ListActivity implements View.OnClickListener

```
Button btnCreate,btnRead,btnUpdate,btnDelete;      // 글쓰기, 읽기, 수정, 삭제 버튼
EditText editTitle,editContent;                    // 제목, 내용 작성 필드
TextView textResult;                               // 결과물 출력 필드
MemoDAO dao = null;                                // 새 메모객체 필드
```

onCreate

※리스트 초기화 및 버튼 객체 생성

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_list);

    initViews();
    initListener();
    init();
}
```

initView( )

※위젯 객체 생성

```
private void initViews(){
    btnCreate = (Button) findViewById(R.id.btnCreate);
    btnRead = (Button) findViewById(R.id.btnRead);
    btnUpdate = (Button) findViewById(R.id.btnUpdate);
    btnDelete = (Button) findViewById(R.id.btnDelete);

    editTitle = (EditText) findViewById(R.id.editTitle);
    editContent = (EditText) findViewById(R.id.editContent);
    textResult = (TextView) findViewById(R.id.textResult);
}
```

initListener( )

※위젯 리스너 호출

```
private void initListener() {
    btnCreate.setOnClickListener(this);
    btnRead.setOnClickListener(this);
    btnUpdate.setOnClickListener(this);
    btnDelete.setOnClickListener(this);
}
```

onClick( )

※버튼별 동작 정의

```
@Override
public void onClick(View view) {
    switch (view.getId()){
        case R.id.btnCreate:
            createAfterRead();
            break;
        case R.id.btnRead:
            read();
            break;
        case R.id.btnUpdate:
            break;
        case R.id.btnDelete:
            break;
    }
}
```

init( )

※새로운 메모객체 생성

```
private void init(){
    dao = new MemoDAO(this);
}
```

* MemoDAO 클래스의 생성자는 Context context 파라미터를 가지고 있으므로, AppCompatActivity를 상속받는 본 클래스 자체의 context를 전달하기 위해 this를 사용한다 > MemoDAO와 DBHelper의 생성자를 호출하게 된다.

createAfterRead( )

※제목, 글 내용 입력 후 create버튼 터치 시 실행되어 데이터 저장 및 refresh

```
private void createAfterRead(){
    Memo memo = getMemoFromScreen(); // Memo 데이터 화면에서 가져오기
    create(memo);                    // 생성
    showInfo("입력되었습니다!!!");   // 결과 안내(토스트 메시지)
    resetScreen();                   // 화면초기화
    read();                          // 목록 갱신
}
```

getMemoFromScreen( )

※ 화면에서 입력한 메모 title, content내용을 받아 저장한 Memo객체 하나를 생성해 return

```
private Memo getMemoFromScreen(){
    // 1. 화면에서 입력된 값을 가져온다
    String title = editTitle.getText().toString();
    String content = editContent.getText().toString();
    // 2. Memo 객체를 하나 생성해서 값을 담는다
    return new Memo(title,content);
}
```

create(Memo memo)

※ 리턴받은 Memo객체의 title, content 내용을 DAO에서 query로 처리해 SQLite에 전달한다.

```
private void create(Memo memo){
    dao.create(memo);
}
```

getMemoFromScreen( )

※ DB저장 완료 후 사용자에게 notice

```
private void showInfo(String comment){
    Toast.makeText(this, comment, Toast.LENGTH_SHORT).show();
}
```

resetScreen( )

※ 입력 완료 후 텍스트 입력 위젯을 빈칸으로 refresh

```
private void resetScreen(){
    editTitle.setText("");
    editContent.setText("");
}
```

read( )

※ DB상에 있는 모든 memo내용을 받아서 textResult에 출력해준다

```
public void read(){
    ArrayList<Memo> data = dao.read();
    // DB처리 클래스(MemoDAO)에서 작업 :
    // List생성 후, DB상에 저장된 객체 별 정보를 add한 후 리턴한다.
    textResult.setText(" ");   // 결과 창 공백화
    for(Memo memo :data){      // textResult창에 DB에서 전달받은 내용 출
        textResult.append(memo.toString());
    }
}
```

Update , Delete 처리 필요

Memo

```
int id;
String title, content, n_date;
```

Constructor

```
public Memo(){

}
```

Constructor

```
public Memo(String title, String content){
    this.title = title;
    this.content = content;
}
```

toString( )

```
@Override
public String toString() {
    return id+"¦"+title+"¦"+content+"¦"+n_date+"\n";
}
```

MemoDAO

* Memo 데이터를 받아 DBHelper에 query를 전달해 DB에 저장되도록 한다
* C,R,U,D 기능 메서드 보유

```
DBHelper helper;        // DBHelper 클래스(query 전달 역할)의 객체를 생성한다.
```

Constructor

```
public MemoDAO(Context context){
    helper = new DBHelper(context);
}
```

* ListActivity 클래스에서 init( ); 을 통해 MemoDAO 객체가 생성되고, 이 과정에서 ListActivity의 context가 전달된다.
* 해당 context를 받아 (SQLiteOpenHelper 클래스를 상속받는) DBHelper 데이터 타입의 helper 객체가 생성된다.

create(Memo memo)

※(C) ListActivity의 create(memo)에서 전달받은 memo객체를 받아 helper에서 query를 실행 시켜 DB에 insert한다.

