;
}

private ContentValues createContentValues(String category, String summary,
    String description) {
    ContentValues values = new ContentValues();
    values.put(KEY_CATEGORY, category);
    values.put(KEY_SUMMARY, summary);
    values.put(KEY_DESCRIPTION, description);
    return values;
}
```

Create content values

```
        KEY_CATEGORY, KEY_SUMMARY, KEY_DESCRIPTION }, null, null, null,
        null, null);
    }

    /**
     * Return a Cursor positioned at the defined todo
     */
    public Cursor fetchTodo(long rowId) throws SQLException {
        Cursor mCursor = database.query(true, DATABASE_TABLE, new String[] {
            KEY_ROWID, KEY_CATEGORY, KEY_SUMMARY, KEY_DESCRIPTION },
            KEY_ROWID + "=" + rowId, null, null, null, null, null);
        if (mCursor != null) {
            mCursor.moveToFirst();
        }
        return mCursor;
    }

    private ContentValues createContentValues(String category, String summary,
        String description) {
        ContentValues values = new ContentValues();
        values.put(KEY_CATEGORY, category);
        values.put(KEY_SUMMARY, summary);
        values.put(KEY_DESCRIPTION, description);
        return values;
    }
}
```

blems @ Javadoc Declaration Console

de.vogella.android.todos.database.TODOAdapter

Resources

```
⊕ /* AUTO-GENERATED FILE. DO NOT MODIFY.⊞  
  
package de.vogella.android.todos;  
  
public final class R {  
    ⊖ public static final class array {  
        public static final int priorities=0x7f040000;  
    }  
    ⊖ public static final class attr {  
    }  
    ⊖ public static final class color {  
        public static final int black=0x7f060001;  
        public static final int listcolor=0x7f060000;  
    }  
    ⊖ public static final class drawable {  
        public static final int icon=0x7f020000;  
        public static final int reminder=0x7f020001;  
        public static final int todo=0x7f020002;  
    }  
    ⊖ public static final class id {  
        public static final int LinearLayout01=0x7f080001;  
        public static final int TextView01=0x7f080006;  
        public static final int category=0x7f080000;  
        public static final int icon=0x7f080005;  
        public static final int insert=0x7f080008;  
        public static final int label=0x7f080007;  
        public static final int todo_edit_button=0x7f080004;  
        public static final int todo_edit_description=0x7f080003;  
        public static final int todo_edit_summary=0x7f080002;  
    }  
}
```

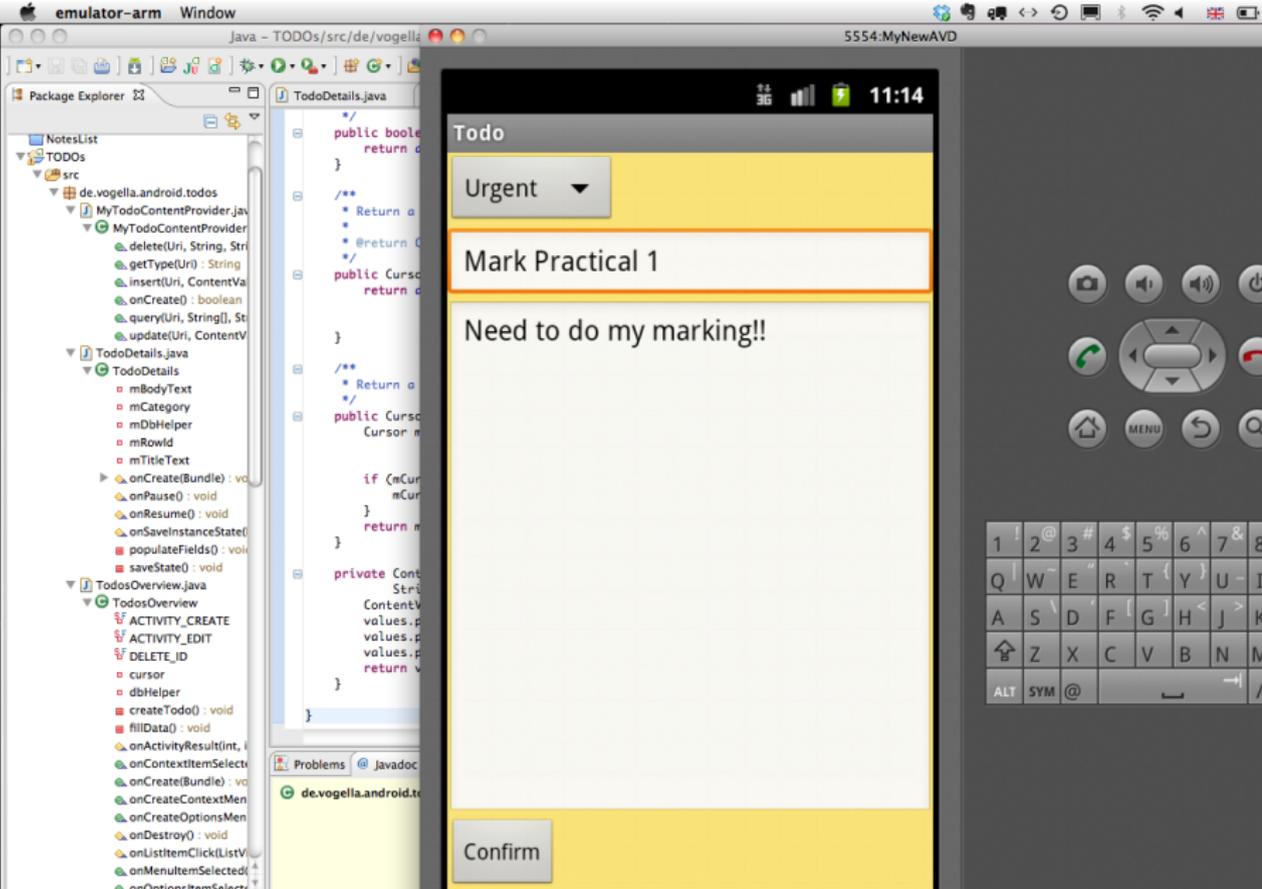
Running the application

When we run the application we are able to create a TODO note, and set its category to Urgent or Reminder.

Running the TODOs application

The screenshot shows the Android Studio interface with an emulator running. The emulator screen displays a yellow background with the text "Mark Practical 1" and a small icon of a document with a checkmark. The IDE background shows the Package Explorer on the left, the Java code for `TodoDetails.java` in the center, and the Android Studio toolbar on the right. The Package Explorer shows the project structure for `de.vogella.android.todos`, including `MyTodoContentProvider.java`, `TodoDetails.java`, and `TodosOverview.java`. The Java code in the center shows the `public boolean` method and the `public Cursor` method. The Android Studio toolbar on the right includes icons for camera, volume, power, home, menu, and search, as well as a virtual keyboard.

Editing a TODO item



Setting category to "Urgent"

The image shows an Android emulator window titled "emulator-arm Window" with the Java IDE interface. The Package Explorer on the left shows the project structure for "de.vogella.android.todos". The main editor displays the code for "TodoDetails.java". The emulator screen shows a "Todo" app with a dropdown menu set to "Urgent". The text "Mark Practical 1" and "Need to do my marking!!" is visible. A dialog box is open with "Urgent" and "Reminder" options, each with a radio button. A "Confirm" button is at the bottom.

