Software Engineering Large Practical

Working with databases in Android

Stephen Gilmore

School of Informatics, University of Edinburgh

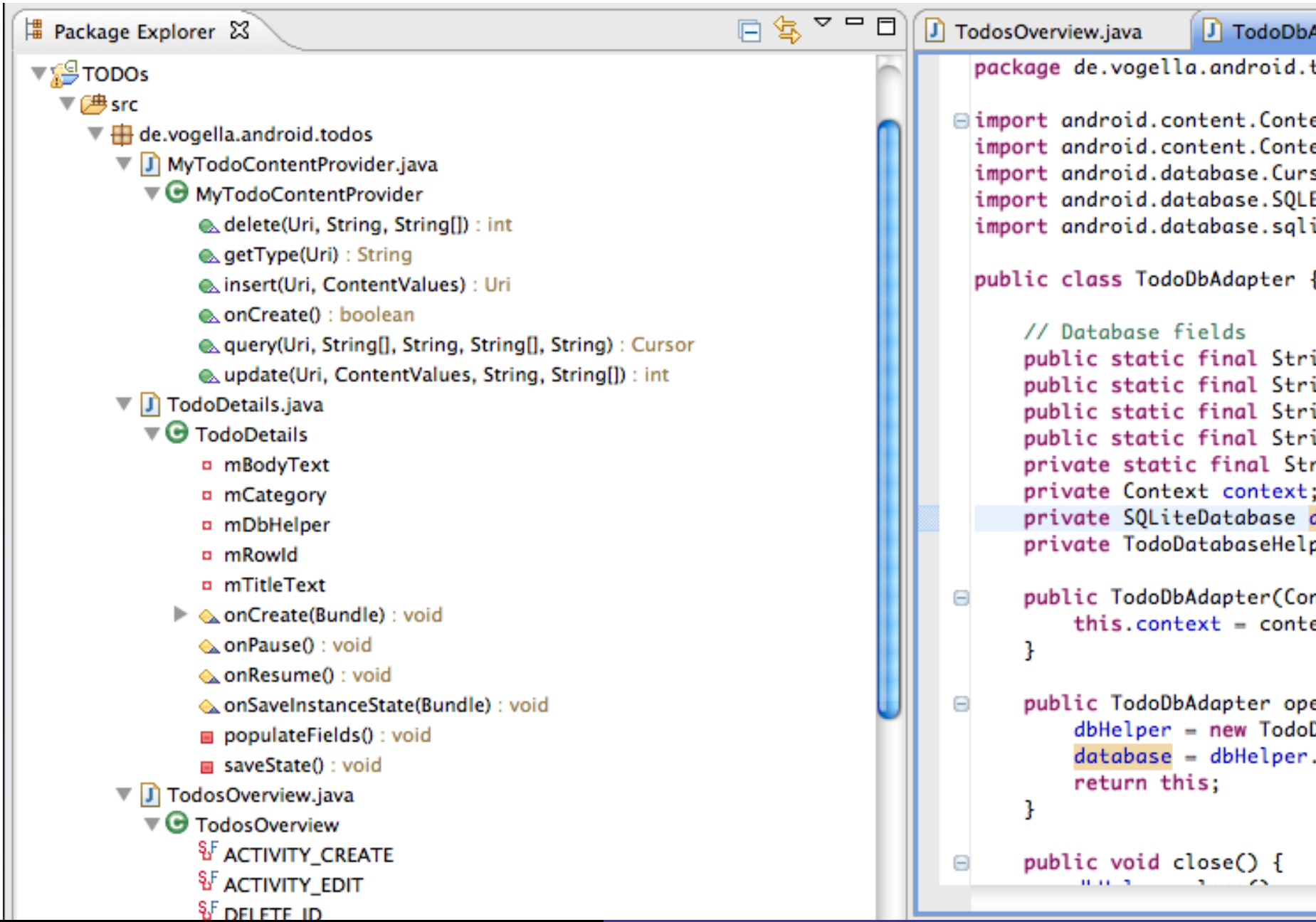
October 16th, 2013

A database example

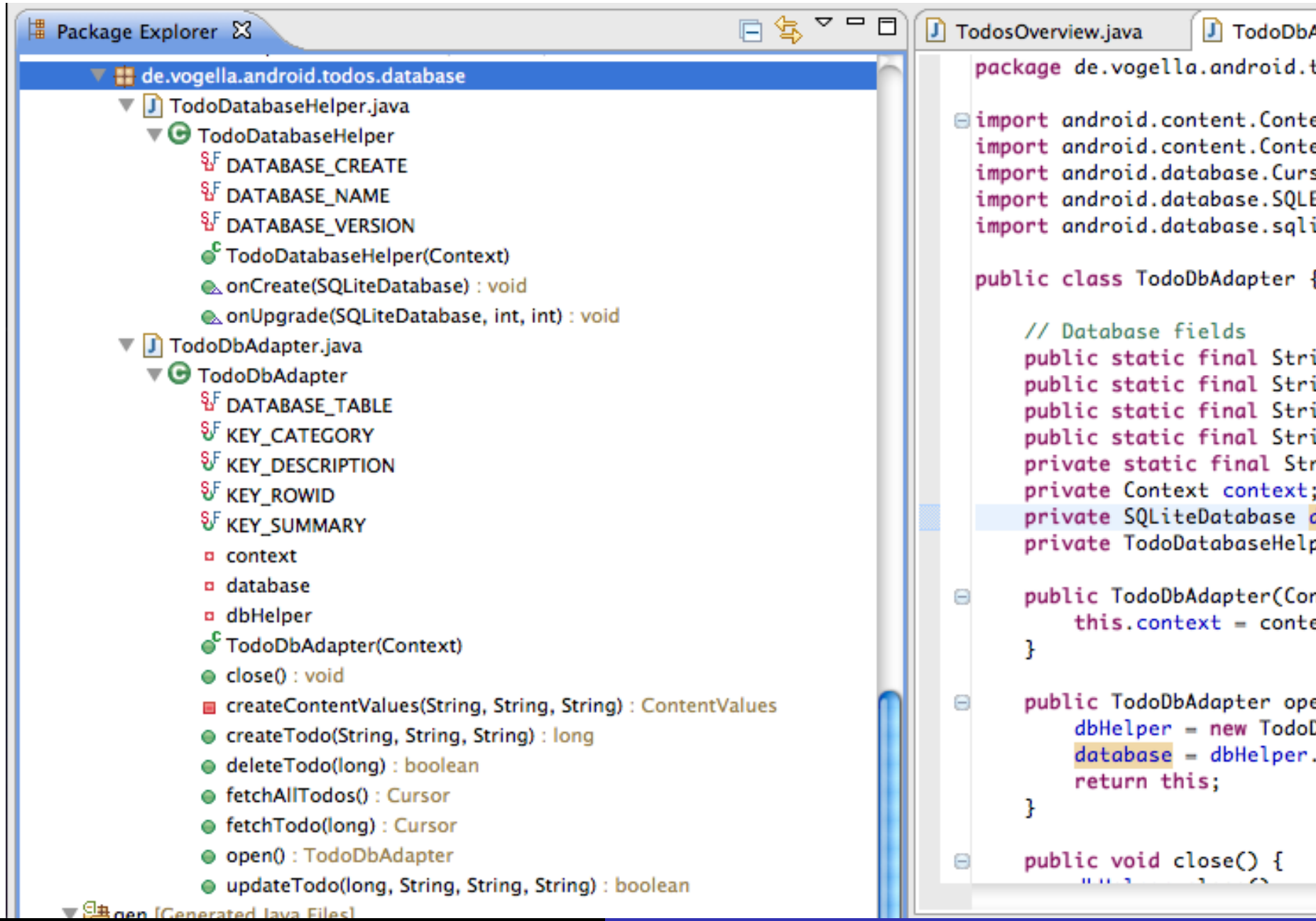
In this lecture we will look at an example Android application which creates a database of TODO notes, with reminders of things which need to be done, and descriptions of these.

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

Lars Vogel example: TODOs



A database example



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())

```
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);
    }
}
```

oblems | @ Javadoc | Declaration | Console

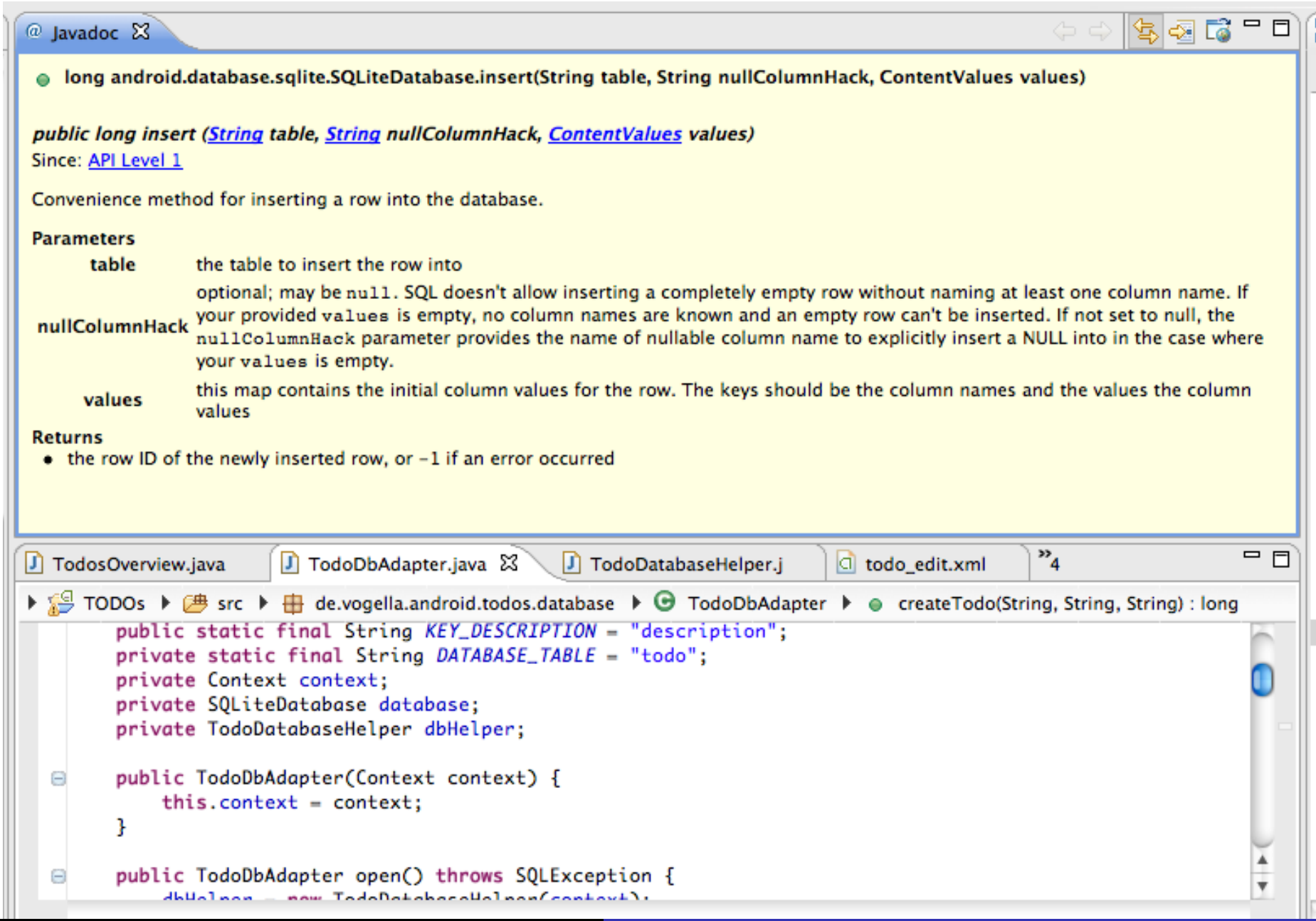
TodoDatabaseAdapter (open(), close())

```
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.
```

