activity_list.xml

ListView widget

list_item.xml

Layout ID - itemView
No. ID - itemNo
Date ID - textDate

*List View에서 출력된 1개의 객체 단위

ListActivity

ListView listView; // activity_list.xml에 생성된 List View widget
Button btnPost; // activity_list.xml에 생성된 글쓰기 버튼
private static final int WRITE_ACTIVITY = 12345; // result code로 사용될 임의의 enum

onCreate

```
initView();                 // 글쓰기 Button, ListView 변수 할당
initListener();             // 글쓰기 Button에 setOnClickListenr 연결
init();                     // 게시판 목록 refresh
```

initView( )

```
private void initView() {
    listView = (ListView) findViewById(R.id.listView);
    btnPost = (Button) findViewById(R.id.btnPost);
}
```

* Xml의 ListView와 글쓰기 Button 변수 생성

initListener( ) ※ ListActivity의 글쓰기 버튼 클릭 시 글쓰기 페이지(WriteActivity)로 이동

```
private void initListener() {
    btnPost.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(ListActivity.this, WriteActivity.class);
            startActivityForResult(intent, WRITE_ACTIVITY);
        }
    });
}
```

* StartActivityForResult를 이용해 intent와 int타입 Request Code 를 전달하며 대상 activity를 실행한다.
* resultCode는 정수 타입(enum)이므로, 실행할 때는 필드에 선언한 WRITE_ACTIVITY에 임의의 정수를 할당해 사용한다.

onActivityResult( ) ※ startActivityForResult를 통해 호출된 activity가 종료될 때 호출되는 메서드

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data)
    switch(requestCode){
        case WRITE_ACTIVITY:
            if(resultCode == RESULT_OK) {
                init();
            }
            break;
    }
}
```

* init( ); 가 없을 경우, 새 post 등록이 되더라도 app을 종료 후 다시 구동해야 refresh가 된다.

init( ) ※ listView를 refrsh한다

```
private void init(){
    ArrayList<Memo> list = loadData();
    ListAdapter adapter = new ListAdapter(this, list);
    listView.setAdapter(adapter);
}
```

ArrayList<Memo> loadData() ※ 파일 목록에서 파일을 하나씩 꺼내 result에 담아 리턴

```
private ArrayList<Memo> loadData(){
    ArrayList<Memo> result = new ArrayList<>();
    for(File item : getFilesDir().listFiles()){
        try {
            String text = FileUtil.read(this, item.getName());
            Memo memo = new Memo(text);
            result.add(memo);
        } catch (IOException e) {
            Toast.makeText(this, "에러:"+e.toString(), Toast.LENGTH_SHORT).show();
        }
    }
    return result;
}
```

ListAdapter extends BaseAdapter

*기본 Adapter를 상속받아 data와 listView를 연결하는 객체

Context context;
ArrayList<Memo> data; // 파일 목록에서 불러온 내역을 저장할 Memo타입 변수

Constructor

```
public ListAdapter(Context context, ArrayList<Memo> data){
    this.context = context;
    this.data = data;
}
```

* ListActivity에서 init( ) 호출 시 사용할 adapter를 위한 생성자

getView

```
@Override
public View getView(int position, View view, ViewGroup viewGroup) {
    Holder holder = null;
    if(view == null){
        view = LayoutInflater.from(context).inflate(R.layout.item_list, null);
        holder = new Holder(view);
        view.setTag(holder);
    }else{
        holder = (Holder) view.getTag();
    }