```
public boolean ...
return ...

/**
 * Return a ...
 * @Return C ...
 */
public Cursor ...
return ...

/**
 * Return a ...
 * @Return a ...
 */
public Cursor ...
return ...

private Cont ...
String ...
ContentV ...
values.p ...
values.p ...
return v ...
}
```

TODOs application code

We look at the application code to see how the Java code interacts with the XML layout and see how the database state is managed.

TodoDetails (imports)

```
package de.vogella.android.todos;

import android.app.Activity;
import android.database.Cursor;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import de.vogella.android.todos.database.TODOAdapter;

public class TodoDetails extends Activity {
    private EditText mTitleText;
    private EditText mBodyText;
    private Long mRowId;
    private TODOAdapter mDbHelper;
    private Spinner mCategory;

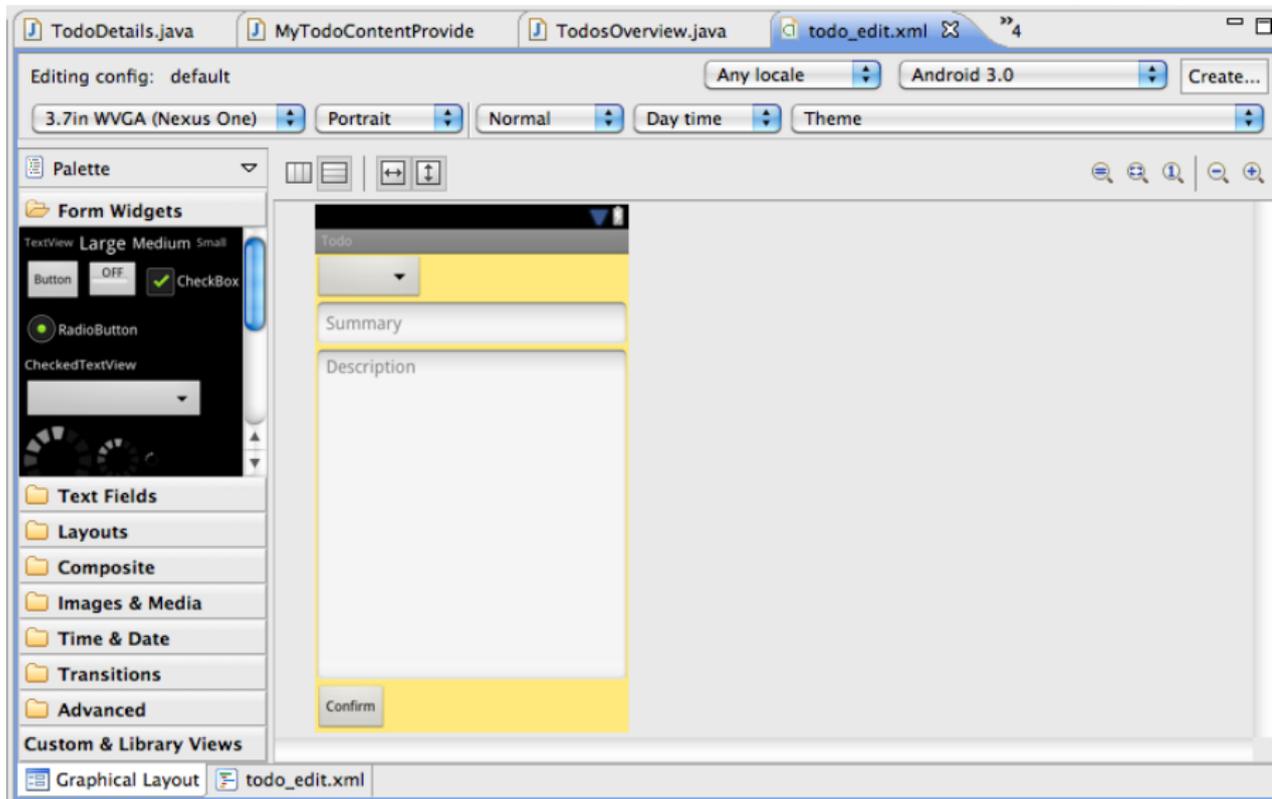
    @Override
    protected void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        mDbHelper = new TODOAdapter(this);
        mDbHelper.open();
        setContentView(R.layout.todo_edit);
        mCategory = (Spinner) findViewById(R.id.category);
        mTitleText = (EditText) findViewById(R.id.todo_edit_summary);
    }
}
```

TodoDetails (onCreate)

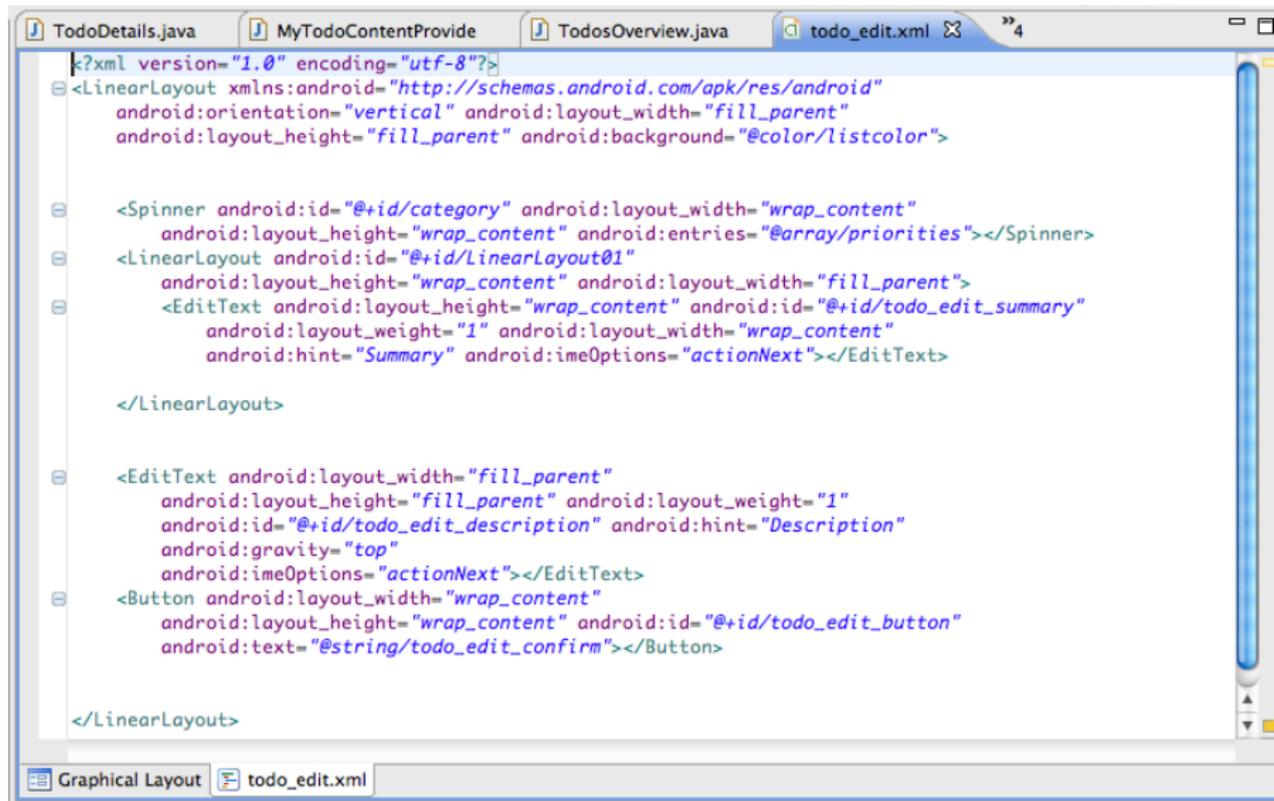
```
@Override
protected void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    mDbHelper = new TodoDbAdapter(this);
    mDbHelper.open();
    setContentView(R.layout.todo_edit);
    mCategory = (Spinner) findViewById(R.id.category);
    mTitleText = (EditText) findViewById(R.id.todo_edit_summary);
    mBodyText = (EditText) findViewById(R.id.todo_edit_description);