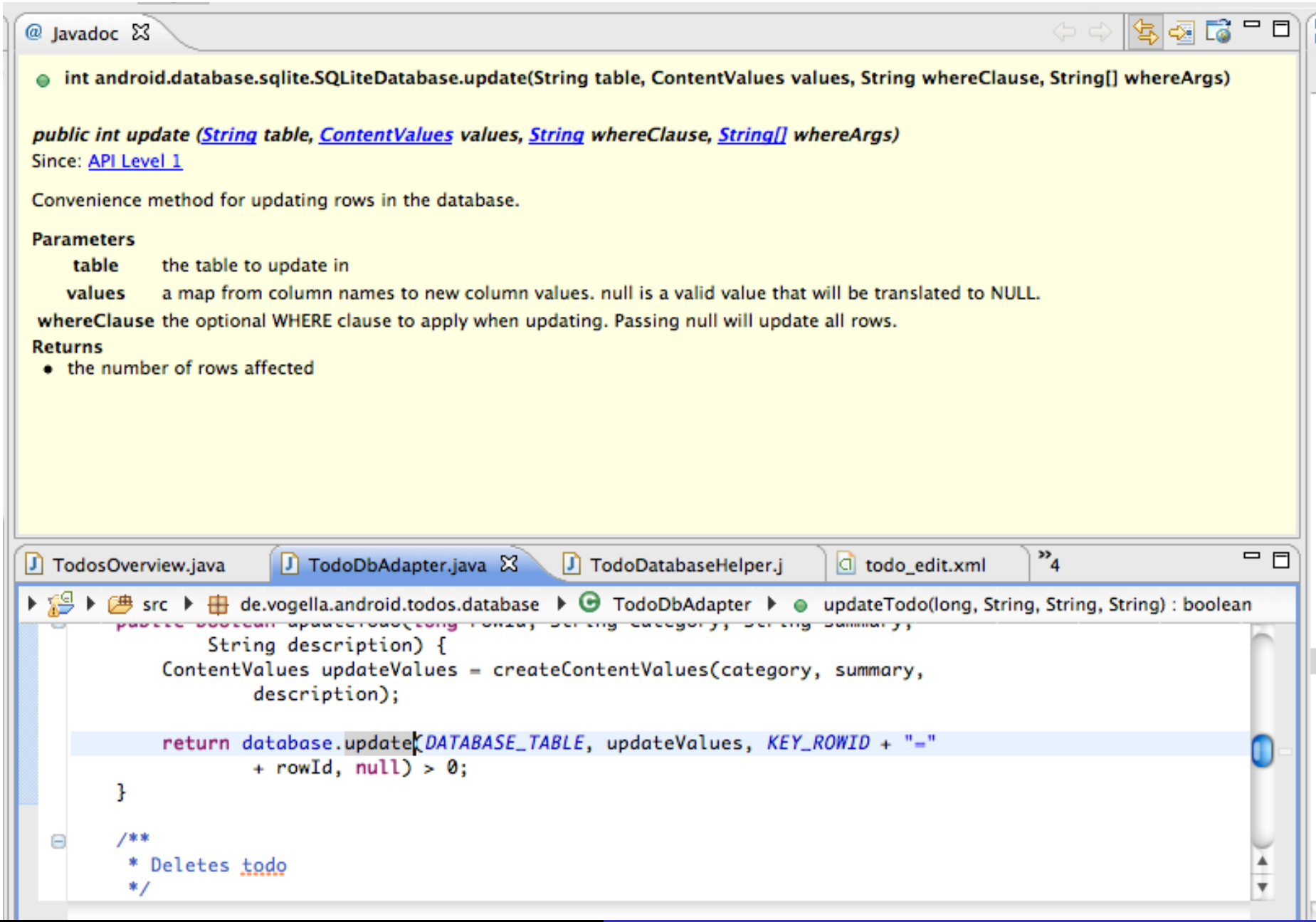
The create, update, and delete methods

```
/**
 * 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
```

