Introdução à plataforma Android

Aula 07 (22/05/15)

Aula 06

- Persistência de dados (Banco de dados SQLite)

Banco de dados SQLite

Referência: goo.gl/PZKWkp

O que é o SQLite

- **SQLite** é uma biblioteca em linguagem C que implementa um banco de dados SQL embutido. Programas que usam a biblioteca SQLite podem ter acesso a banco de dados SQL sem executar um processo SGBD separado.
- SQLite não é uma biblioteca cliente usada para conectar com um grande servidor de banco de dados, mas sim o próprio *servidor*. A biblioteca SQLite lê e escreve diretamente para e do arquivo do banco de dados no disco.

Definindo o SCHEMA do banco

```
public static abstract class UsuarioEntries implements BaseColumns {

    private static final String TABLE_NAME = "usuario";
    private static final String COLUMN_NAME_NOME = "nome";
    private static final String COLUMN_NAME_EMAIL = "email";
    private static final String COLUMN_NAME_CELULAR = "celular";
    private static final String COLUMN_NAME_COR = "cor";

    private static final String TEXT_TYPE = " TEXT";
    private static final String LONG_TYPE = " INTEGER";
    private static final String COMMA_SEP = ",";

    public static final String CREATE_TABLE =
            "CREATE TABLE " + TABLE_NAME + " (" +
                    UsuarioEntries._ID + " INTEGER PRIMARY KEY," +
                    UsuarioEntries.COLUMN_NAME_NOME + TEXT_TYPE + COMMA_SEP +
                    UsuarioEntries.COLUMN_NAME_EMAIL + TEXT_TYPE + COMMA_SEP +
                    UsuarioEntries.COLUMN_NAME_CELULAR + TEXT_TYPE + COMMA_SEP +
                    UsuarioEntries.COLUMN_NAME_COR + TEXT_TYPE +
                    " )";

    public static final String SQL_DELETE_ENTRIES =
            "DROP TABLE IF EXISTS " + UsuarioEntries.TABLE_NAME;
}
```

- Nesta classe declaramos o nome de uma tabela, todas suas colunas, seu script de criação e remoção

Método para salvar um usuário

```
public Usuario salvar(SQLiteDatabase db, Usuario usuario) {

    ContentValues values = new ContentValues();
    if(usuario.getId() != 0) {
        values.put(UsuarioEntries._ID, usuario.getId());
    }
    values.put(UsuarioEntries.COLUMN_NAME_NOME, usuario.getNome());
    values.put(UsuarioEntries.COLUMN_NAME_EMAIL, usuario.getEmail());
    values.put(UsuarioEntries.COLUMN_NAME_CELULAR, usuario.getCelular());
    values.put(UsuarioEntries.COLUMN_NAME_COR, usuario.getCor());

    // insere a nova lina, retornando a primary key como valor
    long newRowId = db.insert(
        UsuarioEntries.TABLE_NAME, // nome da tabela
        null, // coluna que pode ser nula
        values // valores a serem inseridos
    );

    // se salvou corretamente, seta o id
    if(newRowId != -1) {
        usuario.setId(newRowId);
    }

    return usuario;
}
```

- **ContentValues:** uma estrutura chave/valor onde a chave representa o nome de uma coluna e o valor representa o dado desta coluna
- **SQLiteDatabase.insert():** método para inserção de dados no banco. Em nosso exemplo passamos como parâmetro o nome da tabela a receber os dados, e um **ContentValues** que contem os dados a serem inseridos.
- **Resultado final:** podemos imaginar em nosso exemplo que a query final gerada pelo Android será:

INSERT INTO usuario (nome, email, celular, cor) VALUES (“fulano”, “email”, “123456”, “#FF0000”);

Método para alterar um usuário

```
public boolean alterar(SQLiteDatabase db, Usuario usuario) {
    // novos valores para as colunas
    ContentValues values = new ContentValues();
    values.put(UsuarioEntries.COLUMN_NAME_NOME, usuario.getNome());
    values.put(UsuarioEntries.COLUMN_NAME_EMAIL, usuario.getEmail());
    values.put(UsuarioEntries.COLUMN_NAME_CELULAR, usuario.getCelular());
    values.put(UsuarioEntries.COLUMN_NAME_COR, usuario.getCor());

    // selecionar valor da coluna id
    String selecao = UsuarioEntries._ID + " LIKE ?";
    // parametros para o valor da coluna id
    String[] parametrosSelecao = { String.valueOf(usuario.getId()) };

    int count = db.update(
            UsuarioEntries.TABLE_NAME,
            values,
            selecao,
            parametrosSelecao);

    return count > 0;
}
```