    Button confirmButton = (Button) findViewById(R.id.todo_edit_button);
    mRowId = null;
    Bundle extras = getIntent().getExtras();
    mRowId = (bundle == null) ? null : (Long) bundle
        .getSerializable(TodoDbAdapter.KEY_ROWID);
    if (extras != null) {
        mRowId = extras.getLong(TodoDbAdapter.KEY_ROWID);
    }
    populateFields();
    confirmButton.setOnClickListener(new View.OnClickListener() {
        public void onClick(View view) {
            setResult(RESULT_OK);
            finish();
        }
    });
}
```

Graphical layout of todo_edit.xml



Text of todo_edit.xml



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent" android:background="@color/listcolor">

    <Spinner android:id="@+id/category" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:entries="@array/priorities"></Spinner>
    <LinearLayout android:id="@+id/LinearLayout01"
        android:layout_height="wrap_content" android:layout_width="fill_parent">
        <EditText android:layout_height="wrap_content" android:id="@+id/todo_edit_summary"
            android:layout_weight="1" android:layout_width="wrap_content"
            android:hint="Summary" android:imeOptions="actionNext"></EditText>

    </LinearLayout>

    <EditText android:layout_width="fill_parent"
        android:layout_height="fill_parent" android:layout_weight="1"
        android:id="@+id/todo_edit_description" android:hint="Description"
        android:gravity="top"
        android:imeOptions="actionNext"></EditText>
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:id="@+id/todo_edit_button"
        android:text="@string/todo_edit_confirm"></Button>

</LinearLayout>
```

Outline of todo_edit.xml

The image shows a screenshot of an IDE with two windows. The left window displays the XML code for `todo_edit.xml`, and the right window displays the Outline view of the XML structure.

```
roid.com/apk/res/android"
ut_width="fill_parent"
:background="@color/listcolor">

:layout_width="wrap_content"
droid:entries="@array/priorities"></Spinner>
01"
droid:layout_width="fill_parent">
ontent" android:id="@+id/todo_edit_summary"
ayout_width="wrap_content"
ptions="actionNext"></EditText>

"
roid:layout_weight="1"
  android:hint="Description"

ext>

droid:id="@+id/todo_edit_button"
"></Button>
```

The Outline view on the right shows the following structure:

- LinearLayout
 - category (Spinner)
 - LinearLayout01
 - todo_edit_summary (EditText)
 - todo_edit_description (EditText)
 - todo_edit_button - "Confirm"

TodoDetails (populateFields)

```
private void populateFields() {
    if (mRowId != null) {
        Cursor todo = mDbHelper.fetchTodo(mRowId);
        startManagingCursor(todo);
        String category = todo.getString(todo
            .getColumnIndexOrThrow(TodoDbAdapter.KEY_CATEGORY));

        for (int i = 0; i < mCategory.getCount(); i++) {

            String s = (String) mCategory.getItemAtPosition(i);
            Log.e(null, s + " " + category);
            if (s.equalsIgnoreCase(category)) {
                mCategory.setSelection(i);
            }
        }

        mTitleText.setText(todo.getString(todo
            .getColumnIndexOrThrow(TodoDbAdapter.KEY_SUMMARY));
        mBodyText.setText(todo.getString(todo
            .getColumnIndexOrThrow(TodoDbAdapter.KEY_DESCRIPTION));
    }
}
```

Save state, onPause, onResume

```
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    saveState();
    outState.putSerializable(TodoDbAdapter.KEY_ROWID, mRowId);
}

@Override
protected void onPause() {
    super.onPause();
    saveState();
}

@Override
protected void onResume() {
    super.onResume();
    populateFields();
}
```

Save state

```
private void saveState() {
    String category = (String) mCategory.getSelectedItem();
    String summary = mTitleText.getText().toString();
    String description = mBodyText.getText().toString();

    if (mRowId == null) {
        long id = mDbHelper.createTodo(category, summary, description);
        if (id > 0) {
            mRowId = id;
        }
    } else {
        mDbHelper.updateTodo(mRowId, category, summary, description);
    }
}
```

TodosOverview (onCreate)

```
package de.vogella.android.todos;

import android.app.ListActivity;

public class TodosOverview extends ListActivity {
    private TodoDbAdapter dbHelper;
    private static final int ACTIVITY_CREATE = 0;
    private static final int ACTIVITY_EDIT = 1;
    private static final int DELETE_ID = Menu.FIRST + 1;
    private Cursor cursor;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.todo_list);
        this.getListView().setDividerHeight(2);
        dbHelper = new TodoDbAdapter(this);
        dbHelper.open();
        fillData();
        registerForContextMenu(getListView());
    }

    // Create the menu based on the XML definition
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
```

Options menu, item selected

```
// Create the menu based on the XML definition
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.listmenu, menu);
    return true;
}

// Reaction to the menu selection
@Override
public boolean onOptionsItemSelected(int featureId, MenuItem item) {
    switch (item.getItemId()) {
        case R.id.insert:
            createTodo();
            return true;
    }
    return super.onOptionsItemSelected(featureId, item);
}
```