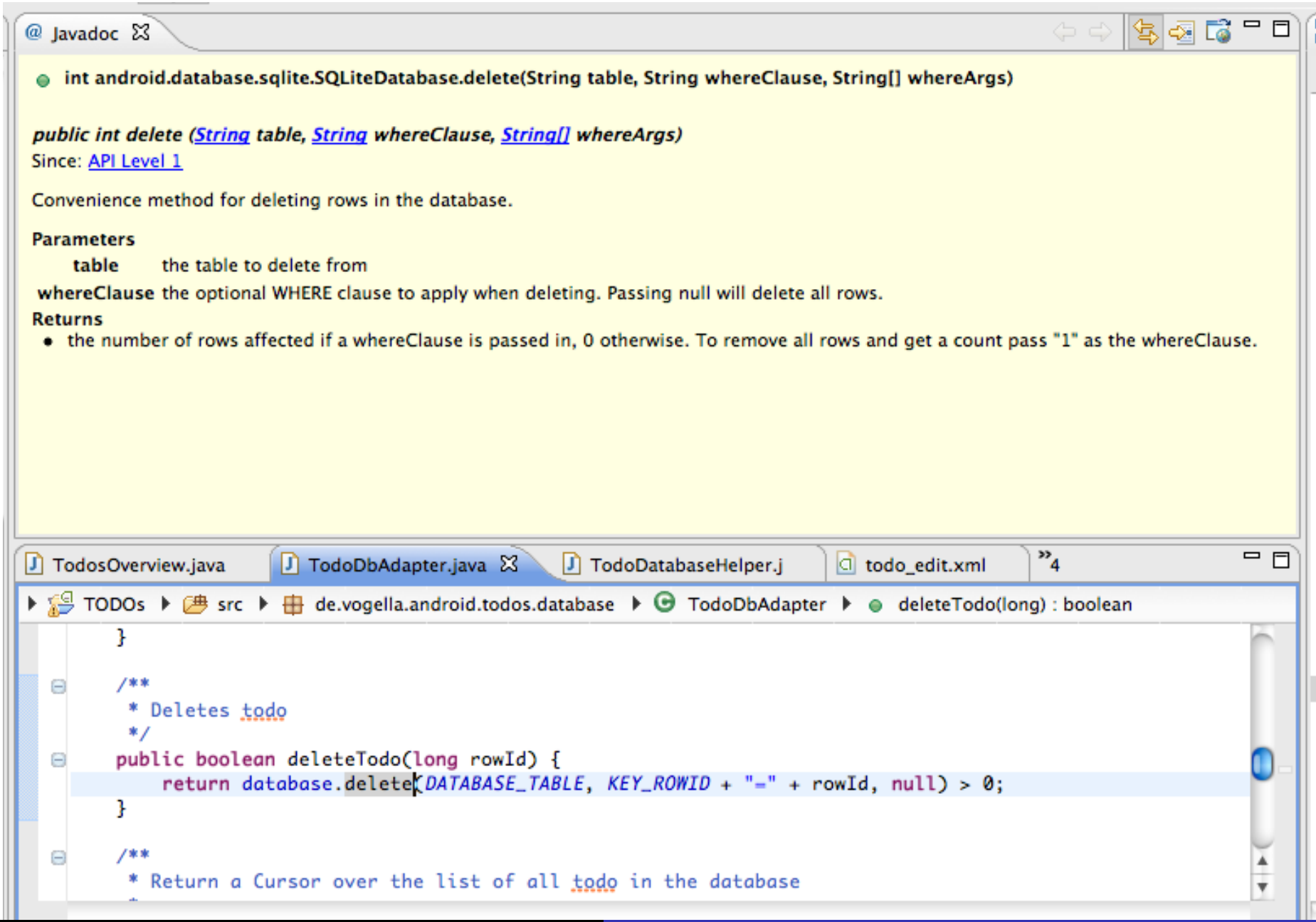
The `insert()` method



The update() method



The delete() method



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;
    }

}
```

oblems | @ Javadoc | Declaration | Console

de.vogella.android.todos.database.TodoDbAdapter

Resources

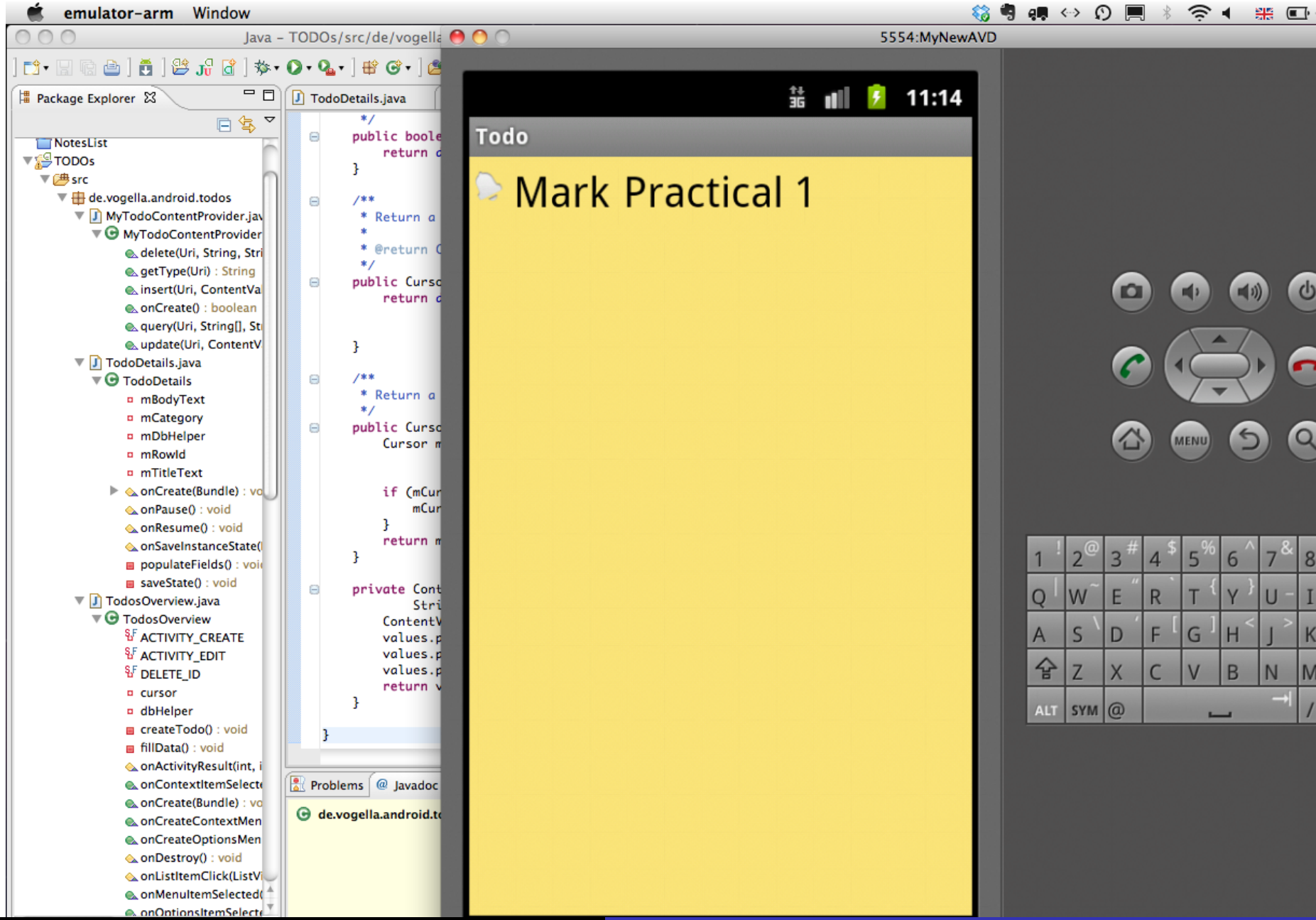
```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.

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;
```

Problems | Javadoc | Declaration | Console

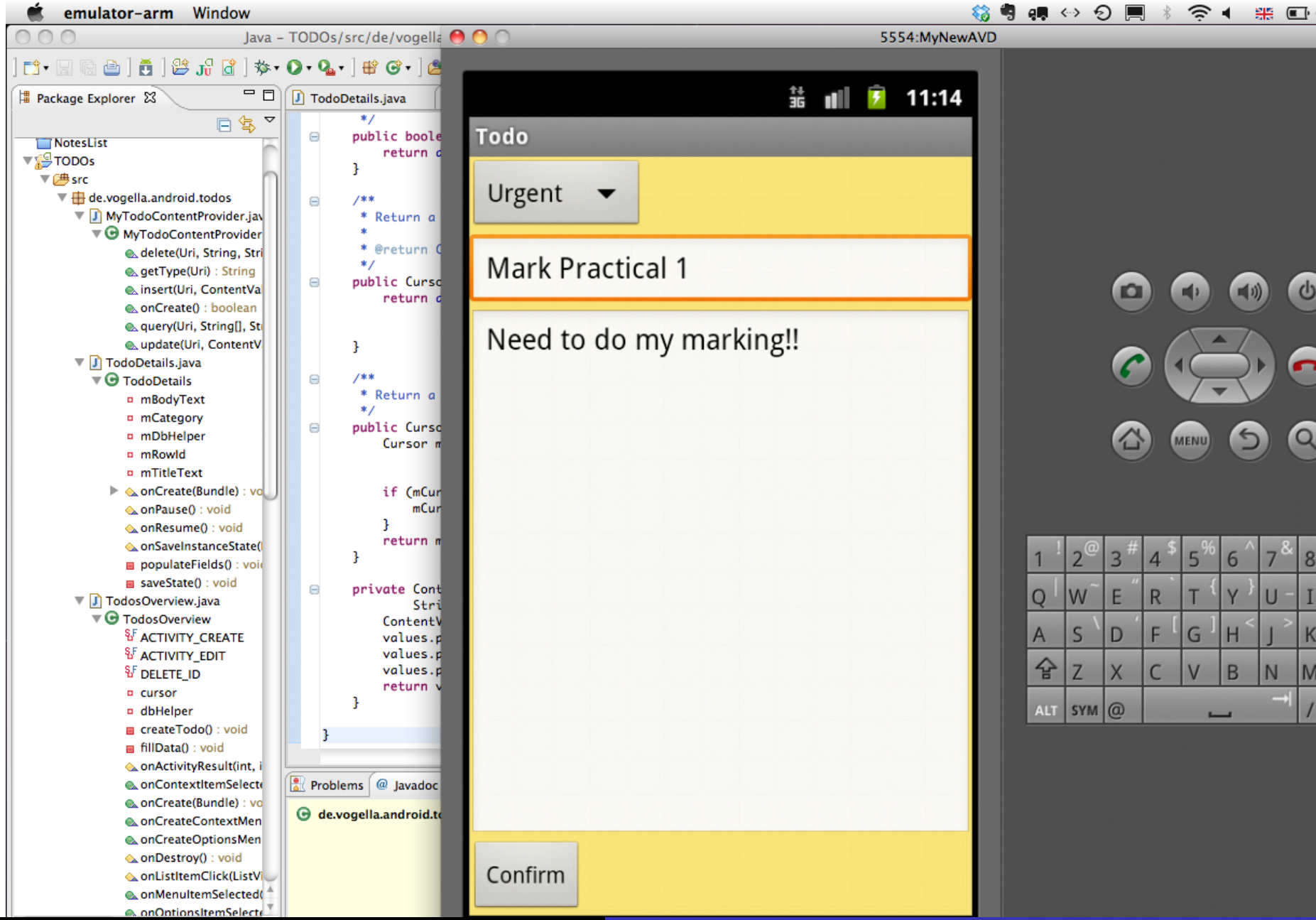
Running the application

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

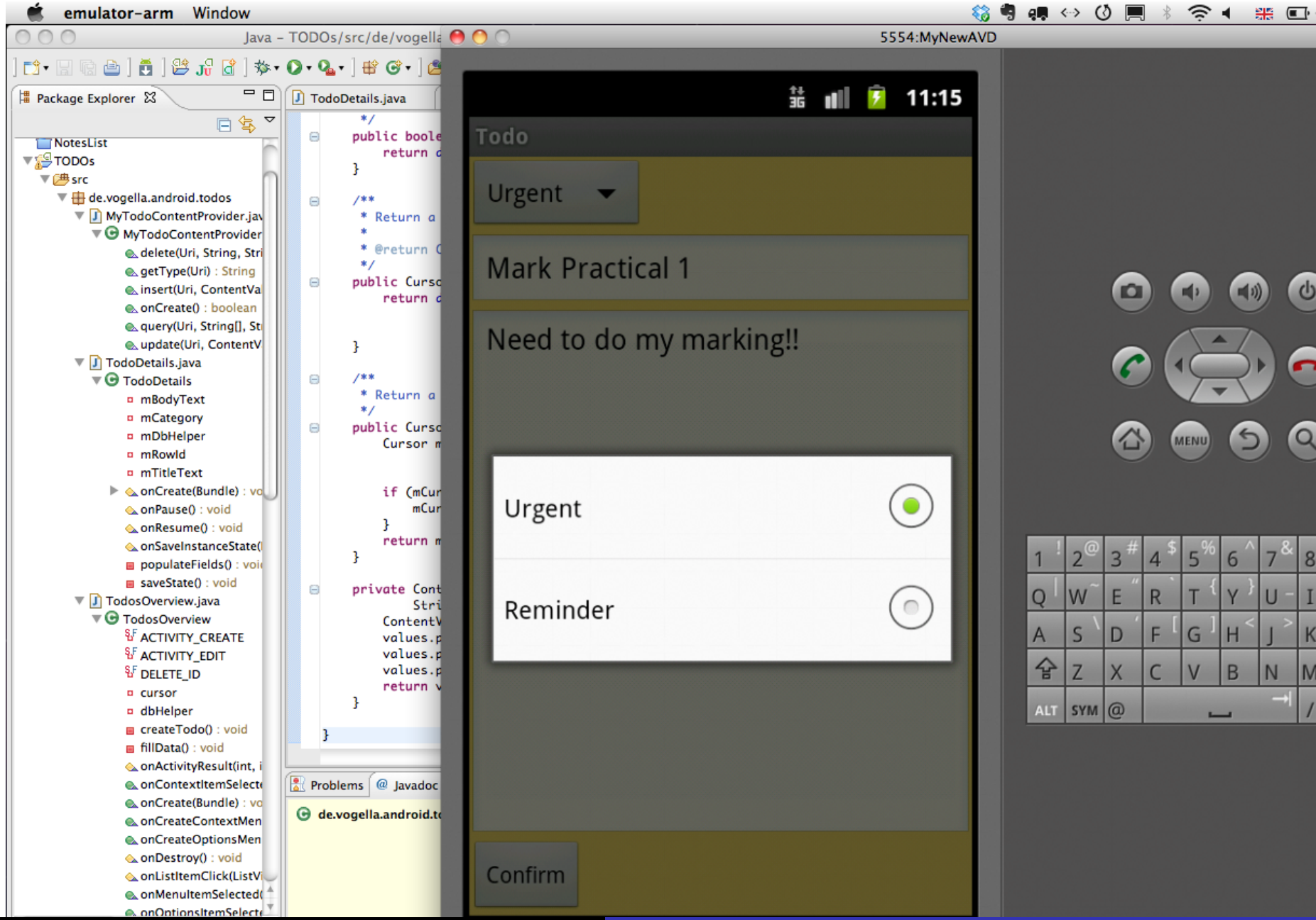
Running the TODOs application



Editing a TODO item



Setting category to "Urgent"



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.

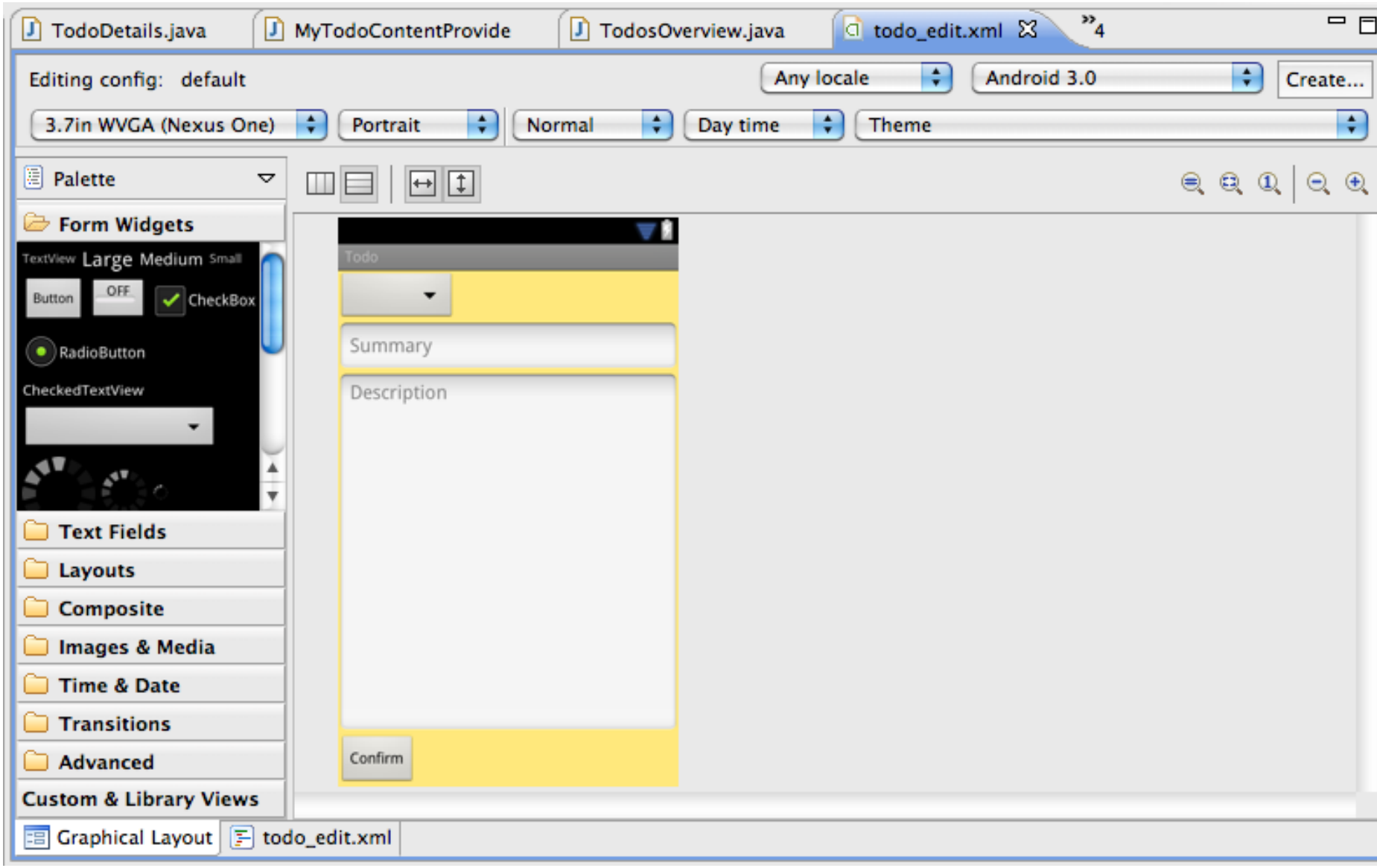
TodoDetails (imports)

