




LẬP TRÌNH ANDROID CƠ BẢN

BÀI 6: SQLITE (T.T)

- ⊙ Kết thúc bài học này bạn có khả năng
 - ⊙ Tổ chức cấu trúc ứng dụng
 - ⊙ Thao tác với database



Phần I: Tổ chức cấu trúc ứng dụng

-  Tổ chức cấu trúc ứng dụng

-  Tạo Class DAO

Phần II: Thao tác với database

-  Xử lý xem, thêm, xóa, sửa trong activity

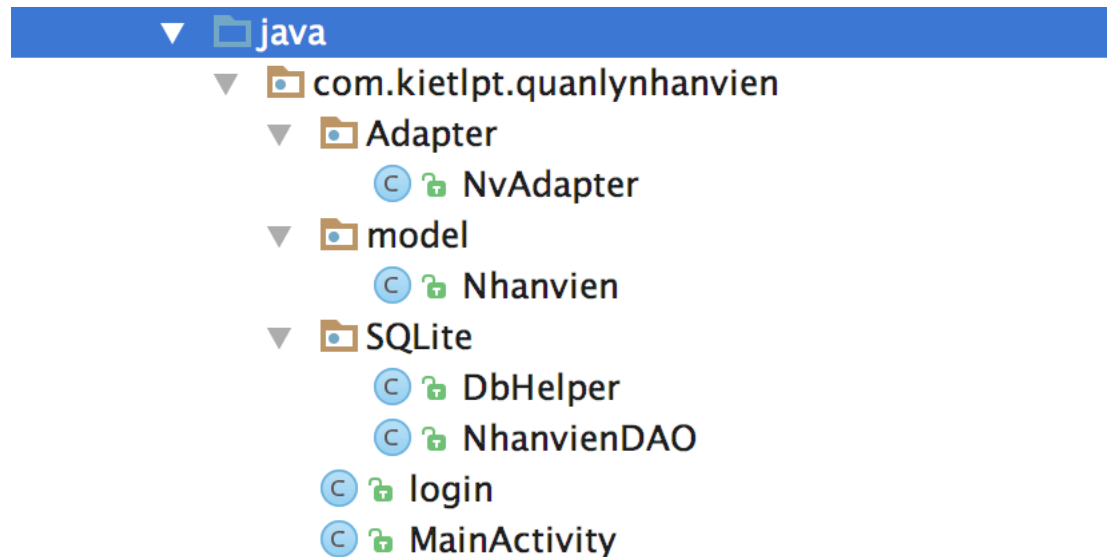




BÀI 6: SQLITE (TT)

PHẦN I: TỔ CHỨC CẤU TRÚC ỨNG DỤNG

- Để dễ viết code và bảo trì chúng ta cần tổ chức cấu trúc ứng dụng theo nhiều package theo cấu trúc sau:



- Adapter: chứa các class dùng để Custom ListView
- Model: chứa các Class quản lý đối tượng
- SQLite: chứa các Class liên quan đến SQLite
- Ngoài ra các thể tạo thêm các package khác chứa các activity..

- Để chuẩn bị truy vấn và thao tác với CSDL, chúng ta cần tạo các Class DAO để thực hiện các thao tác trên.
- Class DAO có hàm tạo phải thực hiện kết nối SQLite (gọi Class DBHelper)
- Tạo phương thức truy vấn (xem)
- Tạo phương thức thêm
- Tạo phương thức xóa
- Tạo phương thức sửa

- Hàm tạo:

```
public class NhanvienDAO {  
    private SQLiteDatabase db;  
  
    public NhanvienDAO(Context context) {  
        DbHelper dbHelper = new DbHelper(context);  
        db = dbHelper.getWritableDatabase();  
    }  
}
```


- Phương thức truy vấn:
 - Sử dụng đối tượng Cursor chứa data đã truy vấn được.
 - Sử dụng phương thức rawQuery của SQLiteDatabaseHelper để truy vấn dữ liệu.
 - Duyệt qua đối tượng Cursor bằng vòng lặp

- Phương thức truy vấn:
 - ❖ Phương thức truy vấn chung:

```

public List<Nhanvien> getNV(String sql, String...selectionArgs) {

    List<Nhanvien> list = new ArrayList<>();

    Cursor c = db.rawQuery(sql, selectionArgs);

    while (c.moveToNext()){

        Nhanvien nv = new Nhanvien();
        nv.id = c.getString(c.getColumnIndex("id"));
        nv.name = c.getString(c.getColumnIndex("name"));
        nv.salary = c.getInt(c.getColumnIndex("salary"));

        list.add(nv);
    }

    return list;
}