Options item selected

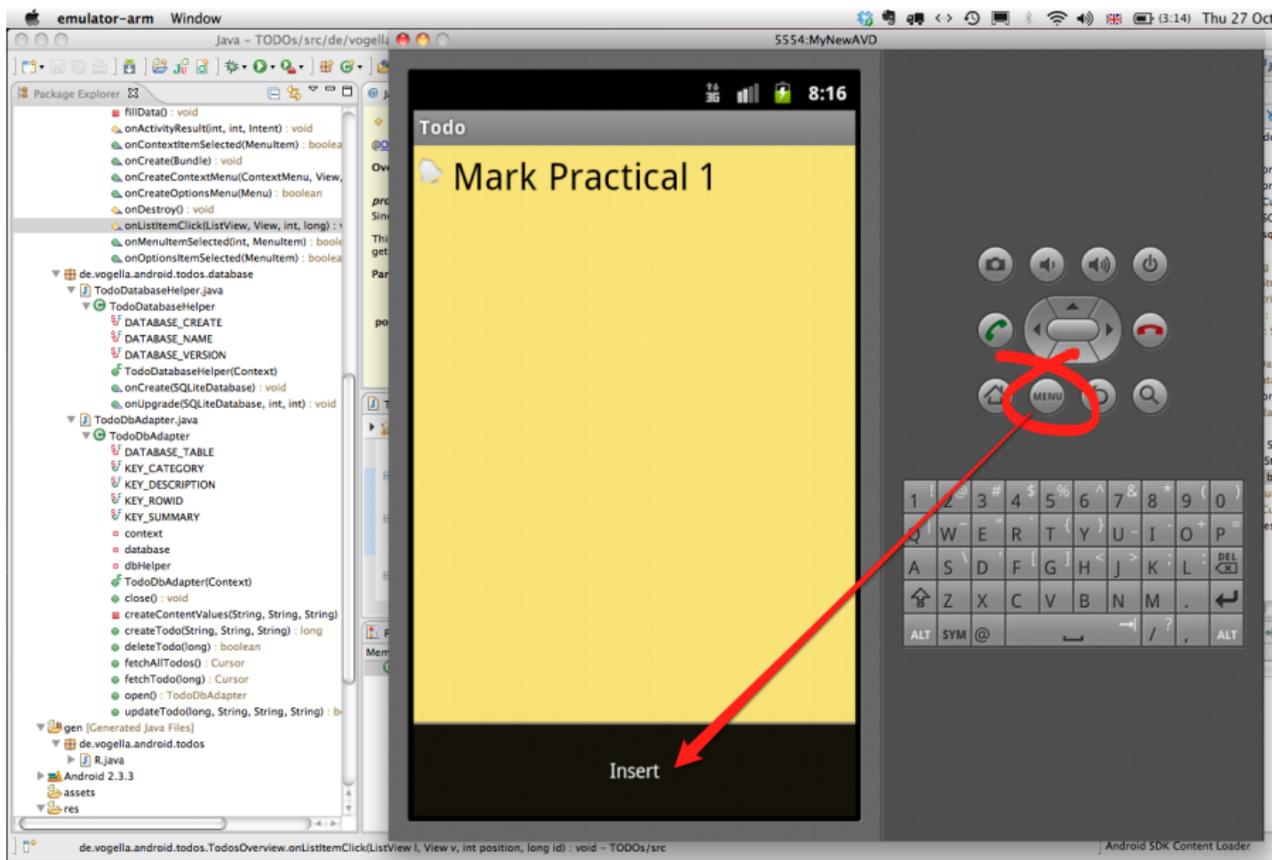
```
// Create the menu based on the XML definition
public boolean onCreateOptionsMenu(Menu menu) {}

// Reaction to the menu selection
public boolean onOptionsItemSelected(int featureId, MenuItem item) {}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.insert:
            createTodo();
            return true;
    }
    return super.onOptionsItemSelected(item);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
```

Accessing the insert menu



Inspecting the database

During development we might have bugs in our code which cause problems with the database. In this case we would like to be able to see the content of the database and check that it is as we expect. We can use the Dalvik Debug Monitor Server (DDMS) to do this and we can also inspect the database using other tools for processing SQLite databases.

Dalvik Debug Monitor Server (DDMS)

The screenshot displays the Eclipse IDE with the DDMS (Dalvik Debug Monitor Server) interface. The top toolbar includes icons for Devices, Threads, Heap, Allocation Tracker, and File Explorer. The main window is divided into three panes:

- Devices:** A table listing virtual devices. The table has columns for Name, PID, and MyNewAVD.
- File Explorer:** A directory tree showing the application's data directory. The tree is expanded to show the 'data' directory, which contains sub-directories for various system and application-specific data.
- LogCat:** A log window showing system messages. The log entries include timestamps, PID, tag, and message text.

The 'Devices' pane shows the following table:

Name	PID	MyNewAVD
emulator-55 Online		
system_proc:61	8600	
com.android:127	8602	
jp.co.omonos:119	8603	
com.android:130	8601	
com.android:142	8604	
com.android:162	8605	
android_proc:180	8606	
com.android:200	8607	
com.android:216	8610	
com.android:233	8616	
com.android:243	8620	
com.android:245	8624	
android_proc:261	8627	
com.android:272	8629	
com.android:292	8631	

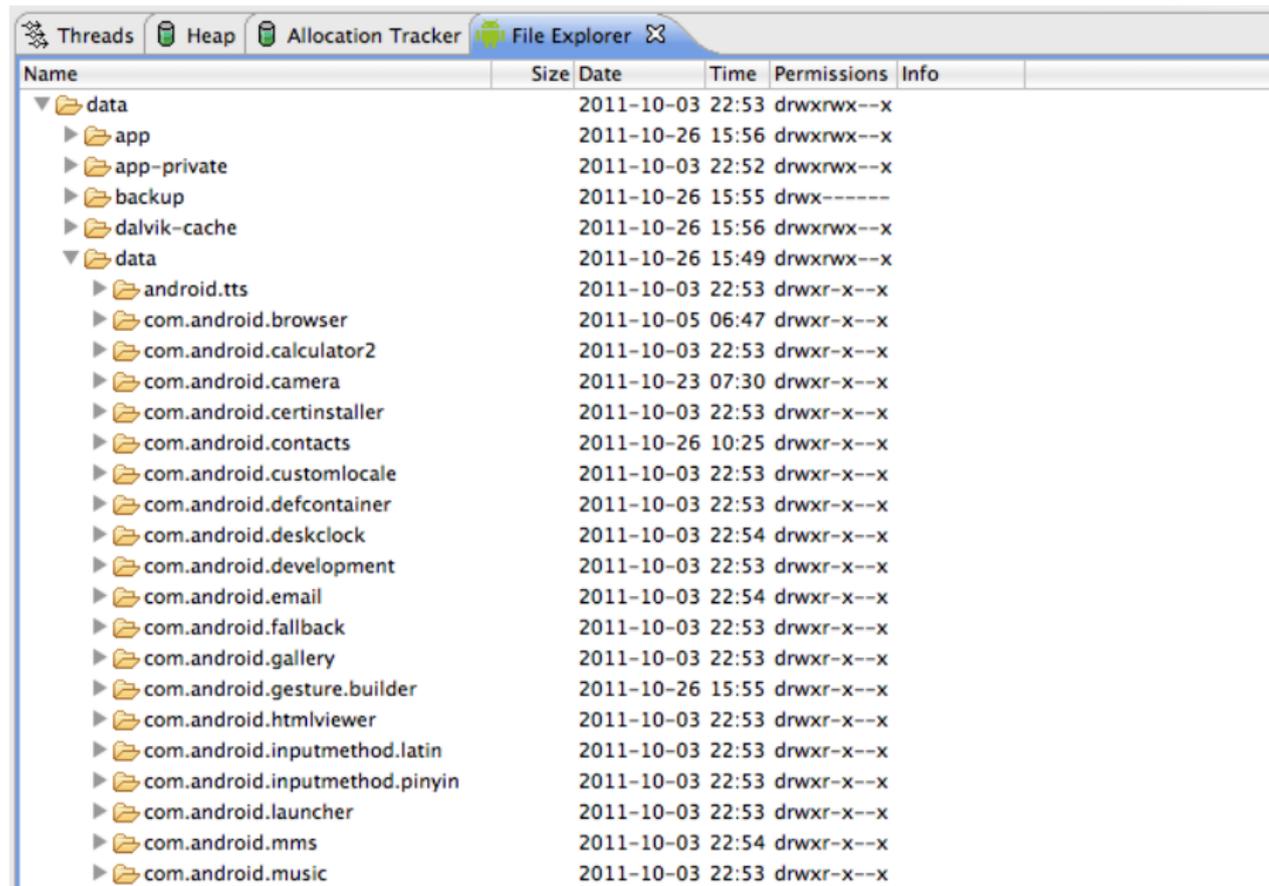
The 'File Explorer' pane shows the following directory tree:

- data
 - app
 - app-private
 - backup
 - dalvik-cache
 - data
 - android.tts
 - com.android.browser
 - com.android.calculator2
 - com.android.camera
 - com.android.certinstaller
 - com.android.contacts
 - com.android.customlocale
 - com.android.defcontainer
 - com.android.deskclock
 - com.android.development
 - com.android.email
 - com.android.fallback
 - com.android.gallery
 - com.android.gesture.builder
 - com.android.htmlviewer
 - com.android.inputmethod.latin
 - com.android.inputmethod.pinyin
 - com.android.launcher
 - com.android.mms
 - com.android.music
 - com.android.netspeed
 - com.android.packageinstaller
 - com.android.phone
 - com.android.protips
 - com.android.providers.applications
 - com.android.providers.contacts
 - com.android.providers.downloads
 - com.android.providers.downloads.ui

The 'LogCat' pane shows the following log entries:

```
Time          pid  tag      Message
10-06 15:35:31 W 30     2proc0  Preloaded drawable resource #01080093 (res/drawable-hdpi/eye_def_app_icon.png) that varies with configuration!
10-06 15:35:31 W 30     2proc0  Preloaded drawable resource #01080092 (res/drawable-hdpi/arrow_down_click.png) that varies with configuration!
10-06 15:35:31 W 30     2proc0  Preloaded drawable resource #01080091 (res/drawable-hdpi/arrow_down_click.png) that varies with configuration!
```

File explorer in DDMS



The screenshot shows the DDMS File Explorer window with the following tabs: Threads, Heap, Allocation Tracker, and File Explorer. The File Explorer tab is active, displaying a table of files and folders. The table has columns for Name, Size, Date, Time, Permissions, and Info. The files are organized into a tree structure, with a 'data' folder expanded to show its contents.

Name	Size	Date	Time	Permissions	Info
data		2011-10-03	22:53	drwxrwx--x	
app		2011-10-26	15:56	drwxrwx--x	
app-private		2011-10-03	22:52	drwxrwx--x	
backup		2011-10-26	15:55	drwx-----	
dalvik-cache		2011-10-26	15:56	drwxrwx--x	
data		2011-10-26	15:49	drwxrwx--x	
android.tts		2011-10-03	22:53	drwxr-x--x	
com.android.browser		2011-10-05	06:47	drwxr-x--x	
com.android.calculator2		2011-10-03	22:53	drwxr-x--x	
com.android.camera		2011-10-23	07:30	drwxr-x--x	
com.android.certinstaller		2011-10-03	22:53	drwxr-x--x	
com.android.contacts		2011-10-26	10:25	drwxr-x--x	
com.android.customloclace		2011-10-03	22:53	drwxr-x--x	
com.android.defcontainer		2011-10-03	22:53	drwxr-x--x	
com.android.deskclock		2011-10-03	22:54	drwxr-x--x	
com.android.development		2011-10-03	22:53	drwxr-x--x	
com.android.email		2011-10-03	22:54	drwxr-x--x	
com.android.fallback		2011-10-03	22:53	drwxr-x--x	
com.android.gallery		2011-10-03	22:53	drwxr-x--x	
com.android.gesture.builder		2011-10-26	15:55	drwxr-x--x	
com.android.htmlviewer		2011-10-03	22:53	drwxr-x--x	
com.android.inputmethod.latin		2011-10-03	22:53	drwxr-x--x	
com.android.inputmethod.pinyin		2011-10-03	22:53	drwxr-x--x	
com.android.launcher		2011-10-03	22:53	drwxr-x--x	
com.android.mms		2011-10-03	22:54	drwxr-x--x	
com.android.music		2011-10-03	22:53	drwxr-x--x	

Devices

Name	State	MyNewAVD
emulator-55	Online	MyNewAVD
system_proc61		8600
com.android.127		8602
jp.co.omrons.119		8603
com.android.130		8601
com.android.142		8604
com.android.162		8605
android.proc.180		8606
com.android.200		8607
com.android.216		8610

File Explorer

Name	Size	Date	Time	Permissions
com.example.android.contactmanag		2011-10-26	15:55	drwxr-x--x
com.example.android.livecubes		2011-10-26	15:55	drwxr-x--x
com.example.android.notepad		2011-10-26	15:55	drwxr-x--x
com.example.android.searchabledic		2011-10-26	15:55	drwxr-x--x
com.example.android.softkeyboard		2011-10-26	15:55	drwxr-x--x
com.example.android.wiktionary		2011-10-26	15:55	drwxr-x--x
com.mad.mick.forum		2011-10-26	15:55	drwxr-x--x
com.svox.pico		2011-10-03	22:53	drwxr-x--x
de.vogella.android.todos		2011-10-26	15:56	drwxr-x--x
databases		2011-10-26	15:57	drwxrwx--x
applicationdata	5120	2011-10-26	15:57	-rw-rw----
lib		2011-10-26	15:56	drwxr-xr-x
jp.co.omronsoft.openwnn		2011-10-26	15:50	drwxr-x--x
dontpanic		2011-10-03	22:52	drwxr-x--x
local		2011-10-03	22:52	drwxrwx--x
lost+found		2011-10-03	22:52	drwxrwx--x
misc		2011-10-03	22:52	drwxrwx--t
property		2011-10-03	22:53	drwx-----
secure		2011-10-03	22:52	drwx-----
system		2011-10-26	15:56	drwxrwxr-x
mnt		2011-10-26	15:55	drwxrwxr-x
system		2011-02-03	22:51	drwxr-xr-x

Emulator Control

Telephony Status

Voice: home Speed: Full

Data: home Latency: None

Telephony Actions

Incoming number:

Voice

SMS

LogCat

```
Android
[2011-10-26 15:55:03 - TODOS] Refreshing resource folders.
[2011-10-26 15:55:03 - TODOS] Starting incremental Pre Compiler: Checking resource changes.
[2011-10-26 15:55:03 - TODOS] Attribute minSdkVersion (9) is lower than the project target API level (10)
[2011-10-26 15:55:03 - TODOS] Nothing to pre compile!
[2011-10-26 15:55:03 - TODOS] -----
[2011-10-26 15:55:03 - TODOS] Android Launch!
[2011-10-26 15:55:03 - TODOS] adb is running normally.
```