    Memo memo = data.get(position);

    holder.setNo(memo.getNo());
    holder.setTitle(memo.getTitle());
    holder.setDate(memo.getDatetime());
    holder.setPosition(position);
    holder.setAuthor(memo.getAuthor());
    holder.setContent(memo.getContent());

    return view;
}
```

*메서드 getView는 listView에 포함된 item 각각을 호출해 화면에 출력하는 역할을 한다.(user 호출이 아닌, system 호출) > 화면에 가시적으로 보이는 리스트(동시에 n개)가 호출되고, 화면에 미 노출되는 item은 호출되지 않는다.

*getView가 실행되면 LayoutInflater를 통해 item_list.xml의 textview객체들이(textNo, textTitle, textDate) view 객체에 대입된다. 1개의 개별 view객체가 생성되는 동시에 , Holder 객체에 동일 view 객체가 대입되며 그 후 해당 view 객체는 view.setTag(holder)를 실행한다.

> 만일, holder 객체가 null이 아닌 view가 대입되어 있다면 해당 holder 객체에는 view.getTag(); 현 값이 저장되어 재활용

*Memo 클래스에서 추출한 no, title, datetime, author, content의 값을 저장한 후 view를 리턴한다.

기타 method

```
@Override
public int getCount() { // 현재 데이터의 총 개수
    return data.size();
}

@Override
public Object getItem(int position) { // 현재 출력할 데이터를 리턴
    return data.get(position);
}

@Override
public long getItemId(int position) {// 뷰의 아이디를 리턴해준다.
    return position;
}
```

Holder

int position;
String author, content;
TextView textNo, textTitle, textDate;

Constructor

```
public Holder(View view){
    textNo = view.findViewById(R.id.textNo);
    textTitle = view.findViewById(R.id.textTitle);
    textDate = view.findViewById(R.id.textDate);

    view.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(view.getContext(), DetailActivity.class);


            intent.putExtra("position", position);
            intent.putExtra("title",textTitle.getText());
            intent.putExtra("author", author);
            intent.putExtra("content", content);
            intent.putExtra("datetime", textDate.getText());

            view.getContext().startActivity(intent);
        }
    });
}
```

```
public void setAuthor(String author){
    this.author = author;
}
public void setContent(String content){
    this.content = content;
}
public void setPosition(int position){
    this.position = position;
}
public void setNo(int no) {
    textNo.setText(no + "");
}
public void setTitle(String title) {
    textTitle.setText(title);
}
public void setDate(Long datetime) {
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy/MM/dd HH:mm");
    String result = sdf.format(datetime);
    textDate.setText(result);
}
```

Memo

private static final String DELIMETER = "//"; **int no;** **String title, author, content;** **long datetime;**

Constructor

```
public Memo(){ }

public Memo(String text){
    String lines[] = text.split("\n");  // 1. 문자열을 줄("\n")단위로 분해
    for(String line : lines){  // 2. 문자열을 행("\n")단위로 분해
        String columns[] = line.split(DELIMETER);
        String key = "";
        String value = "";
        if(columns.length == 2) {
            key = columns[0];
            value = columns[1];
        }else{
            key = "";
            value = columns[0];
        }
        switch(key){
            case "no":
                setNo(Integer.parseInt(value));
                break;
            case "title":
                setTitle(value);
                break;
            case "author":
                setAuthor(value);
                break;
            case "datetime":
                setDatetime(Long.parseLong(value));
                break;
            case "content":
                setContent(value);
                break;
            default:
                appendContent(value);
        }
    }
}
```

```
public int getNo() {
    return no;
}
public void setNo(int no) {
    this.no = no;
}

public String getTitle() {
    return title;
}
public void setTitle(String title) {
    this.title = title;
}

public String getAuthor() {
    return author;
}
public void setAuthor(String author) {
    this.author = author;
}

public String getContent() {
    return content;
}
public void setContent(String content) {
    this.content = content;
}

public long getDatetime() {
    return datetime;
}
public void setDatetime(long datetime) {
    this.datetime = datetime;
}

public void appendContent(String value) {
    this.content += "\n"+value;
}

public String toString(){
    StringBuilder result = new StringBuilder();
    result.append("no").append(DELIMETER).append(no).append("\n");
    result.append("title").append(DELIMETER).append(title).append("\n");
    result.append("author").append(DELIMETER).append(author).append("\n");
    result.append("datetime").append(DELIMETER).append(datetime).append("\n");
    result.append("content").append(DELIMETER).append(content).append("\n");
    return result.toString();
}
public byte[] toBytes(){
    return toString().getBytes(); // 문자열에서 바이트배열로 변환
}
```

DetailActivity

TextView textTitle, textDate, textAuthor, textContent

OnCreate