```
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.TodoDbAdapter;

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

    @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);
```

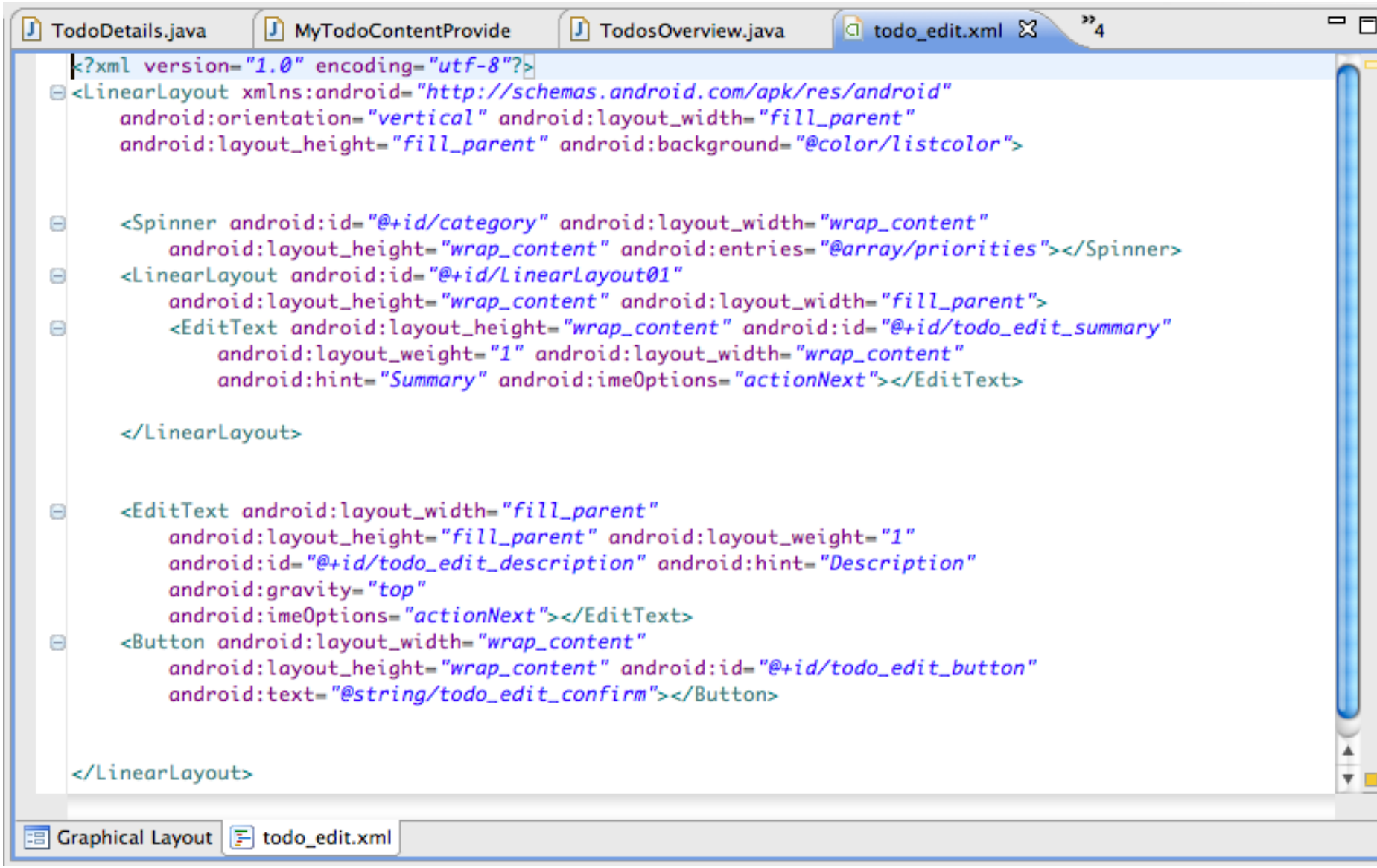
TodoDetails (onCreate)

```
@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();
        }

    });
}
```

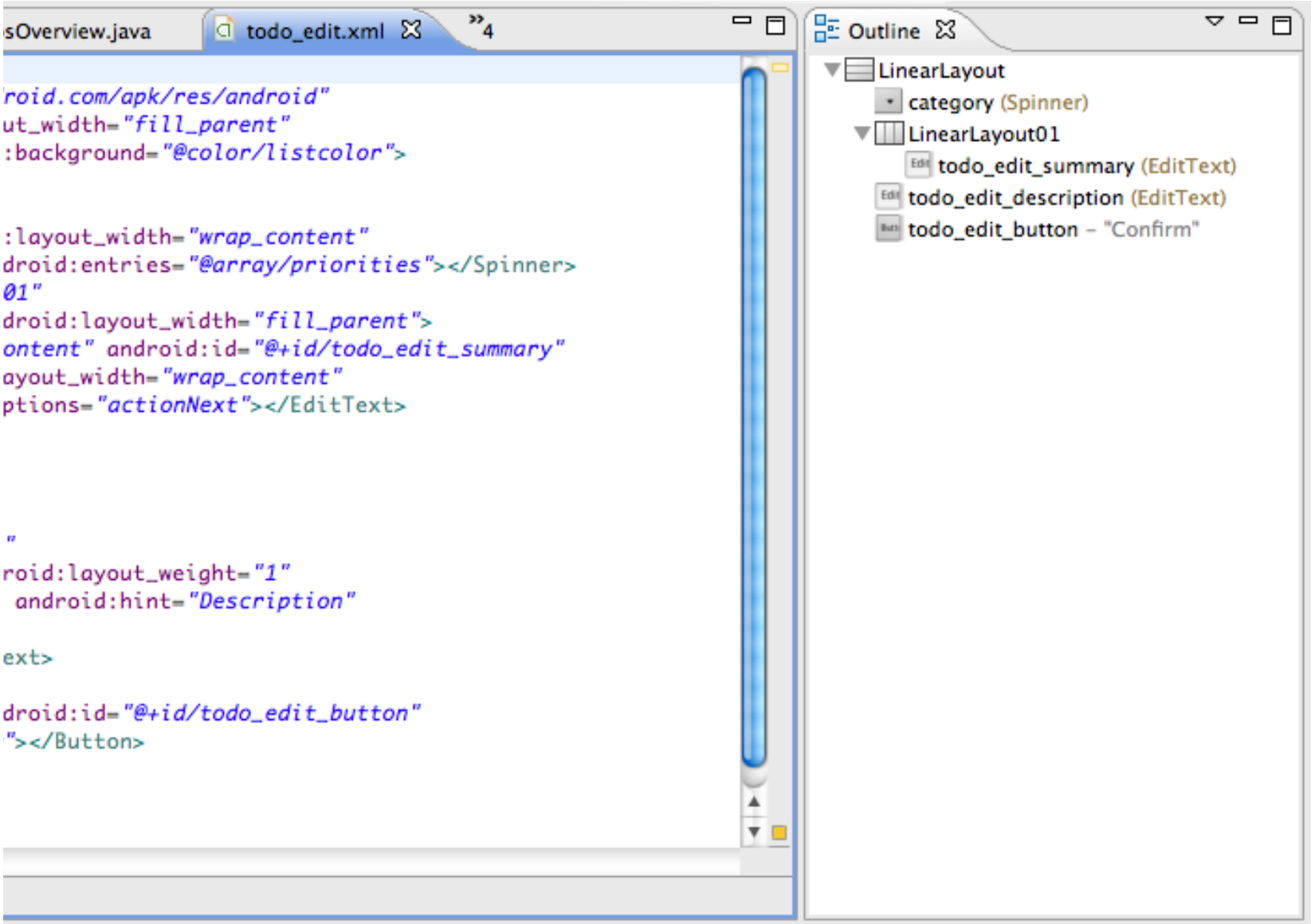
Graphical layout of `todo_edit.xml`



Text of `todo_edit.xml`



Outline of `todo_edit.xml`



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 defintion
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
```