Pulling a file from the device

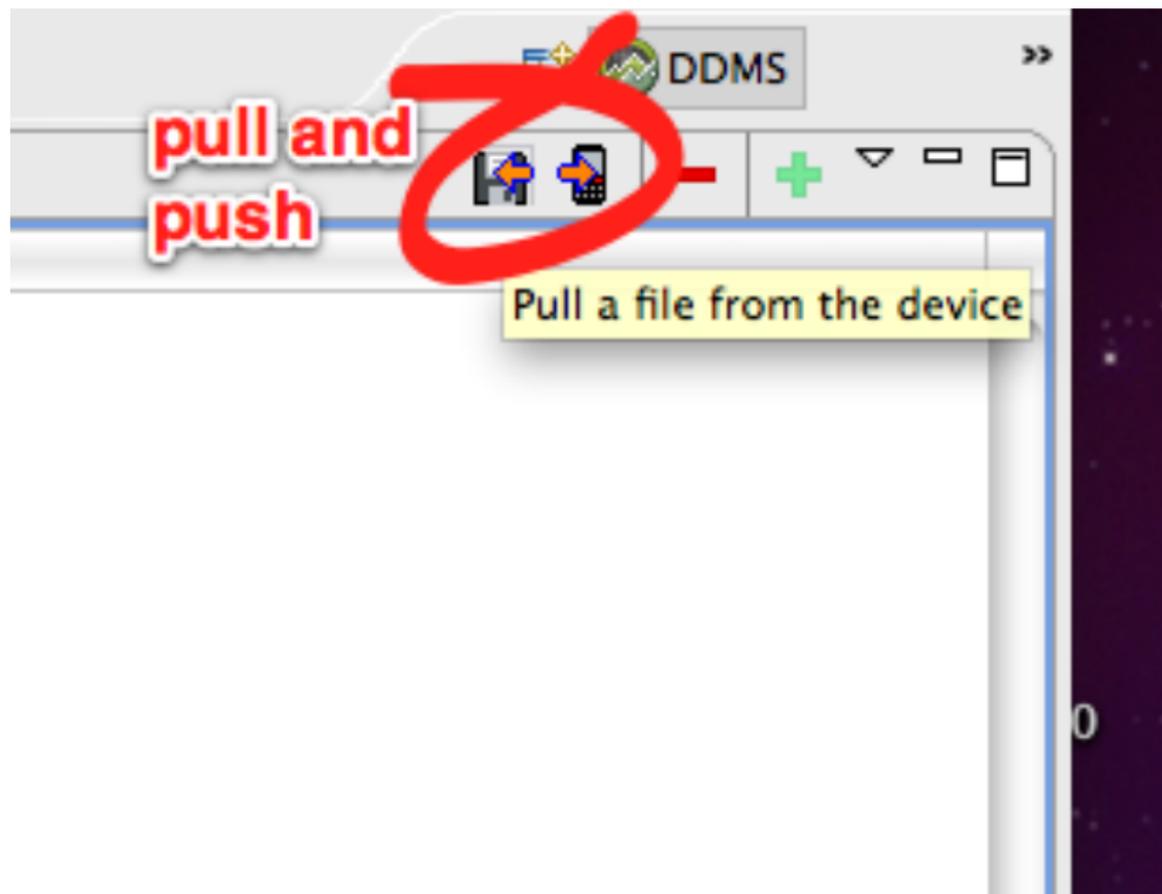
The screenshot shows a File Explorer window with a table of files. The table has columns for Name, Size, Date, Time, Permissions, and Info. A yellow tooltip is visible over the table with the text "Pull a file from the device".

Name	Size	Date	Time	Permissions	Info
com.example.android.contactmanag		2011-10-26	15:55	drwxr-x--x	
com.example.android.livecubes		2011-10-26	15:55	drwxr-x--x	
com.example.android.notepad		2011-10-26	15:55	drwxr-x--x	
com.example.android.searchabledic		2011-10-26	15:55	drwxr-x--x	
com.example.android.softkeyboard		2011-10-26	15:55	drwxr-x--x	
com.example.android.wiktionary		2011-10-26	15:55	drwxr-x--x	
com.mad.mick.forum		2011-10-26	15:55	drwxr-x--x	
com.svox.pico		2011-10-03	22:53	drwxr-x--x	
de.vogella.android.todos		2011-10-26	15:56	drwxr-x--x	
databases		2011-10-26	15:57	drwxrwx--x	
applicationdata	5120	2011-10-26	15:57	-rw-rw----	
lib		2011-10-26	15:56	drwxr-xr-x	
jp.co.omronsoft.openwnn		2011-10-26	15:50	drwxr-x--x	
ntpanic		2011-10-03	22:52	drwxr-x---	
al		2011-10-03	22:52	drwxrwx--x	
it+found		2011-10-03	22:52	drwxrwx---	
sc		2011-10-03	22:52	drwxrwx--t	
operty		2011-10-03	22:53	drwx-----	
cure		2011-10-03	22:52	drwx-----	
stem		2011-10-26	15:56	drwxrwxr-x	
		2011-10-26	15:55	drwxrwxr-x	
m		2011-02-03	22:51	drwxr-xr-x	

ing resource changes.
in the project target API level (10)

```
isOverview activity launch  
lator with compatible AVD 'MyNewAVD'  
vice 'MyNewAVD'
```