- **String "selecao"**: string que representa uma cláusura **WHERE,** para filtrar por colunas. Em nosso exemplo podemos imaginar como "WHERE id = ?"
- **String [] parametrosSelecao**: dados que substituirão a '?' na string de seleção.
- **SQLiteDatabase.update()**: método para atualização de registros no banco. Em nosso exemplo passamos como parâmetro o nome da tabela a receber os dados, um **ContentValues** que contem os dados a serem inseridos, uma string de seleção e um array de parâmetros.
- **Resultado final:** podemos imaginar em nosso exemplo que a query final gerada pelo Android será:

UPDATE usuario SET nome = "ciclano", email = "email@email", celular = "9123456", cor = "#FF1122" WHERE id = 1;

Método para apagar um usuário

```
public boolean apagar(SQLiteDatabase db, Usuario usuario) {
    // define o where
    String selecao = UsuarioEntries._ID + " LIKE ?";
    // define os parametros, em ordem
    String[] parametrosSelecao = {String.valueOf(usuario.getId())};
    // executa o sql
    int count = db.delete(UsuarioEntries.TABLE_NAME, selecao,
            parametrosSelecao);

    return count > 0;
}
```

- **SQLiteDatabase.delete()**: método para remoção de registros do banco. Em nosso exemplo passamos como parâmetro o nome da tabela, uma string de seleção e um array de parâmetros.
- **Resultado final:** podemos imaginar em nosso exemplo que a query final gerada pelo Android será:

SELECT *column_name*,*column_name* FROM *table_name*;

Método para buscar dados (1/2)

```
public List<Usuario> listar(SQLiteDatabase db) {
    // define quais colunas serao projetadas
    String[] projecao = {
            UsuarioEntries._ID,
            UsuarioEntries.COLUMN_NAME_NOME,
            UsuarioEntries.COLUMN_NAME_EMAIL,
            UsuarioEntries.COLUMN_NAME_CELULAR,
            UsuarioEntries.COLUMN_NAME_COR
    };

    // ordenacao
    String ordenacao = UsuarioEntries._ID + " ASC";

    Cursor cursor = db.query(
            UsuarioEntries.TABLE_NAME,  // nome da tabela
            projecao,                   // colunas a serem retornadas
            null,                       // colunas da clausula where
            null,                       // valores da clausula where
            null,                       // nao agrupa os resultados
            null,                       // nao filtra por grupos
            ordenacao
    );
```

- **String[] projecao:** dados a serem projetados no SELECT.
- **String ordenacao:** cláusula ORDER BY
- **SQLiteDatabase.query()**: busca de dados no banco. Em nosso exemplo passamos como parâmetro o nome da tabela, uma string de projecao e uma string de ordenacao.
- **Resultado final:** podemos imaginar em nosso exemplo que a query final gerada pelo Android será:

SELECT nome, email, celular, cor FROM *usuario ORDER BY id ASC*;

Método para buscar dados (2/2)

```
List<Usuario> usuarios = new ArrayList<>();
    while(cursor.moveToNext()) {
        Usuario usuario = new Usuario();

        usuario.setId(
                cursor.getLong(cursor.getColumnIndex(UsuarioEntries._ID)));

        usuario.setNome(

cursor.getString(cursor.getColumnIndex(UsuarioEntries.COLUMN_NAME_NOME)));

        usuario.setEmail(

cursor.getString(cursor.getColumnIndex(UsuarioEntries.COLUMN_NAME_EMAIL)));

        usuario.setCelular(

cursor.getString(cursor.getColumnIndex(UsuarioEntries.COLUMN_NAME_CELULAR)));

        usuario.setCor(

cursor.getString(cursor.getColumnIndex(UsuarioEntries.COLUMN_NAME_COR)));

        usuarios.add(usuario);
    }

    return usuarios;
}
```

- **Cursor**: objeto que contem os dados retornados pela query
- Neste trecho, percorremos todas as linhas retornadas pela query e as convertemos em objetos **Usuario**

Aulas disponíveis em:
bitbucket.org/introducaoandroid/

Obrigado e até a próxima!