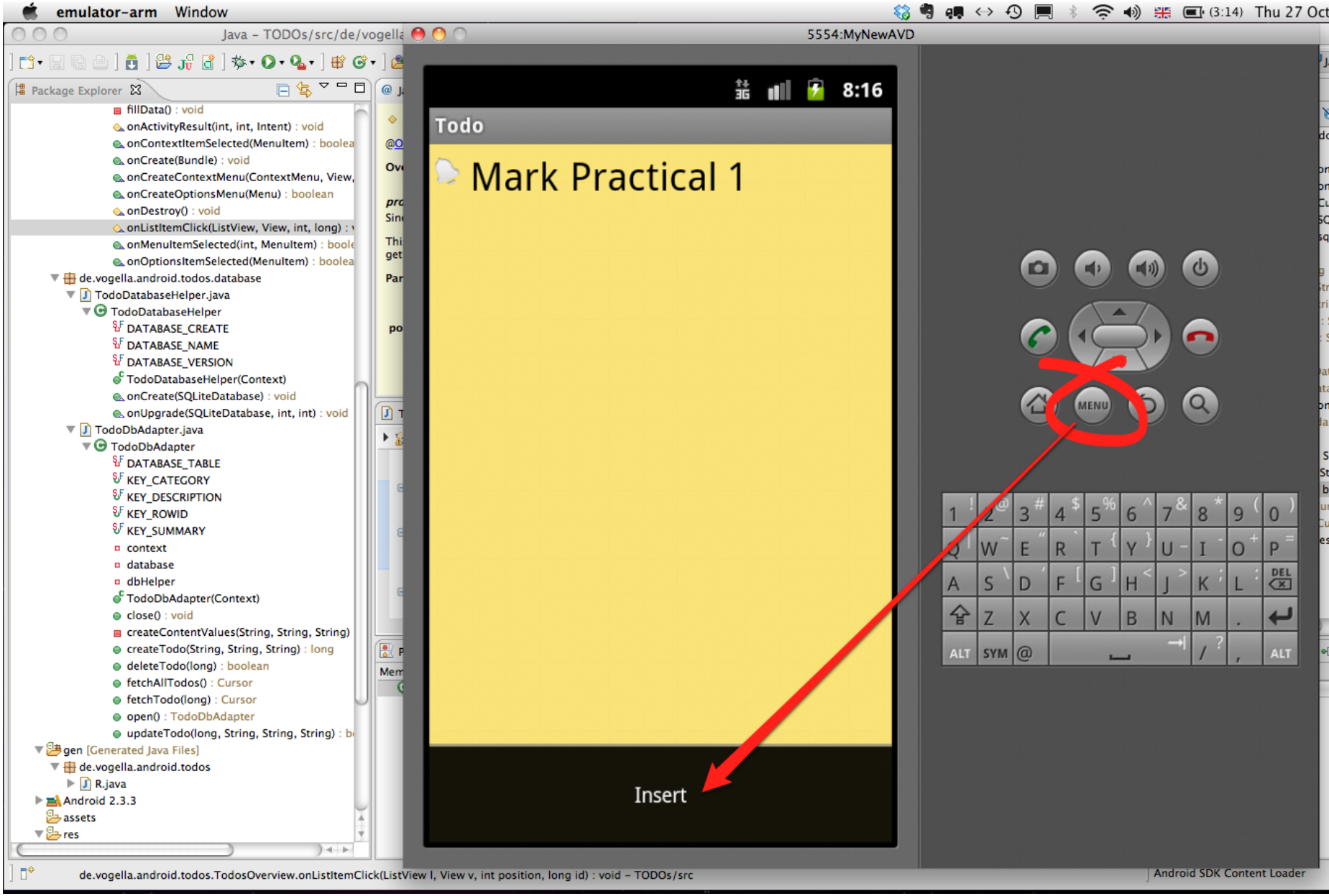
Options menu, item selected

```
// Create the menu based on the XML defintion
@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 onMenuItemSelected(int featureId, MenuItem item) {
    switch (item.getItemId()) {
    case R.id.insert:
        createTodo();
        return true;
    }
    return super.onMenuItemSelected(featureId, item);
}
```