```
initView();
init();
```

InitView( )

```
private void initView() {
    textTitle = (TextView) findViewById(R.id.textTitle);
    textDate = (TextView) findViewById(R.id.textDate);
    textAuthor = (TextView) findViewById(R.id.textAuthor);
    textContent = (TextView) findViewById(R.id.textContent);
}
```

Init( )

```
public void init(){
    Intent intent = getIntent();
    int position = intent.getIntExtra("position", -1);

    String title = intent.getStringExtra("title");
    String author = intent.getStringExtra("author");
    String content = intent.getStringExtra("content");
    String datetime = intent.getStringExtra("datetime");

    textTitle.setText(title);
    textDate.setText(datetime);
    textAuthor.setText(author);
    textContent.setText(content);
}
```

WriteActivity

Button btnPost;
EditText editTitle, editAuthor, editContent;

OnCreate

```
initView();
initListener();
```

InitView( )

```
private void initView() {
    btnPost = (Button) findViewById(R.id.btnPost);
    editTitle = (EditText) findViewById(R.id.editTitle);
    editAuthor = (EditText) findViewById(R.id.editAuthor);
    editContent = (EditText) findViewById(R.id.editContent);
}
```

InitListener( )

```
private void initListener() {
    btnPost.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Memo memo = getMemoFromScreen();
            write(memo);
        }
    });
}
```

Write(Memo memo)

```
private void write(Memo memo){
    try {
        String filename = System.currentTimeMillis() + ".txt";
        FileUtil.write(this, filename, memo.toString());

        setResult(RESULT_OK); // 나를 호출한 액티비티로 성공/실패 값을 넘겨준다.

        Toast.makeText(this, "등록되었습니다!", Toast.LENGTH_SHORT).show();
    } catch (IOException e) {
        Toast.makeText(this, "에러:"+e.toString(), Toast.LENGTH_SHORT).show();
    }

    finish();
}
```

Memo getMemoFromScreen()

```
private Memo getMemoFromScreen(){
    Memo memo = new Memo();
    memo.setNo(1);
    memo.setTitle(editTitle.getText().toString());
    memo.setAuthor(editAuthor.getText().toString());
    memo.setContent(editContent.getText().toString());
    memo.setDatetime(System.currentTimeMillis());
    return memo;
}
```

FileUtil

```
public static String read(Context context, String filename) throws IOException {
    StringBuilder sb = new StringBuilder();
    FileInputStream fis = null;
    BufferedInputStream bis = null;
    try {
        fis = context.openFileInput(filename);
        // 내용을 읽고
        bis = new BufferedInputStream(fis);
        // 한번에 읽어올 버퍼양을 설정
        byte buffer[] = new byte[1024];
        // 총 몇 바이트를 읽는 변수설정
        int count = 0;
        while( (count = bis.read(buffer)) != -1 ){
            String data = new String(buffer, 0, count);
            sb.append(data);
        }
    } catch (IOException e) {
        throw e;
    } finally {
        if(bis != null) {
            try {
                bis.close();
            } catch (IOException e) {
                throw e;
            }
        }
        if(fis != null) {
            try {
                fis.close();
            } catch (IOException e) {
                throw e;
            }
        }
    }
    return sb.toString();
}
```

```
public static void write(Context context,String filename, String content)
        throws IOException {
    FileOutputStream fos = null;
    try {
        fos = context.openFileOutput(filename, MODE_PRIVATE);
        fos.write(content.getBytes());
    }catch(Exception e){
        throw e;
    }finally {
        if(fos != null){
            try {
                fos.close();
            } catch (IOException e) {
                throw e;
            }
        }
    }
}
```