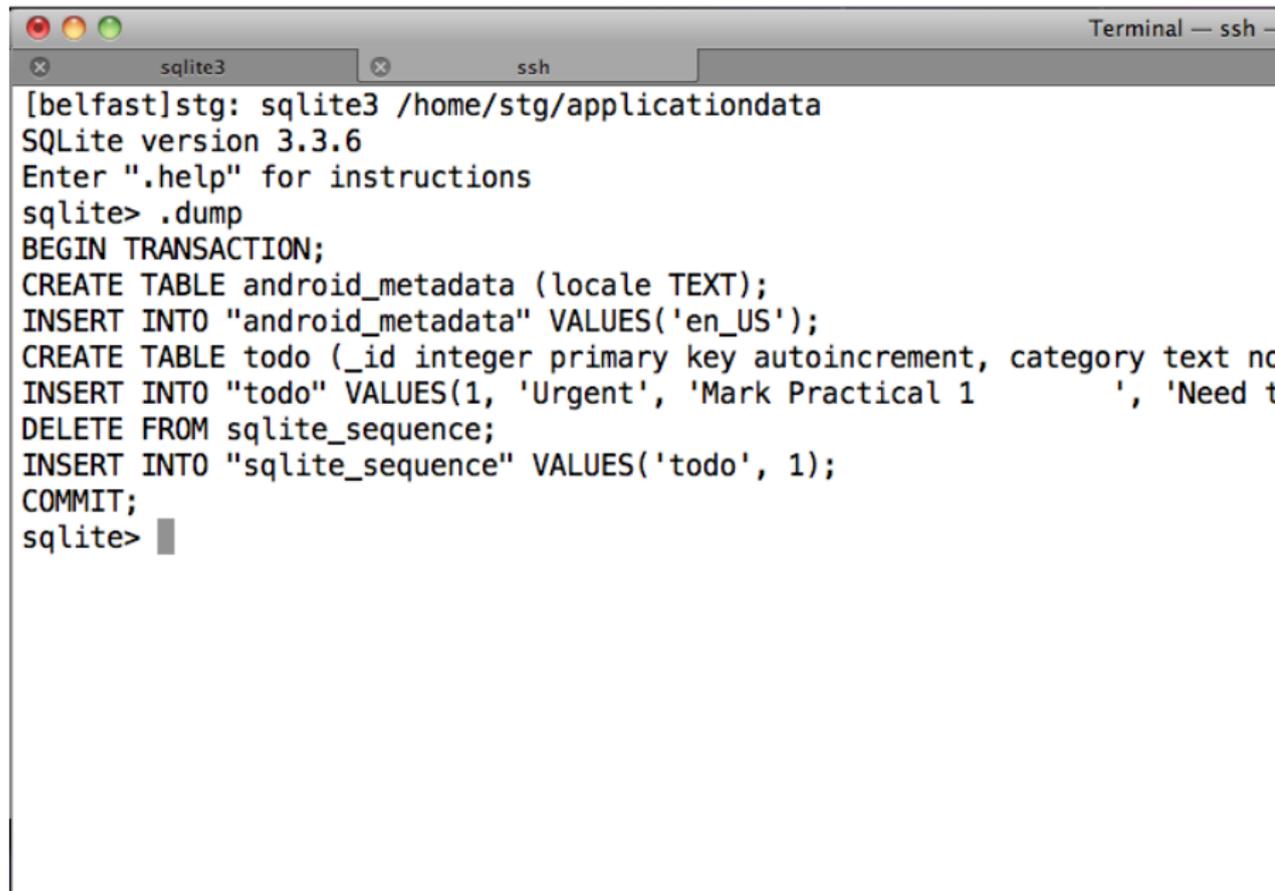
Pulling a file from the device



Inspecting the file with sqlite3

```
Terminal — sqlite3
sqlite3
dhcp-91-122:~ stephengilmore$ sqlite3 Desktop/applicationdata
SQLite version 3.6.12
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .dump
BEGIN TRANSACTION;
CREATE TABLE android_metadata (locale TEXT);
INSERT INTO "android_metadata" VALUES('en_US');
CREATE TABLE todo (_id integer primary key autoincrement, category text no
INSERT INTO "todo" VALUES(1,'Urgent','Mark Practical 1','Need to
DELETE FROM sqlite_sequence;
INSERT INTO "sqlite_sequence" VALUES('todo',1);
COMMIT;
sqlite> █
```

Inspecting the file with sqlite3 on DiCE

A terminal window titled "Terminal - ssh" with two tabs: "sqlite3" and "ssh". The terminal shows the execution of the sqlite3 command on a file. The output displays the SQLite version (3.3.6) and the results of the .dump command, which includes SQL statements for creating and inserting data into tables named android_metadata, todo, and sqlite_sequence. The terminal prompt is currently at sqlite> with a cursor.

```
[belfast]stg: sqlite3 /home/stg/applicationdata
SQLite version 3.3.6
Enter ".help" for instructions
sqlite> .dump
BEGIN TRANSACTION;
CREATE TABLE android_metadata (locale TEXT);
INSERT INTO "android_metadata" VALUES('en_US');
CREATE TABLE todo (_id integer primary key autoincrement, category text no
INSERT INTO "todo" VALUES(1, 'Urgent', 'Mark Practical 1', 'Need t
DELETE FROM sqlite_sequence;
INSERT INTO "sqlite_sequence" VALUES('todo', 1);
COMMIT;
sqlite> █
```

Other tools for inspecting the database

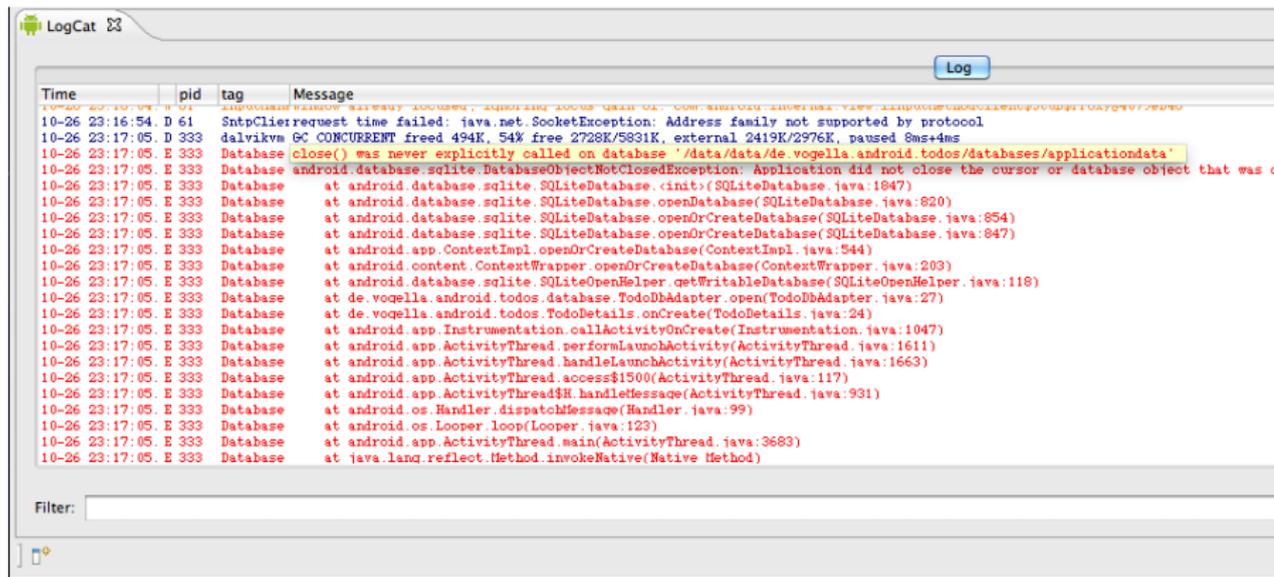
Numerous browsers exist for SQLite databases.

- ▶ The **CellObject SQLite & XML Browser** is an Eclipse plug-in tool built on top of Dalvik Debug Monitor Server (DDMS), intended for Android application developers. It is available from <http://cellobject.net/>
- ▶ **SQLite Manager** is an add-on for Firefox which allows users to browse SQLite databases using the Firefox browser. It is available from <https://addons.mozilla.org/en-US/firefox/addon/sqlite-manager/>
- ▶ **SQLite Database Browser** allows users to create, design and edit database files compatible with SQLite using a familiar spreadsheet-like interface. It is available from <http://sqlitebrowser.sourceforge.net/>

Using LogCat

As usual, the effect of Java exceptions and other problems will appear in the LogCat view.

Debugs and errors displayed in LogCat



The screenshot shows the LogCat window in Android Studio. The window title is "LogCat" with a search icon. A "Log" button is visible in the top right corner. The log messages are displayed in a table with columns for Time, pid, tag, and Message. The messages include a warning about a failed SntpClient request, a debug message about GC memory usage, and several error messages from the Database class, including a prominent one stating "Database close() was never explicitly called on database".