Options item selected

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

    // Reaction to the menu selection
    public boolean onMenuItemSelected(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 onContextItemSelected(MenuItem item) {
```

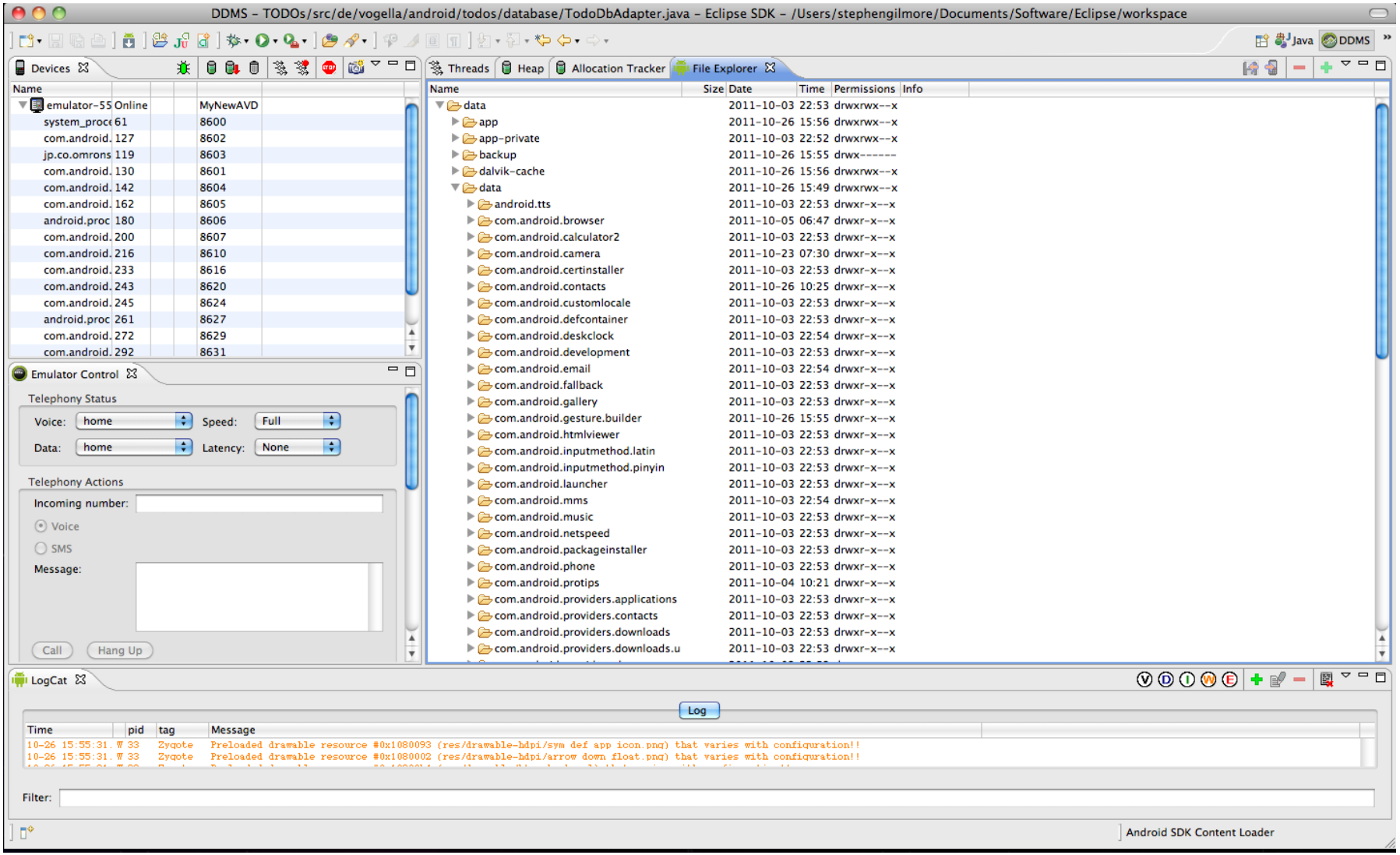
Accessing the insert menu



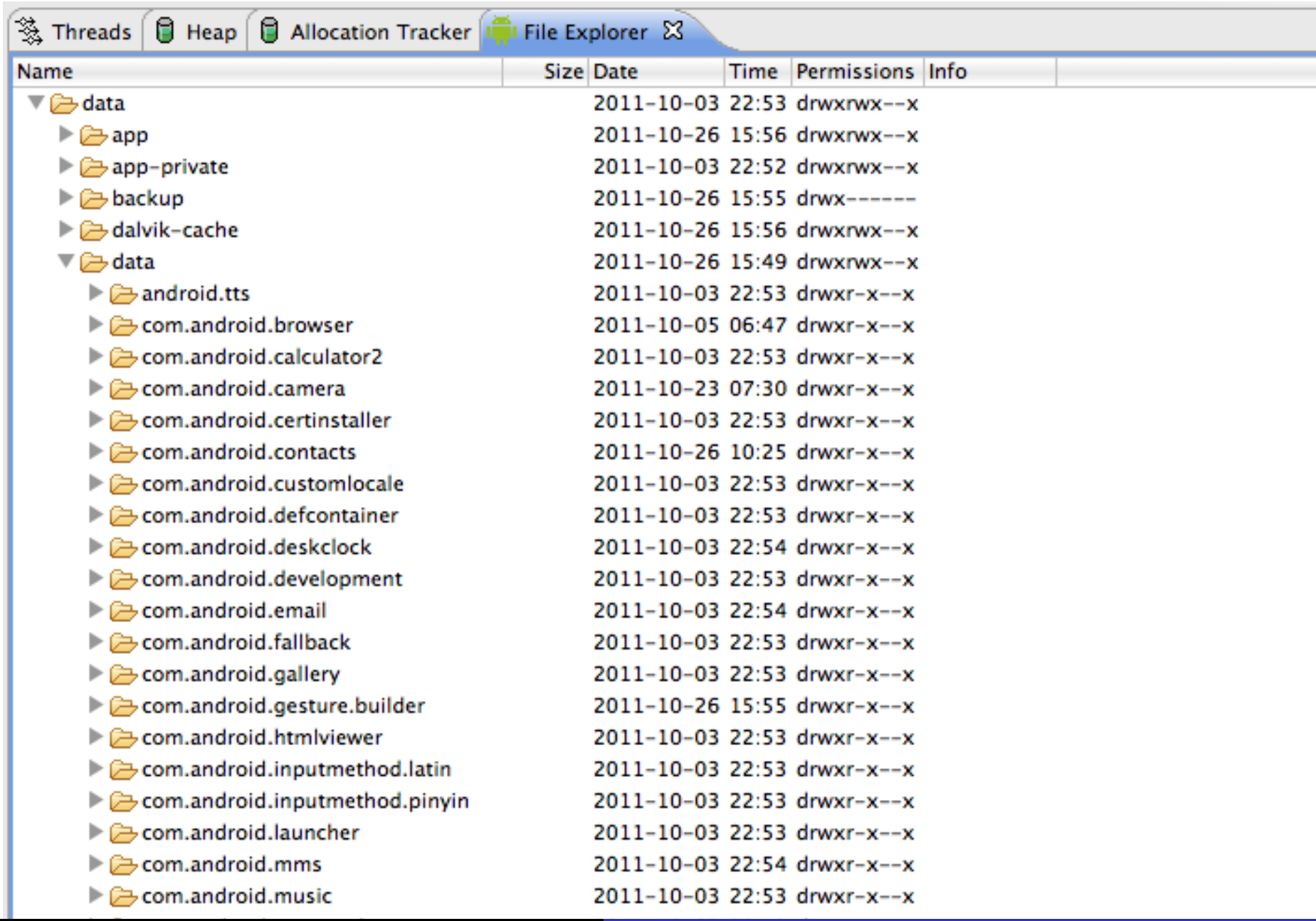
Inspecting the database

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 SQLlite databases.

Dalvik Debug Monitor Server (DDMS)

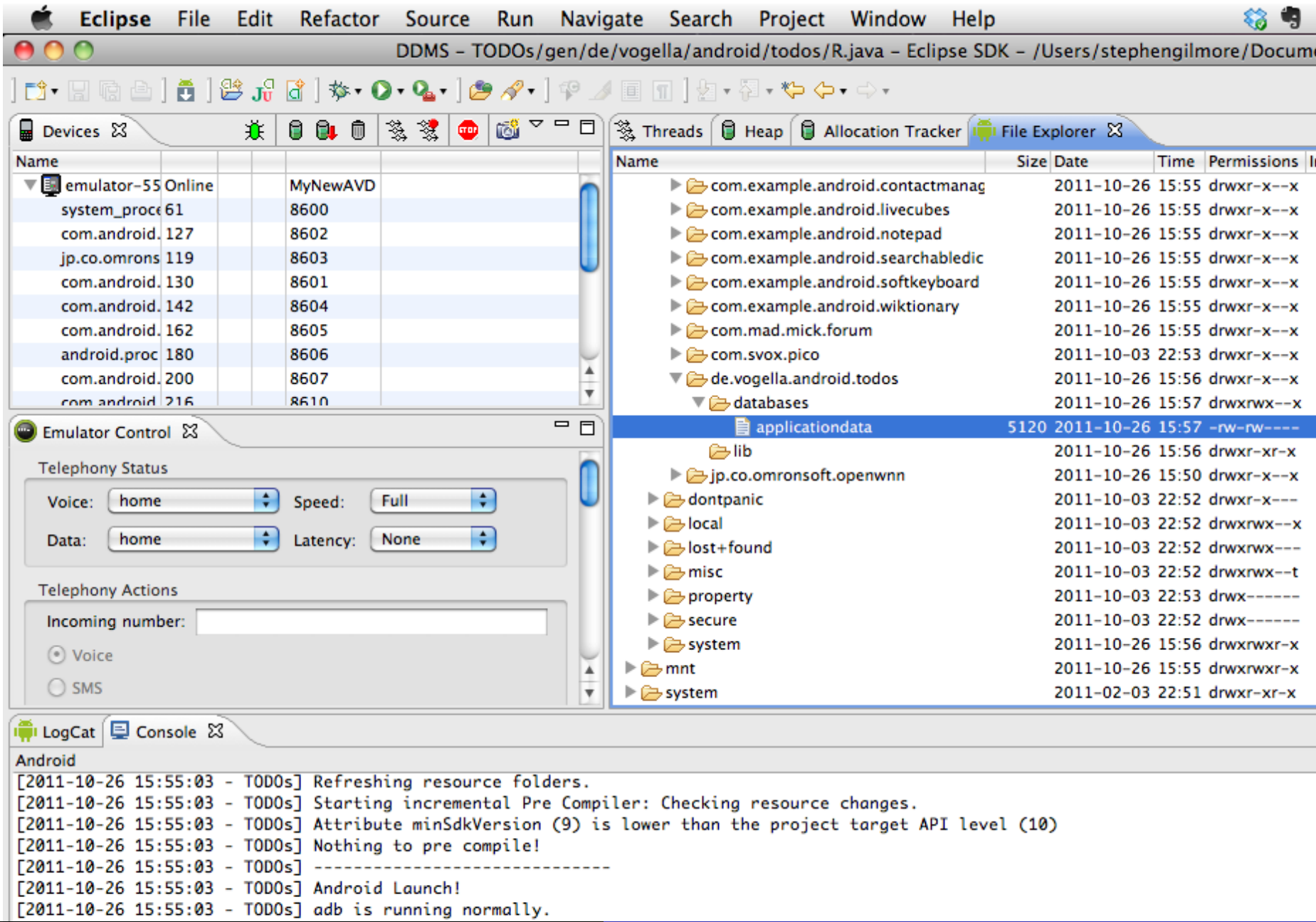


File explorer in DDMS

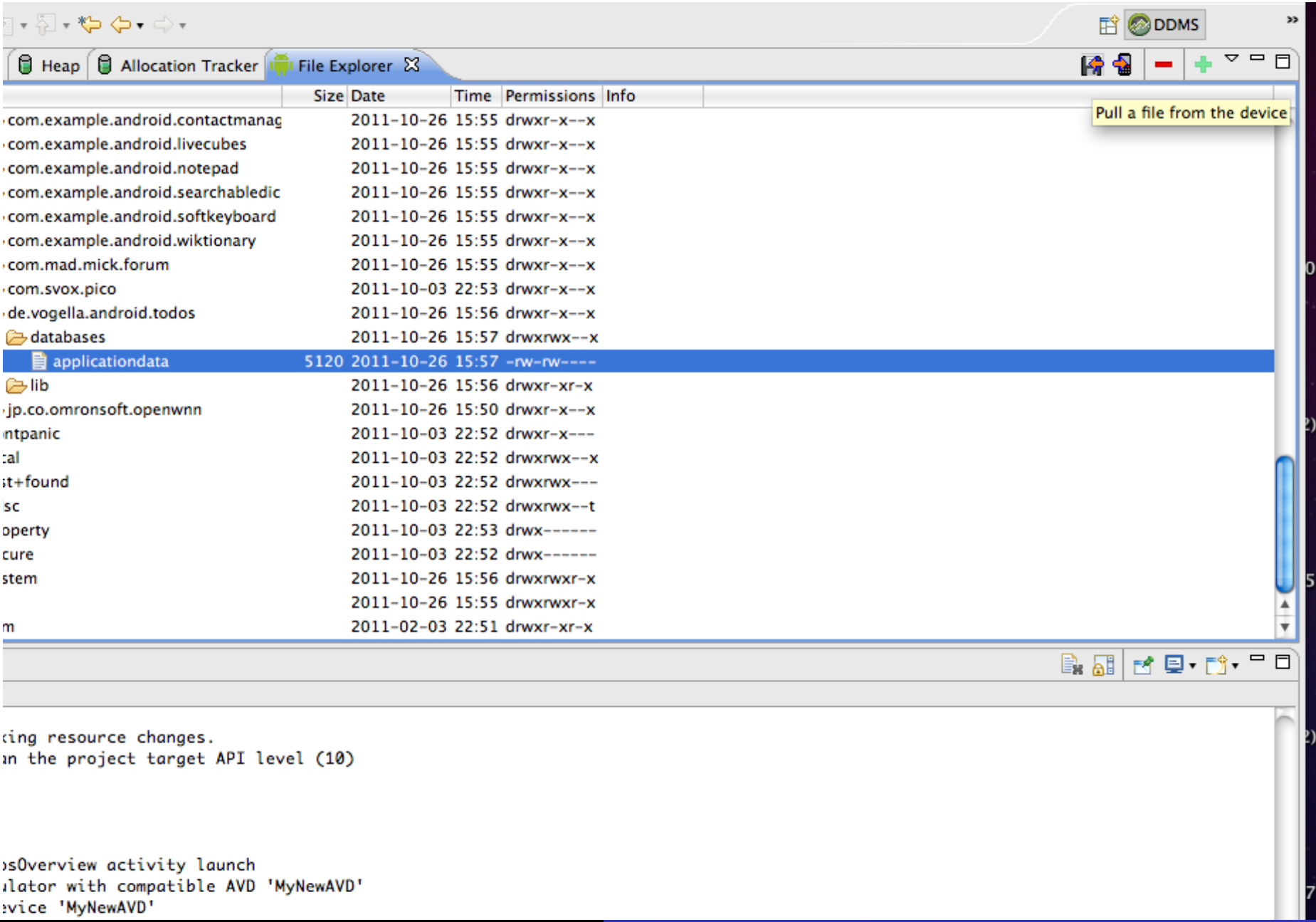


Name	Size	Date	Time	Permissions	Info
data		2011-10-03	22:53	drwxrwx--x	
