03

Inserindo no banco

Transcrição

Já preparamos parte do caminho para um banco de dados. Vamos continuar nesse percurso para fazer com que o aluno seja de fato selecionado quando clicarmos no *check mark*.

Vamos jogar o aluno no banco de dados?

Para alterar o comportamento do *check mark*, vamos voltar na `FormularioActivity`.

O que queremos fazer é simplesmente dizer para o código inserir o aluno, sem termos que nos preocupar com o banco de dados. Vamos na `onOptionsItemSelected` e teremos o seguinte:

```
@Override
public boolean onOptionsItemSelected(MenuItem  item) {
    switch (item.getItemId()) {
        case R.id.menu_formulario_ok;
            Aluno aluno = helper.pegaAluno();
            Toast.makeText(FormularioActivity.this, "Aluno ” + aluno.getNome() + " salvo!”, Toa

            finish();
            break;
        }
```

```
        return super.onOptionsItemSelected(item);
    }
```

Depois do `Aluno aluno = helper.pegaAluno` , damos um "Enter" e na linha de baixo vamos instanciar o objeto `DAO` , responsável por fazer essa abstração. Vamos criar uma referência para o `DAO` e quando acrescentarmos o `AlunoDAO dao = new AlunoDAO` ele pedirá como parâmetro o contexto,aquele que temos na `AlunoDAO.java` , no construtor, isto é, a identificação para o *Android*. O contexto vai ser acrescentado entre parênteses, digitaremos `this` e ficaremos com: `AlunoDAO dao = new AlunoDAO(this)` .

Vamos inserir algo que fale para o `dao` que queremos que os novos nomes sejam adicionados. Digitaremos `dao.insere` e como parâmetro escreveremos `aluno` e `dao.insere(aluno)` . Só que o método ainda não existe, portanto, daremos um "Alt+Enter" e "Create method insere". Ele criará o método na aba `AlunoDAO.java` onde teremos o seguinte:

```
public class AlunoDAO extends SQLiteOpenHelper {
    public class AlunoDAO (Context context) {
        super(context, "Agenda", null, 1);
    }

     @Override
        public void onCreate (SQLiteDatabase db) {
            String sql = "CREATE TABLE Alunos (id INTEGER PRIMARY KEY, nome TEXT NOT NULL, ende
            db.execSQL(sql);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        String sql = "DROP TABLE IF EXISTS Alunos";
        db.execSQL(sql);
        onCreate(db);
    }

    public void insere(Aluno aluno) {
    }
}
```

Para inserir esse aluno podemos inserir uma instrução `String` e utilizar um `INSERT INTO` e discriminar as colunas e o respectivos valores. Teremos o seguinte:

```
  public void insere(Aluno aluno) {
     String sql = "INSERT INTO Alunos (nome, endereco, telefone, site, nota) VALUES (" + aluno.ge
     }
```

O que aconteceria se alguém fosse no nosso formulário e preenchesse com o nome de Paulo seguido de "'); DROP TABLE Alunos;"?

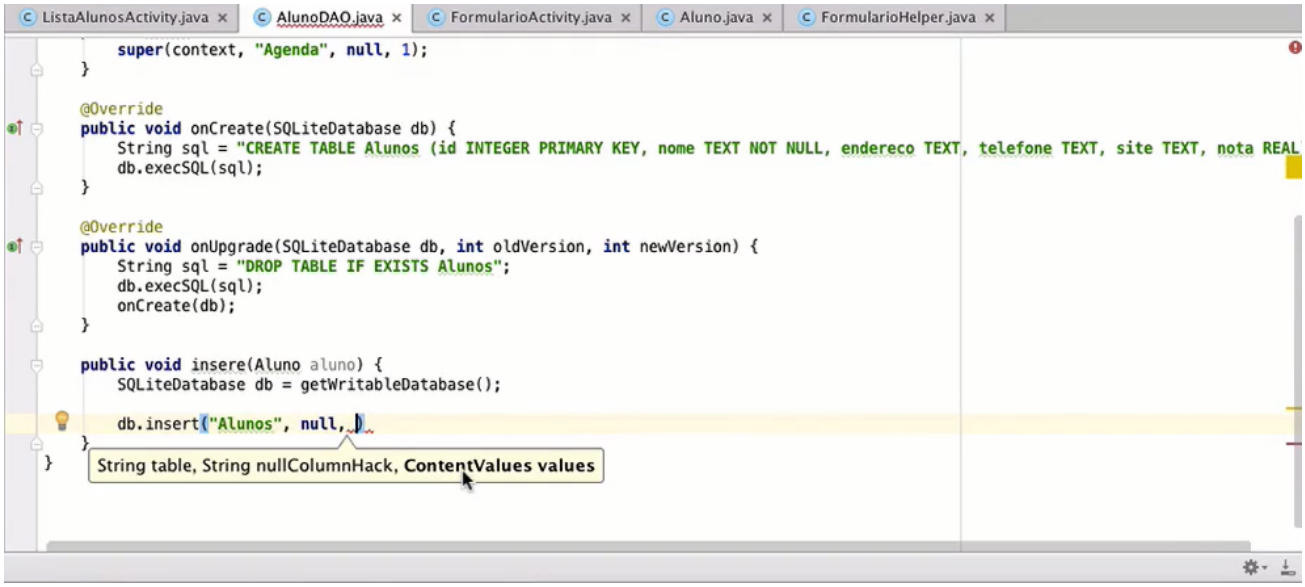
Se o java concatenar o que foi escrito no campo nome, que é "Paulo'); DROP TABLE" teremos mais ou menos o seguinte:

```
public void insere(Aluno aluno) {
   String sql = "INSERT INTO Alunos (nome, endereco, telefone, site, nota) VALUES ('Paulo'); +
   }
```