```
public void create(Memo memo){
    SQLiteDatabase con = helper.getWritableDatabase();
    String query = "insert into memo(title, content, n_date)" +
            " values('"+memo.title+"','"+memo.content+"',datetime('now','localtime'))";
    con.execSQL(query);
    con.close();
}
```

ArrayList<Memo> read()

※(R) ListActivity에서 read() 메서드 내에 ArrayList<Memo> 타입 객체 data를 생성되며, 본 클래스의 read( )에서 리턴받은 data 내용이 ListActivity의 data객체에 저장된다.

```
public ArrayList<Memo> read(){
    String query = "select id, title, content, n_date from memo";

    // 반환할 결과타입 정의
    ArrayList<Memo> data = new ArrayList<>();
    SQLiteDatabase con = helper.getReadableDatabase();
    Cursor cursor = con.rawQuery(query, null);

    while(cursor.moveToNext()){
        Memo memo = new Memo();
        //int index = cursor.getColumnIndex("id"); //id 이름의 컬럼이 몇번째인지
index를 가져오고
        //위에서 가져온 index로 실제 값을 가져와서 저장
        memo.id = cursor.getInt(0);       // query 에서 가져올 컬럼이 몇번째 지정
되었는지 확인
        memo.title = cursor.getString(1);
        memo.content = cursor.getString(2);
        memo.n_date = cursor.getString(3);

        data.add(memo);
    }
    con.close();
    return data;
}
```

close( )

※사용한 DAO를 닫아준다.

```
public void close(){
    helper.close();
}
```

Update , Delete 처리 필요

DBHelper extends SQLiteOpenHelper

* SQLite를 통한 DB 통신 실행작업을 관장하는 클래스
* 본 클래스는 해당 프로그램에 단 하나만 생성되어 작동해야 하는 조건이 있음

```
static final String DB_NAME = "sqlite.db";   // DB파일 명(바탕화면 생성)
static final int DB_VERSION = 1;             // DB버전
```

Constructor

```
public DBHelper(Context context){
    super(context, DB_NAME, null, DB_VERSION);
}
```

* ListActivity 클래스에서 init( ); 을 통해 MemoDAO 객체가 생성되고, 이 과정에서 ListActivity의 context가 전달된다.
* 해당 context를 받아 (SQLiteOpenHelper 클래스를 상속받는) DBHelper 데이터 타입의 helper 객체가 생성된다.
* 필드에 선언된 DB명과 DB version을 파라미터로 받아 생성자가 실행되며, 해당 DB파일이 생성되었는지 체크해
  → 파일이 없으면 onCreate를 호출
  → 파일이 있으면 버전 체크 → 버전이 기 생성된 DB보다 높을 경우 onUpgrade 호출

onCreate(SQLiteDatabase sqLiteDatabase)

* 최초로 생성할 테이블을 정의한다.
* DB가 업데이트 될 경우에 모든 히스토리가 쿼리에 반영 되어야 함(settings.config)

```
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {
    String createQuery = "CREATE TABLE `memo` ( \n" +
            "`id` INTEGER PRIMARY KEY AUTOINCREMENT, \n" +
            "`title` TEXT, \n" +
            "`content` TEXT, \n" +
            "`n_date` TEXT " +
            "'modified` TEXT )";
    sqLiteDatabase.execSQL(createQuery);   // 쿼리를 실행해서 테이블을 생성한다.
}
```

onUpgrade(SQLiteDatabase , int old, int newVersion)

* 변경된 버전과 현재를 비교 / 히스토리 반영 필요
* revision.config

```
@Override
public void onUpgrade(SQLiteDatabase sqLiteDatabase, int old, int newVersion) {
    // revision.config
    // 변경된 버전과 현재버전을 비교해서
    // 히스토리를 모두 반영해야 한다.
    if(old < 2) {
        // version 2
        // String alterQuery2 = "ALTER TABLE `memo` ( \n" +
        //                      Add Column modifed text);
    }
    if(old < 3) {
        // version 3
        // String alterQuery3 = "ALTER TABLE `memo` ( \n" +
        //                      Add Column count text);
    }
    if(old < 4) {
        // version 4
        // String alterQuery4 = "ALTER TABLE `memo` ( \n" +
        //                      Add Column members text);
    }
    if( old < 5) {
        // version 5
        // String alterQuery5 = "ALTER TABLE `memo` ( \n" +
        //                      Add Column found text);
    }
}
```