app		2011-10-26	15:56	drwxrwx--x	
app-private		2011-10-03	22:52	drwxrwx--x	
backup		2011-10-26	15:55	drwx------	
dalvik-cache		2011-10-26	15:56	drwxrwx--x	
data		2011-10-26	15:49	drwxrwx--x	
android.tts		2011-10-03	22:53	drwxr-x--x	
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.customlocale		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	

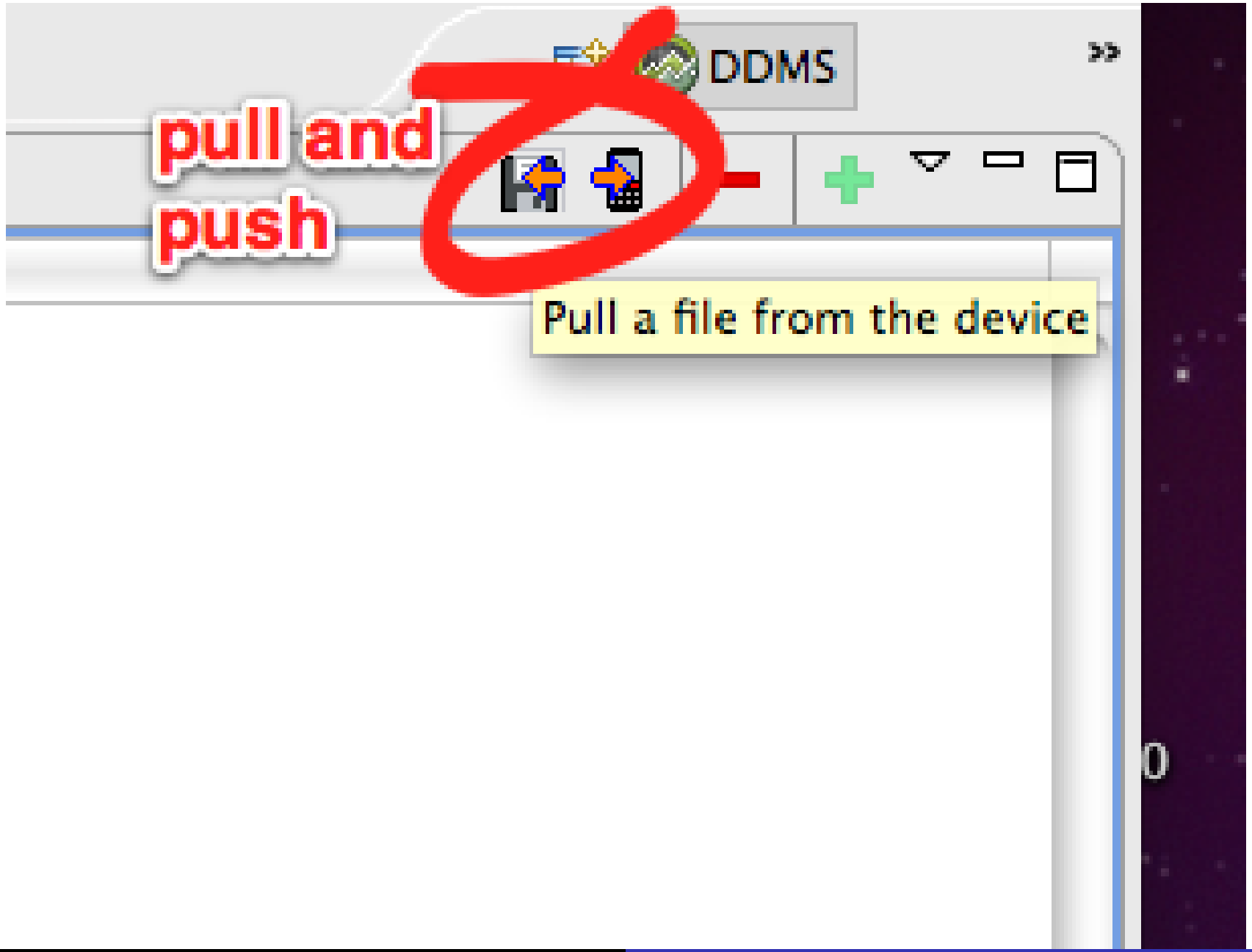
/data/data/de.vogella.android.todos/...



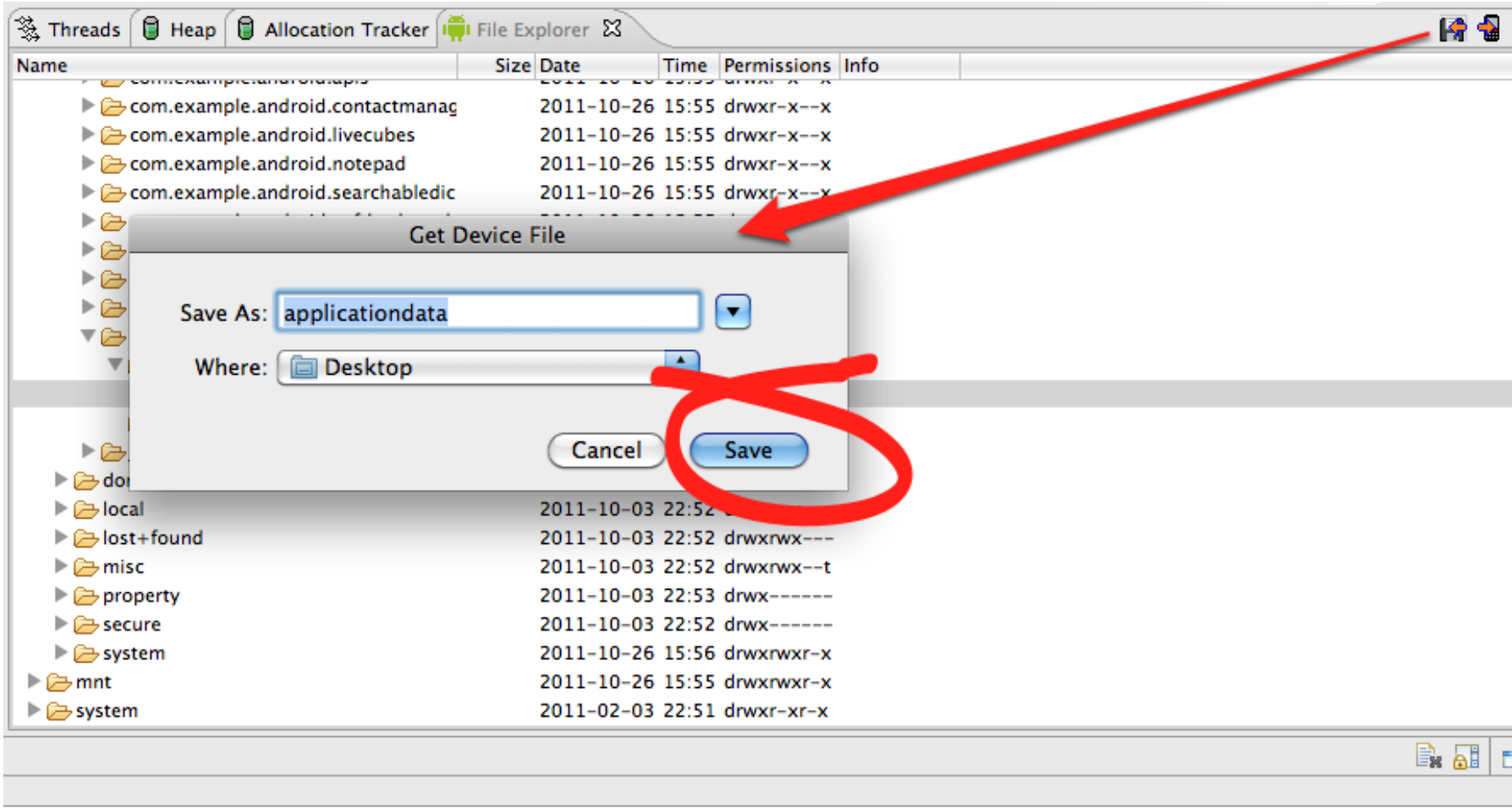
Pulling a file from the device



Pulling a file from the device

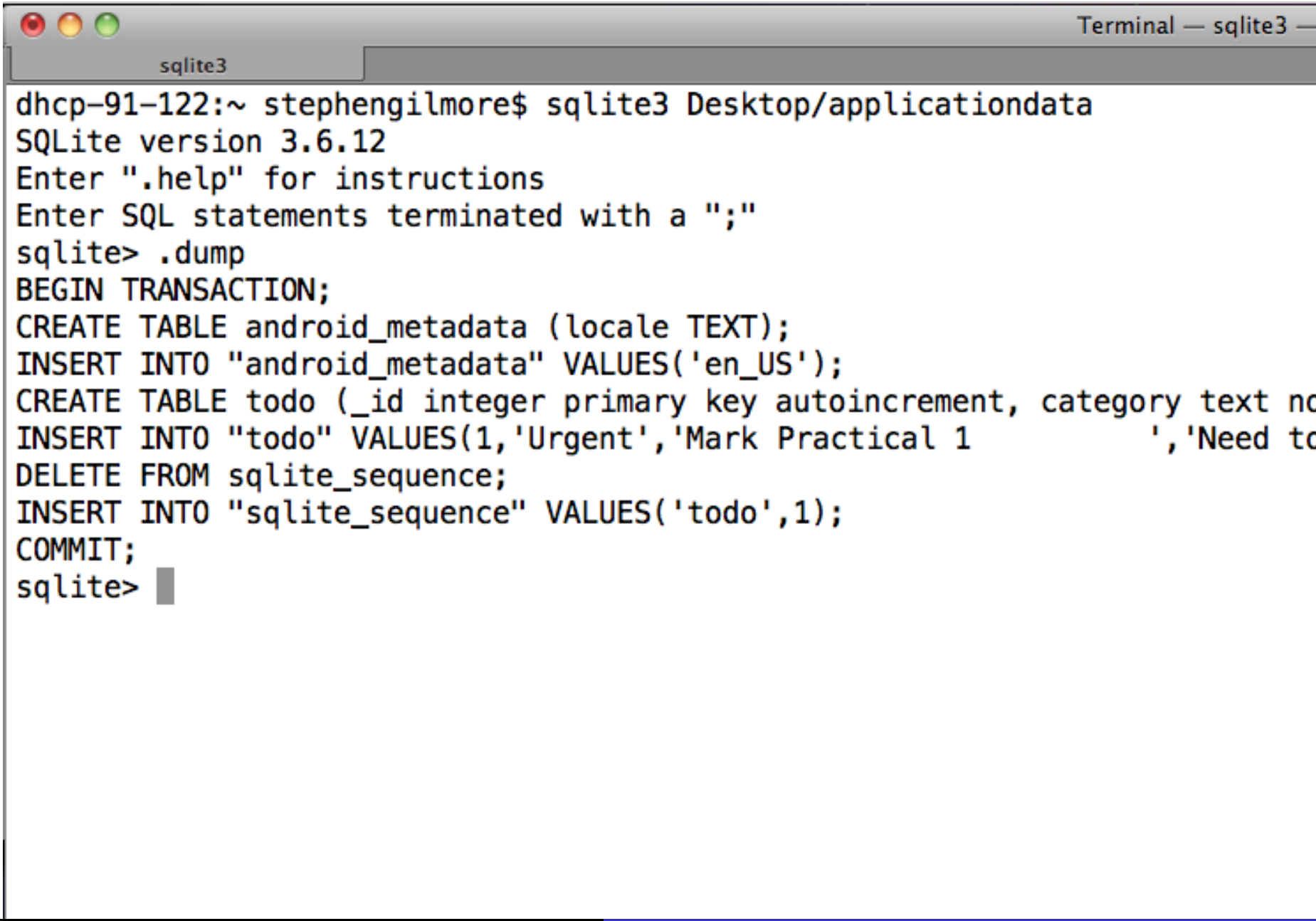


Get the device file

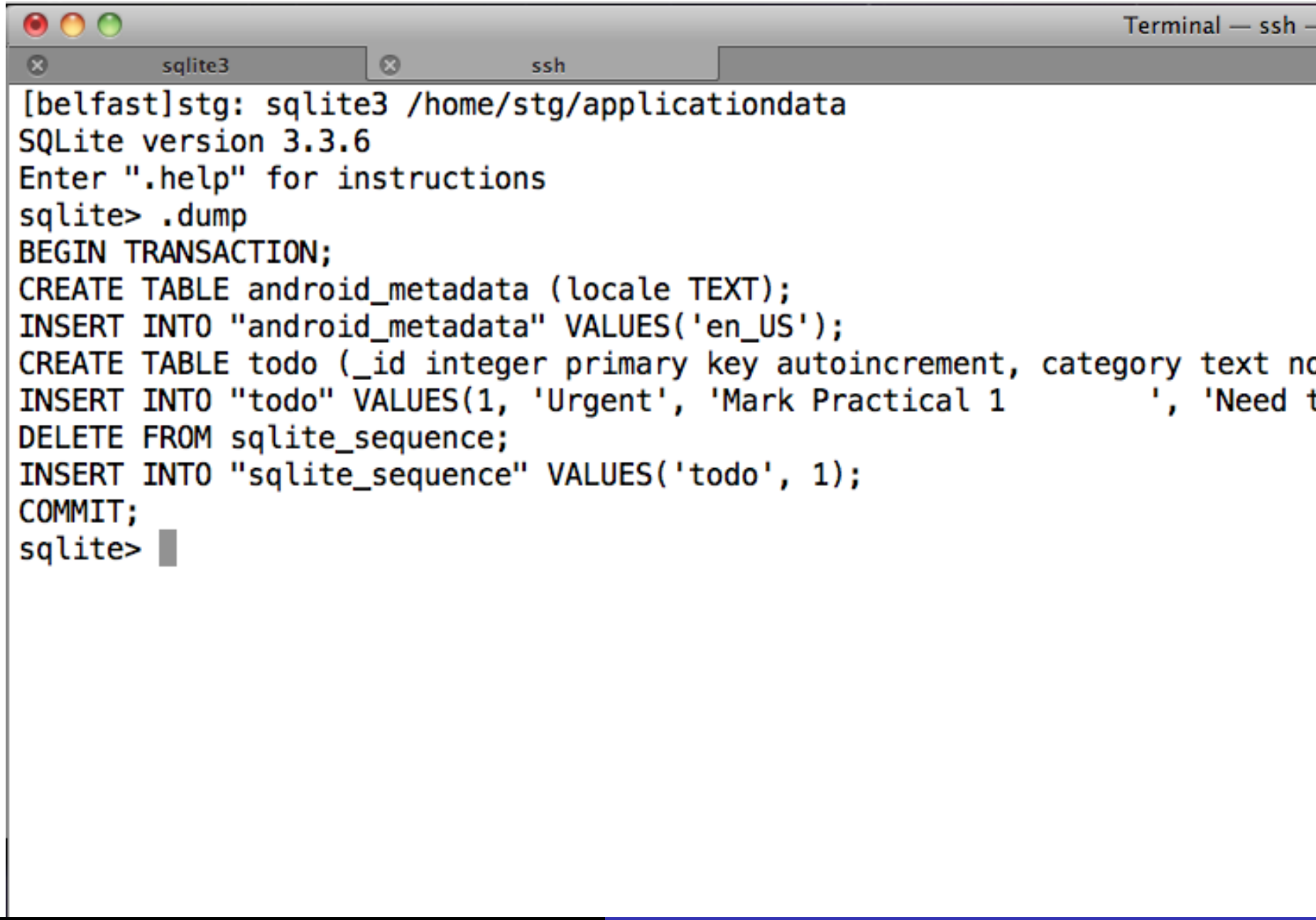