Executando a `sql` vai rodar a primeira instrução e depois a segunda, que é `DROP TABLE Alunos` e que joga fora a tabela inteira que construímos. Isto é, o banco de dados é manipulado! Essa prática é bem comum para quem gosta de destruir aplicações por aí...

Existem formas de cuidar para que isso não ocorra! Em vez de buscarmos essas ações vamos pedir para que a 'SQLiteOpenHelper' faça isso por nós, que ele faça a operação de inserção de novos alunos.

Para isso, vamos inserir no `insere(Aluno aluno)` um método `getWritableDatabase` cuja referência é o `Database` para que justamente possamos escrever nele. Vamos dar um "Alt+Enter" e "Introduce local variable" e chamaremos um `db`. Agora, temos a referência, falta inserir um `db.insert` para introduzir algo, vamos colocar o primeiro parâmetro "Aluno", o segundo deixaremos como "null", pois não utilizaremos ele normalmente. O `null` é usado quando queremos uma linha em branco, mas não queremos isso aqui! Por fim, ele pede alguns valores, como um "Content Value".



Vamos cria-lo em cima do `db.insert("Alunos", "null", )` . Vamos criar um `ContenValues` , chamar de dados e instancia-lo, teremos `ContentValues dados = new ContentValues()` . Agora, ele vai funcionar como uma espécie de *mapping* do *java*. Isto é, a ideia de chave e valor. Na chave nome, guardaremos o valor nome do aluno, na chave endereço, o endereço e assim por diante. Por isso, acrescentaremos abaixo de `ContentValues` o que queremos guardar, os `dados` , e `put` para colocar o que queremos colocar, o `nome` , `aluno` e `getNome` . Teremos, `dados.put("nome", aluno.getNome())` . Faremos o mesmo com o resto dos dados que queremos guardar, como o endereço, site, telefone e nota. E passamos o `ContentValues` como parâmetro no `insert` .

Temos o seguinte:

```
public void insere(Aluno aluno)  {
SQLiteDatabase db = getWritableDatabase();

ContentValues dados = new ContentValues();
dados.put("nome", aluno.getNome());
dados.put("endereco", aluno.getEndereco());
dados.put("telefone", aluno.getTelefone());
dados.put("site", aluno.getSite());
dados.put("nota", aluno.getNota());

db.insert("Alunos", null, dados );
}
```

Vamos voltar no `FormularioActivity.java` para ver como ficou, instanciamos o `dao` e pedimos para ele inserir logo na linha abaixo. O `dao` não precisa saber nada do banco de dados. Ele simplesmente pede para inserir o "aluno". Quando fazemos uma operação com o `dao` é preciso lembrar de fechar a conexão com o banco de dados. Para isso, vamos utilizar um método `close` . Ficamos com, `dao.close()` . Lembrando que conseguimos chamar o método porque ele já está na `AlunoDAO` na `SQLiteOpenHelper` . Teremos o seguinte:

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

        case R.id.menu_formulario_ok:
                Aluno aluno = helper.pegaAluno();
                AlunoDAO dao = new AlunoDAO(this);
```

```
                dao.insere(aluno);
                dao.close();

                Toast.makeText(FormularioActivity.this, "Aluno ” + aluno.getNome() + " salvo!”,

                finish();
                break;
        }


        return super.onOptionsItemSelected(item);
    }
```

Por enquanto, ainda não conseguimos visualizar o que fizemos através do emulador. Ficaremos com esse início de caminho traçado.