Time	pid	tag	Message
10-26 23:16:54	D 61		SntpClient request time failed: java.net.SocketException: Address family not supported by protocol
10-26 23:17:05	D 333		dalvikvm GC CONCURRENT freed 494K, 54% free 2728K/5831K, external 2419K/2976K, paused 8ms+4ms
10-26 23:17:05	E 333	Database	close() was never explicitly called on database '/data/data/de.vogella.android.todos/databases/applicationdata'
10-26 23:17:05	E 333	Database	android.database.sqlite.DatabaseObjectNotClosedException: Application did not close the cursor or database object that was...
10-26 23:17:05	E 333	Database	at android.database.sqlite.SQLiteDatabase.<init>(SQLiteDatabase.java:1847)
10-26 23:17:05	E 333	Database	at android.database.sqlite.SQLiteDatabase.openDatabase(SQLiteDatabase.java:820)
10-26 23:17:05	E 333	Database	at android.database.sqlite.SQLiteDatabase.openOrCreateDatabase(SQLiteDatabase.java:854)
10-26 23:17:05	E 333	Database	at android.database.sqlite.SQLiteDatabase.openOrCreateDatabase(SQLiteDatabase.java:847)
10-26 23:17:05	E 333	Database	at android.app.ContextImpl.openOrCreateDatabase(ContextImpl.java:544)
10-26 23:17:05	E 333	Database	at android.content.ContextWrapper.openOrCreateDatabase(ContextWrapper.java:203)
10-26 23:17:05	E 333	Database	at android.database.sqlite.SQLiteOpenHelper.getWritableDatabase(SQLiteOpenHelper.java:118)
10-26 23:17:05	E 333	Database	at de.vogella.android.todos.database.TODOOpenHelper.open(TODOOpenHelper.java:27)
10-26 23:17:05	E 333	Database	at de.vogella.android.todos.TODODetails.onCreate(TODODetails.java:24)
10-26 23:17:05	E 333	Database	at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1047)
10-26 23:17:05	E 333	Database	at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:1611)
10-26 23:17:05	E 333	Database	at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:1663)
10-26 23:17:05	E 333	Database	at android.app.ActivityThread.access\$1500(ActivityThread.java:117)
10-26 23:17:05	E 333	Database	at android.app.ActivityThread\$H.handleMessage(ActivityThread.java:931)
10-26 23:17:05	E 333	Database	at android.os.Handler.dispatchMessage(Handler.java:99)
10-26 23:17:05	E 333	Database	at android.os.Looper.loop(Looper.java:123)
10-26 23:17:05	E 333	Database	at android.app.ActivityThread.main(ActivityThread.java:3683)
10-26 23:17:05	E 333	Database	at java.lang.reflect.Method.invokeNative(Native Method)

Filter:

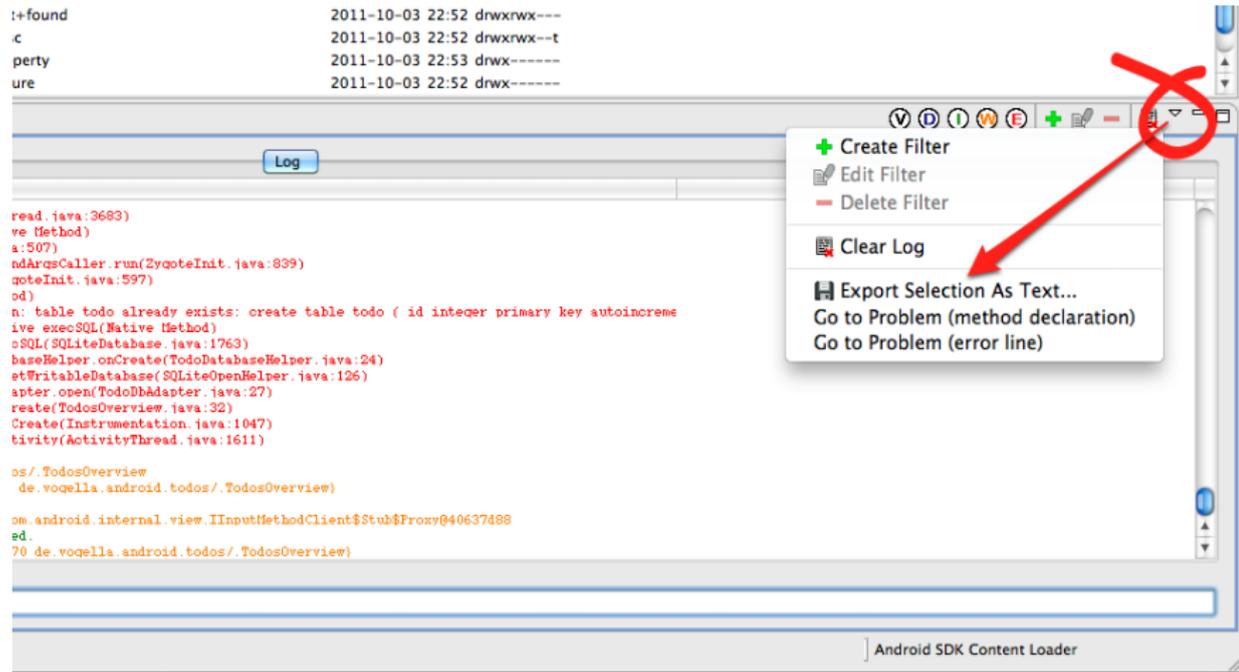
Can filter messages displayed in LogCat

The screenshot shows the LogCat window with the following data:

Time	pid	tag	Message
10-26 15:57:19	E 333	Database	close() was never explicitly called on database '/data/data/de.vogella.android.todos/databases/applicationdata'
10-26 23:17:05	E 333	Database	close() was never explicitly called on database '/data/data/de.vogella.android.todos/databases/applicationdata'
10-26 23:17:05	E 333	Database	close() was never explicitly called on database '/data/data/de.vogella.android.todos/databases/applicationdata'
10-26 23:17:05	E 333	Database	close() was never explicitly called on database '/data/data/de.vogella.android.todos/databases/applicationdata'
10-27 15:07:19	E 333	Database	close() was never explicitly called on database '/data/data/de.vogella.android.todos/databases/applicationdata'

Filter: applicationdata

Can use view menu to export messages



The screenshot shows the Android Studio interface with the Logcat window open. The log contains several lines of text, including timestamps and stack traces. A context menu is open over the log, with a red circle around the 'Export Selection As Text...' option and a red arrow pointing to it. The menu also includes options for 'Create Filter', 'Edit Filter', 'Delete Filter', 'Clear Log', 'Go to Problem (method declaration)', and 'Go to Problem (error line)'. The 'Log' button is visible above the log text.

```
!+found      2011-10-03 22:52 drwxrwx---
c            2011-10-03 22:52 drwxrwx--t
perty       2011-10-03 22:53 drwx-----
ure         2011-10-03 22:52 drwx-----
```

Log

```
read.java:3683)
ve(Method)
a:507)
ndArgsCaller.run(2yqoteInit.java:839)
qoteInit.java:597)
od)
n: table todo already exists: create table todo ( id integer primary key autoincrem
ive execSQL(Native Method)
oSQL(SQLiteDatabase.java:1763)
baseHelper.onCreate(TodoDatabaseHelper.java:24)
stWritableDatabase(SQLiteOpenHelper.java:126)
apter.open(TodoDbAdapter.java:27)
reate(TodosOverview.java:32)
Create(Instrumentation.java:1047)
tivity(ActivityThread.java:1611)
```

os/.TodosOverview
de.voqella.android.todos/.TodosOverview)

om.android.internal.view.IInputMethodClient\$Stub\$Proxy@940637488
ed.
70 de.voqella.android.todos/.TodosOverview)

Android SDK Content Loader