```
ler: Checking resource changes.
 lower than the project target API level (10)


todos.TodosOverview activity launch
ng new emulator with compatible AVD 'MyNewAVD'
/irtual Device 'MyNewAVD'
```

Inspecting the file with `sqlite3`



Inspecting the file with `sqlite3` on DiCE



Other tools for inspecting the database

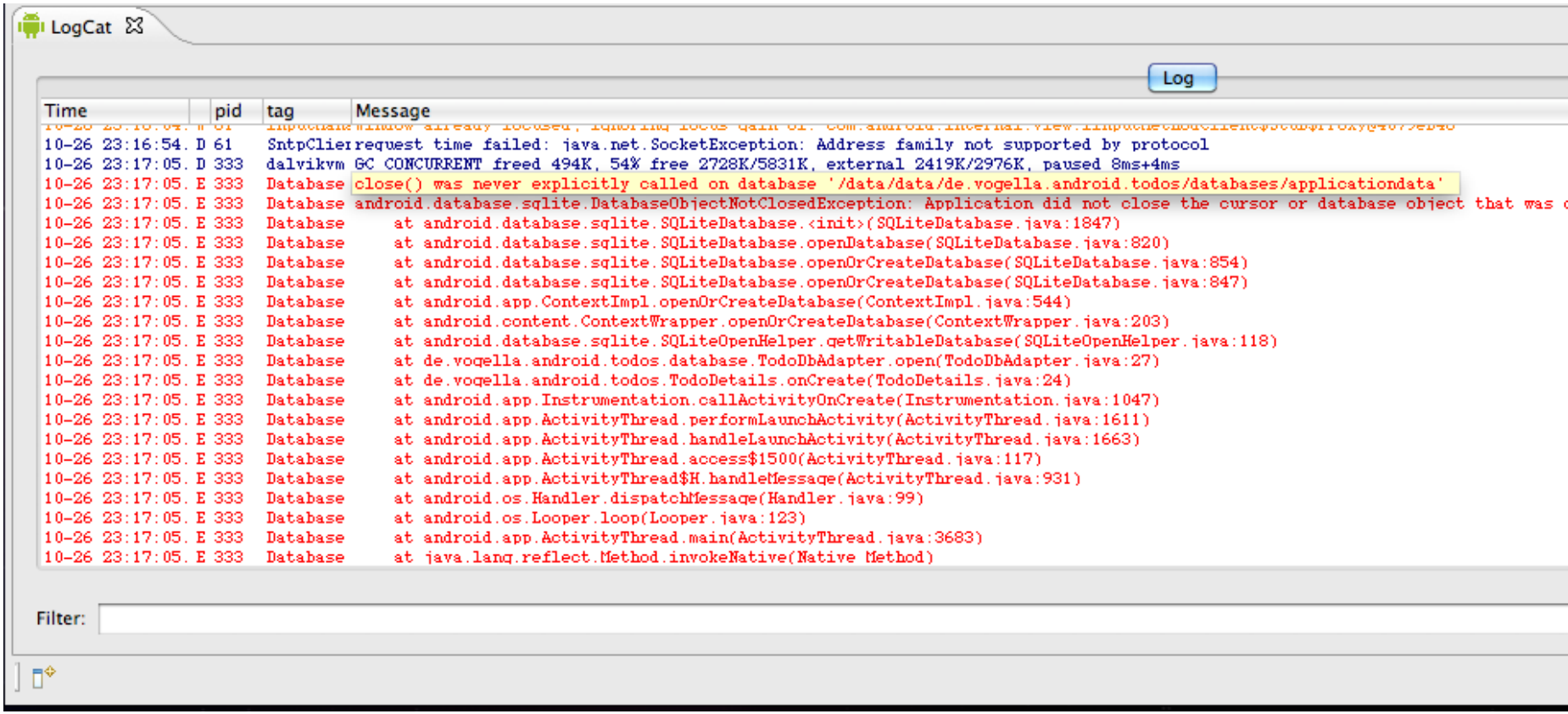
Numerous browsers exist for SQLite databases.

- 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/`

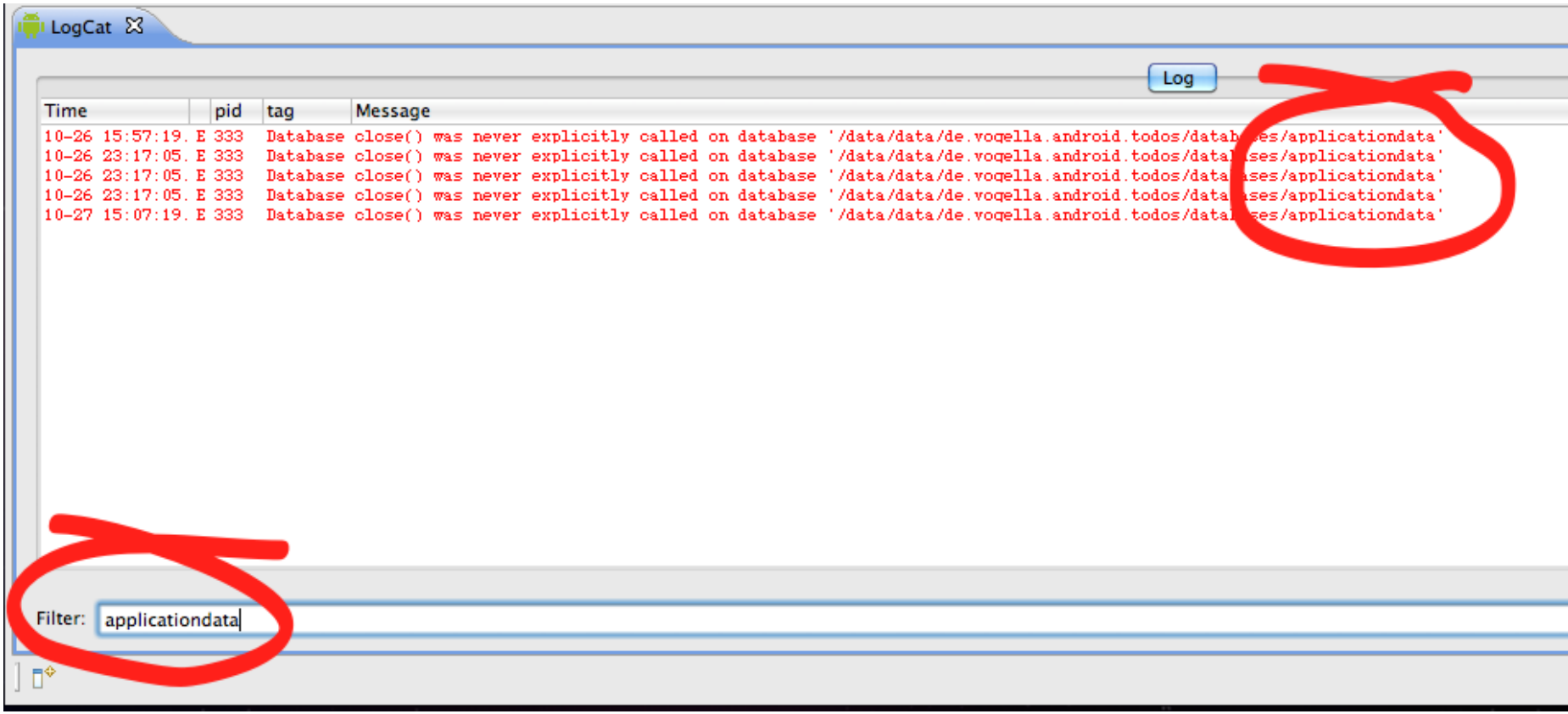
Using LogCat

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

Debugs and errors displayed in LogCat



Can filter messages displayed in LogCat



Can use view menu to export messages