```

- Phương thức truy vấn:
 - ❖ Phương thức truy vấn All:

```
public List<Nhanvien> getNhanVienAll() {  
    String sql = "SELECT * FROM nhanvien";  
  
    return getNV(sql);  
}
```

- Phương thức truy vấn:
 - ❖ Phương thức truy vấn theo điều kiện:

```
public Nhanvien getNhanVenID(String id) {  
    String sql = "SELECT * FROM nhanvien WHERE id=?";  
  
    List<Nhanvien> list = getNV(sql,id );  
  
    return list.get(0);  
}
```

- Phương thức thêm:

- ❖ Sử dụng đối tượng ContentValues chứa giá trị đối tượng cần thêm vào database.
- ❖ Sử dụng phương thức insert của SQLiteDatabaseHelper để thêm dữ liệu.

- Phương thức thêm:

```
public long insert(Nhanvien s) {  
    ContentValues values = new ContentValues();  
    values.put("name", s.name);  
    values.put("id", s.id);  
    values.put("salary", s.salary);  
  
    return db.insert("nhanvien", null, values);  
}
```

- Phương thức sửa:
 - ❖ Sử dụng đối tượng ContentValues chứa giá trị đối tượng cần thêm vào database.
 - ❖ Sử dụng phương thức update của SQLiteDatabaseHelper để sửa dữ liệu.

- Phương thức sửa:

```
public int update(Nhanvien s) {  
    ContentValues values = new ContentValues();  
    values.put("name", s.name );  
    values.put("salary", s.salary);  
  
    return db.update("nhanvien", values, "id=?", new String[]{s.id});  
}
```


- Phương thức xoá:

- ❖ Sử dụng phương thức delete của SQLiteDatabaseHelper để xoá dữ liệu.

```
public int delete(String id) {  
    return db.delete("nhanvien", "id=?", new String[]{id});  
}
```





BÀI 6: SQLITE (TT)

PHẦN II: THAO TÁC VỚI DATABASE

- Show dữ liệu lên ListView đã custom:
 - ❖ Tạo đối tượng từ Class Dao
 - ❖ Gọi phương thức get data trong class Dao
 - ❖ Show lên ListView đã custom

- Show dữ liệu lên ListView đã custom:

```
//Tạo đối tượng Class DAO
```

```
final NhanvienDAO nhanvienDAO = new NhanvienDAO(this);
```

```
//get all Nhan vien
```

```
nhanviens = nhanvienDAO.getNhanVienAll();
```

```
adapter = new NvAdapter(this,nhanviens);
```

```
lv.setAdapter(adapter);
```

- Thêm dữ liệu :
 - ❖ Tạo đối tượng từ Class Dao
 - ❖ Gọi phương thức insert data trong class Dao
 - ❖ Update lại ListView (gọi lại show data)

```
Nhanvien nv = new Nhanvien();  
nv.id = txtId.getText().toString();  
nv.name = txtName.getText().toString();  
nv.salary = Integer.parseInt(txtSalary.getText().toString());  
  
nhanvienDAO.insert(nv);
```

- Sửa dữ liệu :
 - ❖ Tạo đối tượng từ Class Dao
 - ❖ Gọi phương thức update data trong class Dao
 - ❖ Update lại ListView (gọi lại show data)

```
Nhanvien nv = new Nhanvien();  
nv.id = txtId.getText().toString();  
nv.name = txtName.getText().toString();  
nv.salary = Integer.parseInt(txtSalary.getText().toString());  
  
nhanvienDAO.update(nv);
```

- Xoá dữ liệu :
 - ❖ Tạo đối tượng từ Class Dao
 - ❖ Gọi phương thức delete data trong class Dao
 - ❖ Update lại ListView (gọi lại show data)

```
Nhanvien nv = new Nhanvien();  
nv.id = txtId.getText().toString();  
  
nhanvienDAO.delete(nv.id);
```





DEMO

- Xử lý xem, thêm, sửa, xoá
trong activity



Phần I: Tổ chức cấu trúc ứng dụng

 Tổ chức cấu trúc ứng dụng

 Tạo Class DAO

Phần II: Thao tác với database

 Xử lý xem, thêm, xóa, sửa trong activity





Cảm ơn